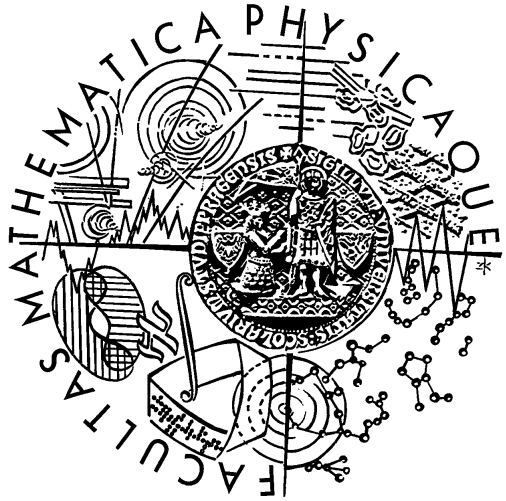
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



David Štěrba

Algoritmy pro průnikové grafy

Katedra aplikované matematiky
Vedoucí diplomové práce: RNDr. Jiří Fiala, PhD.

Studijní program: Informatika

Studijní obor: Diskrétní modely a algoritmy

Praha 2005

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu diplomové práce za řadu podnětných nápadů, pomoc při odstraňování chyb, za náměty pro zdokonalování práce a za řadu cenných rad.

Kontakt

E-mail: *dave@jikos.cz*
Diplomová práce: *http://dave.jikos.cz/diplomka/*

Prohlášení

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

V Praze dne 14. prosince 2005 David Štěrba

Obsah

Název práce: Algoritmy pro průnikové grafy

Autor: David Štěrba

Katedra (ústav): Katedra aplikované matematiky

Vedoucí diplomové práce: RNDr. Jiří Fiala, PhD.

e-mail vedoucího: fiala@kam.mff.cuni.cz

Abstrakt: Cílem této práce je předvést techniky řešení problému hledání maximální nezávislé množiny na průnikových grafech disků v rovině. V průnikovém grafu disků odpovídají vrcholům disky a dva vrcholy jsou sousední, právě když příslušné disky mají neprázdný průnik. Hlavní část práce je věnována aproximačním algoritmům a heuristikám (posouvání mřížky, zakázaný podgraf, omezené okolí vrcholu). Podáme přehled tříd diskových grafů (obecné a jednotkové diskové grafy, grafy s omezeným poloměrem disků). Tyto třídy se studují a modelují se na nich praktické problémy. Krátce zmíníme příklady možného praktického využití (značkování map, facility placement). Součástí práce je program, který demonstruje vybrané algoritmy a heuristiky.

Klíčová slova: algoritmus, heuristiky, průnikové grafy, diskové grafy

Title: Algorithms for intersection graphs

Author: David Štěrba

Department: Department of Applied Mathematics

Supervisor: RNDr. Jiří Fiala, PhD.

Supervisor's e-mail address: fiala@kam.mff.cuni.cz

Abstract: The goal of this thesis is to show techniques of solving the maximum independent set problem on intersection graphs of disks in the plane. An intersection graph is a graph whose vertices are represented by disks and two vertices are adjacent if and only if the corresponding disks have non-empty intersection. The main section of the paper is dedicated to the approximation algorithms and heuristics (grid shifting, forbidden subgraph, bounded neighborhood of a vertex). We will give an overview of disk graph classes (general and unit disk graphs, graphs with bounded radius of disks). These classes are studied and used to model practical problems. We will briefly describe some of these applications (map labelling, facility placement). A program demonstrating several algorithms and heuristics is enclosed with this work.

Keywords: algorithm, heuristics, intersection graphs, disk graphs

Seznam obrázků

Kapitola 1

Úvod

Cílem této práce je předvést techniky a přístupy k řešení obtížných problémů na průnikových grafech geometrických objektů, konkrétně **aproximace maximální nezávislé množiny na průnikových grafech disků**.

Motivací je řešení praktických úloh např. v telekomunikacích nebo v geografii, jak uvidíme v následující sekci.

Centrálním problémem je rozmístění nebo vybrání takových objektů, které se vzájemně nepřekrývají, nezasahují do sebe, jejich vzdálenost je větší než předepsaná hranice, nebo jinak spolu nekolidují/neinterferují. U každého objektu je definována oblast kolize. Objekt a oblast můžeme ztotožnit při vytváření geometrického modelu.

Geometrický objekt kruh/disk v rovině je jednoduše definovatelný. Obecněji se průnikové grafy zabývají průniky přímek, rovnoběžníků, křivek, mnohostěnů, „tlustých objektů“ atd., a to v malých dimenzích (≤ 3) nebo ve vyšších [25].

Průnikový graf je abstrakce, ve které vrcholy v grafu odpovídají jednotlivým objektům a hrana spojuje dva vrcholy, když se protínají objekty příslušné vrcholům. Soubor objektů, jejich tvarů a umístění se nazývá reprezentace průnikového grafu.

V případě, kdy je reprezentace dána spolu se vstupem, lze využít informace o pozicích objektů k zrychlení algoritmů.

Existují případy, kdy není znám algoritmus, který by řešil úlohu bez reprezentace a přitom byl srovnatelný s algoritmem, který reprezentaci využívá.

Nalezení reprezentace daného grafu je NP-těžký problém.

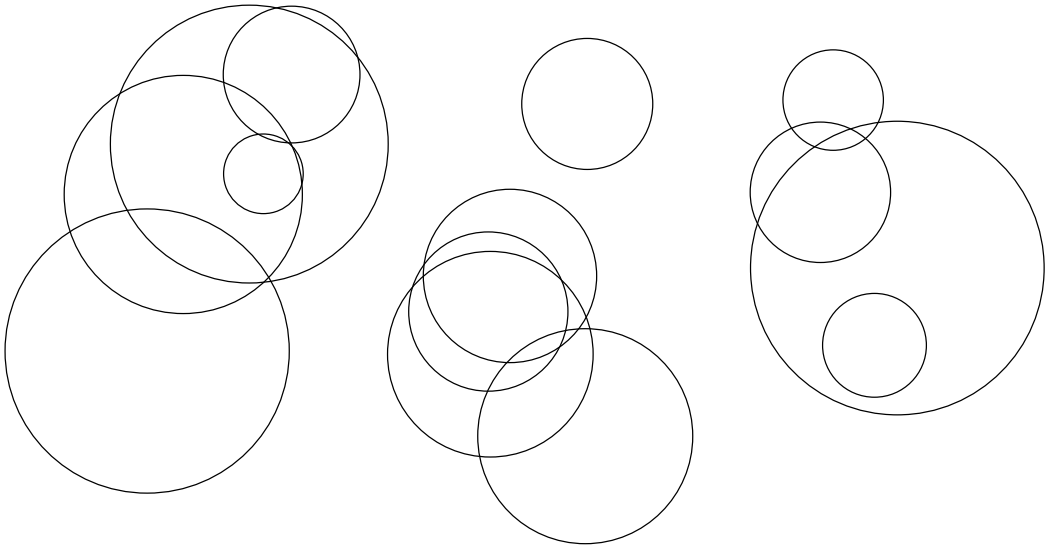
Tento problém je NP-úplný na obecných grafech a NP-úplný zůstává i na nejjednodušší třídě diskových grafů, tzv. jednotkových diskových grafech, ať jsou dány s reprezentací nebo bez.

Budeme se zabývat aproximacemi řešení, neboť z povahy problému neexistuje efektivní algoritmus k vyřešení otázky. Seznámíme se se základními třídami diskových grafů a zmíníme problémy, které se řeší a jaké mohou mít praktické aplikace. V kapitole 4 si ukážeme techniky, které se k aproximacím používají.

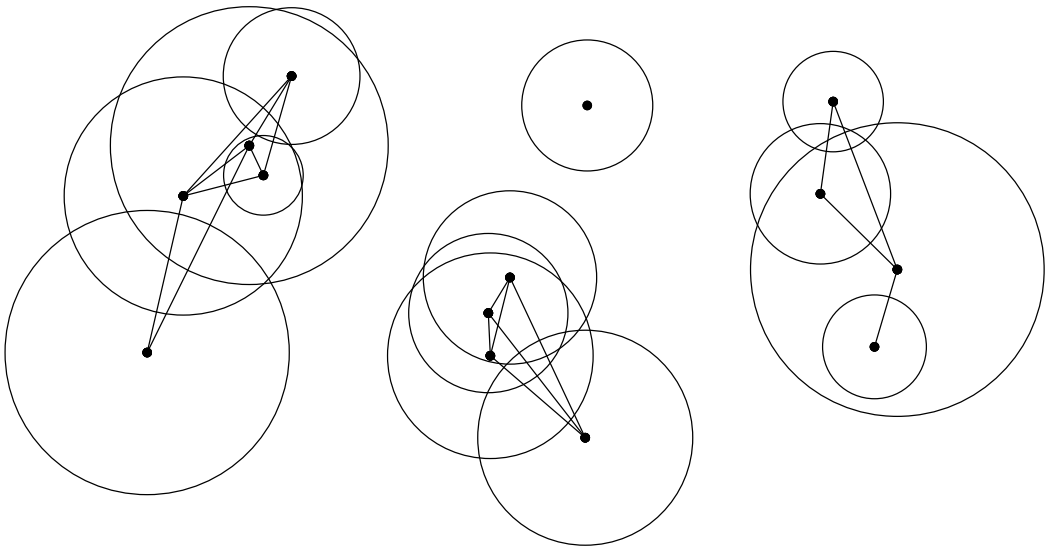
Součástí a přílohou k práci je program, ve kterém jsou demonstrovány některé techniky a algoritmy zmiňované dále.

Předpokládáme, že čtenář má základní znalosti v teorii grafů a výpočetní složitosti. Stručné shrnutí potřebných pojmů je v sekci *Definice a značení*.

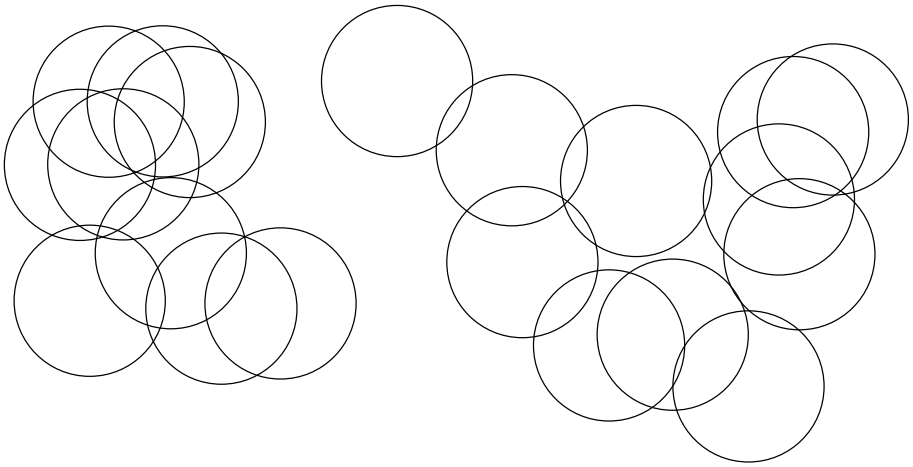
Na následujících stránkách jsou pro názornost příklady průnikových grafů disků s libovolným a jednotkovým poloměrem.



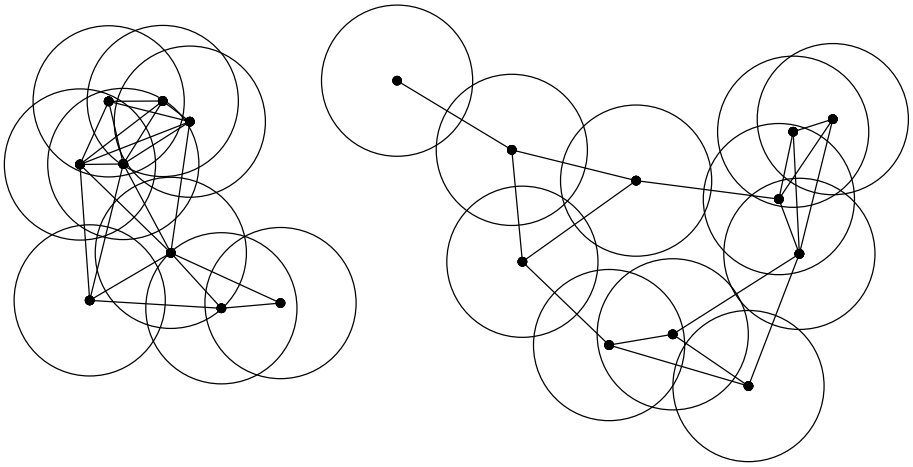
Obrázek 1.1: Příklad množiny disků s libovolným poloměrem



Obrázek 1.2: Příklad průnikového grafu disků s libovolným poloměrem



Obrázek 1.3: Příklad množiny disků s jednotkovým poloměrem



Obrázek 1.4: Příklad průnikového grafu jednotkových disků

1.1 Užití v praxi

Průnikové grafy geometrických objektů jsou teoreticky studovány a používány k modelování praktických problémů. Řešené problémy jsou výpočetně těžké, neexistuje efektivní algoritmus pro nalezení optimálního řešení. Východiskem jsou aproximační algoritmy a obecně použitelné heuristiky nebo naopak konkrétní algoritmy šité na míru jednotlivým situacím.

Obecné techniky mají mnohem širší použití (s drobnými úpravami je lze aplikovat na další typy problémů). Nevyužívají všech strukturálních vlastností, a proto jsou (až řádově) pomalejší než specifické algoritmy. Kvalita řešení těchto dvou přístupů je často srovnatelná. Budeme se zabývat obecnými technikami.

Následují příklady praktických aplikací průnikových grafů se stručným popisem problému (nebo naznačením způsobu řešení) a odkazy na literaturu, kde se lze dočíst více.

Přidělování frekvencí (Frequency assignment problem)
Máme pevně danou konfiguraci vysílačů v rovině. Každý vysílač přenáší radiový signál (nebo více signálů) o dané frekvenci. Signál má nějaký dosah, kde je detekovatelný příjemci. Signály vysílačů mohou interferovat, pokud se překrývají oblasti dosahu, tj. pokud jsou vysílače blízko sebe. Úkolem je vybrat takové frekvence, které jsou dostatečně vzdálené v radiovém spektru, aby bylo možné je rozlišit v oblasti interference. Druhým požadavkem je minimalizace počtu frekvencí. Za předpokladu, že terén je rovinný a signál je šířen rovnoměrně všemi směry, jsou oblasti dosahu vysílačů modelovány jako disky o stejném poloměru. Předpokládáme, že frekvence lze reprezentovat jako celá čísla, která se v případě interference musí lišit. Řešení je založeno na minimálním dobrém obarvení vrcholů průnikového grafu.

Tento problém, jinak známý jako *radio channel assignment* nebo *frequency allocation*, je dobře prozkoumaný, viz např. [23, 28].

Značkování map (Map labelling)
Úkolem je na mapu přiřadit značky k objektům tak, aby se nepřekrývaly. Objekty rozumíme např. hlavní města, řeky, hory. Některé objekty jsou významnější a značka by jim měla být přiřazena přednostně. Příkladem je hlavní město, které chceme znát raději než názvy řek či hor.
Značky jsou obdélníkového tvaru a jsou reprezentovány obdélníkem nebo elipsou. Každá značka má svou váhu, která udává její důležitost na mapě. Řešením jsou značky, které v průnikovém grafu tvoří nezávislou množinu,

tj. ty obdélníky, které se neprotínají. Požadavek na maximální váhu by měl zajistit, že budou vybrány významnější objekty. Samozřejmě, že na zvolení vah velmi záleží.

Odkazy a názorné příklady týkající se tohoto problému můžete najít na internetové stránce [38]. O obecnějších problémech značkování (např. umisťování textu značek na křiku, libovolné natočení značky) pojednává článek [9].

Rozmisťování zařízení (Facility placement)
Zařízení jsou modelována jako disky (buď s jednotkovým nebo libovolným poloměrem). Úkolem je najít co nejvíc z daných míst taková, na která se zařízení umístí a nezasahují jedno do druhého [36].
Jinou rozmísťovací úlohou je najít mezi body v rovině množinu, od které bude každý bod vzdálen nejvýše nějakou mez. Jako body lze uvažovat například hasičské základny. Řešením je nalezení minimální dominující množiny (Minimum Dominating Set).

Směrování v ad-hoc sítích a v *sensor* sítích
Při použití bezdrátových ad-hoc sítí a *sensor* sítí je důležitá znalost umístění. Pro geometrické směrování postačuje znát virtuální souřadnice (narozdíl od reálných). Ze znalosti propojení jednotlivých uzlů je možné získat virtuální souřadnice [26, 37]. Samotné směrování dat probíhá pouze mezi přímo viditelnými vysílači. Určení směrovací páteře lze řešit jako problém minimální souvislé dominující množiny (Minimum Connected Dominating Set).

1.2 Definice a značení

V této části uvedeme některé pojmy, které budeme dále používat a které se netýkají přímo průnikových grafů. Termín uvedený v závorce za definovaným pojmem je používaný anglický originál.

Grafy

Graf $G = (V, E)$ je dvojice, kde V je množina vrcholů a E množina hran. Dále budeme označovat n velikost V a m velikost E.

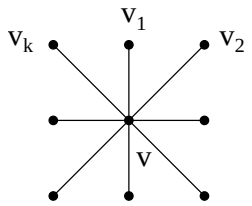
Graf $\overline{G} = (V, E')$ je *doplňkový* ke grafu $G = (V, E)$, ve kterém $e' = \{x, y\} \in E' \Leftrightarrow e = (x, y) \notin E$.

Označíme $G[C]$ *indukovaný podgraf* grafu G na množině $C \subseteq V$: $G[C] = (C, \{\{u, v\} \in E : u, v \in C\})$.

Množinu sousedů vrcholu $v \in G$ v grafu $G = (V, E)$ označíme $N(v) = \{u \in V \ : \{u, v\} \in E\}$.

Podmnožina $C \subseteq V$ se nazývá *nezávislá*, pokud mezi žádnou dvojicí vrcholů z C není hrana. Podmnožina $C \subseteq V$ se nazývá *klika* nebo *úplný graf*, jestliže mezi všemi dvojicemi vrcholů z C jsou hrany.

Bipartitní graf $K_{1,k}$ nazveme *k-hvězda* (*k-claw*), neboť ho můžeme nakreslit jako vrchol, ze kterého vycházejí hrany do zbylých vrcholů (1.5).



Obrázek 1.5: Nakreslení grafu $K_{1,k}$ do hvězdy.

Algoritmy

Stručně zavedeme pojmy týkající se algoritmů a ukážeme, jak je budeme dále používat. Podrobnější pohled na výpočetní složitost lze nalézt v [11].

Abeceda je konečná množina symbolů. *Slovo* je konečná posloupnost symbolů abecedy. *Jazyk* je množina všech slov nad danou abecedou. *Problém* $\mathcal{Q}$ (v abstraktním smyslu) je binární relace na množině instancí problému a množině řešení problému. Instance je slovo ve vstupním jazyce, řešení je slovo ve výstupním jazyce. *Algoritmus* je dobře definovaný výpočetní postup, který na vstupu dostane instanci problému a na výstupu vydá řešení asociované se vstupem. *Časovou složitost* algoritmu měříme v počtu elementárních kroků vykonaných při transformaci vstupu na výstup. Prostorovou složitostí se nebudeme zabývat. Dále v textu budeme *složitostí* myslet časovou složitost.
Algoritmus řešící *dotazovací problém* dostane na vstupu slovo a číslo k. Na výstupu vydá odpověď Ano/Ne, pokud existuje řešení menší než k. Algoritmus řešící *optimalizační problém* dostane na vstupu slovo. Výstupem bude řešení nejmenší velikosti.

Třída P obsahuje všechny problémy, které lze řešit algoritmy se složitostí omezenou polynomem závislým na velikosti vstupu. Problém patří do třídy NP, pokud existuje polynomiální algoritmus A, který ověří, že dané slovo je řešením problému. Problém je *NP-těžký*, pokud existuje polynomiální převod na libovolný problém třídy NP. Problém je *NP-úplný*, pokud je NP-těžký a patří do třídy NP.

Budeme chápat vstup jako graf zadaný běžnou formou (matice sousednosti nebo seznam následníků) plus dodatečné vstupní parametry (konstanty, vrcholy, hrany). Výstupem řešení rozhodovacího problému bude odpověď Ano/Ne, u optimalizačního problému budeme uvažovat např. množinu vrcholů nebo číslo (součet vah).

Hledat aproximační algoritmy má smysl jen u optimalizačních problémů.

Aproximační algoritmus najde přijatelné řešení v polynomiálním čase vzhledem ke vstupu. Algoritmus má aproximační poměr ρ, jestliže pro každý vstup je nalezené řešení nejvýše $\rho \times$ *optimum* (pro minimalizační úlohu) nebo alespoň $1/\rho \times$ *optimum* (pro maximalizační úlohu).

PTAS (Polynomial-Time Approximation Scheme) je algoritmus, který pro danou instanci problému a parametr $\epsilon > 0$ vydá přijatelné řešení, které je nejvýše $(1 + \epsilon)$-krát horší než optimum a čas výpočtu je polynomiální vzhledem k velikosti vstupu, exponent smí však záviset na ϵ.

On-line algoritmus přijímá vstup po částech: v jednom kroku dostane nový vrchol, incidentní hrany a ihned musí vydat řešení. Navíc, toto řešení musí obsahovat předchozí řešení. Algoritmus rozhoduje bez znalosti příštích částí vstupu. Kvalita řešení se udává *kompetitivním poměrem* ρ — řešení je nejvýše ρ-krát horší než optimum.

Kapitola 2

Diskové grafy

V této kapitole se podrobněji seznámíme s třídami diskových grafů, které se teoreticky studují. Zadefinujeme základní diskové třídy a jejich specializace pomocí dodatečných omezujících podmínek. Zmíníme další průnikové modely, se kterými se lze u diskových grafů setkat.

2.1 Definice

Nechť je Σ systém množin a $\mathcal{G}$ je třída grafů. Pro každý podsystém $\mathcal{F} \subseteq \Sigma$ *průnikový graf* z $\mathcal{F}$, označený Ω, je graf s množinou vrcholů $\mathcal{F}$. Vrcholy $U, V \in \mathcal{F} (U \neq V)$ sousedí, pokud jejich průnik je neprázdný. *Průniková třída* Σ značená $\Omega(\Sigma)$ je třídou grafů $\{\Omega(\mathcal{F}) | \mathcal{F} \subseteq \Sigma\}$. Řekneme, že $\mathcal{G}$ je průniková třída, jestliže $\mathcal{G}$ je isomorfní $\Omega(\Sigma)$ pro nějaké Σ. Více o průnikových grafech viz [25]. Formální zavedení průnikových grafů pro nás není zásadní, budeme se dále zabývat třídami, kde Σ jsou uzavřené disky v Euklidovské rovině.

Vzdálenost dvou bodů A a B měříme euklidovskou normou, značíme $dist(A, B)$.

Disk $D = (x, y, r)$ v Euklidovské rovině je množina bodů, které mají vzdálenost nejvýše r od středu $c_D = (x, y)$. Uvažujeme jen uzavřené disky.

Průnikovým grafem disků nazveme graf $G = (V, E)$, ve kterém vrcholům jednoznačně odpovídají disky v rovině. Hrana spojuje dva vrcholy, právě když se odpovídající disky protínají. (Další průnikové modely jsou ukázány v jedné z následujících sekcí.)

Geometrickou *reprezentací* průnikového grafu rozumíme soubor geometrických objektů, ze kterých odvozujeme graf. Pro zjednodušení, nerozlišujeme mezi vrcholem a odpovídajícím objektem.

Graf G se nazývá *diskový*, pokud existuje množina disků $\mathcal{D}$ v rovině, že G je její průnikový graf. Množinu $\mathcal{D}$ nazveme *disková reprezentace*, *diskový model*, *realizace* grafu G. Indukovaný podgraf diskového grafu je též diskový.

Definujme σ jako $\sigma = \frac{\text{poloměr největšího disku}}{\text{poloměr nejmenšího disku}}$, $\sigma \geq 1$.

Zřejmě můžeme všechny poloměry disků přenásobit konstantou tak, že poloměr nejmenšího disku bude vždy 1, nebo jiné číslo, je-li to potřeba. Získáme následující významné třídy diskových grafů:

- jednotkové diskové grafy (*unit disk graphs*, UDG): poloměr všech disků je stejný, např. 1
- obecné diskové grafy (*general disk graphs*, DG): poloměr disků je libovolný, σ není omezené

Další podmínky, které mohou specializovat diskové třídy a o kterých bude řeč dále, jsou tyto:

- σ-omezené (σ-*bounded*, DG_σ): poloměry všech disků jsou z uzavřeného intervalu $[1, \sigma], 1 \leq \sigma$, σ je omezené
- λ-přesné (λ-*precision*, DG_λ): λ je minimální vzdálenost středů dvou disků, poloměr nejmenšího disku je 1
- $\lambda\sigma$-omezené ($\lambda\sigma$-*bounded*, $\mathrm{DG}_{\lambda\sigma}$): spojení předchozích omezení definuje třídu, kde poloměr nejmenšího disku je 1, největšího σ a středy disků jsou vzdáleny nejméně λ

Můžeme zkoumat způsob průniku disků. Pokud se disky pouze dotýkají, hovoříme o *dotykové* reprezentaci *(disk contact graphs)*. Lze se také setkat s označením *interior disjoint graphs*, „vnitřkově disjunktní grafy“. Pro jednotkové dotykové grafy je běžné označení *penny graphs* a pro dotykové grafy s libovolným poloměrem označení *coin graphs*.

Třídám se budeme věnovat hlouběji v dalších oddílech.

2.2 Reprezentace

Reprezentace hraje důležitou roli při navrhování algoritmů pro průnikové grafy. Některé z problémů zmíněných v předchozí sekci lze řešit efektivně s nebo bez reprezentace, avšak u některých má přítomnost reprezentace podstatný vliv. Geometricky lze reprezentaci dělit na menší podúlohy. Setkáme se s heuristikami, které využívají setříděného seznamu disků k rychlému nalezení vhodného kandidáta pro hladový algoritmus.

Když není reprezentace k dispozici, reálnou situaci můžeme modelovat např. průnikovým grafem jednotkových disků. Na takové třídě grafů lze aproximovat problém hledání maximální nezávislé množiny lépe než modelováním na obecných grafech. Dokonce problém hledání maximální kliky je na UDG polynomiálně řešitelný optimálně.

Budeme prezentovat heuristiky, které jsou podobně vystavěny podle tohoto postupu:

1. vypozorujeme lokální nebo globální vlastnost
2. heuristika využije vlastnost
   - *s reprezentací:* najde kandidáta podle geometrických vlastností
   - *bez reprezentace:* kandidát je nalezen podle vlastností v grafu
3. opakovaně je hledán vhodný kandidát, od něhož lze pokračovat

Pro tento typ heuristik je důležité, aby geometrická vlastnost, kterou má kandidát splňovat, měla svůj grafový ekvivalent.

Rozdíl v rychlosti se projeví v nalezení kandidáta. S předzpracováním reprezentace (například setříděním) je nalezení konstantní, zatímco enumerace okolí vrcholu a hledání indukovaného podgrafu je operace řádově pomalejší.

Příkladem heuristiky nevyužívající reprezentace je například polynomiální odhad na počet disjunktních objektů omezené velikosti, které lze vměstnat do omezené oblasti a které se nepřekrývají. Viz sekce 4.4 *Omezené okolí vrcholu*.

2.2.1 Rozpoznávání

Převedení diskové reprezentace na graf je v obecném případě kvadratické vzhledem k počtu disků $O(n^2)$: iterací přes všechny dvojice disků se otestuje průnik.

U řídkých grafů lze tento odhad vylepšit až na $O(n + m)$. V základní verzi algoritmu *sweep line* se disky setřídí podle x-ové souřadnice středů a rovina se rozdělí na pruhy ve směru x-ové osy. Disky protínající pruhy se udržují ve vhodné datové struktuře. Průniky disků se testují vždy ve dvou sousedních pruzích. Složitost tohoto algoritmu je $O(n \log n + m)$.
Více viz [5], kde se diskutují různé verze algoritmu *sweep line*.

Opačná otázka, tj. zda k danému grafu lze nalézt jeho diskovou reprezentaci — rozpoznatelnost třídy grafů — je výpočetně těžká. Už pro nejjednodušší třídu diskových grafů, UDG, je tato otázka NP-těžká [6]. Totéž platí pro disky s omezeným poloměrem [4] a pro obecné diskové grafy [15].

Dosud se neví, zda je tento problém ve třídě NP a je-li tím pádem NP-úplný. V současnosti je tato otázka intenzivně studována. Ví se, že rozpoznávání UDG je ve třídě PSPACE [6]. Naivní přístup „uhodnout reprezentaci a ověřit“ nefunguje, neboť není známo, jestli je vždy možné souřadnice a poloměry disků vyjádřit polynomiálním počtem bitů.

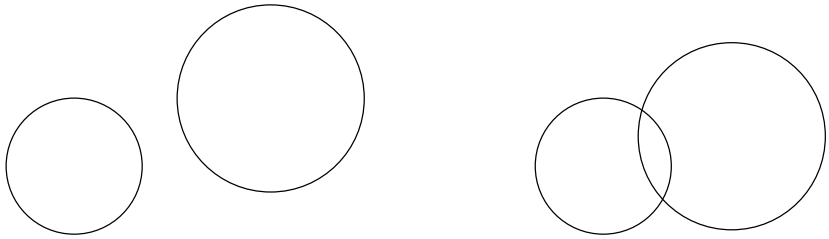
Pro praktické účely je důležitá otázka nalezení přibližných (virtuálních) souřadnic pro zadaný graf. Nalezení virtuálních souřadnic reprezentace UDG lze přeformulovat jako lineární, kvadratické nebo semidefinite programování [32] (*feasibility of linear/quadratic/semidefinite program*). Tento problém je stále NP-těžký (důkaz se opírá o techniku z [6]).

2.2.2 Průnikové modely

Dosud uvažovaný způsob průniku disků byl přirozený: disky se jednoduše protnou. Podívejme se na další průnikové modely, se kterými se lze setkat.

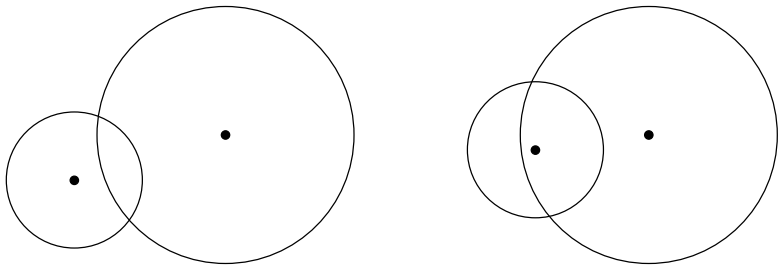
Nejběžnější a nejvíce studovaný je *intersection model*. Hrana spojuje 2 vrcholy, právě když jsou středy disků vzdáleny nejvýše $(r_1 + r_2)$ (obr. 2.1). Dotyk disků považujeme za průnik.

Druhým způsobem definujeme hranu mezi dvěma vrcholy, právě když alespoň jeden disk obsahuje střed druhého. Také se na to lze dívat tak, že středy jsou



Obrázek 2.1: Průniky v *intersection* modelu. V případě a) se disky neprotínají, v b) protínají

od sebe vzdáleny nejvýše $\max\{r_1, r_2\}$, tzv. *proximity model* (obr. 2.2).



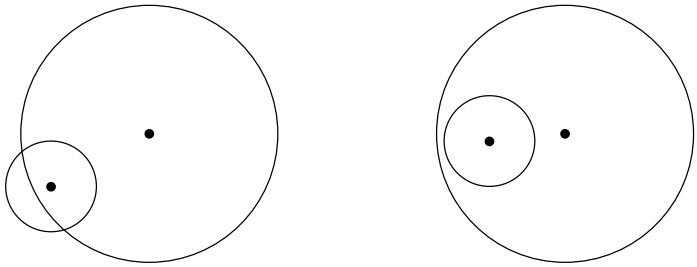
Obrázek 2.2: Průniky v *proximity* modelu. V případě a) se disky neprotínají, v b) protínají

Třetím způsobem definujeme hranu, pokud je jeden disk obsažen v druhém, tzv. *containment model*. Středy disků jsou od sebe vzdáleny nejvýše $\max\{r_1 - r_2, r_2 - r_1\}$ (obr. 2.3).

Pro realističtější model vysílačů bylo nutné odlišit oblast, ve které lze signál přijímat a oblast (obvykle větší), kde signály interferují mezi sebou. Vzájemné rušení vysílačů nastane, pokud se protne oblast vysílání jednoho vysílače s oblastí interference druhého vysílače.

Odvozený model je nazván *double disk* — dva disky jsou přiřazeny jednomu vrcholu (obr. 2.4). Disk v tomto modelu je definován jako $D = (x, y, r, R), r \leq R$. Hrana spojuje vrcholy, právě když středy disků jsou od sebe vzdálené nejvýše $\max\{r_1 + R_2, R_1 + r_2\}$. Tento model zobecňuje všechny předchozí.

Double disk model má pro UDG vlastní název *d-quasi UDG* (*d*-QUDG). Poloměr vnějšího disku je pevně dán 1 a $r \equiv d$. Definice přítomnosti hrany



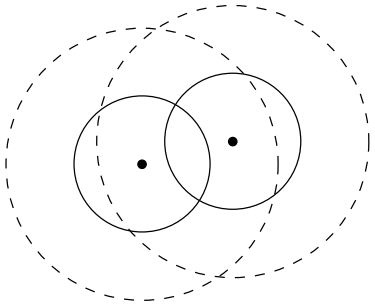
Obrázek 2.3: Průniky v *containment* modelu. V případě a) se disky neprotínají, v b) protínají

Obrázek 2.4: Průniky v modelu *double disk*. V případě a) se „disky“ neprotínají, v b) protínají

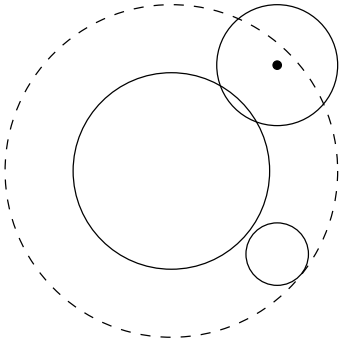
není striktně vymezena. Hrana v grafu není, pokud se vnější disky neprotínají (vzdálenost středů je > 2). Hrana spojuje vrcholy, jestliže vzdálenost středů je $\leq 2d$. Ve zbylém případě, vzdálenost středů je v intervalu $(2d, 2]$, není explicitně řečeno, zda hrana v grafu je či není.

Ve třídě UDG jsou na sebe vzájemně převoditelné modely *intersection* a *proximity*. (Model *containment* nemá smysl brát v úvahu.) Převod modelu *intersection* na *proximity* se provede zdvojnásobením poloměrů. Podmínka průniku disků *proximity* modelu bude splněna: střed disku je obsažen v druhém disku. Převod opačným směrem je zřejmý. Jelikož mají všechny disky stejný poloměr, je převod korektní.

Pro obecné DG má smysl uvažovat *containment* model. Jednotlivé modely na sebe nejsou převoditelné transformací poloměru. Změnu poloměru disku nelze provést uniformně tak, aby byly zachovány průniky. Protipříkladem nepřevoditelnosti *proximity* na *intersection* model je graf $K_{3,3}$, který lze realizovat v *proximity* modelu, a nikoli v *intersection* modelu. Předpokládá se, že existuje i graf, který není *proximity* a je *intersection* -model graf. Doposud



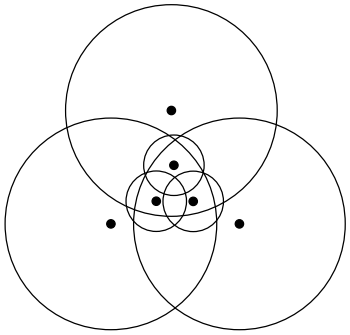
Obrázek 2.5: Převod *intersection* na *proximity* model v UDG



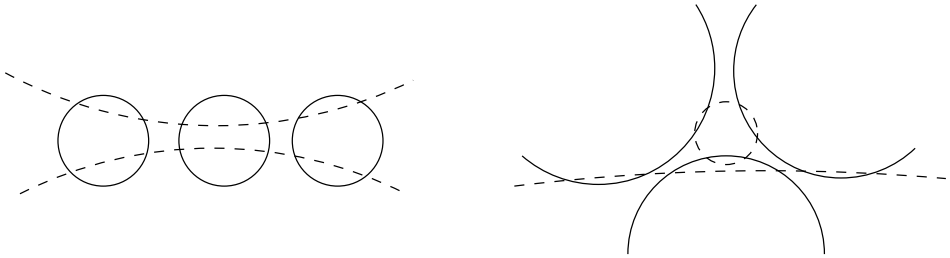
Obrázek 2.6: Převod *intersection* na *proximity* model není pro DG možný jednoduchou transformací poloměrů. Při zvětšení poloměru se může objevit průnik, který v původním modelu není.

není znám.

Nadále budeme uvažovat **intersection model**, jelikož je to model nejvíce prozkoumaný. Aproximační algoritmy a heuristiky uvedené dále v této práci jé možné přizpůsobit i ostatním modelům [22].



Obrázek 2.7: Realizace grafu $K_{3,3}$ v *proximity* modelu



Obrázek 2.8: Dvě konfigurace disků, do kterých se má přidat po jednom disku, aby bylo dosaženo realizace $K_{3,3}$ v *intersection* modelu. Disk musí protínat nečárkované a nesmí protínat čárkované disky. Z obrázků je zřejmé, že to nelze.

2.3 Verze problémů

U jednotlivých tříd budou popsány zejména vlastnosti a řešení dílčích úloh, které budeme využívat v heuristikách. Struktura tříd je bohatá: třídy bez omezení (DG, UDG) nebo závislé na parametru (q-UDG, DG_λ, DG_σ). Uvedeme je v pořadí od obecnějších k specializovaným.

2.3.1 Obecné diskové grafy (DG)

Nejobecnější třída průnikových grafů disků v rovině s libovolným poloměrem. Nemá žádná omezení na poloměry nebo rozmístění disků. Rozpoznatelnost třídy a problém hledání maximální nezávislé množiny jsou NP-úplné. Poznáme jen jednu jednoduchou heuristiku, která využívá lokálních vlastností vrcholu. Existuje PTAS algoritmus pro řešení problému hledání maximální (vážené) nezávislé množiny, který potřebuje geometrickou reprezentaci. Otevřeným problémem zůstává, zda existuje i PTAS, který reprezentaci nepotřebuje.

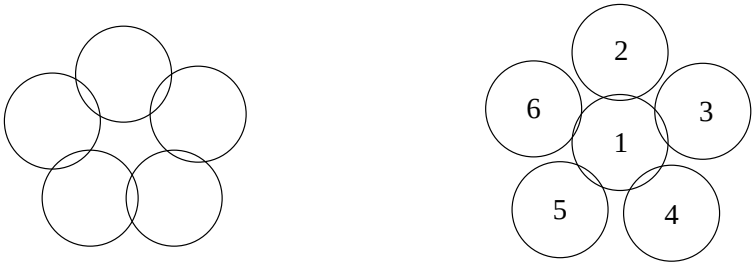
2.3.2 Jednotkové diskové grafy (UDG)

Jednotkové diskové grafy jsou průnikovým modelem disků se stejným poloměrem. Tato třída je nejjednodušší z diskových grafů (bez dalších omezujících přívlastků) a můžeme na ní ukázat některé vlastnosti, které platí pro obecné diskové grafy.

Víme, že není *perfektní*, neboť obsahuje pěticyklus (obr. 2.9a). Obsahuje *kliku libovolné velikosti*, tedy i nerovinné grafy a nelze použít efektivní algoritmy pro rovinné grafy. Třída rovinných grafů omezuje zdola složitostní odhady pro UDG. Tato třída neobsahuje 6-hvězdu (*6-claw free*) [31].

Hledání maximální nezávislé množiny je na této třídě NP-těžký problém [8]. Je známo, že nalezení maximální kliky v UDG s reprezentací je polynomiální $O(n^{4.5})$ [8]. Využitím speciálních vlastností párování v geometrických grafech se získá $O(n^{3.5} \log n)$ [5]. Vylepšením prvního algoritmu lze dosáhnout časové složitosti $O(n\omega^3)$ [23]. (Problém lze řešit polynomiálně i bez reprezentace, viz dále.)

Uvedeme algoritmus z [8] a později ho využijeme jako proceduru v heuristice



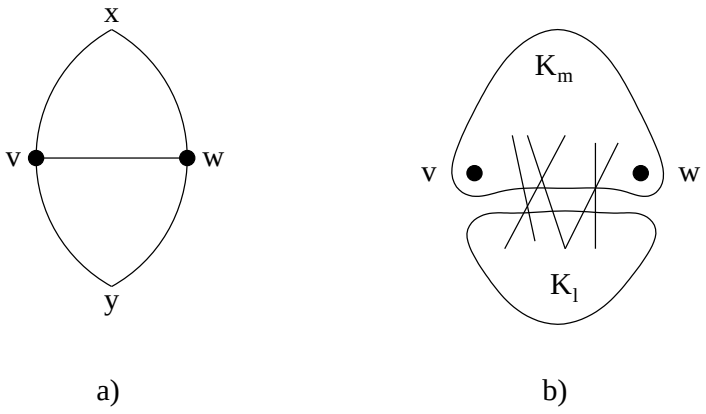
Obrázek 2.9: UDG obsahuje pěticyklus (a), okolí disku v UDG nemůže obsahovat IS velikosti 6 (b)

založené na trhání klik.

Maximální klika v UDG s reprezentací (podle [8]) Řešení vychází z následujících dvou pozorování (obr. 2.10). Oblouky $\overset{\frown}{xwy}$ a $\overset{\frown}{yvx}$ mají poloměr $dist(v, w)$ a středy v v a w.

Pozorování 1: jsou-li v, w sousední vrcholy, množina $C \subseteq V$ nějaká maximální klika, pak $C \subseteq H_{vw} = \{a \in V : \text{střed disku } a \text{ leží v oblasti } \widehat{vxwy}\}$.

Pozorování 2: pro každé v, w sousední vrcholy je graf indukovaný na H_{vw}, doplněk bipartitního grafu.



Obrázek 2.10: Lokální situace v okolí maximální kliky v UDG. Sousední vrcholy v, w a jejich geometrické okolí (a). Indukovaný podgraf (doplněk bipartitního grafu) (b).

Vrcholy z oblasti $\widehat{vxw}$ indukují úplný podgraf K_m (symetricky $\widehat{vwy}$ K_l) spojené nějakými hranami (2.10b). Doplňkový graf je bipartitní.

Podle pozorování 1 existuje dvojice vrcholů v', w' takových, že graf indukovaný na $H_{v'w'}$ je maximální klika. Najdeme ji jako největší ze všech maximálních klik na všech indukovaných podgrafech G_{vw}, pro všechny dvojice sousedních vrcholů v, w. Implementováno podle [23] v čase $O(n^{4.5})$: klika v doplňkovém (bipartitním) grafu je maximální nezávislá množina, kterou lze získat z každého maximálního párovaní. Problém maximálního párování v bipartitním grafu je řešitelný technikou Hopcroft-Karpa [17] (se složitostí $O(m\sqrt{n})$) v čase $O(n^{2.5})$. Celkem $O(n^{4.5})$.

Důsledkem je fakt, že maximální stupeň vrcholu v UDG je $6\omega - 6$, kde ω je velikost maximální kliky v G. Vylepšení uvedené v [23] pracuje na celém okolí zvoleného vrcholu a párování hledá postupným rozšiřováním již nalezených.

Hledání kliky na UDG je polynomiální i bez reprezentace [30]. Poněkud rozporuplný výsledek říká, že uvedený (robustní) algoritmus najde maximální kliku na grafu nebo vydá certifikát, že graf nepatří do třídy UDG.

Maximální klika v UDG bez reprezentace (podle [30]) Algoritmus pracuje zhruba takto: najde eliminační uspořádání hran takové, že okolí hrany indukuje doplněk bipartitního grafu (pozorování 2). Eliminací seznamu hran najde největší kliku na okolí hrany. Podle pozorování 1 existuje dvojice vrcholů (spojená hranou e), které jsou v maximální klice, a hrana e je v eliminačním seznamu. Tolik k myšlence algoritmu. Podívejme se podrobněji, jak algoritmus pracuje.

Pro danou posloupnost hran $L = e_1, \ldots, e_m$ v grafu G označme $G_L[i]$ podgraf G sestávající z hran $e_i, \ldots, e_m$. Pro každou hranu $e_i = \{u, v\}$ definujeme $N_{L,i}$ jako množinu vrcholů, které jsou sousední s u a v v grafu $G_L[i]$. Označme $N_{uv} = G[N(u) \cup N(v)]$, indukovaný podgraf na sousedech vrcholů hrany $\{u, v\}$.
Posloupnost hran $L = e_1, \ldots, e_m$ má vlastnost CNEEO (*cobipartite neighborhood edge elimination ordering*), jestliže pro každou hranu $e_i = \{u, v\}$ indukuje $N_{L,i}$ doplněk bipartitního grafu v G.

Algoritmus najde hladově CNEEO L v G. Začne s prázdným seznamem L. Postupně vyzkouší všechny hrany $e_i = \{u, v\}$ z $G_{L,i}$ a vybere první, pro kterou N_{uv} indukuje doplněk bipartitního grafu v G. Iterací a eliminací hran L najde největší kliku na $N_{L,i}$ výše uvedeným způsobem. Podle pozorování 1, pro nějaké i je $N_{L,i}$ maximální klika.
Pokud v grafu existuje CNEEO L, pak ho algoritmus najde. Certifikát, že graf není UDG, algoritmus vydá, pokud L v grafu neexistuje. Každý UDG graf má CNEEO. O opačné implikaci [30] nemluví, takže lze očekávat, že

existuje graf $G \notin UDG$, který má CNEEO. Připomeňme, že rozpoznatelnost třídy UDG je NP-těžká.

2.3.3 d-quasi jednotkové grafy (d-QUDG)

Zobecněním třídy UDG jsou d-quasi jednotkové grafy, q-QUDG. Poloměr disků je 1 a parametr d říká, v jaké minimální vzdálenosti středů dvou disků nastane průnik.

Přítomnost hrany v grafu nemá striktní definici. Jsou-li u, v vrcholy grafu, D_u, D_v příslušné disky, pak:

- $\{u, v\} \in E : dist(c_{D_u}, c_{D_v}) \leq 2d$
- $\{u, v\} \notin E : dist(c_{D_u}, c_{D_v}) > 2$
- libovolně: $dist(c_{D_u}, c_{D_v}) \in (2d, 2]$

Pro třídu UDG je $d = 1$.

Třída d-QUDG více vyhovuje modelování praktických problémů. Nalezení realizace grafu třídy d-QUDG je NP-těžké pro $d \geq 1/\sqrt{2} \approx 0.707$ [21]. Lze to chápat jako dolní odhad na to, jak dobře se dají vypočítat virtuální souřadnice z grafu zadaného seznamem hran a vrcholů. V článku je uvedena metrika kvality přibližné realizace d-QUDG grafu.

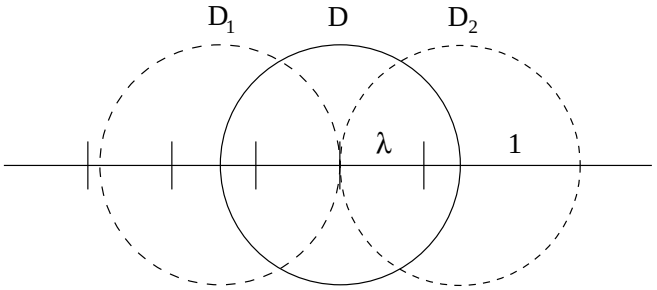
2.3.4 λ-přesné (λ-precision) DG

Předpokládáme, že poloměry disků jsou přenásobeny konstantou tak, aby nejmenší byl 1. Pak λ je minimální vzdálenost dvou středů disků.

Spíše než třída grafů je λ-*precision* omezující podmínka, která může být přidána k diskové třídě, a to spolu s jinými podmínkami (např. omezení na poloměr disků).

Parametr λ může být zadán jako požadavek na rozmístění disků. Naopak v případě, když jsou disky v rovině rozmístěny s nějakou (byť jen přibližnou) pravidelností (např. čtvercová, šesterečná mřížka), pak za λ můžeme považovat minimální vzdálenost mezi všemi dvojicemi středů.

Podmínka minimální vzdálenosti středů disků má omezující vliv i na vlastnosti průnikového grafu. Ukažme si příklad aplikace na třídu UDG. Velikost maximální kliky je $O(\frac{1}{\lambda^2})$. Zafixujeme-li disk D, pak středy disků, se kterými může tvořit kliku K, leží právě uvnitř D. Každé dva disky D_1 a D_2 z K musí splňovat $dist(c_{D_1}, c_{D_2}) \leq 2$. Nejhustší rozmístění středů tvoří trojúhelníkovou mřížku. Počet mřížových bodů, které leží v D můžeme shora odhadnout počtem mřížových bodů ve čtverci se stranou délky $\frac{2}{\lambda}$, což je $O(\frac{1}{\lambda})$.



Obrázek 2.11: λ-precision disky v UDG modelu. Čárkované disky D_1 D_2 jsou nejvzdálenější disky, které mohou náležet s D do jedné kliky.

Na třídu DG_λ lze aplikovat stejný odhad na velikost maximální kliky. Snadno nahlédneme tak, že přenásobíme poloměry disků konstantou, aby největší měl poloměr 1. Hodnota λ je též přenásobena, jelikož jsme definovali třídu DG_λ s jedničkovým poloměrem nejmenšího disku.

Vlastnost minimální vzdálenosti můžeme použít k odhadu velikosti okolí disku v UDG_λ i v DG_λ (předpokládáme poloměry přenásobené na hodnoty nejvýše 1). Mějme disk D a jemu odpovídající vrchol v. Velikost $N(v)$ je $O(\frac{1}{\lambda^2})$. Středy disků, se kterými se D může protnout, leží v kruhu K s poloměrem $\frac{2}{\lambda}$. Stejným způsobem jako v výše odhadneme počet bodů (a tedy i počet disků) v kruhu K čtvercem se stranou $\frac{4}{\lambda}$, což je $O(\frac{1}{\lambda^2})$.

Aplikací podmínky $\lambda = 1$ na UDG se získá třída tzv. *penny grafů*, neboli dotykových grafů jednotkových disků.

2.3.5 σ-omezené (σ-bounded) DG

Parametrem σ označujeme poměr poloměrů největšího a nejmenšího disku. Nejdůležitější vlastností σ-omezenosti je možnost odhadnout počet disjunktních disků, které lze umístit do dané geometrické oblasti. Této vlastnosti

využijeme v technice založené na *lokálních omezeních* (sekce 4.3). Aplikace σ-omezenosti definuje novou zúženou třídu diskových grafů. Lze kombinovat spolu s jinou omezující podmínkou.

Pro velké hodnoty σ má třída DG_σ zhruba stejné vlastnosti jako DG a aproximační algoritmy dosahují stejných aproximačních poměrů.

Pro $\sigma < \dfrac{1}{\sin(\pi/10)} \approx 2.236$ je DG_σ třída se zakázaným podgrafem 10-hvězda (*10-claw free*).

Podívejme se na lokální vlastnost, kterou využijeme v heuristice. Okolí disku D může obsahovat nejvýše $O(\sigma^2)$ disjunktních disků. Všechny sousední disky D musí mít střed nejvýše ve vzdálenosti 3σ od středu D. Plocha kruhu ohraničujícího okolí D je $\pi \cdot 9\sigma^2$. Disky mají poloměr alespoň 1 a plochu alespoň π. Okolí D může obsahovat nejvýše $\dfrac{\pi \cdot 9\sigma^2}{\pi} = 9\sigma^2 = O(\sigma^2)$ disků.

2.3.6 $\lambda\sigma$-omezené DG

Třída grafů $\mathrm{DG}_{\lambda\sigma}$ spojuje dohromady podmínky minimální vzdálenosti a omezeného poloměru disků. Poloměry disků jsou normalizovány do intervalu $[1, \sigma]$ a minimální vzdálenost středů dvou disků je alespoň λ.

Poznámka k omezujícím podmínkám diskových tříd

Výše uvedené omezující podmínky a jejich kombinace slouží k specializaci algoritmů a navržení pokročilých technik, kterými se nebudeme zabývat [1]. Například vlastnost minimální vzdálenosti omezuje velikost maximální kliky a k řešení problémů lze použít techniky analogické u rovinných grafů, jmenujme metodu geometrického separátoru.

2.3.7 Dotykové grafy disků (disk contact graphs)

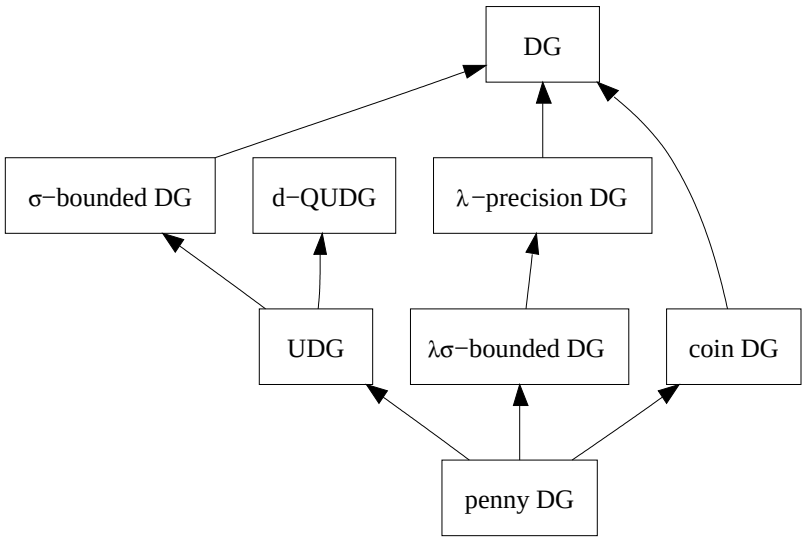
Dotykové diskové grafy jsou speciálním případem třídy DG, kde se dva protínající disky smí pouze dotýkat — odtud název dotykové grafy disků (*disk contact graphs*). Můžeme dále rozlišit, zda se jedná o disky s libovolným poloměrem, tzv. *coin* grafy, nebo s jednotkovým poloměrem, tzv. *penny* grafy.

Zajímavý (a dávný) výsledek říká, že třída rovinných grafů je totožná s dotykovými grafy [20, 33]. Znamená to, že každý rovinný graf lze nakreslit do roviny tak, že vrcholům odpovídají disky (libovolného poloměru) a hrany jsou reprezentovány dotykem dvou disků. Rozpoznatelnost třídy rovinných grafů je lineární [18], a tím pádem rozpoznatelnost dotykových grafů (bez omezení poloměru).

Rozpoznatelnost třídy dotykových grafů s omezeným poloměrem je NP-těžká [5], což platí pro speciální případ — třídy *penny* grafů. Prolbém MIS je na třídě *penny* grafů NP-těžký [7].

2.4 Shrnutí

Nyní máme základní přehled diskových tříd a vlastností, které budeme využívat v heuristikách. Inkluzní diagram diskových tříd:



Obrázek 2.12: Inkluzní diagram diskových tříd. Šipka vede od podtřídy k nadtřídě.

Kapitola 3

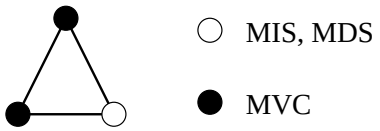
Úlohy

Jak bylo naznačeno v úvodní části, problém hledání maximální nezávislé množiny je jeden ze fundamentálních problémů v teorii grafů a zaslouží si pozornost i díky praktickým potřebám. Na průnikových grafech disků je zkoumán z teoretického hlediska, kdy se hledají polynomiální aproximace a heuristiky, a z praktického hlediska, kde jde především o rychlé řešení konkrétních případů.

3.1 Maximální nezávislá množina (MIS)

Zkratkou MIS (Maximum Independent Set) označíme problém hledání maximální nezávislé množiny. V obecnosti je to NP-úplný problém a je známý jako těžko aproximovatelný. Pro obecné grafy s n vrcholy neexistuje polynomiální aproximační algoritmus pro MWIS (Maximum Weighted Independent Set) s poměrem $n^{1-\epsilon}$ pro libovolné $\epsilon > 0$, za předpokladu $NP \neq \text{co}-RP$ [14].

I na UDG s reprezentací je tento problém NP-těžký [8]. Z toho plyne, že je NP-těžký i pro obecné diskové grafy, σ-omezené diskové grafy a též v případě, že není k dispozici reprezentace. Zajímají nás aproximační algoritmy pro tento problém. Nalezení maximální nezávislé množiny je například jedním z počátečních kroků algoritmu pro nalezení přibližné reprezentace UDG [27].



Obrázek 3.1: Příklad řešení problémů MIS, MVC a MDS

3.1.1 MIS s ohodnocením vrcholů (MWIS)

U problému můžeme rozlišit váženou a neváženou variantu. Přívlastkem „vážená“ myslíme takovou instanci problému, kde každému vrcholu je přiřazena nezáporná váha.

Neváženou variantu lze převést na váženou tak, že všem vrcholům dáme stejnou váhu, například 1.

U varianty MIS s ohodnocenými vrcholy, označované MWIS (Maximum Weighted Independent Set), hledáme nezávislou množinu s největším součtem vah. Taková množina nemusí být největší co do počtu vrcholů. Není asymptotický rozdíl ve složitosti přesného řešení obou variant — vždy musíme enumerovat všechny nezávislé množiny.

3.2 Příbuzné problémy

Vybrali jsme ke studiu jeden konkrétní problém — MIS. Spolu s ním se v praxi řeší následující problémy, o kterých bychom se měli zmínit: Minimum Vertex Cover, Minimum Dominating Set, Minimum Coloring. Na obecných grafech jsou to NP-úplné problémy a těžko aproximovatelné. Na diskových grafech jsou NP-těžké [8], nicméně mohou být snáze aproximovatelné.

Některé z problémů jsou na sebe převoditelné s malou režií (jinak jistě polynomiální) a nebudeme se jim věnovat hlouběji.

3.2.1 Minimální vrcholové pokrytí (MVC)

Úkolem je najít v grafu $G = (V, E)$ takovou množinu $C \subseteq V$, že každá hrana má alespoň jeden konec v této množině.

Můžeme si to představit tak, že vybíráme ty vrcholy, ze kterých je „vidět“ na všechny ostatní. Je-li $C \subseteq V$ řešením problému MVC, pak $C' = V \setminus C$ je řešením MIS (a naopak). Tento vztah platí i v případě ohodnocených vrcholů. Aproximační faktor získaný řešením jednoho problému nemusí být nutně zachován při převedení na druhý problém. Viz například PTAS algoritmus pro nalezení MWVC (Minimum Weighted Vertex Cover) na DG v [34].

3.2.2 Minimální dominující množina (MDS)

Cílem je najít množinu $C \subseteq V$ takovou, kde každý vrchol z $V \subseteq C$ má souseda v C. Řešení problému MIS je zároveň řešením MDS.

Další varianty tohoto problému kladou dodatečné požadavky na výslednou množinu:

- souvislá (Minimum Connected Dominating Set) - pro příklad uveďme, že MCDS modeluje problém nalezení páteře pro směrování mezi vysílači
- nezávislá (Minimum Independent Dominating Set)

Pro ilustraci uvedeme, že např. problém nalezení vysílačů pro tísňová volání v ad-hoc síti lze modelovat jako MDS [35].

3.2.3 Minimální dobré obarvení

Problém nalezení dobrého obarvení znamená rozdělit V na k množin $V_1, V_2, \ldots, V_k$ tak, že V_i jsou nezávislé a k je nejmenší možné. Například opakovaným řešením MIS takové rozdělení najdeme. Jak již bylo zmíněno, barvení diskových grafů se v telekomunikacích využívá při řešení problému přidělování frekvencí. Problém je rozsáhle studován v různých verzích, viz [1, 12, 24, 10].

Kapitola 4

Techniky řešení

Jak již bylo zmíněno, neexistuje efektivní algoritmus pro řešení MIS, a proto nás zajímají aproximační algoritmy tohoto problému. Ukážeme si techniky, jejichž myšlenka je jednoduchá, a náročnější algoritmy zmíníme s odkazem na literaturu.

Pro návrh algoritmů má zásadní význam, zda je vstup algoritmu zadán jako disková reprezentace (seznam disků), nebo jako graf (seznam vrcholů a hran). Některé problémy lze řešit efektivně v obou případech (např. maximální klika v UDG).

Hlavním požadavkem na algoritmy je polynomiální (efektivní) výpočetní složitost a relativně dobrý aproximační poměr. Kritéria rychlosti a přesnosti jsou protichůdná a záleží, které preferujeme:

- *rychlost* za cenu aproximační nepřesnosti
- *aproximační přesnost* za cenu rychlosti

Rychlé nepřesné algoritmy mají konstantní aproximační faktor a jsou snadno implementovatelné. Jedná se většinou o hladové algoritmy.

Aproximační algoritmy typu PTAS (Polynomial-Time Approximation Scheme), se kterými se setkáme, řeší MIS s libovolnou přesností výměnou za rychlost.

Algoritmy budeme schematicky zapisovat jako posloupnost kroků popsaných slovně.

O implementaci vybraných algoritmů pojednává příloha A.

Dále v kapitole budeme hovořit o grafu $G = (V, E)$. Výstupem algoritmů je množina $I \subseteq V$, která je na počátku algoritmu prázdná. Optimální řešení problému MIS (MWIS) na grafu G budeme značit I^*.

4.1 Posouvání mřížky (grid shifting)

Technika posouvání mřížky byla poprvé použita k aproximaci NP-úplných problémů na rovinných grafech [2, 16]. Geometrická separace způsobí i grafovou separaci. Úloha se vyřeší na podproblémech a jejich sjednocení je výsledné řešení.

Jednodušší varianta (pro UDG) pracuje zhruba takto: rovina je rozdělena mřížkou na buňky, v jednotlivých buňkách se spočítají dílčí řešení MIS, která se zpět složí do celku [19]. Mřížka se postupně posouvá vlevo a nahoru, dokud nesplyne s výchozí. Disky protínající mřížku se před výpočtem odstraní. Za řešení se prohlásí maximální mezi všemi posuny mřížky.

Složitější varianta (pro DG) pracuje s více vnořenými mřížkami. Řešení na jedné úrovni používá výsledky z nižších úrovní a pomocí dynamického programování hledá nejlepší řešení [34].

Přesnost a časová složitost algoritmu závisí na rozteči mřížky. Větší rozteč znamená přesnější aproximaci a pomalejší běh algoritmu.

Algoritmus řeší problém MIS nebo MWIS. Rozdíl je v řešení dílčího problému uvnitř buňky mřížky. Buď se vybere množina s největším počtem disků, nebo množina s největším součtem vah. Uvažujeme váhovou funkci $w : V \to \mathbb{R}^+$.

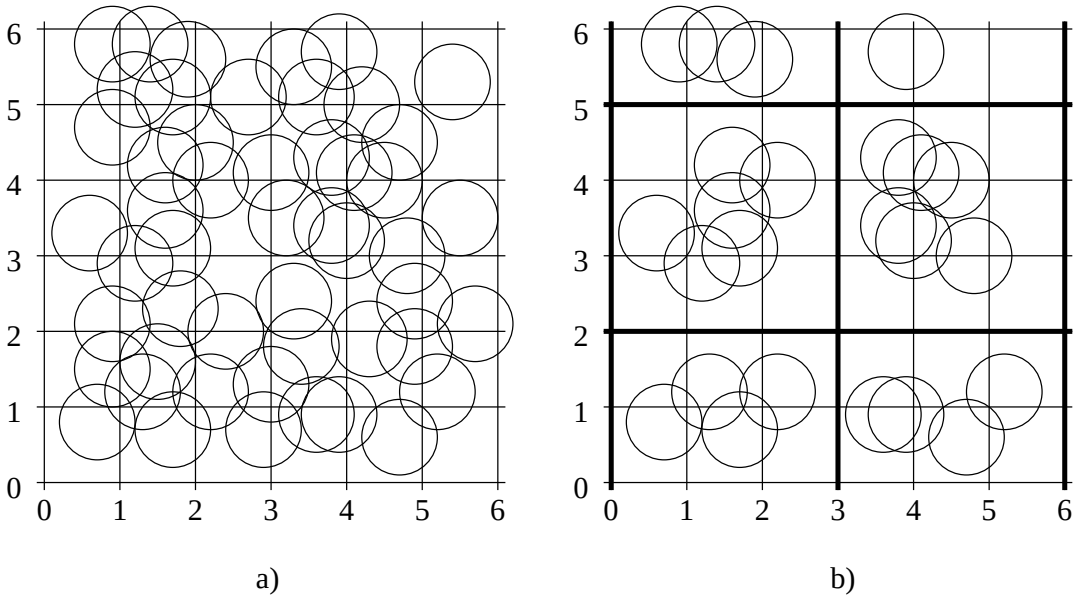
4.1.1 Použití na UDG

Pro třídu UDG je technika snadno použitelná vzhledem k uniformnosti disků: rovina je rozdělena jednou mřížkou.

Vstupem algoritmu je číslo $k \geq 3$ (určující přesnost) a soubor jednotkových disků $\mathcal{D}$ s poloměrem $\frac{1}{2}$. Takto zvolený poloměr zajistí, že jeden disk zaplní celou buňku mřížky s roztečí 1. Mřížka se skládá ze všech horizontálních a vertikálních přímek protínajících osy v celočíselných souřadnicích. Můžeme

předpokládat, že žádný disk nemá střed na souřadnicích uprostřed dvou přímek mřížky. Pak se žádný disk mřížky nedotýká, ale vždy protíná právě jednu přímku v horizontálním a právě jednu přímku ve vertikálním směru.

Uvažujme množinu $\mathcal{D}_{i,j} \subseteq \mathcal{D}$ pro všechny dvojice $(i,j), 0 \leq i,j \leq k-1$, která obsahuje všechny disky neprotínající vertikální přímky $x = i+kp$ (pro nějaké $p \in \mathbb{Z}$) a neprotínající horizontální přímky $y = j+kp$ (pro nějaké $p \in \mathbb{Z}$). Množina $\mathcal{D}_{i,j}$ vypadá jako shluky disků ve čtvercové síti (obr. 4.1b).



Obrázek 4.1: Ukázka techniky *posouvání mřížky* pro $k = 3, i = 0, j = 2$. a) Soubor disků a výřez mřížky, která ho týká. b) Množina $\mathcal{D}_{0,2}$. Silně jsou vyznačeny vertikální přímky $x = 3p$ ($p \in \mathbb{Z}$) a horizontální přímky $y = 2+3p$ ($p \in \mathbb{Z}$). Jsou smazány disky, které tyto přímky protínaly.

Každý z disků z $\mathcal{D}_{i,j}$ je celý obsažen v nějakém čtverci s délkou strany k. Řešením MIS na $\mathcal{D}_{i,j}$ je sjednocení řešení MIS ze všech čtverců.

Maximální nezávislá množina v jednom čtverci může mít nejvýše $C = O(k^2)$ prvků, neboť do plochy čtverce se vejde nejvýše k^2/π disků, aniž by se protínaly. Maximální nezávislou množinu lze nalézt enumerací všech množin velikosti nejvýše C v čase $O(|S|^C)$, kde S je množina disků jednoho čtverce.

Pro pevné k lze spočítat maximální nezávislou množinu $I_{i,j}$ na $\mathcal{D}_{i,j}$ v polynomiálním čase. Výstupem algoritmu je největší $I_{i,j}$ přes všechny dvojice (i,j).

Pro názornost si ukažme schéma algoritmu ($\mathcal{A}$):

1. najdi množinu $D_{i,j}$, tak že vymažeš z množiny $\mathcal{D}$ disky protínající mřížku
2. spočti dílčí řešení $I_{\text{dílčí}}$ uvnitř všech buněk mřížky
   (a) enumeruj všechny množiny M velikosti $\leq C$
   (b) testuj nezávislost M
   (c) aktualizuj lokální maximum I_S: vyber nejpočetnější (MIS) nebo s největším součtem vah (MWIS)
3. aktualizuj globální maximum I podle současného řešení $I_{\text{dílčí}}$
4. posuň mřížku a pokračuj krokem 1

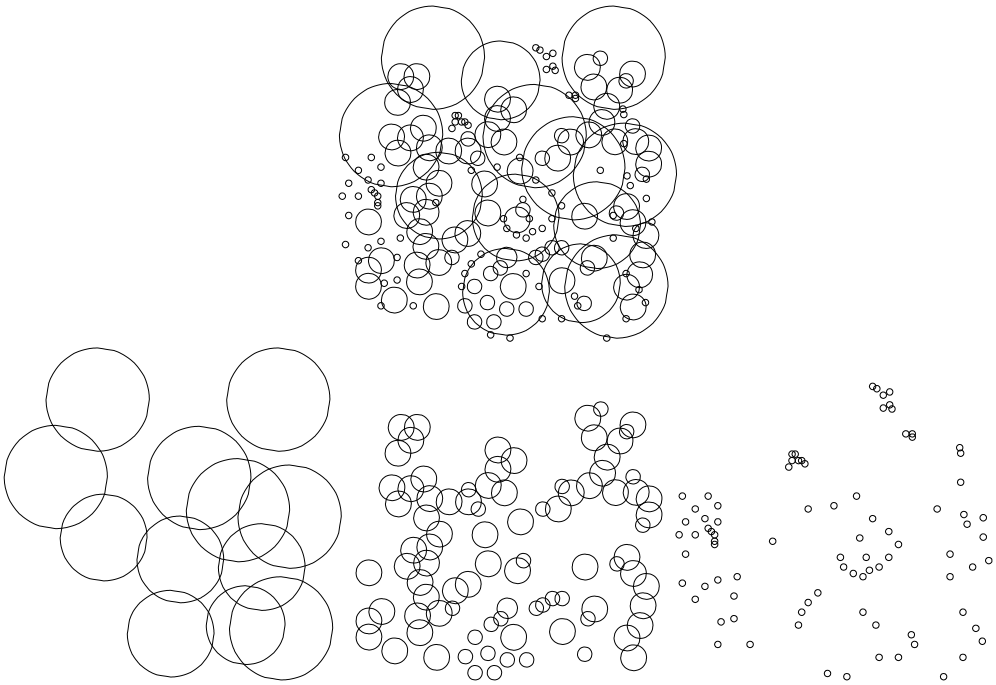
Algoritmus využívající tuto techniku pro nalezení MIS na UDG [19] je typu PTAS s aproximačním faktorem $1 - \frac{2}{k}$. Vzhledem k optimu I^* je velikost řešení alespoň $(1 - \frac{2}{k})|I^*|$. Každý disk protíná právě jednu vertikální a horizontální přímku (mřížky s roztečí 1). Existuje takové i, že nejvýše $|I^*|/k$ disků protíná vertikální přímky $x = i + kp$ ($p \in \mathbb{Z}$). Podobně pro horizontální přímky existuje j, že nejvýše $|I^*|/k$ disků je protnuto. Množina $\mathcal{D}_{i,j}$ stále obsahuje maximální nezávislou množinu velikosti alespoň $(1 - \frac{2}{k})|I^*|$. Algoritmus spočítá MIS na všech $\mathcal{D}_{i,j}$, a tedy výsledná množina musí mít velikost alespoň $(1 - \frac{2}{k})|I^*|$.

Pro požadovaný aproximační faktor $\epsilon > 0$ zvolíme $k = \lceil \frac{2}{\epsilon} \rceil$. Složitost algoritmu $\mathcal{A}$ je $O(k^2 n^{O(k^2)})$.

Složitost lze snížit na $O(kn^{O(k)})$. Z množiny $\mathcal{D}$ se vynechají pouze disky protínající horizontální přímky $y = j + kp (p \in \mathbb{Z})$ a použitím dynamického programování se najde optimum v každém pruhu šířky k mezi dvěma horizontálními přímkami [19, 24].

4.1.2 Použití na DG

U třídy DG je situace složitější. Disky je potřeba rozdělit do vrstev podle velikosti. Každá vrstva má mřížku odpovídající velikosti disků na své úrovni. Posouvání probíhá na všech vrstvách současně užitím mřížek různé zrnitosti.



Obrázek 4.2: Příklad DG bez rozdělení na vrstvy a rozdělení na vrstvy podle poloměru disků.

Pomocí dynamického programování najde nejlepší řešení na dané vrstvě a všech vrstvách již zpracovaných. Postupuje se od nejmenších disků k větším.

Vstupem algoritmu je číslo $k \geq 2$, které udává požadovanou přesnost aproximace. Předpokládejme, že poloměr největšího disku je 1 a nechť $d_{\min}$ je poloměr nejmenšího disku. Množinu $\mathcal{D}$ rozdělíme do $L+1$ vrstev, kde $L = \lfloor \log_{k+1}(1/d_{\min}) \rfloor$. Vrstvu l tvoří množina disků D_l, jejichž průměr d splňuje $\frac{1}{(k+1)^{l+1}} < d \leq \frac{1}{(k+1)^l}$. Velikost poloměrů ve vrstvách exponenciálně klesá.

Ke každé vrstvě l přísluší mřížka, jejíž rozteč je $\frac{1}{(k+1)^l}$. Je tvořena horizontálními přímkami $y = h(k+1)^l,\ (h \in \mathbb{Z})$ a vertikálními přímkami $x = v(k+1)^l,\ (v \in \mathbb{Z})$. Čísla h a v nazveme *index* přímky v mřížce. Pro zjednodušení požadujeme, aby každý disk protínal právě jednu vertikální a horizontální přímku ve své vrstvě. (Dotýká-li se disk horizontální přímky, pak to za průnik považujeme, pokud se disk přímky dotýká shora, analogicky u vertikální přímky dotyk zleva.)

Dvojicí (r, s) označíme relativní posun mřížky vůči počátku, $0 \leq r, s \leq k$. Horizontální (resp. vertikální) přímku v mřížce nazveme *aktivní* pro (r, s), pokud je její index roven $k \mod r$ (resp. $s \mod s$). Aktivní přímky v mřížce na úrovni l jsou aktivní i v (řidší) mřížce na úrovni $l' < l$. Z toho též plyne, že řidší mřížka nepůlí buňky hustší mřížky.

Definujme množinu $\mathcal{D}^l_{r,s}$ jako množinu všech disků na úrovni l, které neprotínají aktivní přímky. Definujme $\mathcal{D}_{r,s}$ jako sjednocení $\mathcal{D}^l_{r,s}$ pro $0 \leq l \leq L$.

Uvažujme úroveň $0 \leq j \leq L$. Aktivní přímky rozdělí rovinu na čtverce, které nazveme *j-čtverce*. Pokud j-čtverec S obsahuje alespoň jeden disk $D \in \mathcal{D}^j_{r,s}$, nazveme D *relevantní pro* S. Každý j-čtverec obsahuje $(k+1)^2$ $(j+1)$-čtverců. Vnoření čtverců tvoří stromovou strukturu (les $\mathcal{L}$). Pokud hladina h ve stromě neobsahuje žádný relevantní j-čtverec, pak je vynechána. Přesněji to znamená, že mezi j_1-čtvercem S_1 a relevantním j_2-čtvercem S_2 ($j_1 < j_2, S_1 \subset S_2$) je stromová hrana, pokud pro všechna j ($j_1 < j < j_2$) neexistuje relevantní j-čtverec S. Sjednocení všech stromů s kořeny v relevantních j-čtvercích tvoří les.

Algoritmus zpracovává všechny relevantní čtverce počínaje úrovní L. Pro každý j-čtverec S je metodou dynamického programování vyplněna tabulka T_S.

Pro každou nezávislou množinu I disků protínajících čtverec S na úrovních $\leq j$ obsahuje záznam tabulky $T_S(I)$ disjunktní disky úrovně $> j$ obsažené v S a disjunktní s I. Navíc do $T_S(I)$ uložíme množinu s maximální váhou.

Sestavení tabulky je klíčové pro pochopení algoritmu, proto se podívejme, co předchozí definice vlastně říká. Na tabulku se můžeme dívat takto: indexujeme *čtvercem* S a *množinou „velkých" disků* I. Hodnotami tabulky jsou pak „malé" disky z S disjunktní s I.

Algoritmus vyplňuje tabulku počínaje úrovní L, tj. nejmenšími disky. Na této úrovni nelze využít řešení z menších podúloh. Hodnoty $T_S(I)$ jsou nalezeny hrubou silou tak, že se enumerují všechny nezávislé množiny I velikosti nejvýše C L-disků pro daný čtverec S.

Konstanta C je závislá na k a udává maximální počet disjunktních disků úrovně nejvýše j protínajících čtverec S. Disky na úrovni j mají průměr alespoň $d = \frac{1}{(k+1)^{j+1}}$. Strana čtverce S má $(k+1)^2$ jednotek d. Do S se vejde alespoň $((k+1)^2)^2 = O(k^4)$ j-disků. Z vnějšku protíná S nejvýše $4 \cdot ((k+1)^2 + 1) = O(k^2)$ j_1-disků ($j_1 \leq j$), neboť všechny j_1-disky jsou větší

nebo rovné j-diskům. Dohromady vyjde $C = O(k^4)$.

Algoritmus $\mathcal{S}$ pro vyplnění daného čtverce S na úrovni j postupuje dle následujícího schématu:

1. vyber z $\mathcal{D}_{r,s}$ disky úrovně $\leq j$ protínající S do množiny R
2. enumeruj všechny disjunktní podmnožiny $M \subseteq R$ a $|M| \leq C$
3. vyber do množiny J všechny j-disky z M
4. pro každý relevantní podčtverec $S' \subset S$:
   (a) vyber do $I' \subseteq M$ disky, které protínají S'
   (b) přidej do J množinu $T_{S'}(I')$
5. vyber do množiny $I \subseteq M$ disky z úrovní $< j$ (velké disky)
6. nastav hodnotu $T_S(I)$ na M, pokud $w(M) > w(T_S(I))$ (nebo není-li ještě definována)

Podívejme se, jak probíhá vyplňování tabulky a slévání řešení z menších podúloh. V kroku 3 se postupně vyzkouší všechny podmnožiny j-disků a vždy se k nim připojí optimální řešení z menších podúloh (krok 4). Takto vybraná množina je kandidátem na optimum pro čtverec S. V kroku 5 se vyberou velké disky, kterými se indexují hodnoty v tabulce T.

Výstupem algoritmu je $T_S(\emptyset)$ pro všechny čtverce S, které jsou kořeny v lese čtverců $\mathcal{L}$.

Shrňme zde celý algoritmus $\mathcal{B}$:

1. rozděl množinu $\mathcal{D}$ do úrovní
2. pro dané aktivní přímky (r, s) najdi les čtverců $\mathcal{L}$
3. počínaje úrovní L aplikuj algoritmus $\mathcal{S}$ na všechny relevantní čtverce
4. ulož dílčí řešení pokud je maximem

Algoritmus $\mathcal{B}$ je typu PTAS. Alespoň pro jednu dvojici (r, s), $0 \leq r, s < k$, platí, že $\mathcal{D}_{r,s}$ je nejvýše $(1 - \frac{1}{k})$-krát horší než optimum. Argument je podobný jako v případě jednoduché mřížky. Nechť S_r je množina disků, které na svojí úrovni protínají vertikální přímku s indexem r modulo k. Pro různá r jsou množiny S_r disjunktní, a tedy alespoň jedna z nich tvoří nejvýše $\frac{1}{k}$-část optima. Pro tuto množinu nechť $T = \mathit{optimum} \setminus S_r$, pak T má velikost

alespoň $(1-\frac{1}{k})$-násobku optima. Dále, nechť T_s jsou disky z T, které protínají horizontální přímky s indexem s. Opět, pro alespoň jedno s je velikost T_s nejvýše $\frac{1}{k}$-část T. Zbývající množina $U = T \setminus T_s$ má velikost alespoň $(1-\frac{1}{k})^2$-násobku optima.

Pro pevné k běží v polynomiálním čase. Relevantních čtverců může být nejvýše n (tolik co disků). Les $\mathcal{L}$ lze vytvořit v polynomiálním čase. Algoritmus $\mathcal{S}$ enumeruje pro každý relevantní čtverec S $O(n^C)$ množin. V každé z nich provede $O(n)$ vyhledání v podčtvercích (vyhledání je polynomiální). Algoritmus je proveden (k^2)-krát pro všechny dvojice (r, s). Celkový čas běhu je $O(n^{O(k^2)})$.

Oba algoritmy (pro UDG a DG) je možné přizpůsobit i dalším průnikovým modelům [22].

4.2 Zakázaný podgraf k-hvězda (k-claw free graph)

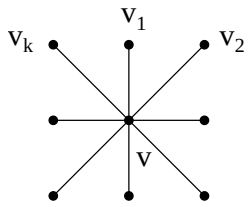
Uvedeme dva algoritmy, založené na absenci 6-hvězdy (*k-claw free*) jako indukovaného podgrafu, k zaručení aproximačního poměru.

Hladový algoritmus $\mathcal{B}$ využívající vlastnosti postupuje podle následujícího schématu:

0. *nepovinný krok: předpočítání dat*
1. vyber libovolný vrchol $v \in V$
2. přidej vrchol v do I a smaž z grafu $N(v)$
3. opakuj 1–3, dokud není G prázdný

Množina I je nezávislá. Její velikost odvodíme podle následujícího tvrzení:

Tvrzení 1: Pokud je graf G bez $(k+1)$-hvězdy, pak algoritmus $\mathcal{B}$ najde MIS s aproximačním faktorem k.
Idea důkazu je jednoduchá. Graf G neobsahuje $K_{1,k+1}$. Mějme graf $K_{1,k}$ (obr. 4.3). Optimum je množina vrcholů $I^* = \{v_1, \ldots, v_k\}$. V nejhorším případě algoritmus přidá do I centrální vrchol v. Podíl $|I^*|/|I| = k$ udává aproximační poměr.



Obrázek 4.3: Graf $K_{1,k}$

Algoritmus $\mathcal{B}$ bez 0-tého kroku běží v čase $O(n+m)$. Je rychlý, ale ne příliš přesný. Jeho vhodné využití je pro první přiblížení řešení MIS.

Existuje algoritmus, který pro $k \geq 4$ a libovolné $\epsilon > 0$ najde $(\frac{k}{2} + \epsilon)$-aproximaci v grafu bez k-hvězdy [13]. Je založen na technice lokálních zlepšení na způsob prodlužování cest omezené délky. Délka cest závisí na parametru ϵ a složitost algoritmu narůstá se snižujícím se ϵ.
Pro třídu UDG získáme aproximační poměr $(2.5+\epsilon)$. V následující sekci uvidíme algoritmus založený na jednodušší myšlence se srovnatelnou časovou složitostí a aproximačním poměrem 3.

5-aproximace na UDG bez reprezentace

Třída UDG neobsahuje 6-hvězdu jako indukovaný podgraf [31]. Zakázaný podgraf se odvodí z geometrické reprezentace (obr. 2.9b).

Podle výše uvedeného tvrzení 1 lze hladově najít MIS v čase $O(n+m)$ s aproximačním faktorem 5. Algoritmus nepotřebuje ke svému běhu reprezentaci a je to on-line algoritmus.

Problém MWIS s váhovou funkcí w lze vyřešit tak, že v 0-tém kroku setřídíme vrcholy sestupně podle vah. V nejhorším případě přidáme do I vrchol v. Váha řešení bude $w(v)$. Optimální řešení může mít váhu až $5w(v)$. Složitost algoritmu je $O(n \log n + m)$.

5-aproximace na DG s reprezentací

Grafy třídy DG nemají tu příznivou vlastnost, že by neobsahovaly k-hvězdu pro nějaké k. Příkladem budiž velký disk, který protíná libovolně mnoho malých disjunktních disků. Přesto lze dosáhnout aproximačního poměru 5

s využitím reprezentace.

V 0-tém kroku setřídíme disky vzestupně podle poloměru. Hladově budeme vybírat vrchol podle následujícího pozorování, které zaručí aproximační faktor 5:

Pozorování: Nechť vrcholu v odpovídá disk D_v s poloměrem r_v. Pak graf $G[N(v)]$ obsahuje nezávislou množinu velikosti nejvýše 5, kterou tvoří disky s poloměrem alespoň r_v.

V okamžiku, kdy algoritmus vybral hladově vrchol v, již nejsou v grafu disky s poloměrem menším než r_v. Buď je v první vybraný vrchol, nebo byly všechny menší disky smazány. Množina $N(v)$ nemůže obsahovat 6-hvězdu: v nejhorším případě jsou všechny zbývající disky stejně velké (analogie s UDG).

4.3 Lokální vlastnosti

V této sekci si ukážeme algoritmy, které nepotřebují ke svému běhu reprezentaci. Prohledáním bezprostředního okolí vrcholu je nalezen vhodný kandidát, jehož výběrem do výsledné množiny se řešení zhorší nejvýše c-krát. Pro UDG je $c = 3$ a pro DG je $c = 5$.

Algoritmus $\mathcal{C}$ využívající vlastnosti postupuje podle následujícího schématu:

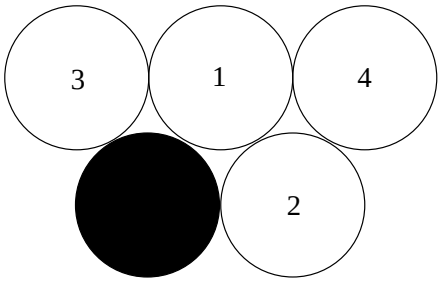
1. najdi vrchol $v \in V$ takový, že $N(v)$ neobsahuje nezávislou množinu velikosti alespoň $(c + 1)$
2. přidej vrchol v do I a smaž z grafu $N(v)$
3. opakuj 1–3, dokud není G prázdný

Časová náročnost kroku 1 je $O(n^{c+2})$: zda $N(v)$ obsahuje nezávislou množinu velikosti $(c+1)$ zjistíme enumerací všech $(c+1)$ prvkových podmnožin $N(v)$ a otestujeme nezávislost. Provedeme pro každý vrchol, $O(n)$. Složitost celého algoritmu je $O(n^{c+3})$.

3-aproximace na UDG

Z geometrické reprezentace UDG můžeme snadno vypozorovat následující [31]: disk s nejmenší x-ovou souřadnicí má ve svém sousedství nezávislou množinu velikost nejvýše 3. Ze symetrie je zřejmé, že tuto podmínku splňují i disky s maximální x-ovou souřadnicí a minimální/maximální y-ovou souřadnicí.

Tvrzení 1: $G \in UDG$, pak existuje $v \in V$ takové, že $N(v)$ neobsahuje nezávislou množinu velikosti 4.



Obrázek 4.4: Okolí disku s největší y-ovou souřadnicí

Viz obr. 4.4. Zkoumáme okolí disku 1. Disky 3 a 4 nelze posunovat směrem nahoru, disk 2 se nesmí dotýkat 4, ale pak už není žádná možnost umístit ●, aby indukovaný podgraf na vrcholech $\{2, 3, 4, ●\}$ byla nezávislá množina.

Je-li k dispozici reprezentace, časová složitost algoritmu se významně sníží. Vhodný kandidát se sám nabídne jako vrchol s nejmenší y-ovou souřadnicí. Čas běhu je tedy $O(n \log n + m)$.

5-aproximace na DG bez reprezentace

Pro zaručení aproximačního poměru 5 použijeme následující tvrzení:

Tvrzení 2: $G \in DG$, pak existuje $v \in V$ takový, že $N(v)$ neobsahuje nezávislou množinu velikosti 6.

Disk s nejmenším poloměrem D_{min}, příslušný vrcholu v_{min}, splňuje tvrzení. V indukovaném podgrafu $G[N(v_{min})]$ najdeme nezávislou množinu velikosti nejvýše 5 (analogie s UDG).

Toto pozorování je použito také v technice 4.2 (5-aproximace na DG s repre-

zentací) s tím rozdílem, že nejmenší disk se najde přímo v reprezentaci.

4.4 Omezené okolí vrcholu

Reprezentaci můžeme využít k rozdělení úlohy na menší části, a to na základě geometrických vlastností (umístění, velikost).

Bez reprezentace je možné (na základě znalostí vlastností třídy grafů) rozsekat graf na bloky, na nich vyřešit problém hrubou silou a sjednocení dílčích řešení vydat jako výsledek. Charakter problému MIS dovoluje dílčí řešení jednoduše sjednotit, jinak by bylo nutné řešení dávat dohromady způsobem, který zaručí, že i sjednocení bude řešením.

Tato technika je použita v PTAS algoritmu pro nalezení MIS v UDG bez reprezentace [29]. Parametr $\rho = 1+\epsilon$ označuje požadovanou přesnost. Ukážeme si jeho myšlenku:

1. vyber libovolný vrchol v
2. zvětšuj okolí N vrcholu v, dokud platí omezující podmínka
3. vyřeš hrubou silou problém na N
4. odstraň z grafu N, ulož si nalezené dílčí řešení $I_{\text{dílčí}}$
5. dokud není graf prázdný, pokračuj dalším vrcholem

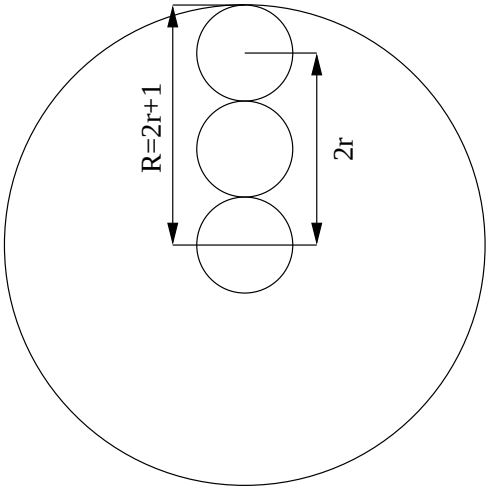
Podmínkou pro aplikovatelnost této techniky je požadavek na omezení velikosti bloků, a to v závislosti na aproximačním faktoru.

Ukažme si na příkladu UDG, jak může vypadat. Definujeme si $N^r(v) = \{u \in V : u$ je vzdáleno od v nejvýše $r\}$, r-té okolí, pro $r = 0, 1, 2, \ldots$. Počínaje N^0 počítáme maximální nezávislou množinu $I_r \subset N^r, r = 0, 1, \ldots$ dokud platí podmínka $|I_{r+1}| > \rho|I_r|$ (zvětšování okolí má přínos).

První r, při kterém je nerovnost porušena, existuje a je omezeno konstantou závislou na ρ.

Nechť funkce $f : V \to \mathbb{R}^2$ určuje střed disku odpovídajícího vrcholu. V kruhu s poloměrem $R = 2r + 1$ a středem v $f(v)$ jsou obsaženy všechny disky z $N^r(v)$ (obr. 4.5). Počet těch, co se neprotínají (a tvoří nezávislou množinu) je shora omezen, $|I_r| < \pi R^2/\pi = O(r^2)$. Z definice r_1 plyne, že pro $r < r_1$

je $|I_r| > \rho|I_{r-1}| > \cdots > \rho^r|I_0| = \rho^r$. Množinu I_r najdeme úplnou enumerací všech podmnožin $N_r(v)$ velikosti maximálně $C = O(r^2)$ v čase $O(n^C)$. $C = O(c^2)$ kde $r \leq r_1 \leq c(\rho) = c$.



Obrázek 4.5: Okolí disku do vzdálenosti r

Takto proběhne krok 2 a 3. Dále z grafu vymažeme N^{r_1+1}, tedy okolí o jednu vrstvu větší. Tím je zaručeno, že řešení nalezené ve zbylém grafu nemá žádnou hranu do I_r.

Stejně lze argumentovat pro objekty podobné jednotkovým diskům, jako například σ-omezené disky, a to i ve vyšších dimenzích (≥ 2). Stačí polynomiální odhad na maximální geometrické okolí vydělené objemem nejmenšího objektu, který přichází v úvahu.

Předchozí úvaha se týkala objektů geometricky podobných jednotkovým diskům. Jiným typem omezující podmínky může být minimální vzdálenost středů objektů. Ve třídě DG_λ je toto splněno a disky mohou mít libovolný poloměr. Velikost r-tého okolí ($|N^r|$) je již polynomiálně omezena v r.

Algoritmus lze také aplikovat na grafy s omezeným stupněm. V [3] je uveden algoritmus na nalezení MIS v grafech s omezeným maximálním stupněm Δ. Pro $\Delta > 2$ a $\epsilon > 0$ je vyřešen problém MIS s aproximačním faktorem a) pro sudé Δ $\dfrac{5}{\Delta + 3} - \epsilon$ a b) pro liché $\dfrac{5}{\Delta + 3.25} - \epsilon$.

4.5 Trhání klik

Heuristika založená na trhání klik vychází z faktu, že z každé maximální kliky (co do inkluze) můžeme vzít nejvýše 1 vrchol. Víme, že maximální kliku lze nalézt v čase $O(n^{4.5})$ na UDG s reprezentací [8] a polynomiálně na UDG bez reprezentace [30].

Schéma algoritmu $\mathcal{D}$:

1. najdi všechny maximální kliky v G $(K^1, \ldots, K^m)$
2. uspořádej kliky sestupně podle velikosti ($|K^i| \geq |K^{i+1}|$), dále pracuj s největší klikou ze seznamu (K)
3. najdi vrchol $v \in K$, jehož všechny hrany vedou do K, jinak vezmi libovolný vrchol z K
4. smaž z grafu $N(v)$ (tedy i K) a přečísluj seznam klik
5. opakuj 2–5, dokud není G prázdný

Složitost algoritmu je $O(n \times n^{4.5} + n \log n + m)$, což je $O(n^{5.5})$, a nepotřebuje k výpočtu reprezentaci.

Aproximační poměr k se nedá dobře odhadnout, protože je silně závislý na vnitřní struktuře grafu. Odhad podle $\omega(G)$ (klikovosti) je hrubý a nevypovídající. Hodnota k může ležet v intervalu $\left[\frac{n}{2}, \frac{n}{\omega(G)}\right]$.

Algoritmus uvádíme jako možný přístup k řešení MIS na UDG. Na jiných třídách nemáme to štěstí, že by bylo možné najít maximální kliku polynomiálně.

Kapitola 5

Závěr

Téma průnikových grafů disků jsme zpracovali přehledově. Rozvedli jsme podrobněji techniky založené na jednoduchých myšlenkách. Přiložený program demonstruje vybrané techniky pro názornost a hlubší porozumění.

Nad úrovní jednoduchých technik je prostor pokročilejších technik, které se opírají o hlubší výsledky z teorie výpočetní složitosti (parametrizovaná složitost) a využívají další geometrické vlastnosti (*problem kernel reduction*, strategie *divide & impera* založená na *geometrické separaci grafu*).

Seznámili jsme se s třídami diskových grafů a s problémy, které se na nich modelují. Přínosem této práce je shrnutí technik aproximačních algoritmů, které řeší obtížné problémy. Věnovali jsme pozornost problému hledání maximální nezávislé množiny, na kterém jsme techniky demonstrovali. V literatuře odkud, jsme čerpali, je technika většinou upravena i pro další problémy (Minimum Vertex Cover, Minimum Dominating Set, ...).

Další práce se může týkat vlastností geometrických objektů, které zkoumáme: rozšíření geometrie na více dimenzí (*d-unit balls*), zobecnění disků na elipsy a dále na tzv. „tlusté“ objekty.

Literatura

[1] J. Alber and J. Fiala. Geometric separation and exact solutions for the parameterized independent set problem on disk graphs. 2002.

[2] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *J. Assoc. Comput. Mach.*, 41:153–180, 1994.

[3] P. Berman and M. Fürer. Approximating maximum independent set in bounded degree graphs. In *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'94 (Arlington, Virginia, January 23-25, 1994)*, pages 365–371, Philadelphia, PA, 1994. ACM SIGACT, SIAM, Society for Industrial and Applied Mathematics.

[4] Breu and Kirkpatrick. On the complexity of recognizing intersection and touching graphs of disks. In *GDRAWING: Conference on Graph Drawing (GD)*, 1995.

[5] H. Breu. Algorithmic aspects of constrained unit disk graphs. Technical Report TR-96-15, Department of Computer Science, University of British Columbia, Sept. 1996. Tue, 22 Jul 1997 22:20:10 GMT.

[6] H. Breu and D. G. Kirkpatrick. Unit disk graph recognition is NP-hard. *Comput. Geom. Theory Appl.*, 9(1-2):3–24, 1998.

[7] M. R. Cerioli, L. Faria, T. O. Ferreira, and F. Protti. On minimum clique partition nad maximum independent set on unit disk graphs and penny graphs: complexity and approximation. *Electronic Notes in Discrete Mathematics*, 18:73–79, 2004.

[8] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Mathematics*, 86:165–177, 1990.

[9] S. Doddi, M. V. Marathe, A. Mirzaian, B. M. Moret, and B. Zhu. Map labeling and its generalizations. In *Proceedings of the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 148–157, 1997.

[10] T. Erlebach and J. Fiala. Independence and coloring problems on intersection graphs of disks, 2001.

[11] M. R. Garey and D. S. Johnson. *Computer and Intractability.* W. H. Freeman and Company, New York, 1979.

[12] A. Graf, M. Stumpf, and G. Weisenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998.

[13] M. M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'95 (San Francisco, California, January 22-24, 1995)*, pages 160–169, Philadelphia, PA, 1995. ACM SIGACT, SIAM, Society for Industrial and Applied Mathematics.

[14] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *FOCS '96; 37th Annual Symposium on Foundations of Computer Science (FOCS '96)*, pages 627–636, Washington - Brussels - Tokyo, Oct. 1996. IEEE.

[15] Hliněný and Kratochvíl. Representing graphs by disks and balls (a survey of recognition-complexity results). *DMATH: Discrete Mathematics*, 229, 2001.

[16] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in robotics and VLSI. In *Symposium of Theoretical Aspects of Computer Science*, volume 166 of *lncs*, pages 55–62, Paris, France, 11–13 Apr. 1984. Springer.

[17] J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973, December.

[18] J. E. Hopcroft and R. E. Tarjan. Efficient planarity testing. Technical Report TR73-165, Cornell University, Computer Science Department, Apr. 1973.

[19] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *Journal of Algorithms*, 26(2):238–274, Feb. 1998.

[20] P. Koebe. Kontaktprobleme der konformen Abbildung. *Berichte über die Verhandlungen der Sächsischen Akad. d. Wiss., Math.–Physische Klasse*, 88:141–164, 1936.

[21] F. Kuhn, R. Wattenhofer, and T. Moscibroda. Unit disk graph approximation, Aug. 11 2004.

[22] X.-Y. Li and Y. Wang. Simple heuristics and PTASs for intersection graphs in wireless ad hoc networks, Sept. 11 2002.

[23] E. Malesinska and V. F. Mathematik. Graph-theoretical models for frequency assignment problems, July 11 1997.

[24] T. Matsui. Approximation algorithms for maximum independent set problems and fractional coloring problems on unit disk graphs, Aug. 13 2000.

[25] T. A. McKee and F. R. McMorris. Topics in intersection graph theory. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[26] Moscibroda, O'Dell, Wattenhofer, and Wattenhofer. Virtual coordinates for ad hoc and sensor networks. In *DialM: Proceedings of the Discrete Algorithms and Methods for Mobile Computing & Communications; later DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, 2004.

[27] T. Moscibroda, R. O'Dell, M. Wattenhofer, and R. Wattenhofer. Virtual Coordinates for Ad hoc and Sensor Networks. In *Proceedings of 2nd Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2004.

[28] R. A. Murphey, P. M. Pardalos, and M. G. C. Resende. Frequency assignment problems. In D.-Z. Du and P. M. Pardalos, editors, *Handbook of combinatorial optimization*, volume Supplement Volume A. Kluwer Academic Publishers, 1999.

[29] Nieberg, Hurink, and Kern. A robust PTAS for maximum weight independent sets in unit disk graphs. In *WG: Graph-Theoretic Concepts in Computer Science, International Workshop WG*, 2004.

[30] Raghavan and Spinrad. Robust algorithms for restricted domains. *ALGORITHMS: Journal of Algorithms*, 48, 2003.

[31] D. J. Rosenkrantz, H. B. H. Iii, H. Breu, M. V. Marathe, and S. S. Ravi. Simple heuristics for unit disk graphs, 1995.

[32] T. Rusterholz. Report On the Approximation of Unit Disk Graph Coordinates. `http://dcg.ethz.ch/theses/ss03/virtualCoordinates_report.pdf`, 2003.

[33] H. Sachs. Coin graphs, polyhedra, and conformal mapping. In *Proceedings of the 2nd Slovenian conference on Algebraic and topological methods in graph theory*, pages 133–138, Amsterdam, The Netherlands, The Netherlands, 1994. Elsevier Science Publishers B. V.

[34] E. Seidel, K. Jansen, and T. Erlebach. Polynomial-time approximation schemes for geometric graphs, Nov. 09 2001.

[35] C. Toregas, R. Swain, C. Revelle, and L. Bergeman. The location of emergency service facilities. *Oper. Res.*, pages 19:1363–1373, 1971.

[36] D. W. Wang and Y.-S. Kuo. A study on two geometric location problems. *Information Processing Letters*, 1988.

[37] M. Wattenhofer, R. Wattenhofer, and T. Moscibroda. Virtual coordinates for ad hoc and sensor networks, Aug. 11 2004.

[38] A. Wolff and T. Strijk. The Map-Labeling Bibliography. `http://i11www.ira.uka.de/map-labeling/bibliography/`, 1996.

Příloha A

EDGAR - Editor of Graphs and Representations

Součástí práce je program Edgar, který umožňuje editaci grafu, diskových grafů, aplikaci algoritmů na daný graf a v neposlední řadě implementuje některé algoritmy zmíněné v textu práce s možností krokování a vizualizace jejich fungování.

Vlastnosti programu:

- prostředí pro ruční editaci reprezentací diskových grafů
- prostředí pro ruční editaci geometrických grafů
- aplikace několika algoritmů na daný graf
- prostředí pro vizualizaci algoritmů na diskových grafech

Primárním účelem bylo napsat program pro editaci reprezentací a vizualizaci popisovaných algoritmů. Jako vedlejší produkt vznikla platforma pro editaci geometrických grafů grafů s možností aplikace algoritmů buď vlastnoručně napsaných, či připojených z dalších knihoven (viz dále).

Program je multiplatformní, jeho vývoj probíhal na systému Windows a Linux. Uživatelské rozhraní je postaveno na toolkitu Qt a grafové struktury využívají knihovny Boost, resp. její části Boost Graph Library.

Zdroje:
Qt – *http://www.trolltech.com*
Boost – *http://www.boost.org*

A.1 Implementované algoritmy

V hlavní části textu jsme popsali algoritmy, které aproximují Maximum Independent Set na diskových grafech. V programu jsou implementovány následující algoritmy:

- Greedy 3-aproximace na UDG s reprezentací
- Greedy 3-aproximace na UDG bez reprezentace
- Greedy 5-aproximace na UDG bez reprezentace
- Greedy 5-aproximace na DG s reprezentací
- Greedy 5-aproximace na DG bez reprezentace
- PTAS k-aproximace na UDG s reprezentací
- Optimální řešení

Algoritmy jsou implementovány v rychlé (neanimované) a v animované verzi (kromě algoritmu pro optimální řešení). Neanimované verze jsou vhodné ke srovnání výsledků aproximací.

V následující sekci se podíváme, jak pracovat s programem.

A.2 Uživatelská dokumentace

V této části popíšeme funkce ovládacích prvků v programu a dále ukážeme, jak krok za krokem vyzkoušet algoritmy. Předpokládáme, že je program nainstalován.

A.2.1 Ovládací prvky

Po spuštění se objeví okno, v jehož levé části je kreslicí plocha, vpravo ovládací prvky a nahoře menu. Pravý panel slouží k ovládání kreslicí plochy a v menu se nachází funkce pro práci s grafem a algoritmy.

Pravý ovládací panel nabízí tyto volby:

- *Left button* – funkce levého tlačítka myši: nový disk, přesun, smazání, změna poloměru.
- *Right button* – funkce pravého tlačítka myši: přesun, smazání, změna poloměru.
- *Object shape* – zobrazení disků jako kružnice, kruh, žádný.
- *Redraw* – obnovení kreslicí plochy.
- *Algorithm* – zkratka k položce v menu *Apply algorithm*.
- *Show graph* – zobrazí hrany půnikového grafu a též středy disků. Při nastavení tvaru objektu jako *Empty* je vidět průnikový graf.
- *Show centers* – zobrazí středy disků.
- *Show edges* – zobrazí hrany průnikového grafu.
- *Unit Disk Graph* – přepnutí typu diskového grafu na jednotkový či obecný.
- *Width*, *Height* – šířka a výška kreslicí plochy.
- *Zoom* – přiblížení či oddálení kreslicí plochy.
- *Apply* – provedení změn v rozměrech a přiblížení plochy.
- *Fit visible* – nastavení velikosti plochy podle rozměrů okna.

Popis funkcí v menu *Graf*:

- *Make everything visible* – posune disky tak aby nepřekračovaly okraj kreslicí plochy, případně plochu zvětší.
- *Redraw* – obnovení kreslicí plochy.
- *Center view* – je-li skutečná velikost kreslicí plochy větší než viditelná oblast, posune se tak, že je vidět střední oblast.
- *Clear graph* – Smaže disky.
- *Order ascending by color* – seřadí objekty na ploše tak, že nižší barva překrývá vyšší.
- *Order descending by color* – seřadí objekty na ploše tak, že vyšší barva překrývá nižší.
- *Open disc graph* – otevře novou záložku s geometrickým grafem odvozeným z reprezentace.
- *Apply algorithm* – zobrazí seznam algoritmů k aplikaci na graf.

- *Run animated algorithm* – zobrazí seznam animovaných algoritmů a poté animační prostředí.
- *Show graph stats* – otevře okno s informacemi o grafu (počet disků, hran, počet disků podle barev).

Animační prostředí vyvolané z menu *Graf*, položka *Run animated algorithm*, zobrazí seznam disků a následné vybrání otevře animační prostředí s následujícími funkčními prvky:

- *Step* – provede jeden krok ve schématu.
- *Next* – pokračuje v algoritmu až do další fáze.
- *Finish* – pokračuje v algoritmu až do konce.
- *Restart* – nastaví animaci do počátečního stavu.
- *Play* – kroky algoritmu následují v intervalech daných hodnotou v poli vedle tlačítka (v milisekundách).
- *Information* – otevře dialog s informacemi o grafu (počty disků).
- *Close* – zavře animační prostředí.

A.2.2 Krok za krokem

Zvolíme, jaký typ diskové reprezentace chceme vytvořit, zda pro *Unit Disk Graf* nebo *Disk Graf*.

Diskovou reprezentaci lze získat dvěma způsoby:

- ruční editací
- vygenerováním náhodné množiny

Ruční editace probíhá tak, že myší jednoduše klikáme v kreslicí oblasti. Pro názornost lze zobrazit hrany, pokud se disky protínají (tlačítko *Show edges*). S disky lze pohybovat tažením, za předpokladu, že je jako funkce tlačítka zvoleno *Move disk*, podobně lze zvolit mazání disků *Delete disk*.

Pro vygenerování množiny disků vybereme v menu *Graf* položku *Apply algorithm*, dále v dialogu algoritmus pro vygenerování daného typu grafu

- *Generate Disk Graf*
- *Generate Unit Disk Graf*

počet disků a poloměry.

Stejným způsobem vybereme algoritmus pro aplikaci na dané množině disků.

Výstupem algoritmu je obarvení disků podle toho, zda disk leží v maximální nezávislé množině, nebo ne. Červené disky tvoří maximální nezávislou množinu, šedé disky byly „smazány“ v průběhu algoritmu. Pro názornost je možné změnit tvar disků z kružnic na disky (tlačítko *Object shape*). Pořadí vykreslení barev (a tedy překrytí disků) lze změnit v menu *Graf*, položky *Order ascending by color* a *Order descending by color*.

Počet disků zobrazených barev je uveden v dialogu *Graph stats*, který lze zobrazit přes *Graf*, položka *Show graph stats*. Zde jsou též údaje o počtu disků, hran a poloměrech.

Třetím výstupem z algoritmu, jímž je velikost nalezené maximální nezávislé množiny, je záznam do logu, který se otevře přes menu *Edgar*, položka *Show log*.

A.2.2.1 Animované algoritmy

Animované verze algoritmů doplňují textovou část práce. Algoritmy jsou rozfázovány podle schémat uvedených v textové části, popřípadě více rozšířeny pro větší názornost. Schéma má dvě úrovně zanoření, pracovně nazvané fáze a kroky. Fáze je blok jednoho či více kroků.

Animační prostředí se vyvolá v menu *Graf*, položka *Run animated algorithm*. Zobrazí se výběr algoritmů s krátkým popisem. Zvolíme nějaký. Otevře se panel pro řízení animace. V horní části je schéma algoritmu, v dolní jsou ovládací prvky, jejichž funkce byla popsána výše.

Právě provedený krok ve schématu je zvýrazněn.

Poznámky

U algoritmu *MIS k-aproximace na UDG s reprezentací* je krok, ve kterém se enumerují všechny nezávislé množiny disků v daném čtverci, což může

být zdlouhavé krokovat. Je možné přeskočit až k výsledné množině tlačítkem *Next*.

A.2.3 Poznámky k implementacím

Optimální řešení MIS

Algoritmus řeší úlohu hrubou silou. Řešení probíhá v každé souvislé komponentě grafu samostatně. Pro malé instance lze řešení najít v krátkém čase. Algoritmus nejde přerušit jinak než ukončením celého programu.

MIS k-aproximace pro UDG bez reprezentace

Řešení nalezené algoritmem nemusí být maximální co do inkluze, neboť disky protínající aktivní přímky nejsou dále brány v úvahu. Pro úplnost jsme implementovali obohacení původního řešení hladovým algoritmem *MIS 3-aproximace na UDG s reprezentací*.

3-aproximace a 5-aproximace bez reprezentace

U algoritmů *MIS 3-aproximace na UDG bez reprezentace* a *MIS 5-aproximace na DG bez reprezentace* je kandidát předvybírán podle vlastnosti na jeho okolí. Přednost mají vrcholy s nízkým stupněm, neboť jejich okolí neobsahuje velkou nezávislou množinu.

A.3 Programy a grafové knihovny

V době psaní práce nebyly k dispozici žádné programy, které by se zabývaly editací či algoritmy na průnikových grafech.

Při průzkumu jsme narazili na tyto programy a knihovny, o kterých bychom se chtěli zmínit.

A.3.1 Pigale

(Public Implementation of Graph Algorithm Library and Editor)

Jak název napovídá, jedná se o knihovnu grafových algoritmů a editor. Obsahuje implementaci mnoha algoritmů pro generování, nakreslení a práci s nakreslením grafu.

Několik algoritmů bylo použito v programu Edgar jako ukázka možnosti rozšíření o algoritmy z externích knihoven.

Zdroj: *http://pigale.sourceforge.net*

A.3.2 Boost Graph Library

Boost je knihovna šablon a funkcí v C++, která implementuje široký okruh užitečných algoritmů a datových struktur. My jsme použili část Boost Graph Library, která implementuje datové struktury pro grafy a několik algoritmů.

Velkou výhodou této knihovny je její rozmanitost a odladěnost. Datové struktury jsou dobře navržené a snadno použitelné.

Zdroj: *http://www.boost.org*

A.4 Instalace a spuštění

Na přiloženém CD je binární spustitelný soubor pro Windows a Linux pro rychlé spuštění. Zdrojové kódy jsou též k dispozici. Obsah CD je podrobněji rozepsán v souboru `readme`.

Windows

Binární verze programu se nachází v adresáři `binaries/windows/` CD. Pokud nefunguje spuštění z CD, nakopírujte adresář na disk a spusťte z disku.

Linux

Zdrojové kódy programu a potřebných knihoven najdete v adresáři `sources/` na přiloženém CD. Nakopírujte je na disk a spusťte skript *make-edgar*. Zkompiluje se nejprve Pigale a následně Edgar. Výsledný program bude v adresáři *bin* a lze ho spustit skriptem *run-edgar*.