Reducering av k-apex grafer till planära grafer

Frans Sontag 1901357
Kandidatavhandling i datavetenskap
Handledare: Johan Lilius
Fakulteten för Naturvetenskaper och Teknik
Åbo Akademi
2022

Innehåll

1 Introduktion

Inom datavetenskapen existerar det många problem som är av både teoretiskt och praktiskt intresse men som inte har någon effektiv lösning, antingen på grund av att ingen ännu funnits eller för att problemet bevisligen inte går att lösa deterministiskt inom polynomial tid. Trots det så kan man i flera fall tillämpa olika tekniker för att avgränsa problemet på ett sådant vis att problemet har en lösning som är tillräckligt effektiv för att vara praktisk.

Då det gäller grafteori är det ett väl etablerat faktum att grafer som innehar vissa egenskaper gör det möjligt att lösa flera av dessa problem mycket effektivt på dem trots att det i det allmänna fallet inte existerar en effektiv lösning. Det är då alltså naturligt att fråga sig om det är möjligt att tillämpa en algoritm att finna en graf ur den ursprungliga grafen som innehar någon önskad egenskap. Denna fråga är vad avhandlingen har som mål att undersöka. samt att se på exempel var en lösning kunde tillämpas.

1.1 Problemspecificering

Då det gäller att modifiera en graf genom att ta något steg som minskar på grafen för att skapa en så kallad minora graf finns det tre sätt att gå tillväga. Dessa är nodradering, kantradering och kantkontraktion. Fokuset i avhandlingen kommer att ligga på lösningar vars basis är nodradering. De två andra metoderna kommer att tillämpas i processen men det slutliga resultatet kommer att bygga på nodradering.

Det finns ett flertal olika egenskaper man kunder önska att en graf innehar som skulle försnabba lösningar och approximation på olika problem, exempelvis att grafen vore ett träd eller att grafen vore planär. Eftersom det inte existerar lösningar för ett antal av dessa eller eftersom de inte har tillräcklig praktisk nytta så kommer avhandlingen att begränsa sig till att visa teoretiska lösningar till hur planaritet kan uppnås i enkla grafer.

1.2 NP-kompletthet av nodraderingsproblemet

Det bevisas i [21] att nodraderings problemet är NP-komplett för icke-triviala ärftliga grafegenskaper som kan testas i polynomial tid. Eftersom planaritet är en egenskap som faller i denna kategori så betyder det även att problemet avhandlingen fokuserar på också är det. I det här skedet kunde man då anta att problemet inte har någon effektiv lösning men det visar sig att det existerar en metod som tillåter oss att hitta en lösning i polynomial tid.

I [22] visas det med hänvisning till metoder ur [8] att man kan genom parameterisering

hitta en lösning till problemet i polynomial tid så att lösningen är beroende på en funktion $f(k)$ för parametern k. Parametern k betyder i fallet av vårt problem hur många noder som behöver raderas. Parameteriserad komplexitetsteori utvecklas vidare i sektion 2.11 då det behövs för att förstå effektiviteten av lösningarna som presenteras i sektion 3.

2 Matematisk bakgrund

Före algoritmerna presenteras behöver vi först en hel del bakgrunds teori för att komplettera läsarens matematiska kunskaper till den grad att innehållet kan förstås. Som förhandskunskaper för innehållet antas elementära kunskaper inom mängdlära, logik och formella språk samt bekantskap med stora O-notation. I alla andra fall kommer det matematiska innehållet att utvecklas i texten.

Observera redan i det här skedet att så gott som all teori i den här delen inte är väldigt rigoröst definierat även om det definieras formellt. Mycket saknas då målet har varit att hålla teorin så kortfattad som möjligt, så det kan vara värt att se på källorna.

2.1 Grundläggande grafteori

Inom graf teori då det talas om grafer så menas ett ordnat par av två mängder, en mängd av noder och en mängd av kanter. Det vill säga att en graf G skulle skrivas som $G = (V, E)$ där V är mängden noder och E är mängden kanter där varje kant är ett par av noder tillhörande V [1, s. 2]. Definitionen kan förlängas ytterligare men det är inte nödvändigt för avhandlingens ändamål. Igenom hela texten bör man anta att då det talas om grafer så betyder det mera specifikt enkla grafer. Det vill säga att grafen är oriktad, oviktad och ändlig. Detta gäller om inte annat skrivits.

Det finns ett antal grundläggande men speciella grafer som vi kommer att behöva då vi utvecklar det matematiska innehållet senare i texten. Dessa är bipartita grafer, kompletta grafer och träd. De andra slags speciella grafer vi behöver kommer att beskrivas närmare inom sina egna sektioner då de inte är grundläggande men ändå centrala till avhandlingens innehåll. Dessutom måste vi även se på ekvivalenta grafer samt delgrafer.

Den första av dessa, alltså bipartita grafen beskriver vi som en graf där nod mängden V kan delas in i två mängder (X, Y) så att $\forall e \in E(G)$ så gäller det att $e = (x, y)$, $x \in X$, $y \in Y$. Det vill säga att kanter inte kan existera mellan noder som tillhör samma nod mängd partition [1, s. 4].

Den andra slags grafen, kompletta grafen beskriver en graf där kant mängden är maximalt stor för sin nod mängd. Alltså att varje nod är granne med varje annan nod, eller att varje nod har $n-1$ kanter den är incident med [1, s. 4]. Dessa grafer betecknas som K_n för den kompletta grafen med n noder. Definitionen av kompletthet kan dessutom förlängas ytterligare att passa för bipartitia grafer. Det vill säga varje nod inom sin nod mängds partition är granne med varje nod i den andra partitionen. Dessa betecknas då som $K_{n,m}$ för den kompletta bipartita grafen med n och m noder i respektive partition.

Slutligen vad gäller speciella grafer så behöver vi ännu se på träd grafer. Vi definierar ett träd som en sammanhängade graf där det inte existerar en enda cykel. Alltså, det ska då enbart finnas en enda stig mellan vilka två noder som helst i grafen, stigar definieras närmare följande sektion. Stämmer inte detta så är grafen inte ett träd. Det är värt att notera att då man talar om träd så menas ofta rotat träd där man har en bestämd nod, men i texten så antas detta att inte vara fallet ifall annat inte nämnts.

Då man inom mängdlära talar om delmängder så menar man att en mängd X' är en delmängd av X om och endast om alla dess element även finns i X. Det kan förlängas till grafer på det vis att delgrafens nodmängd och kantmängd är delmängder av den ursprungliga grafen. Alltså G' är en delgraf av en graf G om $V(G') \subseteq V(G)$ och $E(G') \subseteq E(G)$.

Den sista grundläggande delen inom graf teori som behövs för att utveckla den resterande teorin är det som kallas för graf isomorfism. Det vill säga att vi behöver något vis att se om två grafer är strukturellt ekvivalenta. En graf G och en graf H sägs vara isomorfa om det existerar en bijektion $\phi : V(G) \to V(H)$ så att två noder som bildar en kant i G också gör så i H. Det betcknas som $G \cong H$ [1, s. 12].

2.2 Linjära graf sekvenser

Några rätt så grundläggande men viktiga termer att hålla koll på är stigar och cykler(Eng. path, cycle). En stig är en graf som kan ordnas i en linjär sekvens där två noder ligger efter varandra om de är grannar i grafen, alltså att de delar en kant. Stigar betecknas antingen som P^k för en stig med k noder eller v_iPv_k för stigen $v_i, v_{i+1}, \ldots, v_k$ [12, s. 7]. En viktig observation är att en nod kan enbart förekomma en gång i en stig, vi ser snart varför.

En cykel är väldigt liknande till en stig i och med att de är sekvenser av noder. Skillnaden mellan de två är dock att det sista noden i en cykel är granne med den första noden, de bildar då en loop eller cykel. Dessa betecknas som C^k för cykeln med k noder.

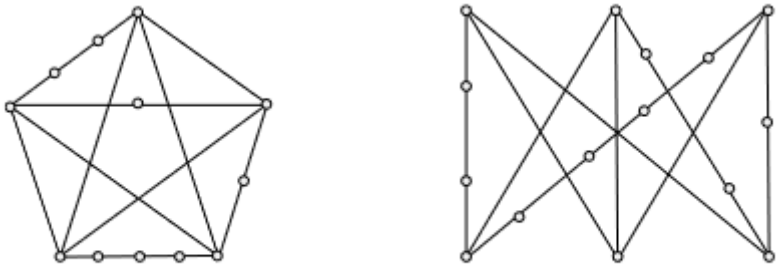
Det som vi slutligen ser på är det som kallas för en vandring(Eng. walk). Till skillnad från cykler och stigar som är nod sekvenser med kanter mellan varje par är en vandring en icke-tom alternerande sekvens av noder och kanter $W^k = v_0e_0v_1e_1...e_{k-1}v_k$.

2.3 Homeomorfism

Homeomorfism är rätt centralt vad gäller planära grafer då det ingår i ett av två krav på planaritet, nämligen Kuratowkis sats. Vi säger att två grafer G_1 och G_2 är homeomorfa om och endast om de har isomorfa expansioner(Eng. subdivision)[1, s. 246] [12, s. 20].

En sådan här expansion fås genom att genomföra en serie kant uppdelningar på grafen G på det vis att man lägger till noder i en kant. Alltså om vi har en kant $e \in E(G) = (x_1, x_3)$ så delar vi upp den genom att lägga till noden x_2 emellan x_1 och x_3, då bildas kanterna (x_1, x_2) och (x_2, x_3).

Vi kallar dessutom expansioner av K_5 och $K_{3,3}$ för Kuratowski expansioner [1, s. 268]. Och definitionen kan ännu förlängas till delgrafer sådant att om grafen G har en delgraf H som är homeomorfisk K_5 eller $K_{3,3}$ så är H en Kuratowski delgraf.



Figur 1: En expansion av a) K_5 och b) $K_{3,3}$ [1, s. 246]

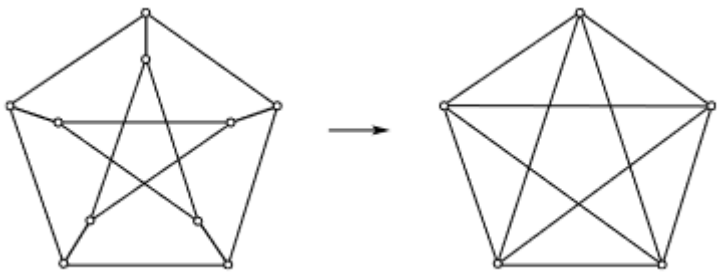
2.4 Minora grafer

Minora grafer(Eng. Graph Minor) är otroligt viktiga då de både teoretiskt motiverar mycket samt att vi rent praktiskt försöker skapa sådana grafer. De är rätt så lika delgrafer men betydligt mer generaliserade. En graf H är en minora graf av en graf G om den kan skapas genom att radera ett antal noder eller kanter, eller igenom att sammandra kanter [1, s. 268]. Vi betecknar dessa även som $H \preceq G$.

Nod och kant radering är ganska lätta och intuitiva att förstå. De utesluts helt enkelt ur sina respektive mängder, $V(G) - v$ och $E(G) - e$. Det som dock är lite tvetydligt i litteraturen är det att vissa källor menar att enbart isolerade noder kunde raderas medan andra inte tar ställning till saken. I texten kommer vi att tillåta radering av noder oavsett med den motivation att då man kan isolera noden genom kant raderingar på förhand. För att klargöra så kan vi med nodradering få $K_5 \preceq K_6$ genom en nod radering där alla incidenta kanter implicit raderas.

Raderings operationerna är rätt så intuitiva att se vad de åstadkommer, det samma kan kanske inte sägas om kant sammandragning. Då man sammandrar en kant $e = (v_1, v_2)$ får man en ny nod v_3 som nu är incident med alla kanter som både v_1 och v_2 var incidenta med. Resulterar sammandragningen i en situation där vi får en multi graf, alltså en som tillåter flera kanter mellan två noder så slås dessa kanter ihop.

Som det nämndes tidigare så är minora grafer betydligt mer generaliserad än delgrafer, detta gäller till den grad att för varje delgraf $G' \subseteq G$ så gäller även $G' \preceq G$. Det intressanta är att om en graf G innehar en egenskap som är delgraf-sluten eller minora-sluten så gäller det att alla $H \subseteq G$ och $H \preceq G$ också innehar den egenskapen, det är då frågan om en ärftlig egenskap vilket nämndes kort i sektion 1.2. Detta blir utvecklas vidare i sektion 2.7 då det är det bästa sättet att formulera dessa grafer.



Figur 2: Kant sammandragning av Petersen grafen till K_5 [1, s. 269]

2.5 Planaritet

En graf sägs vara planär om och endast om det existerar en inbäddning $\tilde{G}$ i planet. Det vill säga att alla noder $V(G) \subseteq \mathbb{R}^2$ och varje kant ska kunna ritas i planet $\mathbb{R}^2$ på så vis att det inte finns några kanter som korsar varandra. Det gäller alltså då att korsningstalet för

grafen G är $Cr(G) = 0$ [1, s. 244].

Vi introducerar konceptet med graf sidor som är nära förknippat med planära grafer. Vi säger att sidor i en planär graf G är ytor i planet $\mathbb{R}^2$ som är uppdelad av kanterna i graf inbäddningen. Dessa ytor betecknas som mängden $F(G)$.

Detta är i sig väldigt lätt att förstå intuitivt vad planaritet innebär men det är trots det inte väldigt rigoröst vad gäller matematiken så vi behöver något mer formellt att ställa som krav. Lyckligtvis finns det två starka krav vi kan använda som bygger på att utesluta de grafer som innehåller de minsta möjliga icke-planära grafer.

2.5.1 Kuratowskis och Wagners satser

De krav som ställs för att en graf G skall vara planär är att den uppfyller kraven som ställs i Kuratowskis och Wagners satser. Satserna bygger på teorin som beskrevs i sektionerna 2.3 samt 2.4.

Kuratowskis sats [1, s. 268]
En graf G är planär om och endast om det inte existerar någon delgraf H av G som är en Kuratowski delgraf.

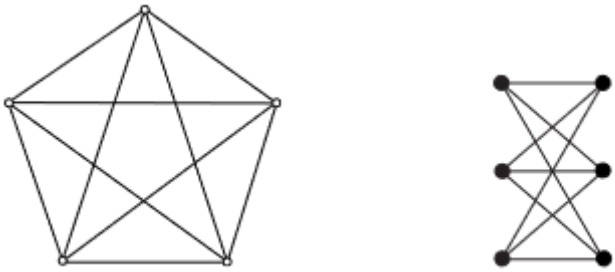
Wagners sats [1, s. 269]
En graf G är planär om och endast om den inte har $K_{3,3}$ eller K_5 som en minora graf.

Om någon av dessa två satser inte följs så är det inte möjligt att grafen är planär. Dessa två krav är utmärkta i teorin men de är inte särskilt praktiska då det gäller att testa för planaritet. Därför kommer det senare i texten att presenteras en effektiv algoritm för att testa planaritet samt återge en inbäddning i $\mathbb{R}^2$ effektivt. Dessutom ges några lätta formler som kan användas för att testa grafer för planaritet i följande del, observera dock att de inte nödvändigtvis kan tillämpas av datorer enkelt.

2.5.2 Algebraiska planaritets krav

Det finns några ganska bra algebraiska krav som kan ställas för planära grafer som är lättare att arbeta med än Kuratowski och Wagners satser.

1) Om $|V(G)| \geq 3$ så är $|E(G)| \leq 3|V(G)| - 6$ [1, s. 260]
2) $\forall v \in V(G)$, $deg(v) \leq 5$ [1, s. 260]



Figur 3: K_5 och $K_{3,3}$ [1, s. 269] [12, s. 17]

3) $|V(G)| - |E(G)| + |F(G)| = 2$ [1, s. 259]

2.6 Apex- och k-Apex-grafer

Eftersom vi vill uppnå planaritet ur vår graf så behövs någon slags klassificering av de grafer som algoritmerna arbetar med för att uppnå planaritet. Dessa grafer kallas för k-apex grafer där $k \in \mathbb{N}$ kallas för apex talet av grafen G. Det är frågan om grafer var raderingen av högst k noder resulterar i att grafen blir planär. Det vill säga att vi identifierar en mängd $X \subset V(G)$ var $|X| \leq k$ så att $V(G) - X$ är planär[19][22]. Om egenskapen håller för grafen G så säger man att $G \in Apex(k)$. I fallet där $k = 1$ så kallas grafen för en apex graf. Det är värt att notera i det här skedet att då $k = 0$ så har vi en planär graf och den behöver inte modifieras.

Ett koncept är relaterat till apex-grafer är det som kallas för nästan planära grafer. I [25] definieras termen som en apex-graf där oavsett vilken nod $v \in V(G)$(samt dess incidenta kanter) man raderar, resulterar $V(G) - v$ i en planär graf. Ett exempel på en graf med denna egenskap är K_5 eftersom oavsett vilken nod som försvinner resulterar det i att grafen blir K_4 som har en planär inbäddning $\tilde{K_4}$. Man bör observera att termen är rätt så tvetydig då den har i litteraturen använts både för att beskriva egenskapen som definierad i [25] och för att beskriva apex-grafer överlag.

2.7 Förbjudna minora grafer

Som det nämndes i Wagners sats så kan man formulera en planär graf på det viset att den inte innehåller K_5 eller $K_{3,3}$ som minora grafer. Detta är dock aningen begränsande då det enbart gäller planära grafer och vi har ett intresse av flera graf klasser. I [24] bevisar de något som kallas för Wagners konjektur vilket i dagens läge kallas för graf minora satsen. Den lyder något som följande:

Graf minora satsen
Oriktade grafer som är partiellt ordnade under graf minora förhållandet bildar en väl-kvasi-ordning(well-quasi-ordering).
En väl-kvasi-ordning definieras som en reflexiv samt transitiv binär relation för en oändlig sekvens element $x_1, x_2, x_3, ... \in X$ som innehåller par $x_i \leq x_j$ där $i < j$ [12, s. 316].

Ett mer intuitivt men ekvivalent sätt att tolka satsen är att vi kan beskriva en graf klass K utgående från en ändlig mängd uteslutna eller förbjudna grafer. I vårt fall så beskriver vi dessa grafer utgående från förbjudna minora grafer. Detta var fallet med planära grafer som inte fick ha K_5 eller $K_{3,3}$ som minora grafer och det kan då generaliseras till flera graf klasser. Trots att vi vet det här nu så måste vi ännu observera att dessa ändliga mängder inte är fullständigt kända för många graf klasser och att det är målet för en hel del forskning inom området [16]. Det är dock nödvändigt att ta upp satsen i och med att det existerar sådana minora grafer som lösnings algoritmer behöver se upp för.

2.8 Inducerade matchningar och oberoende nod mängder

Två stycken relativt ovanliga graf strukturer som kommer att behövas för att förstå lösnings algoritmen är det som kallas för inducerad matchningar(Eng. induced matching) samt oberoende nod mängder(Eng. stable set). Trots att dessa förekommer ganska sällan så är de inte komplicerade.

En inducerad matchning M en mängd kanter i grafen G sådant att varje nod tillhörande en kant i matchningen M är exklusiv till enbart den kanten i matchningen [9]. Det är då intuitivt att tolka det som så att att varje kant i matchningen är separerad från varje annan kant av åtminstone en kant utanför matchningen.

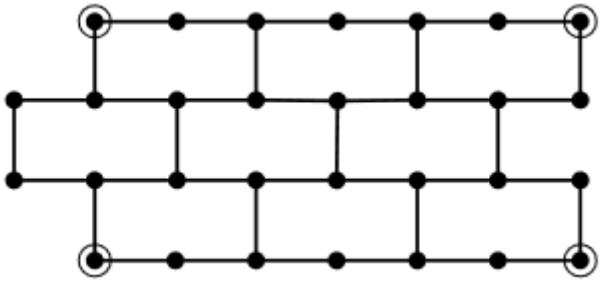
En oberoende nod mängd S är en nod mängd så att inget par av noder i den bildar en kant i grafen G eller i andra ord att inget par av noderna är grannar [1, s. 295]. Det finns alltid då minst en annan nod emellan två noder tillhörande denna mängd.

2.9 Graf väggar och galler

Väggar och galler är speciella slags grafer med intressanta egenskaper ur graf minora teorins perspektiv. Dessa behövs då vi återger Kawarabayashis algoritm för att uppnå planaritet. Till att börja med så definierar vi ett $r \times s$-galler(Eng. $r \times s$-grid) med nod mängden $r \times s$ sådant att paret (i, j) är granne med (i', j') om och endast om $|i - i'| + |j - j'| = 1$,

följande bör även gälla $r,s \geq 2$ [20].

Utifrån den definitionen kan vi nu definiera en elementär r-vägg som en minora graf som fås ur $2r \times r$-galler genom att radera alla kanter med änd-noderna $2i-1,2j-1$ och $2i-1,2j$ för alla $i = 1,2,...,r$ och $j = 1,2,...,(r-1)/2$ samt att radera resulterande noder där graden är 1. Slutligen så definieras en r-vägg som en elementär r-vägg där man utför kant uppdelningar så att de utgör en stig istället [20].



Figur 4: En elementär 4-vägg [20]

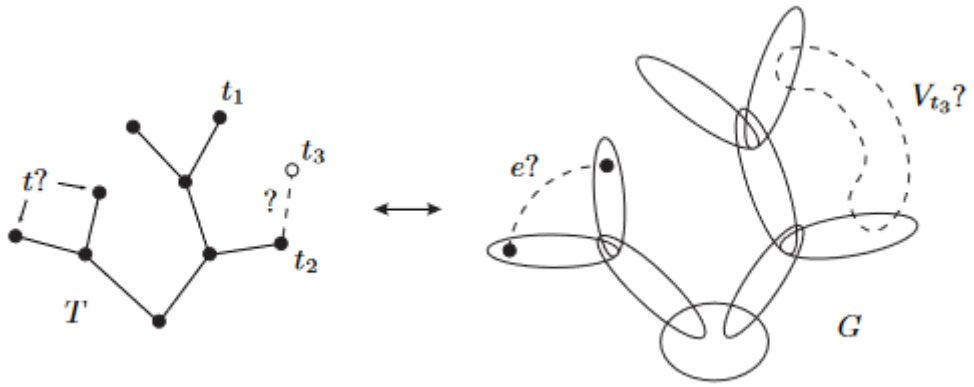
2.10 Trädbredd och Träd-dekomposition

En intressant egenskap vad gäller graf algoritmer är det vi kallar för trädbredd. Sådär informellt kunde vi säga att det är ett mått på hur nära en graf är till ett träd. Då man har ett träd så är trädbredden $tw(G) = 1$ medan om grafen inte är det så gäller det att $tw(G) > 1$.

Intresset vi har av denna egenskap ligger i faktumet att då vi vet eller kan begränsa detta värde så är det möjligt i flera fall att effektivera algoritmer på sådana grafer. Men för att det ska vara möjligt att beräkna detta värde så behöver vi definiera det som kallas för träd-dekomposition.

Vi definierar träd dekomposition som en avbildning av en graf till en träd liknande struktur. Mera formellt så definieras det som följande [12, s. 319].
Låt G bara en godtycklig graf, T ett träd och låt $\mathscr{V} = (V_t)_{t \in T}$ vara en familj nod mängder $V_t \subseteq V(G)$ indexerat av av noderna t i T. Paret $(T, \mathscr{V})$ är en träd dekomposition om den uppfyller följande krav:
1: $V(G) = \bigcup_{t \in T} V_t$
2: För varje kant $e \in E(G)$ existerar det en nod $t \in T$ så att båda noderna i e är i V_t
3: $V_{t1} \cap V_{t3} \subseteq V_{t2}$ då $t_1,t_2,t_3 \in V(T)$ fyller kravet $t_2 \in t_1 T t_3$

Nu då vi har en bättre överblick av vad en träd-dekomposition är så kan vi definiera trädbredd ordentligt. Först definierar vi bredden av $(T, \mathcal{V})$ som $max\{|V_t| - 1, t \in T\}$. Utgående från det kan vi sedan definiera $tw(G)$ som den minsta bredden av alla potentiella träd-dekompositioner [12, s. 321].



Figur 5: Träd dekomposition där delar utesluts efter kraven 2 och 3 [12, s. 319]

2.11 Parameteriserad komplexitetsteori

Slutligen gällande teorin ger vi en starkare definition av parameteriserad tidskomplexitet. I klassisk komplexitets teori bryr vi oss huvudsakligen om den största faktorn ur ett matematiskt uttryck vilket ger oss en uppskattning på vad körtiden är i det värsta fallet. Den stora skillanden mellan klassisk och parameteriserad komplexitet ligger i att man nu utnyttjar ytterligare en parameter för att beskriva komplexiteten, den kallas oftast för k.

Nyttan i att man använder en till parameter så är det att man med den informationen kan avgränsa problemet på det vis att man genom antaganden kan minska på operationer som behöver genomföras eller möjligheter som behöver övervägas.

Exempelvis, om vi har ett NP-komplett problem med komplexiteten $O(2^n)$ och $n = 1000$ som testar varje möjlighet så kommer den att testa $2^{1000} \approx 1.07 * 10^{301}$. Då om vi kan bestämma exempelvis $k \leq 10$ så får vi istället resultatet $\binom{1000}{10} \approx 2.63 * 10^{23}$ vilket är betydligt effektivare även om det är otroligt stort fortfarande [11].

Nu då det har utvecklats intuitivt så ges ännu en formell definition ur [11] på fixerad-parameter-hanterliga(Eng. Fixed-parameter-tractable) problem vilket är de problem som tas upp i avhandlingen. Vi kallar dessa för FPT efter sitt engelska namn.

Definition 1.
Ett parameteriserat problem är det formella språk $L \subseteq \Sigma^* \times \mathbb{N}$ var Σ är ett bestämmt, ändligt alfabete. I en instans $(x,k) \in \Sigma^* \times \mathbb{N}$ kallas k för parametern.

Definition 2.
Ett parameteriserat problem $L \subseteq \Sigma^* \times \mathbb{N}$ tillhör FPT om det existerar en fixerad parameter algoritm $\mathscr{A}$, en beräknelig funktion $f : \mathbb{N} \to \mathbb{N}$ samt en konstant c given sådant att för en instans $(x,k) \in \Sigma^* \times \mathbb{N}$ så bestämmer algoritmen $\mathscr{A}$ deterministiskt om $(x,k) \in L$ inom tids begränsningen $f(k) * |(x,k)|^c$.

3 Lösnings algoritmer

Med då den nödvändiga matematiken har utvecklats tillräckligt för att bilda sig en förståelse kan lösnings algoritmerna presenteras. Som det beskrevs i sektion 1.2 så är problemet NP-komplett då antalet noder som behöver raderas är varierande. Men då vi arbetar med k-apex-grafer så kan man naturligt parameterisera problemet för att uppnå en polynomial tids lösning beroende av någon funktion $f(k)$.

Det första beviset på en polynomial tids lösning kan härledas ur resultaten presenterade i [23] och [24]. Utgående från dessa resultat existerar det en $O(n^3)$ lösning för varje fixerat k värde. Men en lösning med en tids komplexitet på $O(n^3)$ som dessutom är beroende av någon funktion $f(k)$ är inte särskilt effektiv och det har därför funnits mycket utrymme för förbättring.

Den första förbättringen på lösningen publicerades i [22] år 2008 då de hittade en lösning i $O(n^2)$ vilket är betydligt bättre men fortfarande inte särskilt snabbt. Året därefter publicerades i [19] en algoritm som körs i $O(n)$ tid. Båda algoritmerna är parameteriserade för att fungera på k-apex-grafer med relativt stora värden på $f(k)$ vilket betyder att de snabbt tappar effektivitet då k växer.

3.1 Planaritetstestning och inbäddning

Innan själva lösnings algoritmen presenteras kommer vi att behöva en algoritm för att testa för planaritet samt att återge en planär inbäddning. Detta problem har länge studerats och har gett upphov till ett stort antal lösningar och lösningsmetoder. Den algoritmen som presenteras är inspirerad av PQ-träd vilket är en datastruktur skapad av Booth och Lueker

[5] med målet att tillåta effektiv testning av olika egenskaper.

År 2004 publicerades Boyer och Myrvold [6] en linjär tids algoritm för planaritets testning och inbäddning. Den bygger på att lägga till kanter till inbäddningen på ett sådant vis att det minimerar hur många kanter man behöver göra det med. Vi kommer inte att gå djupt in i hur algoritmen exakt fungerar men vi återger pseudokoden för huvudalgoritmen, koden för funktionerna WU och WD(WalkUp, WalkDown) hittas i bilagorna i orginal artikeln.

Man kan tillämpa andra algoritmer för samma ändamål då det behövs i Kawarabayashis algoritm. Då han gav sin algoritm hänvisades det till algoritmerna i [10] och [17] för steget där man skapar den planära inbäddningen. Med andra ord så finns det flera algoritmer man kan tillämpa och det här är en av dem som kan åstadkomma det.

3.1.1 Algoritm beskrivning

Indata: En simpel graf G med $n \geq 2$ noder och $m \leq 3n-5$ kanter
Utdata: Ett boolskt värde på om grafen G är planär samt en inbäddning $\tilde{G}$ om grafen är planär, annars en Kuratowski delgraf av G i $\tilde{G}$
Körtid: $O(n)$

Lösnings steg:
1 Utför en djupet först sökning samt lågpunkts beräkningar för G
2 Skapa $\tilde{G}$ baserat på G samt en separeradDFSBarnLista för varje nod sorterad efter barn nodens lågpunkt
3 För varje nod v från $n-1$ till 0
4 för varje DFS barn c av v i G
5 Inbädda träd kanten (v^c, c) som en bisammanhängande komponent i $\tilde{G}$
6 För varje bakre kant av G incident med v och en efterföljande w
7 WU($\tilde{G}, v, w$)
8 För varje DFS barn nod c av v i G
9 WD($\tilde{G}, v^c$)
10 För varje bakre kant av G incident med v och en efterföljande w
11 Om $(v^c, w) \notin \tilde{G}$
12 IsoleraKuratowskiDelgraf($\tilde{G}, G, v$)
13 returnera $Falskt, \tilde{G}$
14 HämtaPlanärInbäddning($\tilde{G}$)
15 returnera $Sant, \tilde{G}$

3.2 Kawarabayashis algoritm

I [19] presenteras och bevisas en parameteriserad algoritm för att uppnå planaritet från en k-apex graf genom nod radering i linjär tid. Detta är en stor förbättring över lösningen av Marx och Schlotter [22].

Algoritmen kan i kort beskrivas med några steg. Generera en sekvens grafer ur G genom att sammandra inducerade matchningar eller genom att radera en oberoend nod mängd. Sedan behöver vi identifiera den så kallade universala apex nod mängden A^U och tillämpa den för att radera irrelevanta noder och minska på trädbredden av grafen. Upphäv slutligen alla matchningar i enlighet med lösnings hypotesen och skapa inbäddningen av grafen.

A^U definieras som följande: [19]
1) Låt A^U vara den maximala nod mängden i G så att oavsett hur vi bäddar in G i planet så innehålls A^U i en apex nod mängd.
2) Låt A_P vara en mängd noder i G så att $A_P \cap A^U = \emptyset$ men för varje nod $v \in V(A_P)$ existerar det en nod mängd A' där $x \in A'$ sådan att $G \backslash A'$ kan bäddas in i planet. Observera att $A^U \subseteq A'$.
Vi definierar dessutom irrelevanta noder på det vis att noden v i G är irrelevant om $v \notin (A^U \cup A_P)$. Det gäller dessutom att en planär inbäddning $\tilde{G}$ i $\mathbb{R}^2$ förblir planär då noden v läggs till i inbäddningen $\tilde{G}$.

Det är viktigt att identifiera A^U, vi återger nu all nödvändiga satser och antaganden som behövs för att kunna åstadkomma detta. Bevisen av stegen kommer inte att återges här men de finns detaljerade till största delen i [19]. Observera att det finns enstaka bevis i artikeln som är inte är kompletta som då bör tas med en nypa salt.

Till att börja med behöver ännu några termer definieras ur [19]. En sido-vandring(Eng. facial walk) av en sida är en vandring tagen medsols runt sidan sådan att man får en vandrings sekvens. Antag att grafen G är inbäddad i planet med en delmängd U av dess noder, mängden $\mathscr{F}$ av sido-vandringar av G är en sido-täcka(Eng. Face cover) av U om varje element i U tillhör en sido-vandring i $\mathscr{F}$. Låt även $\tau(U)$ vara den minsta av sido-täckorna av U. Sido avståndet(Face-distance) av två punkter u, v i grafen G är det minsta antalet sidor den kortaste kurvan mellan u och v passerar. Punkterna behöver dock inte nödvändigtvis vara noder utan de kan även vara kanter. Slutligen definieras sido avstånds radien som radien i allmänhet används men avståndet mäts som sido avståndet istället.

Målet med följande tre satser är att motivera följande påstående:
För varje $v_i \in A$ gäller det att $v_i \in A^U$ om sido-täckorna av dess grannar i $G-A$ är stora och att det existerar många disjunkta sidor som har en granne av v_i i $G-A$.

Sats 3.1
Låt G en 3-sammanhängande graf och låt F vara en mängd sido cykler av G. F innehåller då en delmängd F' sådan att avsett vilket par a cykler i F' så är de antingen disjunkta eller så delar de högst en nod. Det gäller även att $|F'| \leq \frac{|F|}{40}$.

Sats 3.2
Låt G en 3-sammanhängande graf och låt $U \subseteq V(G)$. Vi skapar en graf G' genom att lägga till en nod v i G så att v är granne med varje nod i U. Antag då att det behövs $2k+1$ antal parvis disjunkta sidor F_i som behövs för att täcka alla noder ur $U \in G$. Nu gäller det antingen att oavsett hur vi raderar k noder så existerar det en Kuratowski minora graf(en minora graf innehållande K_5 eller $K_{3,3}$) i G', eller så gäller det att det existerar $2k$ skivor(Eng. disk) i G där deras innehållande grafer har en sido avstånds radie av högst 8 sådant att de täcker varje nod i U.

Sats 3.3
Det existerar en icke-minskande funktion $f_1 : \mathbb{N} \to \mathbb{N}$ sådant att följande gäller. G är en 3-sammanhängande graf med $U \subseteq G$. Skapa grafen G' genom att lägga till en nod v så att den är granne med varje nod i U. Antag sedan att det finns minst $f_1(k)$ sidor som krävs för att täcka alla noder i U. Det gäller isåfall att oavsett hur vi raderar k noder så existerar det en Kuratowski minora graf i G', eller så gäller det att det existerar $2k$ skivor i G där deras innehållande grafer har en sido avstånds radie av högst 9 sådant att de täcker varje nod i U.

Ur satserna 3.1, 3.2 och 3.3 är det möjligt att identifiera en universal apex nod mängd. Med den vill vi nu radera irrelevanta noder för en planär delgraf Q av $G-A^U$ med målet att minska dess trädbredd i linjär tid. Irrelevanta noder kan som nämndes tidigare återläggas i en planär inbäddning utan att göra grafen icke-planär, alltså $U \cap A_P = \emptyset$.

Sats 3.4
Antag att A^U av G är given där $|A^U| \leq k$. Antag att $G-A^U$ innehåller en planär delgraf Q samt att C är den yttre cykeln av inbäddningen $\tilde{Q}$. Antag sedan att att varje nod i $Q \backslash C$ att dess grannar i grafen $G-A^U$ innehålls i Q. Om Q innehåller en h-vägg W där $h \geq 2k+1$ så gäller det att nod mängden $X \subseteq W$ som har sido avståndet minst k i väggen W från yttre

cykeln C är irrelevant.

Satsen bör tolkas som att om det existerar en $2k+1$-vägg i planära grafen Q given som tidigare så kan vi radera irrelevanta noder i mitten av $2k+1$-väggen tills det inte finns en sådan längre i grafen Q. Detta kan åstadkommas i linjärtid med följande lemman.

Lemma 3.5
Antag att A^U av G är given där $|A^U| \leq k$. Antag att $G - A^U$ innehåller en planär delgraf Q samt att C är den yttre cykeln av inbäddningen $\tilde{Q}$. Antag sedan att att varje nod i $Q \backslash C$ att dess grannar i grafen $G - A^U$ innehålls i Q. Det existerar då en linjär tids algoritm för att hitta en maximal nod mängden $X \subseteq V(Q)$ sådant att grafen $Q - X$ inte innehåller en (2k+1)-vägg, att radering av X ur $G - A^U$ inte orsakar någon förändring i att hitta en inbäddning av $G - A$ i planet för $A^U \subseteq A$ där $|A| \leq k$ och $A \cap X \neq \emptyset$ genom flera tillämpningar av sats 3.4 och slutligen att oavsett hur $G - A - X$ bäddas in i planet så gäller det för vilken som helst nodmängd A där $A^U \subseteq A,\ |A| \leq k$ och $A \cap X \neq \emptyset$ att $G - A - X$ kan förändras av noder i Q så att varje nod i X innehålls i en skiva av inbäddningen $G - A - X$, vi kan då sätta tillbaka noderna av X i någon skiva så att inbäddningen av $G - A$ är planär.

Lemma 3.6
Låt $l \geq 0$ vara ett fixerat heltal samt att A^U av G är given där $|A^U| \leq k$. Anta som tidigare att $G - A^U$ innehåller en planär delgraf Q med yttre cykeln C av en planär inbäddning $\tilde{Q}$, det gäller då att för varje nod i $Q \backslash C$ att dess grannar i grafen $G - A^U$ innehålls i Q. Låt $F_1, ..., F_l$ vara disjunkta sidorna av grafen Q. Det existerar då en linjär tids algoritm för att hitta den maximala nod mängden $X \subseteq V(Q)$ sådant att varje nod i X är omgiven av $2k$ väggar i $Q - X$ som inte innehåller $F_1, ..., F_l$, eller att raderingen av noderna ur X i $G - A^U$ inte orsakar någon förändring i att hitta en inbäddning av $G - A$ i planet för $A^U \subseteq A$ där $|A| \leq k$ och $A \cap X \neq \emptyset$ genom flera tillämpningar av sats 3.4, eller att $Q - X$ inte innehåller $2kl$-vägg och slutligen att oavsett hur $G - A - X$ bäddas in i planet så gäller det för vilken som helst nodmängd A där $A^U \subseteq A,\ |A| \leq k$ och $A \cap X \neq \emptyset$ att $G - A - X$ kan förändras av noder i Q så att varje nod i X innehålls i en skiva av inbäddningen $G - A - X$, vi kan då sätta tillbaka noderna av X i någon skiva så att inbäddningen av $G - A$ är planär.

Slutligen behöver vi en lösnings hypotes för att låta oss skapa delgrafer G'_i med universala apex nod mängden A_i^U för något värde på i. I algoritmen kommer steg 1 att skapa en sekvens grafer $G = G_0, G_1, \ldots, G_b$ där G_{i+1} fås genom att sammandra en stor inducerad matchning M_i eller genom att radera en oberoende nod mängd vars grad är $l \leq k+3$ i vilket fall $M_i = \emptyset$. Vi antar följande hypotes för att skapa G'_i och A_i^U.

Hypotes 3.7
Låt apex talet av grafen G_{i+1} vara $l \leq k$. Vi får grafen G_i genom att upphäva matchningen M_i ur G_{i+1} där $M_i \geq \varepsilon|G_i|$ för en liten konstant $\varepsilon > 0$. Vi har dessutom en delgraf G'_{i+1} av G_{i+1} samt mängden A^U_{i+1} som uppfyller följande krav: A^U_{i+1} är universala apex nod mängden av G_{i+1} vilket fås ur slutsatsen i sats 3.3 och apex talet av G'_{i+1} är l, G'_{i+1} har en begränsad trädbredd $tw(G'_{i+1}) \leq h(k)$ där vi kan säga att $h(k) = O(k^3)$ samt slutligen att $G_{i+1} - G'_{i+1}$ är planär.

Det som vi vill uppnå med hypotesen är att börjar med G_{i+1}, G'_{i+1} och A^U_{i+1} som uppfyller alla tre krav och då konstruera G_i som uppfyller kraven i hypotesen med universala apex nod mängden A^U_i eller att hitta en förbjuden minora graf för k-apex grafer. Detta ska ske i tiden $O(|G_i|)$. Observera att vi klassar denna som en hypotes då ett bevis saknas tillsvidare i den kopia av artikeln som skribenten hade tillgång till men ingen som har citerat artikeln har motsatt sig resultat vilket ger det trovärdighet.

3.2.1 Algoritm beskrivning

Vi återger nu den slutliga algoritmen [19]. **Indata:** En graf G av storlek n och ett tal k där $k \in \mathbb{N}$
Utdata: En inbäddning av G i planet efter radering av $l \leq k$ noder eller en minora av G som inte tillhör $Apex(k)$ och är minimal med denna egenskap.
Körtid: $O(f(k)n)$ för en funktion $f : \mathbb{N} \to \mathbb{N}$

Lösnings steg:
Steg 0: Radera all noder vars grad är mindre än eller lika med 1. Det gäller härefter att $\forall v \in V(G),\ \ deg(v) \geq 2$. All dessa noder kan återinföras senare i $\mathbb{R}^2$ så att kanten de är incident med inte korsar en annan kant.

Steg 1: Sök en sekvens av grafer $G = G_0, G_1, \ \ldots\ , G_b$ sådan att grafen G_i i sekvensen fås ur grafen G_{i-1} antingen genom att sammandra en inducerad matching med åtminstone $\varepsilon|G_{i-1}|$ kanter för någon liten konstant $\varepsilon > 0$ eller genom att radera en oberoende mängd noder I var $|I| = \varepsilon|G_{i-1}|$ och det gäller att varje nod har graden $l \leq k+3$. I fallet då man raderar oberoende mängden I så gäller det för varje nod v i I att:
1) v har samma grannar som åtminstone l andra noder i I.
2) Det existerar $k+3$ noder i G_{i-1} vilket inte är i I som har exakt samma grannar som v.
Om operationen resulterar i en minimal förbjuden graf minora för k-apex grafer stannar algoritmen och returnerar den. Om inte så fortsätter algoritmen tills b steg har tagits där

b är den minsta positiva heltalet så att $|V(G_b)|$ har mindre än B noder för en konstant B. Det gäller alltså då att $b \leq log_{1/\varepsilon} n$ och $\sum_{i=0}^{b} |G_i| = O(n)$. Det går i varje steg i att hitta en inducerad matchning eller oberoende mängd i $O(|G_i|)$ tid vilket implicerar att hela steget i algoritmen tar linjär tid.

Steg 2: Tillämpa en naiv algoritm för att hitta antingen inbäddning av G_b eller en minimal förbjuden graf minora för k-apex grafer i G_b.

Steg 3: Rekursivt hitta delgrafen G_i' av G_i samt A_i^U som uppfyller kraven i hypotes 3.7 för $i = b, b-1, ..., 0$ eller hitta en minimal förbjuden minora graf för k-apex grafer.

Steg 4: Förläng inbäddningen av G_0' till G_0. Här bör man notera att varje nod i A_0^U är även i $V(G_0')$. Därmed kan detta steg åstadkommas genom att tillämpa planaritets algoritmen ur sektion 3.1.

3.2.2 Uppskattning och förbättring av $f(k)$

Med algoritmen färdigt presenterad så behöver vi ännu få en uppskattning av $f(k)$. För att Kawarabayashis algoritm ska vara snabbare än Marx och Schlotters algoritm så bör dess beroende på funktionen $f(k)$ inte vara avsevärt mycket större. I båda algoritmerna får vi att $f(k)$ kan uppskattas vara ungefär dubbel exponentiell av k [19] [22]. Det betyder att algoritmerna blir betydligt långsammare då k växer. Mera specifikt får vi ur [18] att Marx och Schlotters algoritm har ett beroende på k som är $2^{k^{O(k^3)}}$ medan Kawarabayashi algoritm har beroendet $\Omega(2^{k^{\Omega(k^3)}})$.

Ett intressant resultat ur [18] är att vi får en algoritm som kör i linjär tid för samma problem med beroendet på $f(k)$ som är $2^{O(k \log k)}$. Det noteras att det är rätt nära ett optimalt resultat och det lämnades som ett öppet problem om det går att skapa en algoritm med beroendet på $f(k) = 2^k$ som också kör i linjär tid.

4 Algoritmiska tillämpningar

Utgående från det som presenterades i sektion 3 vet vi nu att det är möjligt att effektivt hitta planära minora grafer ur någon graf G då det gäller att $G \in Apex(k)$. Frågan återstår då vad för användning vi kan ha av detta både teoretiskt och praktiskt sett. Vi kommer nu att se på några exempel där planära grafer kan tillämpas praktiskt eller för att hitta väldigt effektiva lösningar på aningen mer teoretiska problem.

4.1 Algoritmer på planära grafer

Det nämndes kort i introduktionen att vi kan i många fall lösa svåra problem på olika specialfalls grafer, i det här fallet planära grafer. Vi kommer att presentera några problem som har effektiva lösningar i planära grafer då de i allmänna fallet kan även vara NP-kompletta. Utöver det ser vi på hurdan potential det existerar för att generalisera sådana lösningar till nästan planära grafer.

Eppstein presenterade i [14] ett antal problem som har effektiva lösningar i planära grafer. Av dessa ser vi huvudsakligen på kortaste stig problemet(Eng. Shortest path), delgrafs isomorfism problemet samt h-kluster problemet.

Resultatet som presenteras för kortaste stig problemet i planära grafer är att vi kan hitta den kortaste stigen mellan två noder u och v i grafen G som är högst l lång i $O(log\, n)$ efter att vi skapat en rutt datastruktur i $O(n)$ tid. Det här är är en stor förbättring över andra algoritmer som exempelvis Dijkstra algoritmen som kör i $O(m + n\, log\, n)$ [15]. Trots att den är snabbare så är det viktigt att komma ihåg begränsningarna att det enbart gäller planära grafer samt att vi behöver parametern l, den kan dock bestämmas vara diametern av grafen som också kan beräknas enligt [14] i linjär tid om vi vet en övre gräns.

Kortaste stig resultatet var en förbättring över allmänt använda algoritmer men det kan vara svårt att se nyttan i att söka efter en planär minora graf om det bästa vi kan åstadkomma är en förbättring från kvasi-linjär tid till linjär och logaritmiskt tid. Därför ser vi nu på problem som är NP-kompletta vilket torde visa nyttan betydligt bättre.

Det är allmänt känt att graf isomorfism har många användningar, det samma gäller delgrafs isomorfism som exempelvis används inom jämförelse av molekylära strukturer eller krets testning [3] [7]. Det nämns i [14] att delgrafs isomofism är NP-komplett för planära grafer men då vi har en delgrafer av en konstant storlek visar det sig att problemet kan lösas i linjär tid. Precis som tidigare så behöver vi komma ihåg att det finns rätt så strikta begränsningar på tillämpningen av detta men det här är ett klart exempel på ett vanligt problem inom många vetenskapliga områden som i planära fallet kan potentiellt lösas effektivt. På samma vis löses även h-kluster problemet, där man vill hitta en h stor nod mängd som maximerar antalent kanter i $O(n)$ tid för ett konstant värde på h i det planära fallet. Detta är också ett NP-komplett problem som det nu visas att har en effektiv lösning som kunde tillämpas.

Som vi ser existerar det många effektiva deterministiska lösningar på svåra problem då

man har en planär graf. Men vi kan förutom deterministiska lösningar hitta effektiva approximationer för problem som inte har en effektiv lösning annars. Se exempelvis [4] där det presenteras några av dessa.

4.2 Generalisering till andra graf typer

Nu då vi sett hur användbart det kan vara att jobba med planära grafer så dyker eventuellt frågan upp om resultaten kan generaliseras till andra graf klasser som är nära relaterade till planära grafer. Det visar sig att vi kan i vissa fall generalisera till andra slags grafer algoritmiskt för att inte nämna det att man intuitivt ser att det kan vara möjligt för exempelvis nästan planära grafer.

Låt oss betrakta minora-slutna graf familjer där vi binder dess trädbredd till diametern. Alltså varje graf G i graf familjen F där vi för en funktion av diametern $f(D)$ får att trädbredden är $tw(f(D))$. För dessa visar det sig att vi kan tillämpa linjär tids approximationer ur [4]. Dessutom kan vi testa delgrafs isomorfism och inducerad delgrafs isomorfism för dessa grafer i $O(n)$ tid [13]. Man bör observera att då vi arbetar med dessa klasser så ökar tids beroendet på delgrafen H vi vill testa [14].

Så det är möjligt i viss mån att generalisera resultat från planära grafer till andra graf familjer. Det finns dock fortfarande andra slags grafer där vi erhåller samma relation mellan graf diameter och trädbredd men som inte är minora-slutna, det är alltså ett öppet problem vilka dessa är.

5 Sammanfattning

Vi kan nu sammanfatta det som tagits upp igenom avhandlingen, se på några av de öppna frågorna som nämndes och även ta upp en par exempel på andra slags grafer vilket man kunde ha intresse av att skapa genom graf modifiering.

Vi har visat att vi kan uppnå det önskade resultat som togs upp introduktionen, detta även inom en rimlig tids användning om vi har små parameter värden. Vi visade även att det finns många tillämpningar av en planär minora graf som tillåter effektiv lösning av komplexa problem, detta även möjligt för NP-kompletta problem inom polynomial eller till och med linjär tid om vi har konstanta parametrar för dem. Approximationer av sådana problem nämndes kort vilket tillåter flera komplexa problem med stor praktisk nytta att lösas effektivt.

Trots att vi visat bra resultat så bör vi komma ihåg faktumet att det ännu finns öppna problem relaterade till vad vi diskuterat vilket för tillfället förhindrar riktig praktisk användning. Vi vet inte alla förbjudna minora grafer för Apex-grafer och därmed inte heller k-Apex-grafer per definition [16]. Det är även öppet om det är möjligt att lösa nod raderings problemet för k-apex-grafer med beroende på $f = 2^k$ [18]. Och slutligen så är det öppet vilka alla icke minora-slutna grafer som erhåller samma relation mellan graf diameter och trädbredd som exempelvis planära grafer har [13]. Detta har betydlig vikt i hur långt vi kan generalisera algoritmer från planära grafer till andra slags grafer.

Slutligen kan det konstateras att det inte finns någon orsak att begränsa sig till att enbart uppnå planaritet i grafer då det gäller graf modifiering. Det existerar andra graf klasser som precis som planära grafer tillåter effektiva lösningar av olika problem. Exempelvis partiella k-träd kunde vara av intresse då de tillåter parameteriserade lösningar av NP-kompletta problem relativt effektivt [2]. Det kunde alltså vara värt att se för vilka graf klasser det existerar effektiva lösningar på svåra problem och om det är möjligt att uppnå dem genom någon slags reducering eller vidare konstruktion av sin ursprungs graf.

Källförteckning

[1] U.S.R. Murty Adrian Bondy. *Graph Theory*. Graduate Texts in Mathematics. Springer London, 2008. ISBN: 978-1-84628-969-9.

[2] Stefan Arnborg och Andrzej Proskurowski. ”Linear time algorithms for NP-hard problems restricted to partial k-trees”. I: *Discrete applied mathematics* 23.1 (1989), s. 11–24.

[3] Peter J. Artymiuk m. fl. ”Similarity searching in databases of three-dimensional molecules and macromolecules”. I: *Journal of Chemical Information and Computer Sciences* 32.6 (1992). PMID: 1474109, s. 617–630. DOI: 10.1021/ci00010a007. eprint: https://doi.org/10.1021/ci00010a007. URL: https://doi.org/10.1021/ci00010a007.

[4] Brenda S Baker. ”Approximation algorithms for NP-complete problems on planar graphs”. I: *Journal of the ACM (JACM)* 41.1 (1994), s. 153–180.

[5] Kellogg S. Booth och George S. Lueker. ”Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms”. I: *Journal of Computer and System Sciences* 13.3 (1976), s. 335–379. ISSN: 0022-0000. DOI: https://doi.org/10.1016/S0022-0000(76)80045-1.

[6] John M. Boyer och Wendy J. Myrvold. ”On the Cutting Edge: Simplified O(n) Planarity by Edge Addition”. I: *Journal of Graph Algorithms and Applications* 8.3 (2004), s. 241–273. DOI: 10.7155/jgaa.00091.

[7] A.D. Brown och P.R. Thomas. ”Goal-oriented subgraph isomorphism technique for IC device recognition”. I: *IEE Proceedings I Solid State and Electron Devices* 135.6 (1988), s. 141. DOI: 10.1049/ip-i-1.1988.0025. URL: https://doi.org/10.1049%2Fip-i-1.1988.0025.

[8] Leizhen Cai. ”Fixed-parameter tractability of graph modification problems for hereditary properties”. I: *Information Processing Letters* 58.4 (1996), s. 171–176. ISSN: 0020-0190. DOI: https://doi.org/10.1016/0020-0190(96)00050-6.

[9] Kathie Cameron. ”Induced matchings in intersection graphs”. I: *Discrete Mathematics* 278.1 (2004), s. 1–9. ISSN: 0012-365X. DOI: https://doi.org/10.1016/j.disc.2003.05.001.

[10] Norishige Chiba m. fl. ”A linear algorithm for embedding planar graphs using PQ-trees”. I: *Journal of computer and system sciences* 30.1 (1985), s. 54–76.

[11] Marek Cygan m. fl. *Parameterized Algorithms*. Springer, Cham, 2016. DOI: 10.1007/978-3-319-21275-3. URL: https://doi.org/10.1007/978-3-319-21275-3.

[12] Reinhard Diestel. *Graph Theory*. Springer Berlin Heidelberg, 2017. DOI: 10.1007/978-3-662-53622-3. URL: https://doi.org/10.1007%2F978-3-662-53622-3.

[13] David Eppstein. "Diameter and treewidth in minor-closed graph families". I: *Algorithmica* 27.3 (2000), s. 275–291.

[14] David Eppstein. "Subgraph Isomorphism in Planar Graphs and Related Problems". I: *Journal of Graph Algorithms and Applications* 3.3 (1999), s. 1–27. DOI: 10.7155/jgaa.00014.

[15] M.L. Fredman och R.E. Tarjan. "Fibonacci Heaps And Their Uses In Improved Network Optimization Algorithms". I: *25th Annual Symposium onFoundations of Computer Science, 1984.* 1984, s. 338–346. DOI: 10.1109/SFCS.1984.715934.

[16] A. Gupta och R. Impagliazzo. "Computing planar intertwines". I: *[1991] Proceedings 32nd Annual Symposium of Foundations of Computer Science*. 1991, s. 802–811. DOI: 10.1109/SFCS.1991.185452.

[17] John Hopcroft och Robert Tarjan. "Efficient planarity testing". I: *Journal of the ACM (JACM)* 21.4 (1974), s. 549–568.

[18] Bart M. P. Jansen, Daniel Lokshtanov och Saket Saurabh. "A Near-Optimal Planarization Algorithm". I: *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '14. Portland, Oregon: Society for Industrial och Applied Mathematics, 2014, s. 1802–1811. ISBN: 9781611973389.

[19] Ken-ichi Kawarabayashi. "Planarity Allowing Few Error Vertices in Linear Time". I: *2009 50th Annual IEEE Symposium on Foundations of Computer Science*. 2009, s. 639–648. DOI: 10.1109/FOCS.2009.45.

[20] Ken-ichi Kawarabayashi, Robin Thomas och Paul Wollan. "A new proof of the flat wall theorem". I: *Journal of Combinatorial Theory, Series B* 129 (2018), s. 204–238. ISSN: 0095-8956. DOI: https://doi.org/10.1016/j.jctb.2017.09.006.

[21] John M. Lewis och Mihalis Yannakakis. "The node-deletion problem for hereditary properties is NP-complete". I: *Journal of Computer and System Sciences* 20.2 (1980), s. 219–230. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(80)90060-4.

[22] Schlotter I. Marx D. ”Obtaining a Planar Graph by Vertex Deletion”. I: *Algorithmica* 62 (2012), s. 807–822. DOI: 10.1007/s00453-010-9484-z.

[23] N. Robertson och P.D. Seymour. ”Graph Minors .XIII. The Disjoint Paths Problem”. I: *Journal of Combinatorial Theory, Series B* 63.1 (1995), s. 65–110. ISSN: 0095-8956. DOI: https://doi.org/10.1006/jctb.1995.1006.

[24] Neil Robertson och P.D. Seymour. ”Graph Minors. XX. Wagner’s conjecture”. I: *Journal of Combinatorial Theory, Series B* 92.2 (2004). Special Issue Dedicated to Professor W.T. Tutte, s. 325–357. ISSN: 0095-8956. DOI: https://doi.org/10.1016/j.jctb.2004.08.001.

[25] K. Wagner. ”Fastplättbare Graphen”. I: *Journal of Combinatorial Theory* 3.4 (1967), s. 326–365. ISSN: 0021-9800. DOI: https://doi.org/10.1016/S0021-9800(67)80103-0.