Övertäckningsdesigner och extremala hypergrafer

Lars Lindqvist
Kandidatuppsats i matematik
Handledare: Klas Markström
Institutionen för matematik och matematisk statistik, Umeå Universitet
20 juni 2012

Sammanfattning

I denna uppsats introducerar vi grundläggande teori rörande övertäckningsdesigner och extremal hypergrafsteori för läsaren. Vidare beskriver vi en metod för att hitta alla icke-isomorfa extremalgrafer $\mathrm{EX}(n, K_s^r)$, för givna n, r, och s. Slutligen presenterar vi de resultat som hittats med denna metod och jämför kort hur metoden står sig mot den naiva metoden.

Abstract

In this paper we introduce the reader to the fundamentals of covering designs and extremal hypergraph theory. Further, we describe a method of finding all non-isomorphic extremal hypergraphs $\mathrm{EX}(n, K_s^r)$, for given n, r and s. Lastly, we present the results found by this method and make some comparisons regarding efficiency between this and the naive method.

Innehåll

1 Bakgrund

En övertäckningsdesign, (n, k, t), eller (n, k, t)-design, är en uppsättning av k-delmängder, som kallas block, av en n-mängd sådan att alla t-delmängder ingår i minst ett block. En (n, k, t)-design är minimal om det inte finns någon annan design med färre block, och antalet block i en minimal (n, k, t)-design betecknas $C(n, k, t)$.

Fortsättningsvis kommer vi att låta n-mängden i en design bestå av heltalen $1, 2, ..., n$, och vi identifierar designen med dess block. Man kan enkelt se att de $\binom{n}{k}$ k-delmängderna av $\{1, ..., n\}$ utgör en (n, k, t)-design för alla $t \leq k$. I allmänhet finns det dock övertäckningsdesigner med betydligt färre block än $\binom{n}{k}$.

Exempel 1.1. Figur 1.1 nedan visar en minimal $(5, 3, 2)$-design med $C(5, 3, 2) = 4$ block. Genom inspektion kan man konstatera att alla 2-delmängder av $\{1, ..., n\}$ ingår i något av blocken. Att det inte kan finnas någon $(5, 3, 2)$-design med färre block kan vi i detta fall se genom att varje block innehåller $\binom{3}{2} = 3$ stycken 2-delmängder, medan det finns $\binom{5}{2} = 10$ stycken 2-delmängder som ska övertäckas. Alltså kan tre block omöjligt räcka till en $(5, 3, 2)$-design.

1	2	3
1	4	5
2	3	4
2	3	5

Figur 1.1: Minimal $(5, 3, 2)$-design.

En övertäckningsdesign behöver inte vara minimal även om samtliga block är nödvändiga för att täcka alla t-delmängderna. Detta kan vi se av figur 1.2 som är en $(5, 3, 2)$-design med fem block. Varje block i denna design är ensam om att innehålla en unik 2-delmängd ($\{1, 2\}$, $\{1, 3\}$, $\{4, 5\}$, $\{2, 4\}$ och $\{3, 5\}$ för de respektive blocken). Men som vi sett i figur 1.1 så existerar det $(5, 3, 2)$-designer som är mindre än den i figur 1.2.

1	2	5
1	3	4
1	4	5
2	3	4
2	3	5

Figur 1.2: Ickeminimal $(5, 3, 2)$-design.

En praktisk motivering för att studera övertäckningsdesigner är deras användning i utformning av experiment. Om varje t-delmängd av n produkter ska testas för, säg, inbördes kompatibilitet, och det är möjligt att testa produkterna i grupper om k, så räcker det med $C(n, k, t)$, snarare än $\binom{n}{k}$ tester. Liknande om n stycken produkter ska jämföras under b olika förhållanden i grupper om k på sådant sätt att varje t-delmängd av produkterna jämförs under samma förhållanden minst en gång, så är utgör blocken i en (n, k, t)-design med b block de olika produktgrupperna som ska jämföras.

I verklighetsbaserade sammanhang kan det finnas skäl att låta blocken vara multimängder (att element kan förekomma flera gånger inom ett block) eller att inte alla block har samma storlek. Övertäckningsdesigner generaliseras även som λ-(n, k, t)-designer, där varje t-mängd ska ingå i minst λ block. Här intresserar vi oss endast för designer där $\lambda = 1$ och varje block består av k unika element, och i synnerhet är det minimala designer som vi vill hitta. Se [BC09] för behandling av mer generella designer och diskussion kring deras statistiska optimalitet.

1.1 Konstruktion av övertäckningsdesigner

Det finns många metoder för att konstruera övertäckningsdesigner och även många övre och undre begränsningar för det minimala blockantalet $C(n, k, t)$. Följande konstruktioner är några av dem som Gordon m.fl beskriver i [GKP95].

Om man ökar k med 1 genom att lägga till ett element i varje block i en minimal (n, k, t)-design så uppstår en $(n, k+1, t)$-design där inte nödvändigtvis alla block behövs för att täcka alla t-mängder, och man får olikheten

$$C(n, k+1, t) \leq C(n, k, t).$$

Exempel 1.2. Vi kan till exempel lägga till det lägsta talet som inte redan ingår i ett givet block till varje block i $(5, 3, 2)$-designen i figur 1.1 för att få designen i figur 1.3. Det första och tredje blocket i denna $(5, 4, 2)$-design innehåller precis samma element, så ett av dem kan plockas bort för att skapa en mindre $(5, 4, 2)$-design. Vi får följaktligen att $C(5, 4, 2) < C(5, 3, 2) = 4$.

1	2	3	4
1	4	5	2
2	3	4	1
2	3	5	1

Figur 1.3: $(5, 4, 2)$-design utvidgad från minimal $(5, 3, 2)$-design.

Plockas ett element, v, bort från en (n, k, t)-design, och v byts ut mot ett godtyckligt element i de block som det tillhörde, så får man en $(n-1, k, t)$-design med lika många block, alltså gäller

$$C(n-1, k, t) \leq C(n, k, t).$$

Exempel 1.3. Från designen i figur 1.1 tar vi bort 5:an och byter ut den mot det lägsta elementet som inte redan finns i ett givet block. Vi får då en $(4, 3, 2)$-design där det första och sista blocket innehåller samma element. Alltså är $C(4, 3, 2) < C(5, 3, 2) = 4$.

1	2	3
1	4	2
2	3	4
2	3	1

Figur 1.4: $(4, 3, 2)$-design utvidgad från minimal $(5, 3, 2)$-design.

Genom att lägga till ett nytt element och låta det ingå i varje block så omvandlar man en (n, k, t)-design till en $(n+1, k+1, t)$-design av samma storlek, man får därmed att

$$C(n+1, k+1, t) \leq C(n, k, t)$$

Exempel 1.4. Följande $(6, 4, 2)$-design får vi genom att lägga till en 6:a till varje block i den minimala $(5, 3, 2)$-designen i figur 1.1. Från denna design kan vi inte omedelbart avlägsna ett block, men om 6:an i det tredje blocket byts ut mot en 5:a så blir det fjärde blocket överflödigt. Därmed konstaterar vi att $C(6, 4, 2) < C(5, 3, 2) = 4$.

1	2	3	6
1	4	5	6
2	3	4	6
2	3	5	6

Figur 1.5: $(6, 4, 2)$-design utvidgad från minimal $(5, 3, 2)$-design.

Denna metod kan även generaliseras för att skapa en (n_1, k_1, t)-design från en (n_2, k_2, t)-design, D, där $n_1 < n_2$ och $k_1 < k_2$. Detta genom att plocka bort alla element som är större än n_1 från D:s block och behålla de block som har fler än t element kvar. Om ett block har färre än k_1 element så lägger vi till godtyckliga element tills det har rätt storlek. Och block, $B = \{e_1, e_2, ..., e_x\}$, som har storlek $x > k_1$ byts ut en (x, k_1, t)-design vars x-mängd är B, snarare än $\{1, 2, ..., x\}$.

Exempel 1.5. I figur 1.6a har vi en $(7, 4, 2)$-design, som vi vill omkonstruera till en $(5, 3, 2)$-design. Så vi tar bort 6:or och 7:or från blocken. De två första blocken är då av rätt storlek. Det sista blocket innehåller dock fyra element, vilket är ett för mycket, så det byter vi ut mot en $(4, 3, 2)$-design som täcker $\{2, 3, 4, 5\}$ (se figur 1.6b). Detta ger upphov till $(5, 3, 2)$-designen i figur 1.6c med 5 block (där det sista blocket kan tas bort för att skapa en minimal design).

1	2	5	6	7
1	3	4	6	7
2	3	4	5	6

(a) $(7, 4, 2)$

2	3	5
2	4	5
3	4	5

(b) $(4, 3, 2)$

1	2	5
1	3	4
2	3	5
2	4	5
3	4	5

(c) $(5, 3, 2)$

Figur 1.6: $(5, 3, 2)$-design utvidgad från $(7, 4, 2)$- och $(4, 3, 2)$-designer.

En (n, k, t)-design kan också skapas av två mindre designer (n_1, k_1, t_1) och (n_2, k_2, t_2), där $n = n_1 + n_2$, $k = k_1 + k_2$ och $t = t_1 + t_2$. Låt D_1 och D_2 vara dessa två mindre övertäckningsdesigner, och låt dem täcka $\{1, ..., n_1\}$ respektive $\{n_1 + 1, ..., n_1 + n_2\}$. För varje t-mängd, α_t, sådan att $\alpha_t = \alpha_{t_1} \cup \alpha_{t_2}$, där $\alpha_{t_1} \subset \{1, ..., n_1\}$ och $\alpha_{t_2} \subset \{n_1 + 1, ..., n_1 + n_2\}$, så finns det block B_1 i D_1 och

B_2 i D_2 som täcker α_{t_1} respektive α_{t_2}. Och vi låter $B_1 \cup B_2$ vara ett block i (n, k, t)-designen som ska konstrueras. Därmed har vi att

$$C(n, k, t) \leq C(n_1, k_1, t_1) \cdot C(n_2, k_2, t_2).$$

För att kunna konstruera övertäckningsdesigner enligt ovanstående metoder så krävs det att man redan känner till befintliga designer. Följande metoder kan i vissa fall användas för att konstruera designer från grunden. Vi börjar med att betrakta en förhållandevis enkel metod, men som inte nödvändigtvis går att använda för alla n, k, t.

Givet ett godtyckligt block, $B_1 = \{e_1, e_2, ..., e_k\}$, så skapar vi nästa block, B_2, genom att addera 1 till varje element i B_1 och ta resultatet modulo n. Detta upprepar vi sedan tills vi har n block. Om blocken inte täcker alla t-delmängder så börjar vi om med ett annat startblock.

Exempel 1.6. Hur pass bra övertäckningsdesigner som skapats på detta sätt varierar väldigt mycket. Figur 1.7 visar en cyklisk $(8, 5, 3)$-design som är minimal[GKP95], medan $C(8, 5, 2) = 4$ vilket innebär att vi med denna metod får dubbelt så många block som är nödvändigt, och $C(8, 5, 4) = 20$ alltså är det omöjligt att konstruera en $(8, 5, 4)$-design med 8 block.

1	2	3	4	6
2	3	4	5	7
3	4	5	6	8
4	5	6	7	1
5	6	7	8	2
6	7	8	1	3
7	8	1	2	4
8	1	2	3	5

Figur 1.7: Minimal $(8, 5, 3)$-design.

Följande metod, kallad "Hill-climbing", är inte heller garanterad att ge en övertäckningsdesign, men Gordon m.fl. hittade bland annat en minimal $(14, 8, 3)$-design genom metoden.[GKP95] För något givet b, välj slumpmässigt b stycken k-delmängder av $\{1, ..., n\}$ som startblock. Så länge de b blocken inte täcker alla t-delmängder så byter vi ut det block som täcker färst t-delmängder, som inte täcks av något annat block, mot en annan slumpmässig k-delmängd. Om det finns någon (n, k, t)-design med b block så kommer den så småningom att hittas.

Exempel 1.7. För att konstruera en minimal $(5, 3, 2)$-design genom hill-climbing så väljer vi 4 stycken slumpmässiga block av de 10 möjliga (se figur 1.10 för en lista av möjliga block). Figur 1.8a har vi våra 4 initialblock, vilka varken täcker $\{1, 3\}$ eller $\{2, 3\}$, och det tredje blocket är inte ensam om att täcka någon 2-mängd ($\{1, 4\}$, $\{1, 5\}$ och $\{4, 5\}$ täcks av det första, andra och fjärde blocken, respektive). Så det tredje blocket byts ut mot ett annat slumpmässigt valt block, se figur 1.8b, vartefter det andra och fjärde blocket bara täcker en unik 2-mängd vardera ($\{4, 5\}$ respektive $\{1, 3\}$), samtidigt som de två andra blocken täcker två unika 2-mängder vardera. Vi byter då ut block nummer två och får i övertäckningsdesignen i figur 1.8c, då varje 2-delmängd av $\{1, 2, 3, 4, 5\}$ täcks av något block.

2	3	5
1	4	5
1	2	4
2	4	5

(a) steg 1

2	3	5
1	4	5
1	2	4
1	3	5

(b) steg 2

2	3	5
3	4	5
1	2	4
1	3	5

(c) steg 3

Figur 1.8: Minimal $(5, 3, 2)$-design konstruerad med hill-climbing.

En metod som däremot alltid ger en övertäckningsdesign är följande giriga konstruktion. Från den maximala (n, k, t)-designen med $\binom{n}{k}$ block väljer vi successivt ut det block som täcker flest t-delmängder som vi inte redan har täckts, tills alla t-delmängder är täckta.

Exempel 1.8. Figur 1.9 visar hur vi kan konstruera en minimal $(5, 3, 2)$-design med hjälp av den giriga algoritmen. Den vänstra kolumnen i tabellerna visar hur många icke-täckta 2-mängder som blocket innehåller. Till steg 1 spelar det förstås ingen roll vilket block som väljs, då inga 2-mängder har täckts innan man valt något block. Därefter väljer vi det första blocket som täcker flest icke-täckta 2-mängder, tills vi i steg 4 får vår övertäckningsdesign.

	1	2	3
2	1	2	4
2	1	2	5
2	1	3	4
2	1	3	5
3	1	4	5
2	2	3	4
2	2	3	5
3	2	4	5
3	3	4	5

(a) steg 1

	1	2	3
	1	4	5
1	1	2	4
1	1	2	5
1	1	3	4
1	1	3	5
2	2	3	4
2	2	3	5
2	2	4	5
2	3	4	5

(b) steg 2

	1	2	3
	1	4	5
	2	3	4
0	1	2	4
1	1	2	5
0	1	3	4
1	1	3	5
2	2	3	5
1	2	4	5
1	3	4	5

(c) steg 3

	1	2	3
	1	4	5
	2	3	4
	2	3	5
0	1	2	4
0	1	2	5
0	1	3	4
0	1	3	5
0	2	4	5
0	3	4	5

(d) steg 4

Figur 1.9: Konstruktion av minimal $(5, 3, 2)$-design via den giriga algoritmen.

Den senaste metoden är den som gav Gordon m.fl. flest minsta (men inte nödvändigtvis minimala) (n, k, t)-designer, för $t \leq 8$, $k \leq 16$ och $n \leq 16$, i [GKP95]. Där diskuterar de även ovanstående och andra metoder mer ingående.

1.2 Hypergrafer

En hypergraf är en graf där kanter består av ett godtyckligt antal hörn. Vi identifierar kanterna med mängden av hörnen som den innehåller, och om alla kanter i en graf har samma storlek, säg r, så kallar vi den r-likformig eller r-graf. Vanliga grafer är därmed 2-likformiga hypergrafer. Samtliga hypergrafer som behandlas här kommer att vara likformiga.

Definition 1.1. En r-graf, G, består av de två mängderna $V(G)$ och $E(G)$, där $|e| = r$ och $e \subseteq V(G)$ för alla $e \in E(G)$.

Exempel 1.9. Figur 1.10 visar K_5^3 med hörnmängd $\{1, 2, 3, 4, 5\}$ och $\binom{5}{3} = 10$ kanter i lexikografisk ordning. De grafer vi kommer att behandla är alla enkla, så alla 3-grafer på 5 hörn som vi intresserar oss för kommer att bestå av en delmängd, snarare än multimängd, av kanterna i K_5^3.

1	2	3
1	2	4
1	2	5
1	3	4
1	3	5
1	4	5
2	3	4
2	3	5
2	4	5
3	4	5

Figur 1.10: Komplett 3-graf på 5 hörn.

Betraktar vi blocken i en (n, k, t)-design som kanter så kan alltså designen ses som en k-likformig hypergraf.

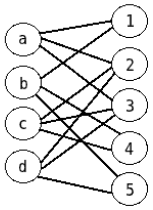
Varje hypergraf kan vi vidare identifiera med en bipartit 2-graf, kallad Levi-graf.

Definition 1.2. Levi-grafen, G', är den unika bipartita 2-grafen som kan konstrueras från r-grafen G genom att låta de två hörnpartitionerna i G' bestå av hörnen respektive kanterna i G. Så varje hörn i G är också ett hörn i G', och varje kant i G har ett korresponderande hörn i G'. För varje kant $e \in G$, så är det motsvarande hörnet $v_e \in G'$ granne med precis de hörn som ingår i e.

Exempel 1.10. Om vi betecknar kanterna i den minimala $(5, 3, 2)$-designen med a,b,c och d (se figur 1.11a) så har vi en grafisk representation av den motsvarande Levi-grafen i figur 1.11b.

a	1	2	3
b	1	4	5
c	2	3	4
d	2	3	5

(a)



(b)

Figur 1.11: Levi-graf för minimal $(5, 3, 2)$-design.

Om F är en given r-graf så säger vi att r-grafen G är F-fri om F inte är en inducerad delgraf av G. En F-fri graf $G = G^r(n, m)$ kallar vi extremal om det inte finns någon F-fri graf på n hörn med fler än m kanter. Om $F = K_s^r$, den kompletta r-grafen på s hörn, och G är en extremal F-fri graf, så kallar vi G även för Turángraf. Antalet kanter i en Turángraf är Turántalet $\mathrm{ex}(n, K_s^r)$, och mängden av Turángraferna betecknar vi $\mathrm{EX}(n, K_s^r)$.

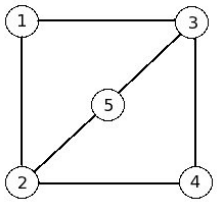
Definition 1.3. Turángraferna / Extremalgraferna $\mathrm{EX}(n, K_s^r)$ är mängden av de K_s^r-fria graferna på n hörn med flest kanter.

Definition 1.4. Turántalet $\mathrm{ex}(n, K_s^r)$ är antalet kanter i de n-hörniga graferna som är extremala med avseende på K_s^r.

I extremalgrafslitteraturen så intresserar man sig också för den asymptotiska andelen av de möjliga kanterna som ingår i extremala K_s^r-fria grafer, $\lim_{n\to\infty} \mathrm{ex}(n, K_s^r)\binom{n}{r}^{-1}$.

Definition 1.5. Turándensiteten $\pi(K_s^r) = \lim_{n\to\infty} \mathrm{ex}(n, K_s^r)\binom{n}{r}^{-1}$.

Exempel 1.11. Följande är den unika K_3^2-fria extremalgrafen på 5 hörn, både i konventionell representation av 2-grafer (figur 1.12a) och som kantlista (1.12b).



(a)

1	2
1	3
2	4
2	5
3	4
3	5

(b)

Figur 1.12: Extremal K_3^2-fri 2-graf på 5 hörn.

Problemet att hitta en minimal övertäckningsdesign är ekvivalent med att hitta en extremal K_s^r-fri hypergraf.

Sats 1.1. *Låt f_1 vara en funktion som omvandlar en r-likformig hypergraf på n hörn och m kanter till en $(n-r)$-likformig hypergraf på lika många hörn och kanter, på sådant sätt att $\tilde{e} \in E(f_1(G))$ om och endast om det finns någon kant $e = \{v_1, v_2, ..., v_r\} \in E(G)$ sådan att $\tilde{e} = V(G)\backslash e$. Och låt f_2 vara komplementsfunktionen, $E(f_2(G)) = E(K_n^r)\backslash E(G)$.*

Om G är en extremal r-graf på n hörn med $\mathrm{ex}(n, K_s^r)$ kanter så är $f_1(f_2(G))$ en minimal $(n, n-r, n-s)$-övertäckningsdesign, och

$$\mathrm{ex}(n, K_s^r) = \binom{n}{r} - C(n, n-r, n-s). \tag{1.1}$$

Bevis. Det är enkelt att se att om G har maximalt antal kanter utan att innehålla K_s^r så är $\bar{G} = f_2(G)$ en r-graf på n hörn som har minimalt, $\binom{n}{r} - \mathrm{ex}(n, K_s^r)$, antal kanter utan att $\bar{K}_s^r = f_2(K_s^r)$ kan induceras från $\bar{G}$. För varje s-mängd, $\kappa_s \subset V(\bar{G})$, så finns det alltså någon kant $\bar{e} \in E(\bar{G})$ sådan att $\bar{e} \subset \kappa_s$.

Tar vi komplementet med avseende på hörnen för varje sådan s-mängd så får vi en $(n-s)$-mängd, $\kappa_{n-s} = V(\bar{G})\backslash\kappa_s$, som då blir en delmängd av hörnkomplementet av den kant $\bar{e} \in \bar{G}$ som var delmängd av κ_s. Eftersom alla κ_s har en tillhörande kant i $\bar{G}$ så har alla $(n-s)$-mängder κ_{n-s} en tillhörande kant $\tilde{\bar{e}} \in \tilde{\bar{G}} = f_1(\bar{G})$.

Så $\tilde{\bar{G}}$ är en $(n-r)$-graf, på n hörn och $\binom{n}{r} - \mathrm{ex}(n, K_s^r)$ kanter, sådan att för varje $(n-s)$-delmängd av $V(\tilde{\bar{G}})$ så finns det en kant som innehåller delmängden.

Alltså är $\tilde{\bar{G}}$ en $(n, n-r, n-s)$-design. Att $\tilde{\bar{G}}$ är minimal följer direkt av att $\bar{G}$ är minimal, och därmed har vi att $\mathrm{ex}(n, K_s^r) = \binom{n}{r} - C(n, n-r, n-s)$.

Q.E.D.

Exempel 1.12. Låt G vara den K_3^2-fria extremalgrafen på 5 hörn i figur 1.12b. Figur 1.13a visar komplementet, $\bar{G}$, av G, och figur 1.13b visar den 3-likformiga grafen, $\tilde{\bar{G}}$, som vi får genom att komplementera de enskilda kanterna i $\bar{G}$. Av sats 1.1 har vi alltså att $\tilde{\bar{G}}$ är en minimal $(5, 3, 2)$-design. (Notera att $\tilde{\bar{G}}$ är designen i figur 1.1, med omvänd ordning på kanterna.)

1	4
1	5
2	3
4	5

(a) $\bar{G}$

2	3	5
2	3	4
1	4	5
1	2	3

(b) $\tilde{\bar{G}}$

Figur 1.13: Konvertering från extremalgraf till minimal övertäckningsdesign.

Det enklaste icke-triviala extremalgrafsproblemet är hitta extremala K_3^2-fria (hyper)grafer, det vill säga, att konstruera de största triangelfria graferna på ett givet antal hörn, n. Då $n \leq 3$ så ser man enkelt att $\mathrm{ex}(n, K_3^2) = n-1$. För $n > 3$ så vill vi att alla cykler ska ha en längd av minst 4, vilket uppfylls av bipartita grafer, och i synnerhet av den kompletta bipartita grafen $K_{a,b}^2$, där $a+b = n$. Vi kan också konstatera att $K_{a,b}^2$ har desto fler kanter ju mindre $|a-b|$ är. Alltså, hörnmängderna ska vara så lika som möjligt. Om $a \geq b+2$ så kan vi flytta ett hörn, x, från a-mängden till b-mängden, eftersom x hade valens b så förlorar vi b kanter, men får i stället $a-1$ nya kanter. Då $a-1-b \geq b+2-1-b = 1$ så har den nya grafen fler kanter än den gamla.

År 1907 så visade Mantel [Man07] att extremalgrafen för K_3^2 är just den kompletta bipartita grafen där partitionernas storlek är så lika som möjligt.

Sats 1.2. ***Mantels sats.***

$$\mathrm{ex}(n, K_3^2) = \left\lfloor \frac{n^2}{4} \right\rfloor$$
$$\mathrm{EX}(n, K_3^2) = \left\{ K^2_{\lfloor n/2 \rfloor, \lceil n/2 \rceil} \right\}$$

Se [Bol98] eller [Wes01] för fullständigt resonemang.

Drygt tre decennier senare så utvidgade Turán [Tur41] Mantels sats genom att bestämma extremalgraferna för kompletta 2-grafer på godtyckligt många hörn.

I en s-partit graf, där hörnen är uppdelade i s partitioner på sådant sätt att hörnen endast har grannar i andra partitioner än sin egen, så kan vi inte ha någon K_{s+1} som delgraf. Den största s-partita grafen på n hörn är uppenbart komplett. Och på likande sätt som för de triangelfria bipartita graferna så ska hörnen vara så jämnt fördelade som möjligt för att maximera kantantalet. Skulle storleken på två partitioner skilja sig med mer än 1 så kan vi skapa en större graf genom att flytta hörn tills alla partitioner har antingen $\lfloor \frac{n}{s} \rfloor$ eller $\lceil \frac{n}{s} \rceil$ hörn.

Vi låter $T_{s,n}$ och $t_{s,n}$ beteckna den största kompletta s-partita grafen respektive antalet kanter i den, och noterar att

$$\frac{n}{2}\left(n - \left\lfloor \frac{n}{s} \right\rfloor\right) \leq t_{s,n} \leq \frac{n}{2}\left(n - \left\lceil \frac{n}{s} \right\rceil\right) \tag{1.2}$$

där likhet gäller när alla hörn har valens $n - \frac{n}{s}$.

Sats 1.3. ***Turáns sats.***

$$\mathrm{ex}(n, K_{s+1}) = t_{s,n}$$
$$\mathrm{EX}(n, K_{s+1}) = \{T_{s,n}\}$$

Bevis. Induktion på antalet hörn.
Låt $s \geq 2$ vara givet. Satsen håller trivialt för $n \leq s+1$, anta att den även gäller för grafer på upp till $n-1$ hörn. Eftersom att $T_{s,n}$ är kantmaximal utan att ha K_{s+1} som delgraf, så är det som behöver visas att om G är en K_{s+1}-fri graf på n hörn och $t_{s,n}$ kanter så måste $G = T_{s,n}$.

Då valensen på hörnen i $T_{s,n}$ skiljer sig så lite som möjligt så har vi att

$$\delta(G) \leq \delta(T_{s,n}).$$

Om x är ett hörn i G med minimal valens, $\deg(x) = \delta(G)$, så ger induktionantagandet att $T_{s,n-1}$ är den inducerade delgrafen $G \backslash x$. Då G inte innehåller K_{s+1} så måste det finnas någon hörnpartition i $T_{s,n-1}$ där x inte har några grannar, och därmed måste G vara s-partit. Eftersom G är s-partit och enligt definition har $t_{s,n}$ kanter så är G den kantmaximala s-partita grafen, alltså $G = T_{s,n}$.
Q.E.D.

Se [Bol98, Die00, Wes01] för fler bevis av Turáns sats, ovanstående är väsentligen satsens tredje bevis i [Bol98].

Av sats 1.3 så känner vi alltså till samtliga minimala $(n, n-2, t)$-designer, och vi kan med hjälp av (1.2) observera följande om Turándensiteten.

Sats 1.4.

$$\pi(K^2_{s+1}) = \frac{s-1}{s}$$

Andra typer av extremalgrafsproblem behandlas också i [Bol98] och [Die00], men då vi i huvudsak intresserar oss för extremalgrafer för kompletta grafer så nämner vi bara kort två av de närliggande resultaten här.

Sats 1.5. ***Erdős och Stones fundamentalsats.*** *Givet $v \geq 2$, $s \geq 1$ och $\epsilon > 0$ så gäller för tillräckligt stora n och $m \geq t_{s,n} + \epsilon n^2$ att alla grafer $G = G(n, m)$ har en komplett $(v+1)$-partit graf med s hörn i varje partition som delgraf.*

Korollarium 1.1.

$$\pi(F) = \frac{\chi(F) - 2}{\chi(F) - 1}$$

Där $\chi(F)$ är F:s kromatiska tal.

Dessvärre är väldigt lite känt om extremala hypergrafer för ens K_s^r, där $s > r > 2$. Till skillnad från fallet med 2-grafer så känner vi bara till $\mathrm{ex}(n, K_s^r)$ för vissa givna r och s, och för $\pi(K_s^r)$ finns det bara förmodade värden och begränsningar. Sidorenko [Sid95] och Keevash [Kee11] sammanfattar kända resultat, här tar vi bara upp en bråkdel av dessa.

En av de bättre generella begränsningarna formulerades först av Schönheim [Sch64] som en undre begränsning för $C(n, k, t)$,

$$\begin{aligned} C(n,k,t) &\geq \left\lceil \frac{n}{k} C(n-1,k-1,t-1) \right\rceil \\ &= \left\lceil \frac{n}{k} \left\lceil \frac{n-1}{k-1} \cdots \left\lceil \frac{n-t+1}{k-t+1} \right\rceil \cdots \right\rceil \right\rceil. \end{aligned} \tag{1.3}$$

I extremalgrafsterminologi så resonerar Frohmader [Fro08] att givet en extremalgraf, $G = G^r(n, \mathrm{ex}(n, K_s^r))$, kan vi plocka bort ett godtyckligt hörn och samtliga kanter som innehåller hörnet. Den resulterande grafen har då $n-1$ hörn och $m \leq \mathrm{ex}(n-1, K_s^r)$ kanter, tar vi det genomsnittliga kantantalet av alla sätt att ta bort ett hörn så får vi att

$$\frac{\mathrm{ex}(n, K_s^r)}{\binom{n}{r}} \leq \frac{\mathrm{ex}(n-1, K_s^r)}{\binom{n-1}{r}},$$

vilket ger motsvarigheten till (1.3) för extremalgrafer

$$\mathrm{ex}(n, K_s^r) \leq \left\lfloor \frac{n}{n-r} \mathrm{ex}(n-1, K_s^r) \right\rfloor. \tag{1.4}$$

Den bästa generella begränsningen som inte är beroende av andra Turántal är de Caens [dC83] olikhet

$$\begin{aligned} \mathrm{ex}(n, K_s^r) &\leq \binom{n}{r} \left(1 - \frac{n-s+1}{n-r+1} \Big/ \binom{s-1}{r-1}\right) \\ C(n,k,t) &\geq \binom{n}{k} \frac{(t+1)(n-t)}{(k+1)(n-k)} \Big/ \binom{k}{t}. \end{aligned}$$

Till skillnad från när $r = 2$ så är de extremala graferna för K_s^r inte heller nödvändigtvis unika när $r \geq 3$. Turán förmodade [Tur61] att

$$\mathrm{ex}(n, K_4^3) = \binom{n}{3} - \begin{cases} x(x-1)(2x-1) & \text{om } n = 3x \\ x^2(2x-1) & \text{om } n = 3x+1 \\ x^2(2x+1) & \text{om } n = 3x+2 \end{cases}$$

vilket har visat sig gälla för $n \leq 13$ [Sid95, ref 19, 37, 47]. Om Turáns förmodan stämmer så har Kostochka och Frohmader visat att

$$|\mathrm{EX}(n, K_4^3)| \geq \begin{cases} 2^{x-2} & \text{om } n = 3x & \text{[Kos82]} \\ \frac{1}{2}6^{x-1} & \text{om } n = 3x+1 & \text{[Fro08]} \\ 6^{x-1} & \text{om } n = 3x+2 & \text{[Fro08]} \end{cases}$$

och Markström [Mar09] har även visat att $|\mathrm{EX}(n, K_5^4)| = 3$, för $n \in \{9, 11, 12\}$.

Vad gäller Turándensiteten så är de Caens övre och Sidorenkos [Sid81] undre begränsning de bästa man känner till för det allmänna fallet

$$1 - \left(\frac{r-1}{s-1}\right)^{r-1} \leq \pi(K_s^r) \leq 1 - \binom{s-1}{r-1}^{-1}$$

För särskilda värden finns dock bättre begränsningar, t.ex. visar de Caen m.fl. [dCKW88] för jämna r att

$$\pi(K_{r+1}^r) \geq \frac{3}{4} - \frac{1}{2^r}.$$

För givna värden på r och s har vi också bättre gränser, så som

$$\pi(K_4^3) \leq \frac{\sqrt{21}-1}{6}$$
$$\pi(K_5^4) \leq \frac{1753}{2380}$$

av Giraud [Gir] respektive Markström [Mar09].

Åter igen, för r-likformiga hypergrafer så är den extremala grafteorin väldigt ofullständig för $r \geq 3$. 1981 erbjöd Erdős [Erd81] en av sina många belöningar ($500 US) för lösningen av $\pi(n, K_s^r)$ för någon given K_s^r, $s > r > 2$. Men 30 år efter Erdős erbjudande, och 50 år efter att Turán [Tur61] formulerat problemet så har vi fortfarande ingen lösning.

2 Problem- och metodbeskrivning

Vårt mål är att finna samtliga icke-isomorfa minimala övertäckningsdesigner för givna n, k, t, eller ekvivalent extremala K_{n-t}^{n-k}-fria $(n-k)$-grafer på n hörn. För att uppnå detta så har vi skrivit en uppsättning datorprogram. Med dessa program utför vi dels en uttömmande kombinatorisk sökning - den naiva metoden. Men vi använder också en lätt modifierad variant av den metod, baserad på linjärprogrammering, som Markström beskriver i [Mar09].

För lösning av linjärprogrammeringsproblem så använder vi Gurobi[GO]. Detta för att undvika vissa stabilitetsproblem som Markström haft med GNUs glpk, och för att göra det möjligt att modifiera linjärprogrammet under körning (se avsnitt 2.2.2 punkt 4).

För att undersöka huruvida de grafer som hittas är isomorfa så använder vi McKays programpaket Nauty[BDM81].

Nedan följer en redogörelse för den metod, baserad på den som Markström beskriver i [Mar09], som vi använder för att konstruera samtliga extremala hypergrafer, $\mathrm{EX}(n, K_s^r)$, för givna n,r, och s.

2.1 Konstruktion av extremala hypergrafer

Enligt samma resonemang som gav oss Schönheims olikhet (1.4) så kan vi konstatera följande.

Lemma 2.1. *"Markströms lemma"*
Om G är en K_s^r-fri r-graf på n hörn och m kanter så finns det en K_s^r-fri r-graf, G' på $n-1$ hörn och minst $m - \left\lfloor \frac{mr}{n} \right\rfloor$ kanter, sådan att $G' = G \backslash v$, för något hörn $v \in G$.

Bevis. Låt G vara en K_s^r-fri graf på n hörn och m kanter. Om v är ett hörn i G med lägst valens, $\deg(v) = \delta(G)$, så är den inducerade grafen $G' = G \backslash v$ uppenbart också K_s^r-fri. Och då $\delta(G)$ inte kan vara högre medelvalensen för G, $\frac{mr}{n}$, så får vi att

$$|E(G')| \geq m - \left\lfloor \frac{mr}{n} \right\rfloor .$$

Q.E.D.

Fortsättningsvis låter vi $S(n, m)$ vara mängden av de icke-isomorfa K_s^r-fria graferna på n hörn och m kanter.

Korollarium 2.1. *Känner vi till $S(n-1, m)$, för alla $M - \lfloor \frac{Mr}{n} \rfloor \leq m \leq M$, så kan vi konstruera $S(n, M)$.*

För att hitta $\mathrm{EX}(n, K_s^r)$, för givna n, r och s, så utgår vi från den triviala extremalgrafen på s hörn, $K^- = K_s^r - e$, för någon godtycklig kant $e \in E(K_s^r)$. Sedan följer vi följande procedur som genererar $S(N, M)$, alla icke-isomorfa K_s^r-fria r-grafer på N hörn och M kanter. Proceduren utförs för alla $N = s+1, ..., n$ med tillhörande övre gräns för $\mathrm{ex}(N, K_s^r)$ från Schönheims olikhet (1.4), $M = \lfloor \frac{N}{N-r} \mathrm{ex}(N-1, K_s^r) \rfloor$. Om $S(S, M) = \emptyset$ så utför vi proceduren med avseende på $(N, M-1)$, $(N, M-2)$, och så vidare, tills en icketom mängd $S(N, m)$ har hittats, och då har vi att $m = \mathrm{ex}(N, K_s^r)$.

`1` För varje $\mu = M - \lfloor Mr/N \rfloor$, ..., M:
`1.1` Om inte $S(N-1, \mu)$ är känd:
`1.1.1` Utför proceduren med avseende på $(N-1, \mu)$.
`1.2` Utvidga alla $G \in S(N-1, \mu)$ till N hörn och M kanter.
`2` Utför isomorfireduktion på de grafer som hittats.

Steg `1.1.1` i proceduren innebär alltså att vi kontrollerar att alla K_s^r-fria grafer på $N-1$ hörn som behövs enligt korollarium 2.1 är kända. I annat fall så kommer de först att konstrueras, vilket i sin tur kan kräva att grafer på $N-2$ hörn och färre än $\mathrm{ex}(N-2, K_s^r)$ kanter behöver skapas, och så vidare. För ett givet n där vi är intresserade av $\mathrm{ex}(n, K_s^r)$ så kommer endast de extremala K_s^r-fria graferna på n hörn att konstrueras. Men för $\nu \leq n-1$ så är det mycket möjligt att proceduren bygger upp $S(\nu, m)$, där m är mycket mindre än $\mathrm{ex}(\nu, K_s^r)$.

Steg `1.2` utförs på två olika sätt, vilka beskrivs närmare i avsnitt 2.2.

I steg `2` så använder vi programpaketet Nauty[BDM81] för att göra oss av med icke-unika grafer. Dock omvandlar vi först graferna till deras motsvarande övertäckningsdesigner enligt sats 1.1. Detta eftersom att $C(n, n-r, n-s)$ i allmänhet är mycket mindre än $\mathrm{ex}(n, K_s^r)$, vilket medför att det är effektivare[1] att etablera isomorfirelationer för hypergraferna som motsvarar små designer än stora K_s^r-fria r-grafer. Övertäckningsdesignerna omformuleras sedan som motsvarande Levi-grafer, då Nauty endast kan hantera tvålikformiga hypergrafer.

[1] För K_{14}^6-fria extremalgrafer på 16 hörn och 8004 kanter så hade vi 720 grafer, varav 3 var unika. I det fallet var tidsåtgången 13 timmar för isomorfireducering via extremala hypergrafer respektive 18 sekunder för de motsvarande minimala övertäckningsdesignerna.

2.2 Grafutvidgning

När en given K_s^r-fri graf, G, på n hörn och m kanter ska utvidgas till $n+1$ hörn och M kanter enligt korollarium 2.1 så använder vi, som nämnts ovan, två olika strategier. Vi beskriver dessa nedan, och diskuterar kort deras effektivitet i avsnitt 4.1.

2.2.1 Kombinatorisk sökning

Om $G = G^r(n, m)$ är den K_s^r-fria graf som vi vill utvidga så lägger vi till ett hörn, v_{n+1}, och för varje sätt att välja $M - m$ kanter som innehåller v_{n+1} så skapar vi en graf G_i' med v_{n+1} och dessa kanter. Vi har då alltså en mängd $U = \{G_i'\}$ av r-grafer på $n+1$ hörn och M kanter. Vi vet dock inte om de är K_s^r-fria, och blir tvungna att undersöka om så är fallet för varje graf i U. Men då samtliga kanter i de nya graferna som saknas i ursprungsgrafen innehåller det nya hörnet så är det endast de $\binom{n}{s-1}$ potentiella K_s^r-delgraferna som har v_{n+1} i sin hörnmängd som måste uteslutas. Vartefter vi hittat de utvidgade grafer som eftersöktes.

Att detta ger $S(n+1, M)$, alla K_s^r-fria grafer på $n+1$ hörn och M kanter, har vi av att utvidgningen görs för alla

$$G \in \bigcup_{m=M-\lfloor \frac{Mr}{n+1} \rfloor}^{M} S(n, m).$$

2.2.2 Linjärprogrammering

Givet en r-graf $G = G^r(n, m)$ så identifierar vi den med en kant-tupel

$$\mathrm{EI}(G) = (x_e)_{e \in E(K_n^r)},$$

där $x_e \in \{0, 1\}$ är 1 om kanten e ingår i G, och $x_e = 0$ annars.

Om $G = G^r(n, m)$ är den K_s^r-fria graf som ska utvidgas till $n+1$ hörn och M kanter så formulerar vi linjärprogrammet för utvidgning av G som ett maximeringsproblem över variablerna x_e, för $e \in E(K_{n+1}^r)$, där följande villkor ska gälla.

Metoden är väsentligen densamma som Markströms i [Mar09], så när som på punkt 2 där vi här utnyttjar att grafen som ska utvidgas redan är K_s^r-fri.

1. Antalet kanter i de resulterande graferna ska vara M, vilket uttrycks av likheten
$$\sum_{e \in E(K_{n+1}^r)} x_e = M.$$

2. Inte av någon s-delmängd $\kappa_s \subset V(K_{n+1}^r)$ som innehåller det nya hörnet ska en K_s^r kunna induceras från de resulterande graferna. För varje $\kappa_s \ni v$ har vi alltså följande olikhet
$$\sum_{e \subset \kappa_s} x_e \leq \binom{s}{r} - 1.$$

3. Av de kanter som inte innehåller det nya hörnet så ska endast kanterna från G ingå i de resulterande graferna, så för varje $e \in E(K_n^r)$ har vi likheten
$$x_e = \begin{cases} 1 & \text{om } e \in E(G), \\ 0 & \text{om } e \notin E(G) \end{cases}.$$

4. När en lösning G' har hittats så lägger vi till en olikhet som uttrycker att åtminstone en av kanterna i G' inte ska ingå i framtida lösningar.
$$\sum_{e \in E(G')} x_e \leq M - 1.$$

Av punkt 3 har vi att linjärprogrammet har frihet i de $|E(K_{n+1}^r)| - |E(K_n^r)| = \binom{n}{r-1}$ variablerna som motsvarar kanter som innehåller det nya hörnet. Detta ger $2^{\binom{n}{r-1}}$ potentiella lösningar, men så som i den kombinatoriska sökningen så begränsar vi oss till grafer på M hörn i punkt 1. Så problemet blir att välja $M-m$ variabler x_e bland $e \in E(K_{n+1}^r) \backslash E(K_n^r)$ att sätta till 1, utan att överträda någon av de $\binom{n}{s-1}$ olikheterna som punkt 2 definierar.

Den stora fördelen vi här har över Markströms glpk-baserade program är att vi efter att lösning hittats och linjärprogrammet modifierats i steg 4 kan låta Gurobi fortsätta körningen med den nya olikheten. Detta innebär dels att linjärprogramslösaren kan behålla eventuell information om att en given lösning är ogiltig, d.v.s. att motsvarande graf innehåller den förbjudna K_s^r. Och slipper då utesluta grafen i fråga varje gång som körningen startas om efter att en giltig lösning har hittats. Dessutom behöver ingen explicit förändring av linjärprogrammet göras på hårddisk så länge programmet är igång, vilket också minskar onödig tidsåtgång.

När det slutligen blir omöjligt att hitta lösningar som inte överträder någon av olikheterna så har vi hittat alla utvidgningar G_i' av G som vi önskade finna.

3 Resultat

För de extremala K_s^r-fria hypergrafer/minimala designer som vi hittat så har designer av samma storlek varit kända sedan tidigare[Gor]. Att $(12, 7, 4)$-designen med 24 block var minimal var dock inte känt av varken [Gor] eller [Mar]. Vårt huvudsakliga bidrag är att samtliga extremalgrafer för givna värden har hittats med undantag för de K_{12}^{11}-fria graferna på 14 hörn där vi bara kunnat konstatera att $|\mathrm{EX}(14, K_{12}^{11})| \geq 323571$, och de K_{11}^8-fria graferna på 13 hörn där vi har att $|\mathrm{EX}(13, K_{11}^8)| \geq 30$.

I tabellerna nedan anger ex värdet för $\mathrm{ex}(n, K_s^r)$, S den övre gränsen för ex enligt Schönheims olikhet, C värdet för $C(n, n-r, n-k)$ och opt-x antalet icke-isomorfa K_s^r-fria grafer med $\mathrm{ex}(n, K_s^r) - x$ kanter.

n	ex	S	C	opt-0	opt-1	opt-2	opt-3
9	123	123	3	4			
10	246	246	6	2	592		
11	451	451	11	1	3	643	2830
12	768	773	24	356			

Figur 3.1: Värden för K_8^5-fria hypergrafer och $(n, n-5, n-8)$-designer.

n	ex	S	C	opt-0	opt-1
14	2000	2000	2	1	
15	3000	3000	3	1	10
16	4362	4363	6	1	

Figur 3.2: Värden för K_{13}^5-fria hypergrafer och $(n, n-5, n-13)$-designer.

n	ex	S	C	opt-0
15	3001	3001	2	1
16	4365	4365	3	1

Figur 3.3: Värden för K_{14}^5-fria hypergrafer och $(n, n-5, n-14)$-designer.

n	ex	S	C	opt-0
16	4366	4366	2	1
17	6185	6185	3	1
18	8563	8563	5	1

Figur 3.4: Värden för K_{15}^5-fria hypergrafer och $(n, n-5, n-15)$-designer.

n	ex	S	C	opt-0
17	6186	6186	2	1
18	8565	8565	3	1
19	11623	11623	5	4

Figur 3.5: Värden för K_{16}^5-fria hypergrafer och $(n, n-5, n-16)$-designer.

n	ex	S	C	opt-0
15	5003	5003	2	1
16	8004	8004	4	3

Figur 3.6: Värden för K_{14}^6-fria hypergrafer och $(n, n-6, n-14)$-designer.

n	ex	S	C	opt-0
17	12374	12374	2	1
18	18561	18561	3	1

Figur 3.7: Värden för K_{16}^6-fria hypergrafer och $(n, n-6, n-16)$-designer.

n	ex	S	C	opt-0
12	492	492	3	1
13	1277	1279	10	≥ 30

Figur 3.8: Värden för K_{11}^8-fria hypergrafer och $(n, n-8, n-11)$-designer.

n	ex	S	C	opt-0
13	71	71	7	1
14	331	331	33	≥ 323571

Figur 3.9: Värden för K_{12}^{11}-fria hypergrafer och $(n, n-11, n-12)$-designer.

4 Diskussion

4.1 Effektivitet av grafutvidgning

I figur 4.1 ser vi tidsåtgång[2] för de olika utvidgningsprogrammen.

K_s^r	n	LP	Komb.
K_3^2	≤ 25	1 s.	40 s.
K_5^2	≤ 23	2 s.	15 s.
K_5^4	≤ 8	1 s.	10 s.
K_5^4	9	70 s.	> 64 h.
K_{14}^4	≤ 17	10 s.	360 s.
K_{14}^4	18	140 s.	> 64 h.

Figur 4.1: Tidsåtgång för att hitta $\mathrm{EX}(n, K_s^r)$

Värt att notera är att den kombinatoriska sökningen inte hittat någon lösning över huvud taget för varken $\mathrm{EX}(9, K_5^4)$ eller $\mathrm{EX}(18, K_{14}^4)$ efter 64 timmars körning. I fallet med K_{14}^4 så har vi att $\mathrm{ex}(17, K_{14}^4) = 2376$ och $\mathrm{ex}(18, K_{14}^4) = 3054$, vilket är den övre gränsen som Schönheims olikhet (1.4) ger. Därmed är det endast extremalgraferna på 17 hörn (dessutom är $|\mathrm{EX}(17, K_{14}^4)| = 1$) som behöver utvidgas till 18 hörn, men ändå hittades ingen extremalgraf på 18 hörn.

För K_5^4 så blir problemet värre då $\mathrm{ex}(8, K_5^4) = 56$, vilket av olikhet (1.4) ger att $\mathrm{ex}(9, K_5^4) \leq 100$. Men $\mathrm{ex}(9, K_5^4) = 96$, och för att hitta $S(9, 96)$ för K_5^4 så behöver vi enligt korollarium 2.1 först känna till $S(8, m)$, för $m = 54, 55, 56$ (vilket i sin tur kräver alla K_5^4-fria grafer på 7 hörn med en kant färre än $\mathrm{ex}(7, K_5^4) = 28$ är kända).

[2] Körningar är gjorda på likvärdiga datorer i sal MA426, MIT-huset, Umeå universitet. `http://support.cs.umu.se/labs/ma426/`

Under körningstiden så har dock inte kombinationssökningen lyckats utesluta ens att $\mathrm{ex}(9, K_5^4)$ skulle vara 100. Då det ska läggas till $100 - \mathrm{ex}(8, K_5^4) = 44$ kanter till extremalgrafen på 8 hörn, och det finns $\binom{8}{3} = 56$ möjliga kanter som innehåller det 9:e hörnet. Vilket ger att vi har $\binom{56}{44} \approx 5 \cdot 10^{11}$ grafer som ska konstrueras och uteslutas innehålla K_5^4, och i detta fall har ungefär 25000 grafer per sekund kunnat genereras. Alltså skulle det ha behövts omkring 620 timmar för att konstatera att $\mathrm{ex}(9, K_5^4) \leq 99$. (Att försöka utvidga från extremalgrafen på 8 hörn till de faktiska extremalgraferna på 9 hörn skulle kräva att $\binom{56}{96-56} \approx 4 \cdot 10^{13}$ grafer skapas. Med liknande genereringsfrekvens skulle detta ta cirka 5 år.)

Referenser

[Bol98] Béla Bollobás, *Modern Graph Theory*, Springer, cop., New York, 1998.

[dC83] D. de Caen, *Extension of a theorem of Moon and Moser on complete subgraphs*, Ars Combinatoria **16** (1983), 5-10. Citerad i [GKP95] och [Sid95].

[dCKW88] D. de Caen, D.L. Kreher, and J. Wiseman, *On constructive upper bounds for the Turán numbers* $T(n, 2r+1, 2r)$, Nineteenth Southeastern Conference on Combinatorics, Graph Theory, and Computing (Baton Rouge, LA, 1988), Congr. Nr. 65 (1988), 277-280. Citerad i [Sid95].

[Die00] Reinhard Diestel, *Graph Theory*, 2nd ed., Springer, New York, 2000.

[Erd81] Pál Erdős, *On the combinatorial problems which I would most like to see solved*, Combinatorica **1** (1981), no. 1, 25-42.

[Fro08] Andrew Frohmader, *More Constructions for Turán's* $(3, 4)$*-Conjecture*, The Electronic Journal of Combinatorics **15** (2008), R137.

[Gir] G. Giraud, *Une minoration de la premiere fonction ternaire de Turán, manuscript.* Citerad i [Sid95].

[GNUa] GNU, *The GNU Compiler Collection.* `http://www.gnu.org/software/gcc/`.

[GNUb] ———, *GNU Make.* `http://www.gnu.org/software/make/`.

[GNUc] ———, *GNU Scientific Library.* `http://www.gnu.org/software/gsl/`.

[GKP95] Daniel M. Gordon, Greg Kupergerg, and Oren Patashnik, *New constructions for Covering Designs*, Journal of Combinatorial Designs **3, 4** (1995), 269-284.

[Gor] D. Gordon, *La Jolla Covering Repository.* `http://www.ccrwest.org/cover.html`.

[GO] Gurobi Optimization, *Gurobi 4.6.* `http://gurobi.com/`.

[Kee11] Peter Keevash, *Hypergraph Turán problems*, Surveys in Combinatorics (2011), 83-140.

[BC09] R.A Bailey and Peter J. Cameron, *Combinatorics of optimal designs*, Surveys in Combinatorics (2009), 19-73.

[Kos82] A. V. Kostochka, *A class of constructions for Turán's* $(3, 4)$*-problem*, Combinatorica **2** (1982), 187-192.

[Man07] W. Mantel, *Wiskundige Opgaven*, 1907.

[Mar09] Klas Markström, *Extremal hypergraphs and bounds for the Turán density of the 4-uniform* K_5, Discrete Mathematics **309, 16** (2009), 5231-5234.

[Mar] ———, *A web archive of Covering Designs.* `http://abel.math.umu.se/~klasm/Data/hypergraphs/coveringdesign.html`.

[BDM81] Brendan D. McKay, *Practical graph isomorphism*, Proceedings of the Tenth Manitoba Conference on Numerical Mathematics and Computing **1** (1981), 45-87. `http://cs.anu.edu.au/~bdm/nauty/`.

[Sch64] J. Schönheim, *On coverings*, Pacific Journal of Mathematics **14** (1964), 1405-1411.

[Sid81] Alexander Sidorenko, *Systems of sets that have the T-property*, Moscow University Mathematics Bulletin **36** (1981), no. 5, 22-26.

[Sid95] ———, *What we know and what we do not know about Turán numbers*, Graphs and Combinatorics **11** (1995), no. 2, 179-199.

[Tur41] Pál Turán, *On an extremal problem in graph theory*, Mat. és Fiz. Lapuk **48** (1941), 436-452. Citerad i [Bol98].

[Tur61] ———, *Research problems*, Magyar Tud. Akad. Mat. Kutato Int. Közl **6** (1961), no. 3, 417-423. Citerad i [Sid95] och [Fro08].

[Wes01] Douglas B. West, *Introduction to Graph Theory*, 2nd ed., Prentice Hall, cop., Upper Saddle River, N.J., 2001.

A Notationsförteckning

$G^r(n,m)$	r-graf på n hörn och m kanter.
$\bar{G}$	Komplement av G.
$\delta(G)$	Minsta valens för hörnen i G.
$\deg(v)$	v:s valens.
$V(G)$	Hörnmängd av G.
$E(G)$	Kantmängd av G.
$\mathrm{EX}(n,F)$	Mängden av maximala F-fria grafer på n hörn.
$\mathrm{ex}(n,F)$	Turántalet för F.
$S(n,m)$	Mängden K_s^r-fria r-grafer på n hörn och m kanter
$\pi(F)$	Turándensiteten för F, $\lim_{n\to\infty} \mathrm{ex}(n,F)\binom{n}{r}^{-1}$.
$T_{s,n}$	Den unika grafen i $\mathrm{EX}(n,K_s^2)$.
$t_{s,n}$	$\mathrm{ex}(n,K_s^2)$.
K_s^r	Kompletta r-grafen på s hörn.
$K_{a,b}^2$	Kompletta bipartita 2-grafen med partitioner om a respektive b hörn.

B Programbeskrivning

De program som har skrivits består av följande filer, vars källkod finns i Bilaga C, turan.sh, turan-brute.sh, expand_graphs-comb.sh, expand_graphs-lp.sh, expand_graphs-lp-solver.sh, split.py, brute.c, ei2cd.c, ei2graph.c, nCk.c, graph.c, graph.h, isoreduce.c, isoreduce_via_CD.c, lpgraph.c, lphead.c, lpsolve.c, seed.c, util.c, util.h och verifycover.py. Kompilering utförs lämpligtvis med GNU make[GNUb] och den medföljande filen Makefile. För kompilering rekommenderas GNU gcc[GNUa]. Koden kan också kompileras med hjälp av filen Makefile-nogcc, vilket medför att GNU-specifika delar av koden tas bort, och då ska det gå att kompilera med vilken C99-kompatibel kompilator som helst, men argumentet `-C` kommer då inte att fungera. Kompilering av programmen kräver att Gurobi 4.6[GO] och GNU Scientific Library[GNUc] finns tillgängligt. Under körning så förväntas programmet shortg från Nauty[BDM81] finnas tillgängligt i användarens `$PATH` och alla kompilerade program samt skript ska finnas i den katalog från vilken körning görs.

B.1 Gemensamma argument

För programmen som beskrivs i avsnitt B.8 och framåt så gäller att de accepterar en delmängd av argumenten i listan nedan. För dessa spelar det ingen roll i vilken ordning argumenten anges. För skript och program som beskrivs innan avsnitt B.8 så anger vi vad som gäller specifikt för det programmet.

- `-q` - Mindre information skrivs ut på skärmen.
- `-D katalog` - Spara filer i katalogen som angivits.
- `-C` - Om det redan finns en fil med angivet filnamn så skrivs den inte över, och inga beräkningar som annars varit nödvändiga kommer att utföras.
- `-a` - Om det redan finns en fil med angivet filnamn så läggs utdata till i slutet av filen, istället för att skriva över det befintliga innehållet.
- `-o filnamn` - Filnamn för fil dit utdata ska skrivas, "-" för standard output.
- `-r #` - Likformighet för grafer.
- `-k #` - Antal hörn i den kompletta graf som inte får ingå i grafer som söks.
- `-n #` - Antal hörn i grafer som läses in.
- `-m #` - Antal kanter i grafer som läses in.
- `-N #` - Antal hörn i grafer som söks.
- `-M #` - Antal kanter i grafer som söks.
- `-f filnamn` - Filnamn för indata, standard input antas om inget anges.
- `-T #` - Tidsgräns i minuter.
- `-t #` - Antal trådar som får användas.
- `-s #` - Minsta antal grafer som ska hittas innan tidsgräns börjar gälla.
- `-S #` - Övre gräns för antal grafer som får hittas.

De program som skriver ut data har standardfilnamn som användas om inget annat anges av argumentet `-o`. Dessa filer sparas i så fall i första hand i katalog som anges av argumentet `-D`, i andra hand miljövariabeln `$GRAPH_DIR` och i tredje hand den katalog som körning görs från. Argumenten `-D`, `-C` och `-a` kan anges för alla sådana program.

För program som läser in data från filer så behöver inte argumenten `-r`, `-k`, `-n`, `-m`, `-N` eller `-M` anges som filnamnet för fil med indata innehåller "-r=#", "-k=#" osv.

Argumentet `-q` kan anges för samtliga program, och de fyra sista argumenten i listan gäller endast för lpsolveB.11.

B.2 Filformat

För linjärprogram med filändelse ".lp" se [GO].

Filer med filändelse .ei innehåller grafer i klartext, endast grafer med samma likformighet, antal hörn och antal kanter ingår i en given fil. En graf $G = G^r(n, m)$ beskrivs per rad, där den lexikografiska sorteringen av kanterna i K_n^r har identifierats med mängden $\{1, 2, ...\binom{n}{r}\}$ och de av K_n^r:s kanter som G innehåller står uppräknade, separerade med mellanslag. Alltså, 3-grafen i figur 1.1 skulle i .ei-formatet beskrivas genom "`1 6 7 8`" då den innehåller de första, sjätte, sjunde och åttonde kanterna av den kompletta 3-grafen på 5 hörn i figur 1.10.

Filer med filändelse .txt beskriver också grafer i klartext, men med en kant per rad på samma sätt som de tabeller som använts ovan. Grafer separeras av textraden "--" i slutet av grafen.

B.3 turan.sh / turan-brute.sh

Dessa skript utför väsentligen samma sak, så när som på metoden de använder för grafutvidgning, där turan.sh använder linjärprogrammeringsmetoden och turan-brute.sh utför kombinatoriskt uttömmande sökning. Skripten tar tre argument: r, s och n_{max}, där ordning spelar roll, och r är likformighet på önskade grafer, s är antal hörn i den kompletta graf som ska undvikas och n_{max} är det maximala antalet hörn för vilken vi vill hitta $\mathrm{EX}(n, K_s^r)$. Vid körning av t.ex. "./turan.sh 4 5 10" så kommer $\mathrm{EX}(n, K_5^4)$ att hittas för $n = 4, 5, ..., 10$. Filer relaterade till lösandet av programmet lagras i den katalog som `$GRAPH_DIR` pekar på, alternativt `/tmp/EX`. De grafer som hittas kommer att heta "graphs-r=#-k=#-n=#-m=#.ei", där # är talet för likformighet, antal hörn i kompletta grafen som undviks, antal hörn i grafer som filen innehåller respektive antalet kanter i filens grafer.

B.4 expand_graphs-lp.sh / expand_graphs-comb.sh

Dessa två skript är i huvudsak ämnade att användas av turan.sh respektive turan-brute.sh. De kan dock köras separat, då med de fyra argumenten r, s, N och M. En körning av "./expand_graphs-lp.sh 4 5 9 96" kommer då att hitta alla K_5^4-fria grafer på 9 hörn och 96 kanter, och om det krävs så anropar skriptet sig självt för att rekursivt konstruera de grafer på färre hörn som kan behövas för att hitta de grafer som söks.

B.5 expand_graphs-lp-solver.sh

Detta skript körs av expand_graphs-lp.sh, men kan också köras separat. Det tar ett argument, en sökväg till ett linjärprogram. Skriptet läser in inställningar från filen `~/.lpconfig.sh`, där följande variabler kan deklareras, med standardvärden inom parenteser:

- `LPTIMEOUT` - Maximal tidsåtgång i minuter per lösningsförsök. (120)
- `LPTHREADS` - Maximalt antal trådar som får användas av Gurobi. (1)
- `LPMAXSOLN` - Maximalt antal lösningar per lösningsförsök. (1000)
- `LPMINSOLN` - Minimalt antal lösningar som ska hittas innan tidsgräns börjar gälla. (0)
- `LPSTOP` - Se nedan. (no)

Om tids- eller lösningsgräns uppnås utan att linjärprogrammet har konstaterats vara olösbart så avbryts körningen och linjärprogrammet delas upp i två nya linjärprogram av skriptet split.pyB.6. Vartefter de två nya linjärprogrammen rekursivt löses, såvida inte variabeln `LPSTOP` är satt till "yes". Alltså om `LPSTOP` inte är "yes" så kommer en djupet-först-sökning att göras på angivet linjärprogram.

B.6 split.py

Detta skript tar ett linjärprogram i .lp- alternativt .lp.gz-format och delar upp det i två delar. I de nya linjärprogrammen så introduceras en ny likhet $x_e = 1$ respektive $x_e = 0$, där e representerar en kant för vilken det inte tidigare var angivet om den skulle ingå i de utvidgade graferna. På så sätt skapas två enklare problem som kan lösas sekvensiellt (standardbeteende för expand_graphs-lp-solver.shB.5) eller parallellt på separata datorer.

B.7 nCk

Programmet tar två argument n och k och skriver ut $\binom{n}{k}$.

B.8 seed

Programmet kräver argumenten `-r` och `-k`. Det sparar grafen K_s^r minus en kant i filen "graphs-r=[r]-k=[s]-n=[r]-m=[$\binom{n}{r} - 1$].ei" (där "[x]" ersätts av värdet för x).

B.9 lphead

Programmet kräver argumenten `-r`, `-k`, `-N` och `-M`, det formulerar de (o)likheter som gäller för samtliga linjärprogrammeringsutvidgningar av K_s^r-fria r-grafer $G^r(N, M)$. Standardfilnamnet som används är "lphead-r=[r]-k=[s]-N=[N]-M=[M].lp".

B.10 lpgraph

Detta program tar indata i .ei-format och skapar den del av ett linjärprogram i .lp-format som beskriver grafen som ska utvidgas. För varje graf som läses in så skapas en enskild fil med filnamn enligt "lpgraph-r=[r]-k=[s]-n=[n]-m=[m]-N=[N]_no=i.lp", där r, s, n och m tas som argument eller deduceras från filnamnet på .ei-fil som läses in, N definieras som $n + 1$ och $i = 0, 1, 2...$ för de olika graferna.

B.11 lpsolve

Detta program kräver endast ett filnamn som argument. Filen ska vara ett giltigt linjärprogram, vilket fås av att sätta samman den fil som lpheadB.9 producerat med en av de filer som lpgraphB.9 skapat. Linjärprogrammet löses av Gurobi och lösningarna (graferna) som hittas sparas i en fil med namnet på linjärprogrammet med ".soln" tillagt på slutet, formatet för .ei-filerB.2 används för dessa grafer.

B.12 brute

Programmet läser in grafer i .ei-format och kräver därtill argumentet `-M`. Till de inlästa graferna läggs ett hörn till, och så många kanter som krävs för att få grafer med M kanter, de av dessa grafer som är K_s^r-fria sparas sedan i filen "graphs-r=[r]-k=[s]-n=[r]-m=[m]_preiso.ei".

B.13 isoreduce / isoreduce_via_CD

Båda dessa program läser in grafer i .ei-format och skriver ut en av de inlästa graferna från varje isomorfiklass till filen "graphs-r=[r]-k=[s]-n=[r]-m=[m].ei". Skillnaden på programmen är att isoreduce_via_CD omvandlar de inlästa graferna till övertäckningsdesigner före de skickas till Nauty.

B.14 ei2graph / ei2cd

Dessa program omvandlar grafer i .ei-format till hypergrafer i läsbart format, antingen som K_s^r-fria hypergrafer eller övertäckningsdesigner. Utfiler får namnen "graphs-r=[r]-k=[s]-n=[r]-m=[m].ei" respektive "covdes-n=[n]-k=[$n-r$]-t=[$n-s$].ei"

C Källkod

Se `http://www8.cs.umu.se/~dv08llt/exjobb/kod/`, alternativt den elektroniska upplagan av detta dokument.