Université de Montréal

# Le jeu de policiers-voleur sur différentes classes de graphes

par

**Jérémie Turcotte**

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en mathématiques

14 décembre 2020

**Université de Montréal**

Faculté des arts et des sciences

Ce mémoire intitulé

**Le jeu de policiers-voleur sur différentes classes de graphes**

présenté par

**Jérémie Turcotte**

a été évalué par un jury composé des personnes suivantes :

*Abraham Broer*

(président-rapporteur)

*Geňa Hahn*

(directeur de recherche)

*Benjamin Seamone*

(codirecteur)

*Iosif Polterovich*

(membre du jury)

# Résumé

Ce mémoire étudie le jeu de policiers-voleur et contient trois articles, chacun portant sur une classe de graphes spécifique.

Dans le premier chapitre, la notation et les définitions de base de la théorie de graphe qui nous serons utiles sont introduites. Bien que chaque article comporte une introduction citant les concepts et résultats pertinents, le premier chapitre de ce mémoire contient aussi une introduction générale au jeu de policiers-voleur et présente certains des résultats majeurs sur ce jeu.

Le deuxième chapitre contient l'article écrit avec Seyyed Aliasghar Hosseini et Peter Bradshaw portant sur le jeu de policiers-voleurs sur les graphes de Cayley abéliens. Nous améliorons la borne supérieure sur le *cop number* de ces graphes en raffinant les méthodes utilisées précédemment par Hamidoune, Frankl et Bradshaw.

Le troisième chapitre présente l'article concernant le *cop number* des graphes $2K_2$-libres. Plus précisément, il est prouvé que 2 policiers peuvent toujours capturer le voleur sur ces graphes, prouvant ainsi la conjecture de Sivaraman et Testa.

Finalement, le quatrième chapitre est l'article écrit avec Samuel Yvon et porte sur les graphes qui ont *cop number* 4. Nous montrons que tous ces graphes ont au moins 19 sommets. En d'autres mots, 3 policiers peuvent toujours capturer le voleur sur tout graphe avec au plus 18 sommets, ce qui répond par la négative à une question de Andreae formulée en 1986. Un pan important de la preuve est faite par ordinateur; ce mémoire contient donc une annexe comprenant le code utilisé.

**Mots clés** : Théorie des graphes, Combinatoire, Jeu de policiers-voleur, *Cop number*, Graphes de Cayley, Graphes $2K_2$-libres, Graphes 4-policiers-gagnants minimaux.

# Abstract

This thesis studies the game of cops and robbers and consists of three articles, each considering a specific class of graphs.

In the first chapter, notation and basic definitions of graph theory are introduced. Although each article has an introduction citing the relevant concepts and results, the first chapter of this thesis also contains a general introduction to the game of cops and robbers and presents some of its major results.

The second chapter contains the paper written with Seyyed Aliasghar Hosseini and Peter Bradshaw on the game of cops and robbers on abelian Cayley graphs. We improve the upper bound on the cop number of these graphs by refining the methods used previously by Hamidoune, Frankl and Bradshaw.

The third chapter presents the paper concerning the cop number of $2K_2$-free graphs. More precisely, it is proved that 2 cops can always catch the robber on these graphs, proving a conjecture of Sivaraman and Testa.

Finally, the fourth chapter is the paper written with Samuel Yvon which deals with graphs of cop number 4. We show that such graphs have at least 19 vertices. In other words, 3 cops can always catch the robber on any graph with at most 18 vertices, which answers in the negative a question by Andreae from 1986. An important part of the proof is by computer; this thesis thus has an appendix containing the code used.

**Keywords** : Graph theory, Combinatorics, Game of cops and robbers, Cop number, Cayley graphs, $2K_2$-free graphs, Minimum 4-cop-win graphs.

# Table des matières

# Liste des tableaux

# Liste des figures

# Remerciements

Je dois avouer que j'ai trouvé bien ardue la rédaction de cette section. Ce n'était pas par manque de gratitude envers les nombreuses personnes m'ayant aidé au cours de ma maîtrise, mais puisqu'il me semblait difficile d'écrire des remerciements reflétant fidèlement ma reconnaissance. À la recherche d'inspiration, j'ai consulté plusieurs autres mémoires, mais je n'y ai découvert que des commentaires exprimés semblait-il avec une aisance dont je ne pouvais qu'être envieux. Je ne peux donc qu'espérer que mes mots pourront similairement illustrer la dette que j'ai envers tous ceux mentionnés ici.

Je veux en premier lieu remercier mes directeurs Geňa Hahn et Ben Seamone pour m'avoir introduit et guidé dans le monde de la théorie des graphes, pour m'avoir accordé une liberté totale dans le choix des projets que je voulais poursuivre et pour leurs nombreux conseils, toujours pertinents et un peu trop souvent en réponse à mes longs courriels.

J'aimerais aussi remercier mes coauteurs dans les articles qui sont présentés ici, Seyyed, Peter et Samuel. Il va sans dire que sans vos excellentes idées et votre travail acharné, rien de ceci ne serait possible. Merci pour vos commentaires et suggestions, qui ont certainement formé ma compréhension bien au delà de nos collaborations. J'espère bien que nous pourrons travailler encore ensemble dans le futur.

Il serait difficile d'écrire des remerciements complets sans mentionner mes collègues du Trio, Simon et Alexis. Je vous compte évidemment parmi mes amis les plus proches. Vous m'avez toujours impressionné avec vos capacités mathématiques, et j'ai sans aucun doute plus appris de vous que de quiconque. J'ai hâte de voir ce que vous accomplirez dans le futur, il est évident que vous pouvez atteindre tout objectif que vous vous donnez.

J'aimerais remercier plus généralement les étudiants que j'ai pu croiser dans mon parcours à l'Université de Montréal, notamment Théo, Paul, Pascale, Victor, Youcef, Alizée, Julien et Frank, dont plusieurs m'ont aussi aidé dans la rédaction de ce travail.

Je me dois aussi de remercier Paul Gauthier, Yvan St-Aubin, Anne Bourlioux, Pierre-Alexandre Mailhot, Geňa Hahn et Marlène Frigon pour les opportunités qu'ils m'ont donné, comme stagiaire, auxiliaire d'enseignement ou chargé de cours, et leurs nombreux conseils.

Merci à tous mes amis, plusieurs que j'ai déjà nommés, pour leur présence, et pour m'avoir écouté parler de toutes ces mathématiques qui ne vous intéressaient probablement pas. Merci à Olivier et Patrick pour nos discussions quotidiennes, qui ne cessent d'être intéressantes, et j'espère qu'elles continueront pour longtemps encore. J'aimerais plus que tout remercier Gabriel, David, Giuseppe et Frédéric, votre présence dans ma vie depuis toutes ces années est irremplaçable, et ma confiance en vous est absolue.

Mes parents mentionnent souvent en blague que je leur dois rien de moins que mon existence. Je crois néanmoins que le support qu'ils m'ont offert dans les 24 dernières années vaut bien plus. Je me dois évidemment de remercier toute ma famille, incluant mes parents et mon frère Jonathan, pour leur amour. Je vous aime aussi.

Finalement, j'aimerais *vous* remercier, le lecteur. Il est plus qu'étrange d'imaginer que peut-être un jour quelqu'un que je n'aurai jamais rencontré lira ces mots comme je les écris maintenant, le 21 novembre 2020 à 0h28. Malgré la distance et le temps qui nous sépare, j'espère que mon travail pourra vous intéresser et vous inspirer, et que vous pourrez percevoir la même beauté inhérente à ces structures mathématiques que j'ai pu y voir.

# Chapitre 1

# Introduction

Dans cette introduction, nous survolons les définitions, concepts et résultats qui sont prérequis pour les articles des chapitres suivants.

## 1.1. Graphes

Nous introduisons tout d'abord les concepts de base de la théorie des graphes. Les définition présentées ici sont standard dans le domaine et sont basées sur ou peuvent être retrouvées dans [**19, 27, 34**].

Nous commençons par définir l'objet mathématique au centre de cette branche.

**Définition 1.1.1.** Un *graphe* $G$ est une paire $(V(G),E(G))$ d'ensembles tels que $E(G) \subseteq \{\{u,v\} : u,v \in V(G), u \neq v\}$. Les éléments de $V(G)$ sont dits les *sommets* de $G$ et les éléments de $E(G)$ sont appelés les *arêtes* de $G$. En général, pour alléger la notation, on écrira $uv$ ou $vu$ pour l'arête $\{u,v\}$.

Dans notre cas, on travaillera toujours avec des graphes finis, c'est-à-dire des graphes tels que $|V(G)| < \infty$.

En général, on peut visualiser un graphe en associant chaque sommet à un point et chaque arête à une ligne. Plusieurs exemples communs sont illustrés dans la Figure 1.1. Dans les prochaines définitions, nous supposerons que $G$ et $H$ sont des graphes.

Comme dans plusieurs autres branches des mathématiques, la théorie des graphes s'intéresse notamment à un certain nombres de propriétés locales et globales (et certaines propriétés qui sont un peu des deux) et comment celles-ci interagissent. La propriété locale la plus simple est celle du voisinage d'un sommet.

**Définition 1.1.2.** Deux sommets $u,v \in V(G)$ sont dits *adjacents* si $uv \in E(G)$. Le *voisinage* (ouvert) de $u \in V(G)$, noté $N(u)$, est l'ensemble des sommets adjacents à $u$. En ajoutant au voisinage ouvert de $u$ le sommet lui-même, on obtient le *voisinage fermé* de $u$, noté $N[u]$. On peut aussi définir plus généralement le *$t$-ième voisinage fermé* de $u$, noté $N^t[u]$, par $N^1[u] = N[u]$ et $N^t[u] = \bigcup_{v \in N^{t-1}[u]} N[v]$.

Malgré qu'il soit souvent pratique de savoir quels sont les voisins d'un sommet donné, il arrive fréquemment que simplement en connaître le nombre est suffisant, ou tout ce que nous savons.

**Définition 1.1.3.** Le *degré* de $u \in V(G)$, noté $d(u)$, est la taille du voisinage (ouvert) de $u$. Le degré maximal d'un graphe $G$ est noté $\Delta(G)$, ou simplement $\Delta$ si le choix de graphe est clair. Similairement, on écrit $\delta(G)$ pour le degré minimal de $G$. Si tous les sommets de $G$ ont le même degré $k$, on dit que $G$ est *$k$-régulier*.

On recherche souvent plusieurs structures dans les graphes. Certaines des plus simples sont les suivantes.

**Définition 1.1.4.**

(1) Une *chaîne* de longueur $t \geq 1$ dans un graphe $G$ est une suite de sommets distincts $(u_1, u_2, \ldots, u_{t+1}) \in V(G)^{t+1}$ tels que $u_i u_{i+1} \in E(G)$ pour $1 \leq i \leq t$. La longueur fait référence au nombre d'arêtes de la chaîne.

(2) Un *cycle* de longueur $t \geq 3$ dans un graphe $G$ est une suite de sommets distincts $(u_1, u_2, \ldots, u_t) \in V(G)^t$ tels que $u_i u_{i+1} \in E(G)$ pour $1 \leq i \leq t-1$ et $u_t u_1 \in E(G)$.

Ces définitions nous permet directement à définir quelques familles communes de graphes.

**Définition 1.1.5.**

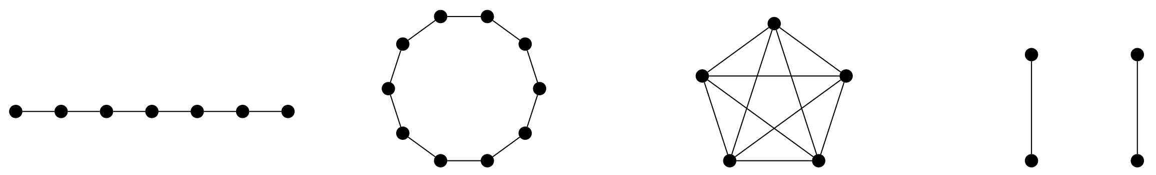
(1) Le graphe $P_t$ est le graphe contenant uniquement une chaîne de longueur $t-1$. On y réfère informellement comme la chaîne de longueur $t-1$.

(2) Le graphe $C_t$ est le graphe contenant uniquement un cycle de longueur $t$. On y réfère informellement comme le cycle de longueur $t$.

(3) Le graphe $K_t$ est le graphe à $t$ sommets dans lequel tous les sommets sont deux à deux adjacents. On l'appelle le *graphe complet* à $t$ sommets.

(4) Pour créer d'autres graphes simples, on peut définir l'addition $G_1 + G_2$ comme l'union disjointe des graphe $G_1$ et $G_2$, en supposant $V(G_1) \cap V(G_2) = \emptyset$. On peut étendre ceci et définir $mG$ comme le graphe formé par $m$ copies distinctes du graphe $G$.

Certains exemples de ces graphes sont présentés dans la Figure 1.1. Notons que bien qu'on en parle comme s'ils étaient uniques, on peut formellement définir chacun de ces graphes une infinité de façon différentes en changeant le nom des sommets dans $V(G)$. Toutefois, ils sont évidemment uniques à isomorphisme prêt, un concept que nous définirons plus bas.



**(a)** $P_7$ **(b)** $C_{10}$ **(c)** $K_5$ **(d)** $2K_2$

**Fig. 1.1.** Exemples de graphes issus de familles communes

La définition des chaînes et des cycles nous permettra de définir plusieurs autres concepts importants.

**Définition 1.1.6.** Soit $u,v \in V(G)$. Soit $C$ une chaîne de longueur minimale entre $u,v$ (pas nécessairement unique). On définit alors la *distance* $d(u,v)$ entre $u$ et $v$ dans $G$ comme la longueur de $C$. Alternativement, la distance peut être définie comme 0 si $u = v$ et sinon le plus petit $t$ tel que $v \in N^t[u]$. On définit le *diamètre* de $G$ comme la distance entre les deux sommets les plus éloignés.

En fait, il est facile de montrer que la paire $(V(G),d)$, où $d$ est la distance sur le graphe $G$, définit un espace métrique discret. Notons que le symbole $d$ est utilisé pour le degré quand il a un paramètre et pour la distance quand il a deux paramètres.

Il est fréquent en mathématiques quand on considère un type d'objet (par exemple un groupe ou un espace topologique) de considérer qu'une restriction pourrait définir un autre objet du même type. La formulation la plus fréquente de cette idée pour les graphes est définie comme suit.

**Définition 1.1.7.**

(1) On dit que $H$ est un *sous-graphe* de $G$ si $V(H) \subseteq V(G)$ et $E(H) \subseteq E(G)$.

(2) On dit que $H$ est un *sous-graphe induit* de $G$ si $H$ est un sous-graphe de $G$ et si $E(H) = \{uv \in E(G) : u,v \in V(H)\}$. Si $S \subseteq V(G)$, le sous-graphe de $G$ induit par $S$, dénoté $\langle S \rangle$, est le sous-graphe induit $H$ tel que $V(H) = S$.

Notons que ces deux définitions sont en général beaucoup moins restrictives que la définition analogue pour les groupes : tous les sous-ensembles de sommets (et pour la première version tous les sous-ensembles d'arêtes) définissent un graphe, tandis que pour qu'un sous-ensemble d'un groupe soit un sous-groupe il faut aussi que l'ensemble soit fermé sous l'opération du groupe (voir plus bas).

Quand nous travaillons à la fois sur un graphe et sur un ou plusieurs de ses sous-graphes, il se peut que nous ajoutions aux symboles définis plus haut un indice pour spécifier à quel graphe on fait référence, par exemple $d_G(u)$, $d_H(u,v)$ ou $N_H(u)$.

Une question naturelle est de demander si sur un graphe donné on peut atteindre tous les sommets en commençant à un certain sommet.

**Définition 1.1.8.** Un graphe $G$ est dit *connexe* si pour toute paire de sommets $u,v \in V(G)$ il existe une chaîne commençant en $u$ et se terminant en $v$. Si un graphe n'est pas connexe, on peut néanmoins partitionner $V(G)$ en *composantes connexes*, c'est-à-dire en sous-graphes maximaux connexes.

Notons que si un graphe n'est pas connexe, on dira que deux sommets dans des composantes connexes distinctes ont une distance infinie, et que le diamètre du graphe est infini.

Nous avons aussi une définition similaire à celle du diamètre mais pour les cycles au lieu des chaînes.

**Définition 1.1.9.** La *maille* (*girth*) d'un graphe $G$ est la longueur d'un plus petit cycle de $G$.

Quand on dit qu'un graphe a maille au moins $m$, on exclura aussi le cas des graphes ne contenant aucun cycle (les cas pour lesquels la maille n'est pas définie). Nous verrons plus loin que cette définition a une importance particulière pour l'étude du jeu de policiers-voleur.

Connaissant la définition des cycles, on peut aussi définir la classe de graphes assez fréquente suivante.

**Définition 1.1.10.** Un graphe $G$ est dit *biparti* s'il ne contient aucun cycle de longueur impaire. De façon équivalente, un graphe $G$ est dit biparti s'il existe une partition des sommets dans des ensembles $A,B$ tel que toute arête doit être entre un sommet de $A$ et un sommet de $B$.

Il est aussi classique en mathématiques de considérer des fonctions entre deux objets du même type qui préservent certaines propriétés (par exemples des homomorphismes de groupe ou des fonctions continues).

**Définition 1.1.11.** Un *homomorphisme* de graphe entre $G$ et $H$ est une fonction $f : V(G) \to V(H)$ telle que $f(u)f(v) \in E(H)$ pour tout $uv \in E(G)$.

Cet outil nous sera utile à plusieurs reprises. Il nous permet aussi de définir ce qu'on considérera comme des graphes identiques comme discuté brièvement plus haut.

**Définition 1.1.12.** Un *isomorphisme* de graphe entre $G$ et $H$ est un homomorphisme $f : V(G) \to V(H)$ qui est bijectif et tel que $\{f(u)f(v) : uv \in E\} = E(H)$. S'il existe un isomorphisme entre $G$ et $H$, on dit qu'ils sont isomorphes et on note $G \simeq H$. Un isomorphisme entre $G$ et lui-même est appelé un *automorphisme*.

Notons que cette définition est différente de celle pour les groupes, car on doit ajouter la condition que l'homomorphisme $f$ est non seulement bijectif sur les sommets, mais qu'il doit induire une bijection sur les arêtes. Intuitivement, le nombre d'automorphismes possible sur un graphe peut être vu comme une mesure du niveau de symétrie du graphe.

Certains sous-graphes ont des noms précis.

**Définition 1.1.13.**

(1) Un *stable* est un ensemble non-vide $S \subseteq V(G)$ tel que tous les sommets dans $S$ sont non-adjacents. On pourrait de façon équivalente demander que $\langle S \rangle \simeq tK_1$ pour un certain $t \geq 1$.
(2) Une *clique* est un ensemble non-vide $C \subseteq V(G)$ tel que tous les sommets de $C$ sont adjacents. On pourrait de façon équivalente demander que $\langle S \rangle \simeq K_t$ pour un certain $t \geq 1$.

Un autre exemple serait de définir une chaîne dans un graphe comme un sous-graphe (pas nécessairement induit) isomorphe à $P_t$ pour un certain $t \geq 1$, et similairement pour un cycle.

Une problème fréquemment étudié en théorie des graphes est de caractériser la structure des graphes pour lesquels un certain sous-graphe est interdit.

**Définition 1.1.14.** On dit que le graphe $G$ est *$H$-libre* s'il ne contient aucun sous-graphe induit isomorphe à $H$.

Par exemple, si un graphe est $K_t$-libre, on peut s'attendre qu'il n'ait pas trop d'arêtes, car autrement on trouverait nécessairement un groupe de sommets de taille $t$ tous adjacents. Plus spécifiquement, le Théorème de Turán (1941) stipule que tout graphe $G$ qui est $K_t$-libre a au plus $\frac{t-2}{2(t-1)}|V(G)|^2$ arêtes (voir par exemple [**27**, Chap. 7.1]).

Dans notre cas, on s'intéressera plus particulièrement dans le deuxième article aux graphes $2K_2$-libres.

Une autre question classique en théorie des graphes est de se demander si on peut dessiner le graphe sans croisements, et si oui quelles sont les propriétés que cela implique.

**Définition 1.1.15.** Un graphe est dit *planaire* s'il peut être dessiné dans le plan sans croisement des arêtes. Plus précisément, on veut que chaque sommet soit un point du plan (distinct des autres sommets), que chaque arête soit une courbe entre les deux sommets qu'elle relie et sans croisements avec elle-même et qu'il n'y ait pas de croisements entre deux arêtes sauf possiblement à leurs extrémités. Similairement, im graphe est dit *toroidal* s'il peut être dessiné sur un tore sans croisement des arêtes. Plus généralement, on dit que le *genre* d'un graphe est le plus petit entier $g$ pour lequel le graphe peut être dessiné sur une surface orientable de genre $g$ (dans [**27**, Exercise 12.53], il est défini qu'une telle surface est similaire à un tore mais avec $g$ poignées) sans croisement des arêtes. Notamment, le genre d'un graphe planaire est 0, et le genre d'un graphe toroidal mais non planaire est 1.

Jusqu'à maintenant, nous n'avons considéré que des graphes non-orientés. Nous définissons maintenant la variante orientée, que nous utiliserons brièvement dans le premier article.

**Définition 1.1.16.** Un *graphe orienté* $G$ est une paire $(V(G),A(G))$ d'ensembles tels que $A(G) \subseteq \{(u,v) : u,v \in V(G), u \neq v\}$. Les éléments de $E(G)$ sont appelés les *arcs* de $G$. On écrira $uv$ pour l'arc $(u,v)$.

Chaque arc a une orientation; on dira donc que l'arc $uv$ *sort* de $u$ et *entre* dans $v$. Notons qu'il est très possible qu'on ait simultanément que $uv,vu \in A(G)$. On peut facilement créer des définitions analogues à la plupart de celles plus haut pour les graphes orientés. Par exemple, on peut définir le *degré sortant* $d^+(u)$ et le *degré entrant* $d^-(u)$ comme respectivement le nombre d'arcs sortants et entrants de $u$. Une *chaîne orientée* sera définie de façon analogue, on demandera simplement qu'on ait jamais deux arcs de la chaîne soient tous deux entrants ou tous deux sortants du même sommet. Pour la connectivité, on dira qu'un graphe orienté est *fortement connexe* si pour toute paire de sommets $u,v$ il existe un chaîne orientée de $u$ vers $v$ et une chaîne orienté de $v$ vers $u$, tandis qu'il sera dit connexe si on peut toujours une chaîne mais en ignorant l'orientation des arcs.

Une variante que nous ne considérons pas ici est celle des *multigraphes*, c'est-à-dire les graphes dans lesquels on peut avoir plusieurs arêtes entre la même paire de sommets, car autoriser les arêtes multiples n'affecterait pas le jeu de policiers-voleur.

En général, nous ne considérons pas les graphes pouvant avoir des boucles, qui sont des arêtes entre un sommet et lui-même. Nous mentionnerons très brièvement les graphes *réflexifs*, soit les graphes ayant une boucle à chaque sommet, car ils peuvent être utiles pour le jeu de policiers-voleur.

## 1.2. Groupes

Dans le premier article, nous travaillerons sur les graphes de Cayley. Afin de pouvoir définir ces graphes, nous avons besoins des bases de la théorie des groupes. Toutes les définitions de cette section peuvent être retrouvées dans [**28**].

**Définition 1.2.1.** Un *groupe* est une paire $(G,\cdot)$ où $G$ est un ensemble et $\cdot$ est une opération $\cdot : G \times G \to G$ vérifiant

(1) $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ pour tous $a,b,c \in G$ (associativité);
(2) il existe un élément $1_G$ tel que $1_G \cdot a = a \cdot 1_G = a$ pour tout $a \in G$ (identité); et

(3) pour tout $a \in G$, il existe un élément inverse $a^{-1} \in G$ tel que $a \cdot a^{-1} = a^{-1}a = 1_G$ (inverse).

De plus, si $a \cdot b = b \cdot a$ pour tous $a,b \in G$, on dit que le groupe est *commutatif* ou *abélien.*

Notons que l'utilisation du symbole $G$ est usuelle autant pour les groupes que les graphes. Lorsque nous travaillerons simultanément avec ces deux concepts, on utilisera plutôt $\Gamma$ comme symbole pour le graphe.

Dans notre cas, on ne considérera que les groupes abéliens. Nous utiliserons donc généralement la notation additive, c'est-à-dire que l'opération sera l'addition $+$, l'élément neutre sera noté $0_G$ et l'inverse de $a$ sera noté $-a$.

Un exemple de groupe abélien est $(\mathbb{Z}/n\mathbb{Z},+)$, soit les entiers modulo $n$ avec l'addition modulo $n$. On peut aussi définir le groupe abélien $(\mathbb{Z}/p\mathbb{Z} \setminus \{0\},\cdot)$, avec $p$ premier, soit les entiers modulo $p$ (sans 0, car il n'a pas d'inverse sous la multiplication) avec la multiplication modulo $p$.

À partir de certains groupes, on peut facilement en former d'autres. La façon la plus simple est le produit de groupes.

**Définition 1.2.2.** Soit $(G,\cdot_G)$ et $(H,\cdot_H)$ des groupes. On définit le *group e produit* $(G \times H, \cdot_{G\times H})$ avec l'opération $(a_1,b_1) \cdot_{G\times H} (a_2,b_2) = (a_1 \cdot_G a_2, b_1 \cdot_H b_2)$. On peut aussi définir le groupe $(G^t, \cdot_{G^t})$ comme le produit du groupe $G$ avec lui même $t$ fois.

Nous aurons aussi besoin du concept de sous-groupes.

**Définition 1.2.3.** Le groupe $(H,\cdot_H)$ est dit *sous-groupe* de $(G,\cdot_G)$ si $H \subseteq G$ et si $\cdot_H$ et $\cdot_G$ coïncident pour les éléments de $H$. On peut alors utiliser le même symbole pour l'opération.

Les graphes de Cayley seront construits à partir d'un groupe et d'un ensemble générateur, que nous avons donc besoin de définir.

**Définition 1.2.4.** Soit un groupe $(G,\cdot)$ et un ensemble $S \subseteq G$. On dit que le *sous-groupe engendré* par $S$, noté $\langle S \rangle$, est le plus petit sous-groupe $H$ de $G$ tel que $S \subseteq H$. On dit que $S$ engendre $G$, ou que $S$ est un *ensemble générateur* de $G$, si $\langle S \rangle = G$.

Un autre concept important est celui des quotients de groupes, que nous ne formulerons ici que pour les groupes abéliens, bien qu'il puisse être défini pour une classe plus large en définissant le concept de sous-groupe normal.

**Définition 1.2.5.** Soit un groupe abélien $(G,+)$ et un sous-groupe $H$. On définit le *groupe quotient* $(G/H,+)$ par $G/H = \{a + H = \{a + h : h \in H\} : a \in G\}$ avec comme opération $(a + H) + (b + H) = (a + b) + H$.

L'exemple $\mathbb{Z}/n\mathbb{Z}$ mentionné plus haut peut en effet être défini comme le quotient du groupe $\mathbb{Z}$ (entiers) par son sous-groupe $n\mathbb{Z}$ (entiers multiples de $n$).

Un résultat élémentaire mais important sur les sous-groupes d'un groupe fini est le suivant, dont nous simplifions légèrement l'énoncé (voir [**28**, Section 3.2]).

**Théorème 1.2.6** (Lagrange)**.** *Si $(G,+)$ un groupe abélien fini et un $H$ un sous-groupe de $G$, alors*

$$|G| = |G/H||H|.$$

Comme mentionné plus haut, il existe aussi un concept d'homomorphisme pour les groupes.

**Définition 1.2.7.** Un *homomorphisme* de groupes entre $(G,\cdot_G)$ et $(H,\cdot_H)$ est une fonction $f : G \to H$ telle que $f(a \cdot_G b) = f(a) \cdot_H f(b)$. Si cette fonction est aussi bijective, on dit alors que c'est un *isomorphisme* et que les groupes $G$ et $H$ sont *isomorphes*, qu'on note $G \simeq H$.

L'exemple le plus important d'homomorphisme dans notre cas sera, pour un groupe abélien $(G,+)$ et un sous-groupe $H$, la projection $p : G \to G/H$ définie par $p(a) = a + H$.

Le dernier concept dont nous avons besoin est celui d'un corps.

**Définition 1.2.8.** Un triplet $(K, +, \cdot)$ est corps si

(1) $(K,+)$ est un groupe abélien;
(2) $(K \setminus \{0\}, \cdot)$ est un groupe abélien; et
(3) $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$ pour tous $a,b,c \in K$ (distributivité).

L'exemple de corps qui nous intéressera sera celui de $\mathbb{Z}/p\mathbb{Z}$ avec l'addition et la multiplication modulo $p$ comme décrit plus haut. Nous aurons besoin du résultat suivant (voir par exemple [**28**, Section 9.5, Proposition 17]).

**Proposition 1.2.9.** *Soit un corps $(K, +, \cdot)$. Un polynôme $f(x)$ de degré $t$ à coefficients dans $K$ a au plus $t$ solutions dans $K$ (incluant les multiplicités).*

## 1.3. Jeu de policiers-voleur

Dans cette section, nous introduisons les règles du jeu de policiers-voleur et présentons certains des résultats majeurs sur ce jeu. Vu le grand nombre d'articles portants sur ce jeu, seule une petite fraction des résultats connus, ceux que je considère sont parmi les plus importants, seront présentés. Afin d'éviter les répétitions, uniquement les résultats qui ne sont pas déjà présentés dans les introductions des articles des prochains chapitres seront mentionnés.

Certaines des preuves, celles qui sont les plus élémentaires, seront présentées afin d'introduire les arguments classiques du domaine. Quand une preuve est présentée, elle est soit très similaire ou identique à sa preuve originale.

### 1.3.1. Règles

Le jeu de policiers-voleur est défini par Quilliot dans [**57**] et par Nowakowski et Winkler dans [**52**]; ces derniers créditent la question à Gabor. Le jeu ne se jouait originalement qu'à un policier; la version à plusieurs policiers, qui est maintenant la norme, est défini par Aigner et Fromme dans [**2**]. Les règles sont les suivantes.

**Définition 1.3.1.** [**2, 52, 57**] Le *jeu de policiers-voleur* est un jeu à tour de rôle impliquant 2 joueurs avec des rôles asymétriques, c'est-à-dire que les objectifs de chaque joueur sont différents. Le premier joueur, celui qui débutera la partie, aura $k$ pièces de jeu nommées les *policiers*, tandis que le deuxième joueur aura une seule pièce de jeu, le *voleur*. La grille de jeu sera un graphe connexe $G$; les cases du jeu seront les sommets de $G$ tandis que les arêtes correspondront aux coups (mouvements) possibles. À leur premier tour, chaque joueur place ses pièces de jeu sur des sommets du graphe; le premier joueur peut en placer plusieurs sur le même sommet s'il le désire. À chacun des tours suivants, chaque joueur peut déplacer un certain nombre (peut-être 0) de ses pièces de jeu vers des sommets adjacents à leurs positions actuelles. Afin de remporter la partie, l'objectif du premier joueur est d'amener un de ses policiers sur le même sommet que le voleur. Si cela n'arrive jamais, le deuxième joueur gagne par défaut.

Notons qu'il n'est pas possible d'inverser l'ordre des joueurs, puisque sinon le jeu serait trivial; il suffirait de placer un des policiers sur le sommet que le voleur a déjà choisi.

Bien que notre objectif ne soit pas de modéliser le comportement de poursuites policières, on veut s'en inspirer. Il est donc raisonnable que les policiers soient placés en premier : on peut voir ceci comme choisir les localisations des postes de police, ou des régions qu'ils patrouillent, puis en sachant cela le voleur décide où commettre un vol.

Évidemment, notre intérêt envers ce jeu n'a pas comme but d'y jouer comme jeu de société, mais plutôt d'analyser ce qui constitue une stratégie optimale. On prendra donc pour acquis que le voleur ne se mettra jamais dans une situation où il peut se faire capturer à moins qu'il n'ait pas d'autre choix (même s'il sait qu'il va perdre éventuellement, il veut survivre le plus longtemps possible), et les policiers n'utiliserons jamais intentionnellement une stratégie perdante s'il en existe une gagnante, c'est-à-dire que les joueurs ne feront jamais d'erreur.

Notre intérêt sera l'entier $k$ mentionné dans la définition du jeu; nous voulons savoir pour quels choix de $k$ le premier joueur gagne, en fonction de la structure du graphe $G$. Ceci nous mène donc à la définition suivante.

**Définition 1.3.2.** [**2**] Étant donne un graphe connexe $G$, le *cop number* est le plus petit nombre de policiers avec lesquels le premier joueur a une stratégie gagnante. Il sera noté $c(G)$.

Nous introduisons aussi la terminologie standard suivante.

**Définition 1.3.3.** Un graphe connexe $G$ est dit *$k$-policier-gagnant* si $c(G) = k$. Pour $k = 1$, on dira simplement que $G$ est *policier-gagnant* (voir [**52**]).

## 1.3.2. Jeu à 1 policier

Comme mentionné plus haut, les premiers résultats sur ce jeu concernent la version avec 1 policier.

Le concept suivant est fondamental dans le domaine.

**Définition 1.3.4.** [**52**] Soit un graphe $G$. On dit que $u \in V(G)$ est un *sommet irréductible* s'il existe $v \in V(G)$, $v \neq u$, tel que $N[u] \subseteq N[v]$.

On voit bien que si le policier est sur $v$ et que le voleur est sur $u$, alors le premier joueur gagne à son prochain tour, et ce peu importe qu'on soit rendu au tour du premier joueur ou du deuxième joueur. Ce concept est souvent nommé un *coin*, mais nous utiliserons ce terme dans le troisième article un peu différemment.

Ce concept nous permet alors de définir la classe de graphes suivante.

**Définition 1.3.5.** [**52**] Un graphe $G$ est dit *démantelable* s'il contient un sommet irréductible $v \in V(G)$ tel que $\langle V(G) \setminus \{v\} \rangle$ est démantelable, ou si $G \simeq K_1$.

Cette classe nous permet alors de caractériser formellement les graphes policiers-gagnants.

**Théorème 1.3.6.** [**52**] *Si $G$ est un graphe connexe, alors $G$ est policier-gagnant si et seulement si $G$ est démantelable.*

Démonstration. Nous présentons la preuve comme formulée par Aigner et Fromme dans [**2**].

On voit bien que $K_1$ est policier-gagnant. Il suffit donc de montrer que $G$ est policier-gagnant si et seulement si $G$ contient un sommet irréductible $v$ et si $G-v$ est policier-gagnant, à l'aide d'un argument inductif.

Supposons tout d'abord que $G$ soit policier-gagnant. Ainsi, il existe une stratégie gagnante pour 1 policier. Ainsi, il existe nécessairement (au moins) une position dans laquelle le voleur se fera prendre peu importe où il se déplace. Notons alors $u$ la position du policier et $v$ la position du voleur, soit un tour avant la victoire du premier joueur. Forcément, $N[v] \subseteq N[u]$, sinon le voleur aurait pu s'enfuir.

On veut montrer qu'il existe une stratégie gagnante sur $G - v$. Il suffit de remarquer que policier peut jouer sur $G$ sans jamais utiliser $v$ (sauf au dernier coup si le voleur se fait capturer sur $v$), car tout coup qu'on peut faire à partir de $u$ est possible à partir de $v$. Jouer sur $G - v$ revient donc à jouer sur $G$ avec cette stratégie spéciale; le fait que le voleur ne vais jamais aller sur $u$ ne change pas que le policier peut gagner, puisque le policier a une stratégie gagnante peu importe les coups choisis par le voleur.

Supposons maintenant que $G-v$ est policier-gagnant, et que $v$ est un sommet irréductible (donc qu'il existe un autre sommet $u$ tel que $N[v] \subseteq N[u]$). Pour capturer le policier sur $G$, le voleur appliquera la stratégie (gagnante) du policier sur $G - v$ quand le voleur n'est pas sur $v$, et quand le voleur est sur $v$ le policier appliqueras cette stratégie en considérant que le voleur était en fait sur $u$. Ceci est bien défini et ne brisera pas la stratégie pour $G-v$ car tout coup possible pour le voleur à partir de $u$ est aussi possible à partir de $v$. Éventuellement, soit le voleur aura été attrapé, ou le policier aura attrapé la position imaginaire du voleur, c'est à dire que le policier est sur $u$ et le voleur est sur $v$, cas dans lequel le premier joueur peut donc gagner un coup plus tard. $\square$

On voit donc que bien que le *cop number* soit une propriété globale du graphe, le cas policier-gagnant semble se décomposer à une propriété relativement locale.

Cette équivalence sera notamment utilisée dans le code présenté à l'Annexe A.2 afin de tester si un graphe est policier-gagnant.

Le même argument sera utilisé dans le deuxième article pour prouver le Lemma 3.4.1. La formulation originale de cet argument dans [**52**] utilise les rétracts, un type d'homomorphisme (voir Definition 4.2.3). Un argument de ce type est en fait utilisé dans [**10**] pour prouver un résultat bien plus général utilisant les rétracts (voir Theorem 4.2.4 dans le troisième article).

### 1.3.3. Bornes supérieures

Nous venons de voir une caractérisation simple des graphes policier-gagnants. Clarke et MacGillivray donnent dans [**26**] une caractérisation des graphes $k$-policiers-gagnants, mais qui ne nous éclaircit que très peu sur la structure de ces graphes. Elle est néanmoins utile pour l'algorithme de calcul du *cop number*, voir Annexe A.1.

En l'absence d'une caractérisation simple du *cop number*, un pan important de la recherche sur le jeu de policiers-voleur porte sur les bornes qu'on peut trouver, autant dans le contexte général que pour des familles de graphes précises. Nous commençons avec certaines des bornes supérieures les plus importantes.

Notons que la structure générale et le choix des résultats présentés dans cette section est similaire à ceux des articles de survol [**8, 17**], qui sont bien plus complets.

Le premier résultat majeur sur le jeu avec plusieurs policiers porte sur les graphes planaires.

**Théorème 1.3.7.** [**2**] *Si $G$ est un graphe planaire connexe, alors $c(G) \leq 3$.*

Une des parties clés de la preuve est le lemme suivant, qui est maintenant un outil de base du domaine.

**Lemme 1.3.8.** [**2**] *Soit $G$ est un graphe connexe, $u,v \in V(G)$, et $P$ une chaîne de longueur minimale entre $u$ et $v$. Il existe une stratégie pour un policier qui, à partir d'un certain moment, interdit au voleur de se déplacer sur un des sommets de $P$.*

Nous ne prouvons pas ce lemme, mais l'idée de sa preuve est simple. Le policier doit commencer par se rendre sur $P$. Ensuite, la tâche du policier sera de suivre la *projection* du voleur sur $P$, c'est-à-dire le policier ira toujours sur le sommet de $P$ dont la distance de $u$ est la même que la distance entre $u$ et le voleur (ou sur $v$ si le voleur est plus $d(u,v)$ sommets plus loin). Puisque $P$ est une plus courte chaîne entre $u$ et $v$, il sera toujours possible pour le policier de suivre ou copier le coup du voleur; le voleur ne pourra jamais aller plus *rapidement* (par rapport à la distance avec $u$) que le policier, car il n'y a aucun raccourci pour $P$.

On voit bien comment ce lemme pourrait être utile dans le cas planaire. En effet, on peut essentiellement découper en morceaux le graphe et la région du plan qu'il occupe; si on choisit bien la chaîne on peut s'assurer que le voleur ne la traverse jamais.

Une version modifiée de ce lemme apparait aussi dans la preuve du prochain résultat prouvé récemment, qui étend le dernier théorème.

**Théorème 1.3.9.** [**46**] *Si $G$ est un graphe toroidal connexe, alors $c(G) \leq 3$.*

On peut donc se demander de façon plus générale si on peut borner le *cop number* en fonction du genre. Les trois résultats suivants montrent l'évolution de la meilleure borne connue en fonction du genre, le dernier résultat étant assez récent.

**Théorème 1.3.10.** [**58**] *Si $G$ est un graphe connexe de genre $g$, alors $c(G) \leq 2g + 3$.*

**Théorème 1.3.11.** [**61**] *Si $G$ est un graphe connexe de genre $g$, alors $c(G) \leq \lfloor \frac{3g}{2} \rfloor + 3$.*

**Théorème 1.3.12.** [**20**] *Si $G$ est un graphe connexe de genre $g$, alors $c(G) \leq \frac{4g+10}{3}$.*

Malgré cette progression, le conjecture suivante est toujours ouverte. Celle-ci est sans doute une des plus importantes du domaine.

**Conjecture 1.3.13** (Schröder)**.** [**61**] *Si $G$ est un graphe connexe de genre $g$, alors $c(G) \leq g + 3$.*

Comme noté dans [**46**], cette conjecture n'est confirmée que pour $g \leq 3$.

Une question encore plus naturelle serait de tenter de borner le *cop number* en fonction de l'ordre du graphe. Il est naturel de croire qu'en général, en l'absence d'autres contraintes, il faut généralement plus de policiers quand le graphe est plus grand. Évidemment, en ne considérant que la taille on risque de perdre beaucoup d'information sur la structure des graphes et donc en arriver avec une borne bien plus élevée que l'actuel *cop number* pour bon nombre de graphes.

La conjecture de Meyniel est considérée comme la conjecture la plus importante portant sur le jeu de policiers-voleur.

**Conjecture 1.3.14** (Meyniel)**.** [**29**] *Si $G$ est un graphe connexe à $n$ sommets, alors $c(G) \in O\left(\sqrt{n}\right)$.*

**Théorème 1.3.15.** [**29**] *Si $G$ est un graphe connexe à $n$ sommets, alors $c(G) \in (1 + o(1))\frac{n \log\log n}{\log n}$.*

Démonstration. Nous présentons la preuve originale de Frankl dans [**29**], avec quelques éléments de la preuve de Baird et Bonato dans [**8**].

Nous commençons par prouver un résultat préliminaire connu. Soit un graphe $G$ qui a degré maximum $\Delta$ et diamètre $D$. On regarde le nombre maximal de sommets qui peuvent être à chaque distance d'un sommet $u$. À distance 0, il y a uniquement $u$. À distance 1, il y a $d(u)$ sommets, et donc au plus $\Delta$ sommets. À distance 2, on a au plus $\Delta(\Delta - 1)$ sommets car on regarde chaque voisin de $u$, et ils ont chacun au plus $\Delta$ voisins, mais cela inclus $u$. Plus généralement, à distance exactement $d \geq 1$ on a au plus $\Delta(\Delta - 1)^d$ sommets. Au total, nous avons donc que

$$|V(G)| \leq 1 + \sum_{d=1}^{D} \Delta(\Delta - 1)^{d-1} = 1 + \Delta\left(\frac{1 - (\Delta - 1)^D}{1 - (\Delta - 1)}\right) = \frac{\Delta(\Delta - 1)^D - 2}{\Delta - 2}$$

tant que $\Delta \neq 2$ en utilisant une identité de sommation fréquemment utilisée. Cette borne (et cet argument) est connue comme la borne de Moore, voir [**8, 68**].

On montre par induction sur $|V(G)|$ que pour tout $b \in \mathbb{R}^{\geq 2}$, $c(G) \leq \frac{|V(G)|}{b} + b^b$. Si $|V(G)| \leq b^b$, l'énoncé est trivial car en plaçant un policier sur chaque sommet on a directement que $c(G) \leq |V(G)|$. On suppose donc que $|V(G)| > b^b$. Si $\Delta(G) \leq 2$, alors $G$ est soit un cycle ou une chaîne, et alors on a bien que $c(G) \leq 2 \leq b^b \leq \frac{|V(G)|}{b} + b^b$. Supposons donc que $\Delta(G) \geq 3$.

Il est impossible que le diamètre et le degré maximum de $G$ soient tous deux au plus $b-1$. En effet, par la borne prouvée plus haut on aurait alors que

$$|V(G)| \leq \frac{\Delta(\Delta-1)^D - 2}{\Delta - 2} \leq \frac{(b-1)(b-2)^{b-1} - 2}{1} \leq b^b.$$

Ceci serait donc une contradiction.

Ainsi, soit il existe un sommet $u$ de degré au moins $b-1$, ou il existe une chaîne $C$ induite de longueur au moins $b-1$. Dans le premier cas, on place un policier sur $u$, qui ne bougera que si le voleur s'aventure dans $N(u)$. Dans le deuxième cas, un policier protège $C$ en utilisant la stratégie du Lemme 1.3.8. Dans les deux cas, on peut donc prendre pour acquis qu'en utilisant 1 policier on peut se restreindre à regarder le reste du graphe. Il n'est pas nécessairement connexe, mais les autres voleurs se rendent dans la composante connexe $G'$ du graphe restant dans laquelle le voleur se situe, et appliquent la stratégie inductive.

Nous savons que $|V(G')| \leq |V(G)| - b$. Ainsi,

$$c(G) \leq 1 + c(G') \leq 1 + \frac{|V(G')|}{b} + b^b \leq 1 + \frac{|V(G)| - b}{b} + b^b = \frac{|V(G)|}{b} + b^b.$$

Ceci complète la preuve par induction.

Afin de prouver le théorème, pour un graphe $G$ choisi de taille $n$, il suffit de prendre

$$b = \frac{\log n}{\log \log n}.$$

On peut prendre pour acquis que $n$ est suffisant grand afin que $b \geq 2$, puisque le $o(1)$ dans la borne recherchée fait en sorte qu'on peut ignorer les petits cas. Ainsi,

$$c(G) \leq \frac{n \log \log n}{\log n} + \left(\frac{\log n}{\log \log n}\right)^{\frac{\log n}{\log \log n}}.$$

On vérifie facilement dans Mathematica que $\lim_{n\to\infty} \frac{\left(\frac{\log n}{\log \log n}\right)^{\frac{\log n}{\log \log n}}}{\frac{n \log \log n}{\log n}} = 0$ et donc $\left(\frac{\log n}{\log \log n}\right)^{\frac{\log n}{\log \log n}} \in o(1)\frac{n \log \log n}{\log n}$, ce qui complète la preuve. □

Des arguments plus avancés furent utilisés pour montrer les deux améliorations suivantes.

**Théorème 1.3.16.** [**24**] *Si $G$ est un graphe connexe à $n$ sommets, alors $c(G) \in O\left(\frac{n}{\log n}\right)$.*

**Théorème 1.3.17.** [**48, 62, 31**] *Si $G$ est un graphe connexe à $n$ sommets, alors $c(G) \in n2^{-(1+o(1))\sqrt{\log n}}$.*

Bien que la véracité de Conjecture de Meyniel ne soit toujours pas connue, on peut se demander si une telle borne tient pour certaines classes naturelles de graphes. Comme noté dans l'introduction de l'article du prochain chapitre, elle tient évidemment pour les classes dont le *cop number* est borné par une constante, comme les graphes planaires. On recherche donc des classes naturelles de graphes avec *cop number* arbitrairement grands, mais croissant asymptotiquement comme $\sqrt{n}$; nous n'en connaissons que quelques unes. La classe des graphes de Cayley dont il s'agira dans le prochain chapitre en est une [**21**]. La conjecture est aussi prouvée presque sûrement asymptotiquement pour les graphes aléatoires [**55, 56**]. Une autre grande classe qui respecte la Conjecture de Meyniel est celle des graphes de diamètre 2 [**48, 67**].

**Théorème 1.3.18.** [**67**] *Si $G$ est un graphe connexe de diamètre au plus 2 à $n$ sommets, alors $c(G) \leq \sqrt{2n}$.*

Démonstration. Nous présentons la preuve de Wagner dans [**67**], légèrement modifiée.

Soit un graphe $G$, mais qui n'est pas nécessairement de diamètre 2. On considère une variante du jeu dans laquelle le voleur joue sur $G$ mais les policiers peuvent bouger sur un graphe $G'$ de diamètre au plus 2 dont $G$ est un sous-graphe induit. On définit $c'(G)$ comme le maximum des *cop number* pour cette variante sur l'ensemble des $G'$ connexes de diamètre 2 dont $G$ est un sous-graphe induit. Il est facile de voir que tout graphe peut être transformé en graphe de diamètre au plus 2 en ajoutant un sommet adjacent à tous les autres et donc que $c'(G)$ existe. On note que $c'(G)$ est toujours fini, en fait $c'(G) \leq c(G)$ car l'ajout de ces sommets additionnels ne peut qu'aider les policiers.

On montre que $c'(G) \leq \sqrt{2n}$ si $G$ a $n$ sommets. La preuve est par induction sur $n$. L'énoncé est trivial pour les petits graphes. Si $n \leq 3$, $c(G) = 1$ et donc l'énoncé tient dans ces cas. Bien qu'on n'en ait pas de besoin, si $n \leq 9$, alors $c(G) = 2$ (voir Theorem 4.2.1), l'énoncé tient aussi directement pour ces graphes.

Soit un $G'$ quelconque avec les conditions plus haut. On veut montrer qu'avec au plus $\sqrt{2n}$ policiers jouant sur $G'$ on peut capturer le voleur qui joue sur $G$.

Si $\Delta(G) \leq \sqrt{2n}$, nous utilisons la stratégie suivante qui utilise au plus $\Delta(G)$ policiers, que nous noterons $P_1, \ldots, P_{\Delta(n)}$. Les policiers prennent des positions originales quelconques. Supposons que le voleur est sur un sommet $x$, et notons son voisinage $N(x) = \{a_1, \ldots, a_{d(u)}\}$. Puisque $G'$ a diamètre au plus 2, pour tout $1 \leq i \leq d(u)$ soit $P_i$ est déjà sur $a_i$, est sur un sommet voisin de $a_i$, ou est sur un sommet ayant un voisin commun avec $a_i$. Le policier $P_i$ aura alors comme tâche de se déplacer vers $a_i$. À la fin de ce tour, chaque voisin de $x$ sera protégé par un policier, c'est-à dire il y aura soit un policier sur $a_i$ ou sur un sommet voisin de $a_i$. Ainsi, le voleur sur $x$ ne pourra pas bouger. Au tour suivant, les policiers pourront se déplacer sur les $a_i$, et au tour suivant le voleur sera capturé.

Supposons maintenant que $\Delta(G) > \sqrt{2n}$ et soit $u$ un sommet de degré maximal dans $G$. On place alors un policier sur $u$, qui ne bougera que pour capturer le voleur s'il s'aventure dans $N(u)$. Il suffit alors de trouver une stratégie pour capturer le voleur dans $G - N[u]$, qui a taille au plus $n - \sqrt{2n} - 1$. On voit alors facilement que $c'(G) \leq c'(G - N[u]) + 1$. En effet, on sait que si on plonge $G - N[u]$ dans $G'$ (c'est aussi un sous-graphe induit dans ce cas), le nombre de policiers nécessaires est au plus $c'(G - N[u])$ pour capturer le voleur s'il ne s'aventure jamais à l'extérieur de $G - N[u]$. Par induction, on a donc que

$$c'(G) \leq \sqrt{2|V(G - N[u])|} + 1 \leq \sqrt{2\left(n - \sqrt{2n} - 1\right)} + 1.$$

On peut vérifier que $\sqrt{2\left(n - \sqrt{2n} - 1\right)} + 1 \leq \sqrt{2n}$, quand cette la racine est définie (pour $n \geq 2 + \sqrt{3}$). Ainsi, $c'(G) \leq \sqrt{2n}$.

Notons que si $G$ est de diamètre au plus 2, alors $c(G) = c'(G)$ en remarquant qu'on peut prendre $G' = G$. Ceci prouve l'énoncé. $\square$

Il est noté dans [**67**] qu'une preuve analogue donne le résultat suivant.

**Théorème 1.3.19.** [**67**] *Si $G$ est un graphe biparti connexe de diamètre au plus 3 à $n$ sommets, alors $c(G) \leq \sqrt{2n}$.*

### 1.3.4. Bornes inférieures

Nous discutons maintenant des bornes inférieures sur le *cop number*. La plupart des bornes se basent sur la maille des graphes afin de pouvoir déterminer du nombre nécessaire, mais pas forcément suffisant, de policiers. Le premier de ces résultats est le suivant.

**Théorème 1.3.20.** [**2**] *Si $G$ est un graphe connexe de maille au moins* 5*, alors $c(G) \geq \delta(G)$.*

Démonstration. Le résultat est trivial pour $\delta(G) = 1$, alors on supposera que $\delta(G) \geq 2$.

Nous montrons que dans un tel graphe, si on joue avec strictement moins de $\delta(G)$ policiers, il existe toujours une façon pour le voleur de s'enfuir. Nous explicitons donc une stratégie gagnante pour le deuxième joueur.

Supposons que le voleur soit sur un sommet $u$. S'il n'y a aucun policier sur un sommet adjacent, alors le voleur ne bougera pas. On peut donc supposer qu'il y a un voleur sur un sommet adjacent et donc que le voleur doit pouvoir bouger sur un sommet non couvert par les policiers. On veut donc montrer qu'il existe forcément un sommet de $N(u)$ auquel aucun policier n'est adjacent.

Puisque $d(u) \geq \delta(G)$ mais qu'on a moins de $\delta(G)$ policiers, il faut qu'il existe au moins un policier couvrant au moins 2 sommets de $N(u)$. Supposons que ce policier soit sur un sommet $x$.

Il existe deux cas possible. Si $x \in N(u)$, on a évidemment que ce policier couvre au moins $x$. Notons $v$ un autre voisin de $u$ couvert par ce policier. Ainsi, $xv$ doit être un arête. On remarque donc que $u$,$x$,$v$ forme un triangle (cycle de longueur 3) dans $G$, ce qui est impossible car la maille est au moins 5.

Supposons que $x \notin N(u)$, et notons $v_1$,$v_2 \in N(u)$ deux sommets couverts par le policier sur $x$. Ainsi, les sommets $u$,$v_1$,$x$,$v_2$ forme un cycle de longueur 4, ce qui est aussi une contradiction à la maille de $G$.

Il nous reste simplement à spécifier quel sommet le voleur doit prendre au début de la partie, en se basant sur le choix de position de départ des policiers, afin d'avoir une stratégie gagnante complète pour le deuxième joueur. Soit $u$ un sommet quelconque. Imaginons pendant un moment que le voleur est sur $u$, l'argument plus haut nous dit alors qu'il existe un voisin où le voleur pourrait s'enfuir sans se faire capturer par les policiers. On peut donc placer le voleur sur ce sommet sans risque d'être soit capturé. $\square$

Nous notons queles hypothèses de ce théorème sont assez fortes. En effet, nous remarquons que la stratégie gagnante du voleur se base sur le fait qu'à tous les coups, dans n'importe quelle position possible, il y a une façon de s'échapper. Il n'y a aucune planification à l'avance dans cette stratégie. En effet, il est clair qu'en général afin d'avoir une stratégie optimale le voleur devrait avoir à décider de ses déplacements en fonction de ce qui pourrait se passer plusieurs coups plus loin. Dans ce cas-ci, nous n'avons qu'une vision locale du graphe, la structure globale du graphe n'est pas considérée. Les meilleures bornes inférieures semblent toutes basées sur ce genre d'argument; comprendre les dynamiques globales du jeu de policiers-voleur semble être une tâche extrêmement difficile.

Dans le théorème précédent, on peut augmenter donc le *cop number* en augmentant le degré minimum du graphe. Andreae prouve dans [**3**] qu'il existe des graphes $k$-réguliers ($k \geq 3$) avec *cop number* arbitairement grands, comme noté par Frankl dans [**29**] avant de de présenter la généralisation suivante du dernier théorème.

**Théorème 1.3.21.** [**29**] *Si G est un graphe connexe de maille au moins* $8t-3$ ($t \geq 1$)*, alors* $c(G) > (\delta(G)-1)^t$.

Le résultat plus fin suivant fut prouvé récemment par Bradshaw et al.

**Théorème 1.3.22.** [**22**] *Si G est un graphe connexe de maille au moins* $4t+1$ ($t \geq 1$)*, alors* $c(G) > \frac{1}{et}(\delta(G)-1)^t$.

On peut aussi se demander si la Conjecture de Meyniel est optimale, c'est-à-dire si le *cop number* peut être borné par une fonction d'ordre inférieur à $O(\sqrt{n})$. Il est prouvé par Prałat dans [**54**] que l'ordre borne dans la Conjecture ne peut pas être réduit (voir aussi la discussion dans [**8**]).

**Théorème 1.3.23.** [**54**] *Il existe une famille de graphes d'ordres arbitrairement grands pour laquelle* $c(G) \approx \sqrt{\frac{n}{2}}$, *où* $n = |V(G)|$.

Ces graphes sont construits comme des graphes d'incidence de plans projectifs, que nous ne définissons pas ici. Il est toutefois noté dans [**8**] que ce sont des graphes bipartis de diamètre 3, montrant que le Théorème 1.3.19 est presque optimal, et que des familles avec *cop number* $\Omega(\sqrt{n})$ de diamètre 2 sont présentées dans [**15**].

De telles familles existent aussi pour les graphes de Cayley, voir [**36**] et l'article du chapitre suivant. Une borne inférieure assez élevée sur le cop number des graphes aléatoire est aussi présentée dans [**13**].

# Chapitre 2

# Cops and robbers on directed and undirected abelian Cayley graphs

par

Peter Bradshaw[1], Seyyed Aliasghar Hosseini[1] et Jérémie Turcotte[2]

([1]) Department of Mathematics, Simon Fraser University, Vancouver, Canada

([2]) Départment de mathématiques et de statistique, Université de Montréal, Montréal, Canada

Cet article a été soumis au journal European Journal of Combinatorics.

Tous les coauteurs ont contribué de façon importante à l'article. Ma contribution est principalement sur les bornes supérieures et se limite à la révision dans le cas des bornes inférieures.

Résumé. Nous montrons que le *cop number* des graphes de Cayley abéliens orientés et non-orientés ont une borne supérieure dans $O(\sqrt{n})$, où $n$ est le nombre de sommets, en introduisant une méthode inductive raffinée. Avec cette méthode, nous améliorons la borne supérieure existante sur le *cop number* des les graphes de Cayley abéliens non-orientés et nous établissons une borne supérieure sur le *cop number* des graphes de Cayley abéliens orientés. Nous utilisons aussi les graphes de Cayley abéliens pour construire de nouvelles *familles de Meyniel extrémales*, qui contiennent des graphes d'ordre $n$ avec *cop number* dans $\Theta(\sqrt{n})$.

**Mots clés :** Graphes de Cayley, Policiers-voleur, Conjecture de Meyniel, Graphes orienté, Famille de Meyniel extrémale

Abstract. We show that the cop number of directed and undirected Cayley graphs on abelian groups has an upper bound in $O(\sqrt{n})$, where $n$ is the number of vertices, by introducing a refined inductive method. With our method, we improve the previous upper bound on cop number for undirected Cayley graphs on abelian groups, and we establish an upper bound on the cop number of directed Cayley graphs on abelian groups. We also use Cayley graphs on abelian groups to construct new *Meyniel extremal families*, which contain graphs of every order $n$ with cop number in $\Theta(\sqrt{n})$.

**Keywords:** Cayley graphs, Cops and robbers, Meyniel's conjecture, Directed graphs, Meyniel extremal family

## 2.1. Introduction

We study the game of cops and robbers, a game in which a team of cops attempts to capture a robber while playing on the vertices of a graph. The game is played on a graph $\Gamma$ which is finite and connected, and can be either undirected or directed. The cops play as a team against the robber. Before the game starts, each cop chooses a starting vertex on $\Gamma$. The robber then does the same. The game alternates between *cop turns* and *robber turns*, with the first turn being a cop turn. On a cop turn, each cop can move to a neighbouring vertex or may choose to pass. The robber then has the same options on a robber turn. There is no restriction preventing two or more cops from sharing the same vertex. If one of the cops ever shares a vertex with the robber, then we will say that the robber is *captured*, and capturing the robber is the cops' objective in order to win the game. On the other hand, if the cops never manage to capture the robber, then we say that the robber wins. The game is played with full information. The cop number of the graph $\Gamma$, written $c(\Gamma)$, is the minimum number of cops needed for a strategy that ensures the cops' victory.

The game of cops and robbers was first introduced for undirected graphs in [**57**] by Quilliot, as well as in [**52**] by Nowakowski and Winkler. The concept of cop number was introduced shortly afterwards by Aigner and Fromme in [**2**]. The cop number is well-studied on many classes of graphs; bounds are known, for example, for graphs of high girth [**29**], Cayley graphs [**21, 29, 30, 35**], intersection graphs [**32**], and graphs excluding certain forbidden subgraphs [**42, 49**].

The game of cops and robbers can also be adapted to directed graphs, or digraphs, by making certain modifications. First, we require the digraphs on which the game is played to be strongly connected. Second, when a cop or the robber moves along an arc to an adjacent vertex, we require that the cop or the robber move in the direction of the arc. This is in contrast to undirected graphs, in which a cop or the robber may move along an edge in any direction, as edges have no orientation. The game of cops and robbers was first considered on digraphs by Hamidoune in [**35**], and this directed version of the game has gained popularity recently; see, for example, [**31, 33, 39, 37, 43**].

Perhaps the furthest reaching and most famous question regarding the cop number is Meyniel's conjecture, which asks whether the cop number of any connected graph on $n$ vertices is in $O(\sqrt{n})$. Frankl first mentions Meyniel's conjecture for undirected graphs in [**29**], and Baird and Bonato ask whether Meyniel's conjecture holds for strongly connected digraphs in [**8**]. Meyniel's conjecture is known, for example, to hold for undirected graphs of diameter 2 [**48, 67**]. The first author has also shown in [**21**] that the cop number of Cayley graphs on abelian groups satisfies Meyniel's conjecture, with an upper bound of $7\sqrt{n}$. Of course, the cop number is bounded above by a constant for many graph classes, such as graphs of bounded genus [**2, 58, 61**], graphs of bounded treewidth [**42**] and graphs without long induced paths [**42**]. In this paper, we will generalize the methods of [**21**] and [**30**] to both improve the upper bound for the cop number of Cayley graphs on abelian groups and show that directed Cayley graphs on abelian groups also satisfy Meyniel's conjecture, which will make these graph classes among the few large classes known to satisfy the conjecture.

Our paper is divided into multiple sections. In Section 2.2, we prove a general lemma about the cop number of Cayley graphs and digraphs on abelian groups. In Section 2.3, we show that the cop number of an undirected Cayley graph on an abelian group of $n$ elements can be bounded by about $0.94\sqrt{n} + \frac{7}{2}$ and show that some improvements are possible by considering the prime decomposition of $n$. In Section 2.4, we use the same methods to bound the cop number of a directed Cayley digraph on an abelian group of $n$ elements by about $1.33\sqrt{n} + 2$. In Section 2.5, we construct, for an infinite number of values $n$, undirected Cayley graphs on abelian groups of $n$ elements with cop number $\frac{1}{2}\sqrt{n}$, and directed Cayley graphs on abelian groups of $n$ elements with cop number $\sqrt{n}$. With a simple modification of these constructions, we obtain families of graphs on $n$ vertices, for any integer $n \geq 1$, with cop number in $\Theta(\sqrt{n})$, which gives new Meyniel extremal families of graphs and digraphs. To the authors' knowledge, the family of digraphs that is obtained has the largest cop number in terms of $n$ of any known digraph construction. Finally, in Section 2.6, we discuss possible improvements and further directions.

## 2.2. Notation and a general strategy

In this section, we will establish some notation and outline our general approach to capturing a robber on a Cayley digraph on an abelian group. All groups that we consider in this paper are abelian. A *directed Cayley graph* on an abelian group is defined as follows:

**Definition 2.2.1.** Let $(G,+)$ be a finite abelian group, and let $S \subseteq G$ be a generating set of $G$ with $0_G \notin S$. The *Cayley graph* $\Gamma$ generated by $G$ and $S$ is defined as follows.

- $V(\Gamma) = G$;
- For any $u,v \in G$, $\Gamma$ contains the arc $(uv)$ if and only if $v - u \in S$.

We often write $\text{Cay}(G,S)$ to refer to the Cayley digraph generated by $G$ and $S$.

We will often refer to directed Cayley graphs on abelian groups as *directed abelian Cayley graphs* or *abelian Cayley digraphs*. In this definition, the requirement that $S$ generate $G$ ensures that the digraph $\text{Cay}(G,S)$ is strongly connected. We recall that in the game of cops on robbers on directed graphs, cops and robbers must traverse edges according to their orientations, so a graph must be strongly connected in order to allow a cop or robber to reach any vertex from any other vertex. We note that for a Cayley graph on an abelian group $G$ generated by a set $S \subseteq G$, if $S = -S$, then all arcs of $\text{Cay}(G,S)$ are bidirectional. In this case, the game of cops and robbers on the directed graph $\text{Cay}(G,S)$ is equivalent to the game on the undirected graph obtained from $\text{Cay}(G,S)$ by replacing each arc with an undirected edge and removing parallel edges. Therefore, when we wish to consider the game of cops and robbers on an undirected Cayley graph on an abelian group, we will require that $S = -S$, and we will regard $\text{Cay}(G,S)$ as an undirected graph. We often refer to an undirected Cayley graph on an abelian group as an *abelian Cayley graph.*

When playing cops and robbers on a Cayley digraph on an abelian group $G$ generated by $S \subseteq G$, we imagine that at each turn, a cop or robber occupies some group element $g \in G$. In the Cayley digraph $\text{Cay}(G,S)$, the vertex $g$ has an out-neighbor $g + s$ for each $s \in S$, and thus we imagine that our cop or robber has a list of possible moves corresponding to the elements of $S$. This cop or robber may choose any element $s \in S$ on its turn and move to the group element $g + s \in G$. We call this *playing the move $s$.* When a cop or robber stays at its current vertex, we say that the cop or robber plays the move $0_G$. To capture the robber, we will let our cops follow a strategy that makes certain robber moves $s \in S$ unsafe for the robber. As we make certain robber moves unsafe, the robber's list of possible moves will become shorter, and the robber's movement options will become more limited.

As the robber's movement becomes more limited, it will become easier for the cops to make even more robber moves unsafe, and we will be able to limit the robber's movement further. Eventually, we will make every move unsafe for the robber, and the robber will have no way to avoid capture. The precise meaning of an unsafe move is discussed below.

The approach of capturing the robber by reducing the number of safe robber moves is introduced by Frankl in [**30**], which is itself inspired by the methods used by Hamidoune in [**35**]. Frankl shows that on an undirected abelian Cayley graph, one cop can usually make two robber moves unsafe, so the number of cops required to capture a robber is about half the size of the graph's generating set, as shown in the following theorem.

**Theorem 2.2.2.** [**30**] *If $\Gamma$ is a Cayley graph on an abelian group with generating set $S$ such that $S = -S$ and $0_G \notin S$, then*

$$c(\Gamma) \leq \left\lceil \frac{|S|+1}{2} \right\rceil .$$

When considering cops and robbers on a Cayley digraph on an abelian group $G$ generated by $S \subseteq G$, we will often define another set $T \subseteq S$ consisting of all of the moves of $S$ that the robber can still play safely, and we will assume that the robber chooses a move from $T$ on each turn in order to avoid unsafe moves. We will call $T$ the robber's *moveset*. In other words, we will often consider a restricted version of the game in which on every turn, the robber is forced to play a move from a set $T$.

The following definition, which originally appears in [**21**] in a slightly different form, is closely related to the concept of limiting the robber's moves.

**Definition 2.2.3.** Let $G$ be an abelian group, and let $S \subseteq G$ and $0_G \notin S$. Given an element $a \in S$, we say that an element $k \in G \setminus \{0_G\}$ *accounts for $a$* (with respect to $S$) if there exists an element $b \in S \cup \{0_G\}$ such that $a - b = k$.

We give some intuition behind the reason that this concept is useful in limiting the moves of a robber on an abelian Cayley digraph. Consider an abelian group $G$ generated by a set $S$, and suppose that a game of cops and robbers is played on Cay($G$,$S$) in which the robber has a moveset $T \subseteq S$. Suppose that some element $k \in G$ accounts for a robber move $a \in T$. Then there exists an element $b \in S \cup \{0_G\}$ such that $a - b = k$. If the robber occupies a vertex $r \in G$, then a cop $C$ at $r + k$ can prevent the robber from playing $a$; if the robber plays $a$, then $C$ can play $b$ to capture the robber. Furthermore, if the robber plays another move $a' \in T$, then $C$ can also play $a'$ and maintain a difference of $k$ with the robber. Thus, on each subsequent turn, the robber must not play $a$, and we see that a cop $C$ at $r + k$ has a strategy to essentially remove $a$ from the robber's moveset. Similarly, a cop $C$ at $r + \gamma k$ for some nonnegative integer $\gamma$ can also essentially remove $a$ from the robber's moveset by considering the following observation. If the robber plays the move $a$ $\gamma$ times, then $C$ can respond with $b$ each time, and $C$ will capture the robber. As $C$ can maintain its “difference" in $G$ with the robber by copying each move $a' \neq a$ that the robber plays, the robber can only play $a$ a finite number of times before being captured, and hence the robber must eventually abandon the move $a$. We illustrate this concept in Figure 2.1. This strategy of using a cop to “copy" the robber's moves and eventually prevent the robber from playing a certain move previously appears in [**30**] and [**35**].

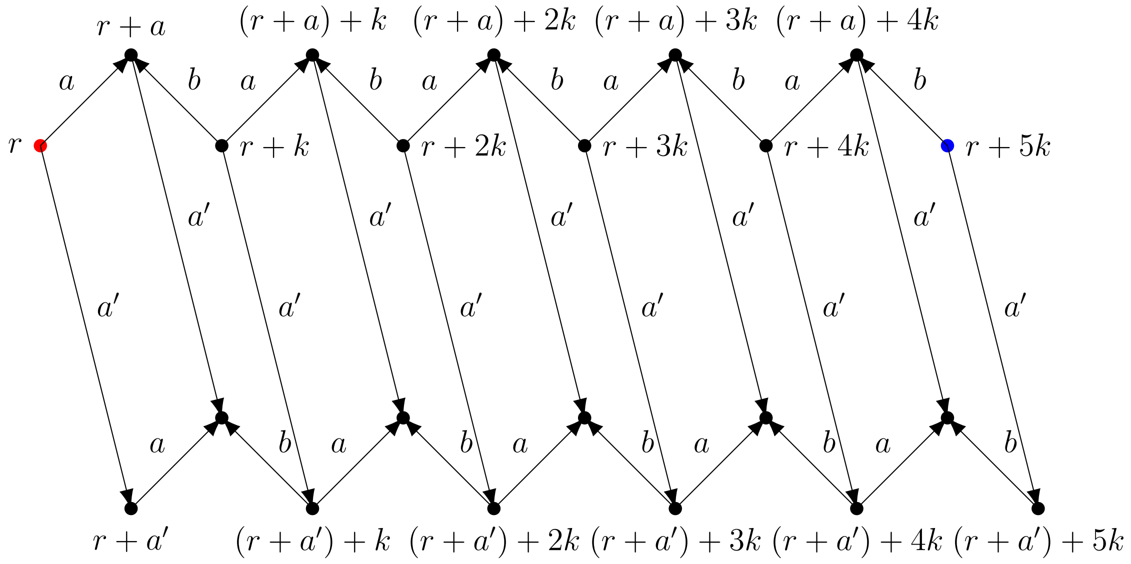


**Fig. 2.1.** The figure shows a cop guarding a robber move in an abelian Cayley graph. The robber's vertex is labelled $r$, and each arc is labelled with its corresponding generating element. The values $a$ and $b$ are generating elements, and $a - b = k$. Here, a cop occupies $r + 5k$, so the difference between the cop and robber's positions is $5k$. If the robber plays $a$, then the cop will play $b$, and the difference between the cop and robber will decrease to $4k$. If the robber continues to play $a$, then the cop may continue to play $b$, decreasing the difference between the cop and robber's positions to $3k$, then $2k$, then $k$, and finally 0. If the robber plays a different move $a'$, then the cop can also play $a'$ and maintain its difference with the robber.

Thus, when we say that a robber move is unsafe, we will mean that a cop is “guarding" this robber move as described above, and the robber can only play this move finitely many times before being captured by the guarding cop. As in [**21**], we will use the fact that when $T$ is large, one element $k$ can account for many elements of $T$. Figure 2.2 shows a local structure that appears in abelian Cayley digraphs with one group element accounting for many generating elements. The figure gives some intuition for how a single group element accounting for many generating elements allows a single cop to guard many robber moves.

To avoid repeating the same conditions, we define the following notation.

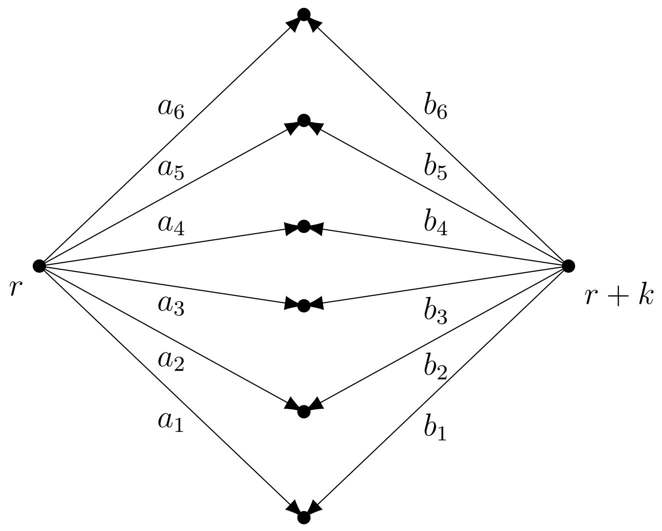


**Fig. 2.2.** The figure shows a subgraph of an abelian Cayley digraph $\Gamma$. The generating set of $\Gamma$ contains six generating elements $a_1, \ldots, a_6$ and six generating elements $b_1, \ldots, b_6$, satisfying $a_1 - b_1 = \cdots = a_6 - b_6$. Therefore, the group element $k = a_1 - b_1$ accounts for all six generating elements $a_1, \ldots, a_6$, and thus if the robber occupies the vertex $r$, a cop at $r + k$ can guard all moves $a_1, \ldots, a_6$.

**Definition 2.2.4.** We define

$$\mathcal{G}_d = \{(G,S,T) : G \text{ is a finite abelian group, } S \text{ is a generating set of } G,\ 0_G \notin S, \text{ and } T \subseteq S\}$$

and

$$\mathcal{G}_u = \{(G,S,T) \in \mathcal{G}_d : S = -S\}.$$

We also define $\mathcal{D} = \{(n,s,t) \in \mathbb{N}^3 : n \geq 1 \text{ and } n-1 \geq s \geq t \geq 0\}$.

The set of triples $\mathcal{G}_d$ corresponds to directed abelian Cayley graphs with specified robber movesets, and the set of triples $\mathcal{G}_u$ corresponds to undirected abelian Cayley graphs with specified robber movesets. The set $\mathcal{D}$ then includes all possible sizes for a triple in $\mathcal{G}_d$ or $\mathcal{G}_u$ (along with some unattainable triple sizes).

We also define the following.

**Definition 2.2.5.** Let $(n,s,t) \in \mathcal{D}$.

- We define $c_d(n,s,t)$ as the maximum, over all triples $(G,S,T) \in \mathcal{G}_d$ of respective sizes $(n,s,t)$, of the number of cops required to capture a robber on $\mathrm{Cay}(G,S)$ when the robber may only play moves in $T$.
- We define $c_u(n,s,t)$ as the maximum, over all triples $(G,S,T) \in \mathcal{G}_u$ of respective sizes $(n,s,t)$, of the number of cops required to capture a robber on $\mathrm{Cay}(G,S)$ when the robber may only play moves in $T$.

Whenever there exists no triple $(G,S,T) \in \mathcal{G}_d$ of respective sizes $(n,s,t) \in \mathcal{D}$, we say that $c_d(n,s,t) = 1$. Similarly, whenever there exists no triple $(G,S,T) \in \mathcal{G}_u$ of respective sizes $(n,s,t)$ for some triple $(n,s,t) \in \mathcal{D}$, we say that $c_u(n,s,t) = 1$. Furthermore, $c_d(n,s,t)$ and $c_u(n,s,t)$ have a trivial upper bound of $n$. Thus, $c_d(n,s,t)$ and $c_u(n,s,t)$ are well-defined for all $(n,s,t) \in \mathcal{D}$.

Note that as $\mathcal{G}_u \subseteq \mathcal{G}_d$, it immediately follows that for any triple $(n,s,t) \in \mathcal{D}$, $c_u(n,s,t) \leq c_d(n,s,t)$. Furthermore, in a standard game of cops and robbers, the robber may choose any move in $S$ on each turn, and hence for a finite abelian group $G$ generated by set $S$, the cop number of $\mathrm{Cay}(G,S)$ is at most $c_d(n,s,s)$ in general, and the cop number of $\mathrm{Cay}(G,S)$ is at most $c_u(n,s,s)$ when $S = -S$.

For technical reasons which will become clear shortly, we also need to define the following.

**Definition 2.2.6.** We define $\mathcal{B} = \{(n,s,t) \in \mathcal{D} : t = 0 \text{ or } s = n-1\}$. We say that triples $(n,s,t) \in \mathcal{B}$ are *boundary values.*

In other words, boundary values give the sizes of triples $(G,S,T) \in \mathcal{G}_d$ for which determining the number of cops required to capture a robber on $\mathrm{Cay}(G,S)$ with moveset $T$ is trivial, as we see in the following observation.

**Lemma 2.2.7.** *If $(n,s,t) \in \mathcal{B}$, then $c_d(n,s,t) = c_u(n,s,t) = 1$.*

Proof. For a triple $(n,s,t) \in \mathcal{B}$, let $(G,S,T) \in \mathcal{G}_d$ such that $(|G|,|S|,|T|) = (n,s,t)$. (If no such triple $(G,S,T)$ exists, then $c_d(n,s,t) = c_u(n,s,t) = 1$ by definition.) Consider a game of cops and robbers on $\mathrm{Cay}(G,S)$ in which the robber's moveset is $T$.

- If $T = \emptyset$, then the robber has no moves, and a single cop may move to the robber's position and capture the robber.
- If $|S| = |G| - 1$, then $\mathrm{Cay}(G,S)$ is a complete digraph, and a single cop can capture the robber after one move.

Thus, we have shown that $c_d(n,s,t) = 1$. As $1 \le c_u(n,s,t) \le c_d(n,s,t)$, it also follows that $c_u(n,s,t) = 1$. $\square$

We note that for a triple $(G,S,T) \in \mathcal{G}_d$ of respective sizes $(n,s,t)$, if $n \le 2$, then $s = n - 1$ must hold, so $(n,s,t) \in \mathcal{B}$. Therefore, when we consider values $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, we may assume that $n \ge 3$.

We are now ready for our main tool, which will be the following lemma. This lemma essentially formalizes a general inductive strategy of capturing the robber by guarding robber moves until no robber move is safe. The lemma generalizes key ideas used by Frankl in [**30**] and uses the idea from [**21**] of having some elements of $S$ which account for many elements of $T$.

We first give an informal description of the lemma. We will have functions $g$ and $h$ taking values in $\mathcal{D} \setminus \mathcal{B}$. The function $h$ will give a lower bound for the number of robber moves that a single cop can make unsafe on an $n$-vertex $s$-regular abelian Cayley graph or digraph, when the robber's moveset is of size $t$. We will show that if $g$ satisfies certain properties that are necessary for the inductive strategy of guarding robber moves described above, then $g(n,s,t)$ gives an upper bound for the number of cops needed to capture a robber on such a graph.

**Lemma 2.2.8.** *Let $(\mathcal{G},c)$ be either $(\mathcal{G}_d,c_d)$ or $(\mathcal{G}_u,c_u)$, and let $g : \mathcal{D} \setminus \mathcal{B} \to \mathbb{R}^{\ge 2}$ and $h : \mathcal{D} \setminus \mathcal{B} \to \mathbb{R}^{>0}$ be functions. Suppose that $g$ and $h$ respect the following conditions for all $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$:*

*(1) For any $(G,S,T) \in \mathcal{G}$ with respective sizes $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, there exists an element $k \in G \setminus \{0_G\}$ accounting for at least $h(n,s,t)$ elements of $T$ with respect to $S$;*

*(2) For $n' \le \frac{n}{2}$, $s' \le s$, $t' \le t$, and $(n',s',t') \in \mathcal{D} \setminus \mathcal{B}$, either $g(n,s,t) \ge c(n',s',t')$, or $g(n,s,t) \ge g(n',s',t')$;*

*(3) If $1 \le t' \le t - h(n,s,t)$, then $g(n,s,t) \ge g(n,s,t') + 1$.*

*Then, if $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, then $c(n,s,t) \le g(n,s,t)$.*

Proof. We fix functions $g$ and $h$ that satisfy the conditions of the lemma.

Suppose that the lemma does not hold for some $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$. We choose our offending triple $(n,s,t)$ with $n$ as small as possible and, subject to $n$ being minimum, with $t$ as small as possible. As the lemma does not hold for $(n,s,t)$, we may choose $(G,S,T) \in \mathcal{G}$ with respective sizes $(n,s,t)$ so that $g(n,s,t)$ cops are not enough to capture a robber on $\mathrm{Cay}(G,S)$, even when the robber may only play moves from $T$. We will show that this gives us a contradiction.

By condition (1), there exists an element $k \in G$, satisfying $k \ne 0_G$, that accounts for at least $h(n,s,t)$ elements of $T$. We would like to show that we can position a cop at a vertex $r + \gamma k$, where $r$ is the position of the robber, and $\gamma$ is some integer. In other words, we would like to show that we can capture the robber "modulo $k$." To this end, we let $\phi : G \to G/\langle k\rangle$

be the natural homomorphism $a \mapsto a + \langle k \rangle$. By the definition of $\phi$, placing a cop at such a vertex $r + \gamma k$ is equivalent to capturing the robber in a game of cops and robbers played on $G/\langle k \rangle$ with cop moveset $\phi(S)$ and robber moveset $\phi(T)$.

We first note that $(G/\langle k \rangle, \phi(S), \phi(T)) \in \mathcal{G}$. In particular, if $S = -S$, then $\phi(S) = \phi(-S) = -\phi(S)$.

We now show that our $g(n,s,t)$ cops have a strategy to capture the robber in the game on $G/\langle k \rangle$. As $k \neq 0_G$, we see that $n' = |G/\langle k \rangle| \leq n/2$, $s' = |\phi(S)| \leq |S| = s$, and $t' = |\phi(T)| \leq |T| = t$. If $(n', s', t') \in \mathcal{B}$, then as shown previously, $c(n', s', t') = 1 < g(n,s,t)$, and our $g(n,s,t)$ cops may capture the robber on $G/\langle k \rangle$. Otherwise, suppose that $(n', s', t') \in \mathcal{D} \setminus \mathcal{B}$. Then $c(n', s', t') \leq g(n,s,t)$ either directly from (2), or from the inequality $c(n', s', t') \leq g(n', s', t') \leq g(n,s,t)$, which follows from the fact that $(n,s,t)$ is a minimal counterexample. Therefore, our $g(n,s,t)$ cops have a strategy by which a cop $C$ can reach a vertex $r + \gamma k$ for some integer $\gamma \geq 0$, where $r \in G$ is the position of the robber.

Next, we show that at this point, $C$ has a strategy to restrict the robber to a moveset of size at most $t - h(G,S,T)$. Let $A = \{a_1, \dots, a_m\} \subseteq T$ be the set of robber moves accounted for by $k$. If the robber plays a move $a' \notin A$, then $C$ plays $a'$, and $C$ will stay at a vertex of the form $r + \gamma k$, where $r$ is the new position of the robber. If the robber plays a move $a_i \in A$, then $C$ has a move $b_i \in S \cup \{0_G\}$ such that $a_i - b_i = k$. After $C$ plays $b_i$, $C$ now occupies a vertex $r + (\gamma - 1)k$, where $r$ is the new position of the robber. Thus we see that whenever the robber plays a move from $A$, which must be accounted for by $k$, the “difference" between the robber and $C$ decreases by exactly $k$. Thus, if the robber plays a move from $A$ sufficiently many times ($\gamma$ times), then the robber will be caught by $C$. Therefore, the robber must eventually stop playing all moves of $A$. The number of moves in $A$ is at least $h(n,s,t)$, and hence $C$ restricts the robber to a moveset $T \setminus A$ of size at most $t - h(n,s,t)$.

We note that when applying the inductive strategy on the quotient graph $\mathrm{Cay}(G/\langle k \rangle, \phi(S))$, it is still possible for the robber to play moves which are not considered safe, but only a bounded number of times. For example, in the paragraph above, we describe the move $a_i$ as unsafe, but the robber may play $a_i$ up to $\gamma - 1$ times without being captured. If the robber plays an “unsafe" move, we pause the inductive strategy; then all cops playing the quotient strategy copy the robber’s move, while the cops guarding this unsafe move advance closer to the robber.

Now, we show that it is possible for at most $g(n,s,t) - 1$ cops to win in the game given by the triple $(G,S,T \setminus A)$. This will give us our contradiction, as we may then capture the robber in the game given by $(G,S,T)$ with $g(n,s,t)$ cops by using one cop to make the moves in $A$ unsafe for the robber and then using the remaining $g(n,s,t) - 1$ cops to win in $(G,S,T \setminus A)$. If $(n,s,t - |A|)$ is a boundary value, then 1 cop is sufficient for the game given by

$(G,S,T\setminus A)$, and then as $g(n,s,t) \geq 2$, we have our contradiction. Note that as $(n,s,t) \in \mathcal{D}\setminus\mathcal{B}$, $(n,s,t-|A|) \in \mathcal{B}$ if and only if $t-|A| = 0$. Otherwise, $t-|A| \geq 1$ and $(n,s,t-|A|) \in \mathcal{D}\setminus\mathcal{B}$, and by the minimality of $(n,s,t)$ and condition (3), $g(n,s,t-|A|)-1$ additional cops are sufficient to capture the robber in the game given by the triple $(G,S,T\setminus A)$. Therefore, in total, we need at most $g(n,s,t)$ cops to capture the robber, which contradicts the minimality of $(n,s,t)$. Thus, our proof is complete. □

## 2.3. Upper bound for undirected abelian Cayley graphs

In this section, we will show that the approach we have outlined in Lemma 2.2.8 gives us an upper bound of $\frac{1}{\sqrt{(\sqrt{2}-1)e}}\sqrt{n}+\frac{5}{2} \approx 0.9424\sqrt{n}+\frac{7}{2}$ on the cop number of undirected abelian Cayley graphs of $n$ vertices. As we consider undirected graphs in this section, whenever we have an abelian group $G$ generated by a set $S$, we will always require that $S=-S$. This way, we may consider $\mathrm{Cay}(G,S)$ as an undirected abelian Cayley graph. Some of the symbolic and optimization computations in this section and the next were done with Mathematica [**41**], but with care and patience, each computation can be checked by hand.

Using our main tool of Lemma 2.2.8, we will aim to define functions $g$ and $h$ that satisfy its conditions for $(\mathcal{G}_u,c_u)$ and such that $g(n,s,s)$, which is an upper bound for cop number, is not too large. Therefore, the main challenge of this section will be choosing suitable functions $g$ and $h$, and moreover, showing that these functions satisfy all the conditions of Lemma 2.2.8.

Before we seek our functions $g$ and $h$ with which to prove an upper bound on the cop number of abelian Cayley graphs, we first note that there is a simple choice of $g$ and $h$ that gives a slightly weaker version of Theorem 2.2.2, as shown in the following proposition. This simple choice of the functions $g$ and $h$ gives an instructive example of how to use Lemma 2.2.8, and this application of Lemma 2.2.8 furthermore shows that our lemma is indeed a generalization of the method of Frankl from [**30**].

**Proposition 2.3.1.** *If $\Gamma$ is a Cayley graph on an abelian group with generating set $S$ such that $S=-S$ and $0_G \notin S$, then*

$$c(\Gamma) \leq \left\lceil \frac{|S|+3}{2} \right\rceil .$$

Proof. We wish to define functions $g$ and $h$ taking values in $\mathcal{D}\setminus\mathcal{B}$ that satisfy the conditions of Lemma 2.2.8 for $(\mathcal{G}_u,c_u)$. We choose

$$g(n,s,t) = \left\lceil \frac{t+3}{2} \right\rceil ;$$

$$h(n,s,t) = \begin{cases} 1 & t = 1,2 \\ 2 & t \geq 3. \end{cases}$$

We claim that $g$ and $h$ satisfy the conditions of Lemma 2.2.8 for $(\mathcal{G}_u,c_u)$. We immediately notice that $h(n,s,t) > 0$ by definition, and for all $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, we have $t \geq 1$, and hence $g(n,s,t) \geq 2$.

We show that condition (1) of Lemma 2.2.8 is satisfied. Indeed, let $(G,S,T) \in \mathcal{G}_u$ with respective sizes $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$. If $t \in \{1,2\}$, then for any $a \in T$, we may let $k = a$; then, as $a - 0_G = k$, $k$ accounts for at least 1 element of $T$ with respect to $S$, namely $a$. If $t \geq 3$, then we may choose any elements $a,b \in T$ with $a \neq -b$ and let $k = a + b$; then, as $a - (-b) = k$ and $b - (-a) = k$, $k$ accounts for at least two elements of $T$ with respect to $S$, namely $a$ and $b$. Hence, condition (1) of Lemma 2.2.8 is satisfied.

Next, we show that condition (2) of Lemma 2.2.8 is satisfied. If we have $n' \leq n/2$, $s' \leq s$, and $t' \leq t$ such that $(n',s',t') \in \mathcal{D} \setminus \mathcal{B}$, then

$$g(n,s,t) = \left\lceil \frac{t+3}{2} \right\rceil \geq \left\lceil \frac{t'+3}{2} \right\rceil = g(n',s',t').$$

Finally, we show that condition (3) of Lemma 2.2.8 is satisfied. Suppose $1 \leq t' \leq t - h(n,s,t)$. As $h(n,s,t) \geq 1$, we must have $t \geq 2$. If $t = 2$, then $t' = 1$, and $g(n,s,t) = 3 = 2 + 1 = g(n,s,t') + 1$. Otherwise, if $t \geq 3$, then $h(n,s,t) = 2$. Thus, if $1 \leq t' \leq t - 2$, then

$$g(n,s,t) = \left\lceil \frac{t+3}{2} \right\rceil \geq \left\lceil \frac{t'+3}{2} \right\rceil + 1 = g(n,s,t') + 1.$$

Hence, as $g$ and $h$ satisfy the conditions of Lemma 2.2.8, it follows that $c_u(n,s,t) \leq g(n,s,t)$ for all $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$. Therefore, for an abelian group $G$ of $n$ elements generated by a set $S \subseteq G$ of $s$ elements satisfying $S = -S$, if $\Gamma = \mathrm{Cay}(G,S)$, one of the following holds: either $(n,s,s) \in \mathcal{B}$ and $c(\Gamma) \leq c_u(n,s,s) = 1 < \lceil \frac{s+3}{2} \rceil$, or $(n,s,s) \in \mathcal{D} \setminus \mathcal{B}$ and $c(\Gamma) \leq c_u(n,s,s) \leq g(n,s,s) = \lceil \frac{s+3}{2} \rceil$. □

In fact, by defining boundary values more carefully, it is possible to obtain the exact result of Theorem 2.2.2 from Lemma 2.2.8. More precisely, if we add the inductive base cases for Frankl's proof from from [**30**] as boundary values in the undirected case, then we may use Lemma 2.2.8 to give the exact same result as Theorem 2.2.2. However, this modification requires that boundary values be defined separately for the directed and undirected cases, and it adds to the already existing technicalities, so we opt for a simpler presentation with a slightly worse additive constant.

In the proof of Theorem 2.2.2, we let a single element of $G$ account for at most two elements of $T$. As discussed earlier, we will see that in general, a single element can account for many more than two elements of $T$. This will allow us to use a similar strategy to find an improved upper bound for the cop number of an abelian Cayley graph.

For the remainder of this section, our goal will be to establish a sharper upper bound on the cop number of an undirected Cayley graph on an abelian group. The main tool for our improved upper bound will be Lemma 2.2.8, so as discussed before, we will seek functions $g$ and $h$ that we can use with Lemma 2.2.8. In the following definition, we define a function $h$ that we will use for the entire remainder of this section. In the definition of $h$, we will use a fixed constant $c > 0$. We will assign a value to $c$ later.

**Definition 2.3.2.** We define the function $h : \mathcal{D} \setminus \mathcal{B} \to \mathbb{R}^{>0}$ by

$$h(n,s,t) = \begin{cases} 1 & t \in \{1,2\} \ \text{ and } t \le c\sqrt{n} \\ 2 & 3 \le t \le c\sqrt{n} \\ \frac{ts}{n-1} & t > c\sqrt{n}. \end{cases}$$

We note that it is important to add that $t \le c\sqrt{n}$ in the first condition. We will sometimes choose $c$ to be as low as about 0.8, so when $n$ is small, it is possible that $c\sqrt{n} < 2$.

**Lemma 2.3.3.** *The function $h$ satisfies condition (1) of Lemma 2.2.8 for $(\mathcal{G}_u, c_u)$.*

Proof. We must show that if $(G,S,T) \in \mathcal{C}_u$ is a triple with respective sizes $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, then there exists an element $k \in G \setminus \{0_G\}$ accounting for at least $h(n,s,t)$ elements of $T$ with respect to $S$.

Suppose that $t \le c\sqrt{n}$. Then $h$ is defined as in the proof of Theorem 2.3.1 given above, and the statement follows from the same argument.

Suppose, on the other hand, that $t > c\sqrt{n}$. We compute a multiset $M$ consisting of all differences $a_i - a_j$, for $a_i \neq a_j$, $a_i \in T$, and $a_j \in S \cup \{0_G\}$. Let $k$ be a most frequently appearing element of $M$. There are $t$ possible choices for $a_i$, and there are $s$ possible choices for each $a_j$, namely $0_G$ and every element of $S \setminus \{a_i\}$.

By the pigeonhole principle, as each element of $M$ is one of $n-1$ possible values, the most commonly occurring element $k$ of $M$ must appear at least $\frac{ts}{n-1}$ times. Therefore, $k$ must account for at least $\frac{ts}{n-1}$ elements of $T$ with respect to $S$. As $k \neq 0_G$, the statement again holds. $\square$

The cutoff at $c\sqrt{n}$ in the definition of $h$ is analogous to the cutoff in the Pairing Algorithm of [**21**]. The idea behind this cutoff is the fact that when $t$ is small, the quantity $\frac{ts}{n-1}$ becomes smaller than 2, and then it is preferable to argue directly that there exists an element of $G \setminus \{0_G\}$ accounting for two elements of $T$. As discussed in Section 2.6, we could, of course, take the ceiling of this function when applying the pigeonhole principle, but this makes analysis of the function $h$ very difficult. One might also be interested in modifying this cutoff to be of the form $t > c\frac{n-1}{s}$ (for some constant $c \geq 1$) in order for $h$ always to be at least 1. For some triples $(n,s,t)$, this would allow $h(n,s,t)$ to become larger while still satisfying condition (1). However, this alternative cutoff of $h$ does not appear to behave nicely when it comes to verifying condition (2).

Next, we will define our function $g$. Our goal for the remainder of this section will then be to show that $g$ and $h$ satisfy the conditions of Lemma 2.2.8 for $(\mathcal{G}_u, c_u)$ and that $g(n,s,s)$ is not too large.

**Definition 2.3.4.** We define the function $g : \mathcal{D} \setminus \mathcal{B} \to \mathbb{R}^{\geq 2}$ by

$$g(n,s,t) = \begin{cases} \left\lceil \frac{t+3}{2} \right\rceil & 1 \leq t \leq c\sqrt{n} \\ \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + \frac{c\sqrt{n}}{2} + \frac{7}{2} & t > c\sqrt{n}. \end{cases}$$

This choice of $g$ may not seem straightforward, so we present the intuition behind this definition. We suppose that for a value $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, we have an abelian group $G$ on $n$ elements generated by a set $S \subseteq G$ (with $S = -S$) of $s$ elements, and a subset $T \subseteq S$ of $t$ elements. We would like to estimate the number of elements of $G$ needed to form a set $K$ such that the elements of $K$ altogether account for each element of $T$, since, as we have discussed, this will help us count the number of cops needed to make every robber move unsafe in a game on Cay$(G,S)$.

In order to estimate the number of elements needed in $K$, we may construct $K$ iteratively. The iterative construction that we describe here is a refinement of the Pairing Algorithm from [**21**]. If $t \leq c\sqrt{n}$, then we may pair the elements of $T$ by the method of Proposition 2.3.1 and obtain a set $K$ of at most $\lceil \frac{t+3}{2} \rceil$ elements, for which the elements of $K$ altogether account for all of $T$ with respect to $S$. On the other hand, if $t > c\sqrt{n}$, we may choose one element $k \in G$ to account for at least $\frac{ts}{n-1}$ elements of $T$, as in the proof of Lemma 2.3.3. More generally, we can use the same idea to define a recursive process that repeatedly adds elements to $K$, and we may run this process until at most $c\sqrt{n}$ elements of $T$ are not accounted for by $K$. We execute our recursive process as follows. We define $z_i$ to be the number of elements accounted for by $K$ after $i$ iterations of our process. We immediately see that $z_0 = 0$, and if we choose $k$ as described earlier during the first iteration of our process, we may let $z_1 \geq \frac{ts}{n-1}$. Additionally, given $z_{i-1}$, there are $t - z_{i-1}$ elements of $T$ not accounted for by $K$, and hence on the $i$th iteration of our procedure, we may add an element to $K$ that accounts for $\frac{s(t-z_{i-1})}{n-1}$ new elements of $T$. Therefore, we obtain a recursive inequality for the number of elements in $T$ accounted for by $K$ after $i$ iterations of our procedure:

$$z_i \geq z_{i-1} + \frac{s(t - z_{i-1})}{n-1} = \frac{n-s-1}{n-1} z_{i-1} + \frac{st}{n-1},$$

which has a closed form of

$$z_i \geq t - t\left(\frac{n-s-1}{n-1}\right)^i.$$

Hence, after $i$ iterations of our recursive procedure, we calculate that there are at most $t\left(\frac{n-s-1}{n-1}\right)^i$ elements of $T$ not accounted for by $K$. As soon as the number of elements in $T$ not accounted for by $K$ is at most $c\sqrt{n}$, we may pair the remaining elements of $T$ as in the proof of Proposition 2.3.1. Therefore, the recursive method we have described will run $i$ times, where $i$ is the smallest integer such that $t\left(\frac{n-s-1}{n-1}\right)^i \leq c\sqrt{n}$. We thus may calculate that

$$i = \left\lceil \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} \right\rceil$$

and hence after the recursive method runs $i$ times, at most $c\sqrt{n}$ elements of $T$ will be left unaccounted for by $K$. At this point, the remaining $c\sqrt{n}$ unaccounted elements of $T$ may be paired into sums, as in the method of Theorem 2.3.1, and each such sum roughly accounts for 2 elements of $T$. We may put these sums into $K$, at which point the elements of $K$ altogether account for all of $T$. In total, our count shows that our set $K$ needs roughly at most

$$\left\lceil \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} \right\rceil + \frac{c\sqrt{n}}{2}$$

elements. This counting method gives us an intuition with which we define the function $g$. The extra additive constant of $g$ is included for technical reasons that will become clear later.

We easily see that our function $g$ is defined on all values in $\mathcal{D} \setminus B$ and bounded below by 2. In the following lemmas, we will bound $g(n,s,t)$ above, and we will show that $g$ and $h$ satisfy conditions (2) and (3) of Lemma 2.2.8.

**Lemma 2.3.5.** *Let $d > 0$. If $d \geq \frac{1}{ce} + \frac{c}{2}$, then $g(n,s,t) \leq d\sqrt{n} + \frac{7}{2}$.*

Proof. We consider two cases :

(1) If $t \leq c\sqrt{n}$, then

$$g(n,s,t) = \left\lceil \frac{t+3}{2} \right\rceil \leq \left\lceil \frac{c\sqrt{n}+3}{2} \right\rceil \leq \frac{c}{2}\sqrt{n} + \frac{5}{2} < d\sqrt{n} + \frac{7}{2}.$$

(2) If $t > c\sqrt{n}$, then we first note that $g(n,s,t) \leq g(n,s,s)$. We wish to find $\alpha$ such that $\frac{\log \frac{s}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} \leq \alpha\sqrt{n}$ for all real values $n \geq 3$ and $1 \leq s \leq n-2$. This inequality can be rewritten as

$$r_{\alpha,c}(n,s) := \frac{c\sqrt{n}}{s}\left(\frac{n-1}{n-s-1}\right)^{\alpha\sqrt{n}} \geq 1. \qquad (*)$$

One calculates that the derivative relative to $s$ is

$$\frac{\partial r_{\alpha,c}}{\partial s} = -\frac{c\sqrt{n}\left(\frac{n-1}{n-s-1}\right)^{\alpha\sqrt{n}+1}(-\alpha\sqrt{n}s+n-s-1)}{(n-1)s^2}.$$

By examining the sign of this derivative, we see that $r_{\alpha,c}(n,s)$ achieves a minimum when $s$ has a value $s^* = \frac{n-1}{\alpha\sqrt{n}+1}$. Therefore, it will suffice to choose a value $\alpha$ such that the inequality $(*)$ holds when $s$ is replaced by $s^*$. We will show that the choice $\alpha = \frac{1}{ce}$ works.

By substituting $s$ with $s^* = \frac{n-1}{\alpha\sqrt{n}+1}$, applying $\alpha = \frac{1}{ce}$, and doing some simplification, we find that

$$w_c(n) := r_{\frac{1}{ce},c}(n,s^*) = \frac{\left(1+\frac{ce}{\sqrt{n}}\right)^{\frac{\sqrt{n}}{ce}}\sqrt{n}\,(ce+\sqrt{n})}{e(n-1)}.$$

In order to show that the inequality $(*)$ holds with $\alpha = \frac{1}{ce}$, it will be enough to show that $w_c(n) \geq 1$.

We will apply the inequality $(1+\frac{x}{y})^y > e^{\frac{xy}{x+y}}$ (for $x,y > 0$) [**44**] [**45**, Section 5.3] with $x = \frac{1}{\sqrt{n}}$, $y = \frac{1}{ce}$, along with the inequality $\frac{\sqrt{n}}{n-1} > \frac{1}{\sqrt{n}}$. With these two inequalities, we find

$$w_c(n) = \frac{\left(1+\frac{ce}{\sqrt{n}}\right)^{\frac{\sqrt{n}}{ce}}\sqrt{n}\,(ce+\sqrt{n})}{e(n-1)} > \frac{e^{\frac{\sqrt{n}}{ce+\sqrt{n}}}\sqrt{n}\,(ce+\sqrt{n})}{e(n-1)} > \frac{e^{\frac{-ce}{ce+\sqrt{n}}}(ce+\sqrt{n})}{\sqrt{n}} =: z_c(n).$$

Furthermore, one calculates that

$$z_c'(n) = -\frac{c^2e^{2-\frac{ce}{ce+\sqrt{n}}}}{2\left(cen^{3/2}+n^2\right)}$$

which is always negative, and that

$$\lim_{n\to\infty} z_c(n) = 1.$$

Therefore, $w_c(n) > z_c(n) > 1$, which confirms that $\frac{\log\frac{s}{c\sqrt{n}}}{\log\frac{n-1}{n-s-1}} \leq \alpha\sqrt{n}$ when $\alpha = \frac{1}{ce}$, for all real values $c > 0$, $n \geq 3$, $1 \leq s \leq n-2$. Thus, we have

$$g(n,s,t) = \frac{\log\frac{t}{c\sqrt{n}}}{\log\frac{n-1}{n-s-1}} + \frac{c\sqrt{n}}{2} + \frac{7}{2} \leq \left(\frac{1}{ce}+\frac{c}{2}\right)\sqrt{n} + \frac{7}{2} \leq d\sqrt{n} + \frac{7}{2}.$$

□

**Lemma 2.3.6.** *Let $d > 0$. If $\frac{c}{2} \geq \frac{d}{\sqrt{2}}$ and $d \geq \frac{1}{ce} + \frac{c}{2}$, then $g$ respects condition (2) of Lemma 2.2.8 for $(\mathcal{G}_u, c_u)$.*

Proof. Consider a choice of $d > 0$ such that $\frac{c}{2} \geq \frac{d}{\sqrt{2}}$ and $d \geq \frac{1}{ce} + \frac{c}{2}$. Let $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$ and $(n',s',t') \in \mathcal{D} \setminus \mathcal{B}$ such that $n' \leq \frac{n}{2}$, $s' \leq s$ and $t' \leq t$. We consider two cases:

(1) If $t \leq c\sqrt{n}$, then

$$g(n,s,t) = \left\lceil \frac{t+3}{2} \right\rceil \geq \left\lceil \frac{t'+3}{2} \right\rceil \geq c_u(n',s',t'),$$

using a result from the proof of Proposition 2.3.1.

(2) If $t > c\sqrt{n}$, then by the previous lemma and our hypotheses on $c$ and $d$,

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + \frac{c\sqrt{n}}{2} + \frac{7}{2} > \frac{c\sqrt{n}}{2} + \frac{7}{2} \geq d\frac{\sqrt{n}}{\sqrt{2}} + \frac{7}{2} \geq d\sqrt{n'} + \frac{7}{2} \geq g(n',s',t').$$

□

**Lemma 2.3.7.** *The functions $g$ and $h$ respect condition (3) of Lemma 2.2.8 for $(\mathcal{G}_u, c_u)$.*

Proof. Let $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$ and $1 \leq t' \leq t - h(n,s,t)$.

We consider four cases :

(1) If $t \leq 2$ and $t \leq c\sqrt{n}$, then as $h(n,s,t) = 1$ we can see that the only case to consider is $t = 2$ and $t' = 1$. In this case, $g(n,s,t) = 3$ and $g(n,s,t') = 2$, so $g(n,s,t) = g(n,s,t')+1$.

(2) If $3 \leq t \leq c\sqrt{n}$, then $h(n,s,t) = 2$, and thus $t \geq t' + 2$. Then,

$$g(n,s,t) = \left\lceil \frac{t+3}{2} \right\rceil \geq \left\lceil \frac{t'+3}{2} \right\rceil = \left\lceil \frac{t'+1}{2} \right\rceil + 1 = g(n,s,t') + 1$$

(3) If $t > c\sqrt{n}$ and $t' \leq c\sqrt{n}$, then

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + \frac{c\sqrt{n}}{2} + \frac{7}{2} > \frac{c\sqrt{n}+3}{2} + 2 > \left\lceil \frac{c\sqrt{n}+3}{2} \right\rceil + 1 \geq \left\lceil \frac{t'+3}{2} \right\rceil + 1 \geq g(n,s,t') + 1$$

(4) If $t,t' > c\sqrt{n}$, we know from Lemma 2.3.3 that $t' \leq t - \frac{ts}{n-1} = t\left(\frac{n-s-1}{n-1}\right)$. Thus,

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + \frac{c\sqrt{n}}{2} + \frac{7}{2} \geq \frac{\log\left(\frac{t'}{c\sqrt{n}} \cdot \frac{n-1}{n-s-1}\right)}{\log \frac{n-1}{n-s-1}} + \frac{c\sqrt{n}}{2} + \frac{7}{2} = \frac{\log\left(\frac{t'}{c\sqrt{n}}\right)}{\log \frac{n-1}{n-s-1}} + 1 + \frac{c\sqrt{n}}{2} + \frac{7}{2} = g(n,s,t')+1.$$

In all cases, $g(n,s,t) \geq g(n,s,t')+1$, so condition (3) of Lemma 2.2.8 is satisfied for $(\mathcal{G}_u,c_u)$. □

We have shown that $g$ and $h$ satisfy the conditions of Lemma 2.2.8, so we are ready for our main result for undirected Cayley graphs on abelian groups.

**Theorem 2.3.8.** *The cop number of any undirected Cayley graph on an abelian group of $n$ elements is at most $\frac{1}{\sqrt{(\sqrt{2}-1)e}}\sqrt{n} + \frac{7}{2} \approx 0.9424\sqrt{n} + \frac{7}{2}$.*

Proof. Let $G$ be an abelian group on $n$ vertices generated by set $S \subseteq G$ (with $S = -S$ and $0_G \notin S$) of $s$ elements. If $(n,s,s) \in \mathcal{B}$, then the result follows directly from Lemma 2.2.7. Otherwise, we assume that $(n,s,s) \in \mathcal{D} \setminus B$.

We first find values $c$ and $d$ satisfying $\frac{c}{2} \geq \frac{d}{\sqrt{2}}$ and $d \geq \frac{1}{ce} + \frac{c}{2}$, which minimize $d$. A simple computation of such values $c$ and $d$ yields $c = \sqrt{\frac{2}{e(\sqrt{2}-1)}} \approx 1.33$ and $d = \sqrt{\frac{1}{e(\sqrt{2}-1)}} \approx 0.9424$.

With these chosen values of $c$ and $d$, the lemmas of this section show that our choices for $g$ and $h$ satisfy all three conditions of Lemma 2.2.8 for the case $(\mathcal{G}_u, c_u)$. Hence, by Lemmas 2.2.8 and 2.3.5, $c(\mathrm{Cay}(G,S)) \le c_u(n,s,s) \le g(n,s,s) \le d\sqrt{n} + \frac{7}{2}$. $\square$

Similarly to Proposition 2.3.1, the additive constant of Theorem 2.3.8 can be improved by 1 to $\frac{5}{2}$ with a more technical definition of boundary values. However, we do not feel that this slight improvement justifies the added technicalities.

We note that Theorem 2.3.8 not only proves that Meyniel's conjecture holds for undirected Cayley graphs on abelian groups (with a smaller multiplicative constant than in [**21**]), but it also proves that Meyniel's conjectured bound holds for these graphs with a coefficient of $\sqrt{n}$ smaller than 1. Indeed, Wagner has conjectured in [**67**] that the coefficient of $\sqrt{n}$ in Meyniel's conjecture should be 1, so Theorem 2.3.8 shows that abelian Cayley graphs satisfy both the conjectured upper bounds of Meyniel and Wagner.

In the next proposition, we show that we may obtain marginal improvements on the coefficient of $\sqrt{n}$ by considering the group structure of $G$. The proposition uses the fact that for a group $G$ of $n$ elements such that the smallest prime divisor of $n$ is a prime $p$, no element of prime order $q < p$ exists in $G$, and moreover, by Lagrange's Theorem and prime factorization, no such element exists in any subgroup or quotient group of $G$.

**Proposition 2.3.9.** *Let $G$ be an abelian group on $n$ elements, and let $S \subseteq G$ be a generating set of $G$ such that $S = -S$ and $0_G \notin S$. Let $p$ be the smallest prime factor of $n$.*

*(1) If $p = 3$, then $c(\mathrm{Cay}(G,S)) \le \sqrt{\frac{3}{2(\sqrt{3}-1)e}}\sqrt{n} + \frac{7}{2} \approx 0.8682\sqrt{n} + \frac{7}{2}$.*

*(2) If $p \ge 5$, then $c(\mathrm{Cay}(G,S)) \le \sqrt{\frac{2}{e}}\sqrt{n} + \frac{7}{2} \approx 0.8578\sqrt{n} + \frac{7}{2}$.*

Proof. (1) In condition (2) of Lemma 2.2.8, we require $n' \le \frac{n}{2}$ because of the bound $|G/\langle k\rangle| \le n/2$ for any element $k \in G$ with $k \ne 0$. However, if $n$ is odd, then we know that $|G/\langle k\rangle| \le n/3$, so we only need to require that $n' \le \frac{n}{3}$ in this condition. Indeed, if 2 does not divide $|G|$, then 2 does not divide $|G/\langle k\rangle|$, and induction may be used. Hence, we may relax the requirement $\frac{c}{2} \ge \frac{d}{\sqrt{2}}$ from Lemma 2.3.6 to $\frac{c}{2} \ge \frac{d}{\sqrt{3}}$. Then, minimizing $d$ with respect to $\frac{c}{2} \ge \frac{d}{\sqrt{3}}$ and $d \ge \frac{1}{ce} + \frac{c}{2}$ yields the solution $c = \sqrt{\frac{1}{(\sqrt{3}-1)e}} \approx 1.0025$ and $d = \sqrt{\frac{3}{2(\sqrt{3}-1)e}} \approx 0.8682$. The result then follows as in Theorem 2.3.8.

(2) As 2 and 3 do not divide $n$, in condition (2) of Lemma 2.2.8, we only need to require $n' \le \frac{n}{5}$. Hence, we may relax the requirement $\frac{c}{2} \ge \frac{d}{\sqrt{2}}$ from Lemma 2.3.6 to $\frac{c}{2} \ge \frac{d}{\sqrt{5}}$. Then, minimizing $d$ with respect to $\frac{c}{2} \ge \frac{d}{\sqrt{5}}$ and $d \ge \frac{1}{ce} + \frac{c}{2}$ yields the solution $c = d = \sqrt{\frac{2}{e}} \approx 0.8578$. The result then follows as in Theorems 2.3.8.

$\square$

We note than no further improvement based on $p$ is possible, as $c = d = \sqrt{\frac{2}{e}} \approx 0.8578$ is the optimal solution when ignoring the constraint $\frac{c}{2} \geq \frac{d}{\sqrt{p}}$. This solution always respects the constraint $\frac{c}{2} \geq \frac{d}{\sqrt{p}}$ when $p \geq 5$, as in those cases $c = d$ and $\sqrt{p} > 2$.

## 2.4. Upper bound for directed Cayley graphs

In this section, we consider the game of cops and robbers on directed abelian Cayley graphs. As we consider directed graphs in this section, whenever we have an abelian group $G$ generated by a set $S$, we no longer require that $S = -S$.

We will show that the approach we have outlined in Lemma 2.2.8 gives us an upper bound of $\sqrt{\frac{2}{(\sqrt{2}-1)e}}\sqrt{n} + 2 \approx 1.3328\sqrt{n} + 2$ on the cop number of directed abelian Cayley graphs of $n$ vertices. In other words, we will show that Meyniel's conjecture still holds for abelian Cayley digraphs, albeit with a worse coefficient than that of Theorem 2.3.8. Our general approach in this section will be very similar to that of Section 2.3. We will define functions $g$ and $h$ that satisfy Lemma 2.2.8 for $(\mathcal{G}_d,c_d)$ and such that $g(n,s,s)$ is not too large. Note that the functions $g$ and $h$ that we will define in this section are not the same as the functions $g$ and $h$ from the previous section. As this section follows the same approach as Section 2.3, our presentation will be terser.

In the following proposition, we use Lemma 2.2.8 with $(\mathcal{G}_d, c_d)$ to establish a directed version of Theorem 2.2.2. The following proposition appears in [**35**], but just like Proposition 2.3.1, we include the proposition as an instructive example of how to apply Lemma 2.2.8 with $(\mathcal{G}_d, c_d)$. Furthermore, we will need a result from the proof of the following proposition to prove the main result of this section.

**Proposition 2.4.1.** [**35**] *If* $\Gamma$ *is a Cayley digraph on an abelian group with generating set* $S$ *such that* $0_G \notin S$*, then*

$$c(\Gamma) \leq |S| + 1.$$

Proof. We wish to define functions $g$ and $h$ taking values in $\mathcal{D}\backslash B$ that satisfy the conditions of Lemma 2.2.8 for $\mathcal{G}_d$ and $c_d$. We choose

$$g(n,s,t) = t + 1;$$

$$h(n,s,t) = 1.$$

We claim that $g$ and $h$ satisfy the conditions of Lemma 2.2.8 for $\mathcal{G}_d$ and $c_d$. We immediately notice that $h(n,s,t) > 0$ by definition, and for all $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, we have $t \geq 1$, and hence $g(n,s,t) \geq 2$.

We show that condition (1) of Lemma 2.2.8 is satisfied. Indeed, let $(G,S,T) \in \mathcal{G}_d$ with respective sizes $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$. For any $a \in T$, $a$ accounts for $a$ with respect to $S$, because there exists $0_G \in S \cup \{0_G\}$ satisfying $a - 0_G = a$. Therefore, $h$ satisfies condition (1) of Lemma 2.2.8.

Next, we show that condition (2) of Lemma 2.2.8 is satisfied. If $n' \leq n/2$, $s' \leq s$, and $t' \leq t$ are values such that $(n',s',t') \in \mathcal{D} \setminus \mathcal{B}$, then

$$g(n,s,t) = t + 1 \geq t' + 1 = g(n',s',t').$$

Finally, we show that condition (3) of Lemma 2.2.8 is satisfied. If $1 \leq t' \leq t - h(n,s,t)$, then

$$g(n,s,t) = t + 1 \geq t' + 2 = g(n,s,t') + 1.$$

Hence, as $g$ and $h$ satisfy the conditions of Lemma 2.2.8 for $\mathcal{G}_d$ and $c_d$, it follows that $c_d(n,s,t) \leq g(n,s,t)$ for all $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$. Therefore, for an abelian group $G$ of $n$ elements generated by a set $S \subseteq G$ of $s$ elements, if $\Gamma = \mathrm{Cay}(G,S)$, then one of the following holds: either $(n,s,s) \in \mathcal{B}$ and $c(\Gamma) \leq c_u(n,s,s) = 1 < s+1$, or $(n,s,s) \in \mathcal{D} \setminus \mathcal{B}$ and $c(\Gamma) \leq c_u(n,s,s) \leq g(n,s,s) = s + 1$. □

We define a function $h$ that satisfies condition (1) of Lemma 2.2.8 for $\mathcal{G}_d$ and $c_d$, and we will use this definition of $h$ throughout the entire section. The definition of $h$ contains a constant $c > 0$ whose value we will decide later.

**Definition 2.4.2.** We define the function $h : \mathcal{D} \setminus \mathcal{B} \to \mathbb{R}^{>0}$ by

$$h(n,s,t) = \begin{cases} 1 & t \leq c\sqrt{n} \\ \frac{ts}{n-1} & t > c\sqrt{n}. \end{cases}$$

**Lemma 2.4.3.** *The function $h$ satisfies condition (1) of Lemma 2.2.8 for $(\mathcal{G}_d, c_d)$.*

Proof. We must show that if $(G,S,T) \in \mathcal{C}_d$ is a triple with respective sizes $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$, then there exists an element $k \in S \cup \{0_G\}$ accounting for at least $h(n,s,t)$ elements of $T$ with respect to $S$. When $t \leq c\sqrt{n}$, the proof follows the method of Proposition 2.4.1. When $t > c\sqrt{n}$, the proof follows the method of Lemma 2.3.3. □

Next, we define our function $g$. Again, we will show that $g$ and $h$ satisfy the conditions of Lemma 2.2.8 and that $g(n,s,s)$ is not too large.

**Definition 2.4.4.** We define the function $g : \mathcal{D} \setminus \mathcal{B} \to \mathbb{R}^{\geq 2}$ by

$$g(n,s,t) = \begin{cases} t + 1 & t \leq c\sqrt{n} \\ \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + c\sqrt{n} + 2 & t > c\sqrt{n}. \end{cases}$$

The following three lemmas are analogues of Lemmas 2.3.5, 2.3.6 ,and 2.3.7.

**Lemma 2.4.5.** *Let $d > 0$. If $d \geq \frac{1}{ce} + c$, then $g(n,s,t) \leq d\sqrt{n} + 2$.*

PROOF. We consider two cases :

(1) If $t \leq c\sqrt{n}$, then

$$g(n,s,t) = t + 1 \leq c\sqrt{n} + 1 < d\sqrt{n} + 2.$$

(2) If $t > c\sqrt{n}$, then by the proof of Lemma 2.3.5, $\frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} \leq \frac{1}{ce}\sqrt{n}$ for all real values $n \geq 3$ and $1 \leq t \leq s \leq n-2$. Thus, we have

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + c\sqrt{n} + 2 \leq \left(\frac{1}{ce} + c\right)\sqrt{n} + 2 \leq d\sqrt{n} + 2.$$

□

**Lemma 2.4.6.** *Let $d > 0$. If $c \geq \frac{d}{\sqrt{2}}$ and $d \geq \frac{1}{ce} + c$, then $g$ respects condition (2) of Lemma 2.2.8.*

PROOF. Consider a choice of $d > 0$ such that $c \geq \frac{d}{\sqrt{2}}$ and $d \geq \frac{1}{ce} + c$. Let $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$ and $(n',s',t') \in \mathcal{D} \setminus \mathcal{B}$ such that $n' \leq \frac{n}{2}$, $s' \leq s$ and $t' \leq t$. We consider two cases:

(1) If $t \leq c\sqrt{n}$, then

$$g(n,s,t) = t + 1 \geq t' + 1 \geq c_u(n',s',t'),$$

using a result from the proof of Proposition 2.4.1.

(2) If $t > c\sqrt{n}$, then by the previous lemma and our hypotheses on $c$ and $d$,

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + c\sqrt{n} + 2 > c\sqrt{n} + 2 \geq d\frac{\sqrt{n}}{\sqrt{2}} + 2 \geq d\sqrt{n'} + 2 \geq g(n',s',t').$$

□

**Lemma 2.4.7.** *The functions $g$ and $h$ respect condition (3) of Lemma 2.2.8 for $(\mathcal{G}_d, c_d)$.*

PROOF. Let $(n,s,t) \in \mathcal{D} \setminus \mathcal{B}$ and $1 \leq t' \leq t - h(n,s,t)$.

We consider three cases :

(1) If $1 \leq t \leq c\sqrt{n}$, then $h(n,s,t) = 1$, and thus $t \geq t' + 1$. Then,

$$g(n,s,t) = t + 1 \geq t' + 2 = g(n,s,t') + 1.$$

(2) If $t > c\sqrt{n}$ and $t' \leq c\sqrt{n}$, then

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + c\sqrt{n} + 2 > c\sqrt{n} + 2 \geq t' + 2 = g(n,s,t') + 1.$$

(3) If $t,t' > c\sqrt{n}$, we know from Lemma 2.4.3 that $t' \le t - \frac{ts}{n-1} = t\left(\frac{n-s-1}{n-1}\right)$. Thus,

$$g(n,s,t) = \frac{\log \frac{t}{c\sqrt{n}}}{\log \frac{n-1}{n-s-1}} + c\sqrt{n} + 2 \ge \frac{\log\left(\frac{t'}{c\sqrt{n}} \cdot \frac{n-1}{n-s-1}\right)}{\log \frac{n-1}{n-s-1}} + c\sqrt{n} + 2 = \frac{\log\left(\frac{t'}{c\sqrt{n}}\right)}{\log \frac{n-1}{n-s-1}} + 1 + c\sqrt{n} + 2 = g(n,s,t') + 1.$$

In all cases, $g(n,s,t) \ge g(n,s,t')+1$, so condition (3) of Lemma 2.2.8 is satisfied for $(\mathcal{G}_d, c_d)$. ☐

**Theorem 2.4.8.** *The cop number of any directed Cayley graph on an abelian group of $n$ elements is at most $\sqrt{\frac{2}{(\sqrt{2}-1)e}}\sqrt{n} + 2 \approx 1.3328\sqrt{n} + 2$.*

PROOF. Let $G$ be an abelian group on $n$ vertices generated by set $S \subseteq G$ (satisfying $0_G \notin S$) of $s$ elements. If $(n,s,s) \in \mathcal{B}$, then the result follows directly from Lemma 2.2.7. Otherwise, we assume that $(n,s,s) \in \mathcal{D} \setminus B$.

We first find values $c$ and $d$ satisfying $c \ge \frac{d}{\sqrt{2}}$ and $d \ge \frac{1}{ce} + c$, which minimize $d$. A computation of such values $c$ and $d$ yields $c = \sqrt{\frac{1}{e(\sqrt{2}-1)}} \approx 0.9424$ and $d = \sqrt{\frac{2}{e(\sqrt{2}-1)}} \approx 1.3328$.

With these chosen values of $c$ and $d$, the lemmas of this section show that our choices for $g$ and $h$ satisfy all three conditions of Lemma 2.2.8 for the case $(\mathcal{G}_d, c_d)$. Hence, by Lemmas 2.2.8 and 2.4.5, $c(\text{Cay}(G,S)) \le c_d(n,s,s) \le g(n,s,s) \le d\sqrt{n} + 2$. ☐

Similarly to Proposition 2.3.9, we may obtain marginal improvements on the coefficient of $\sqrt{n}$ by considering the group structure of $G$.

**Proposition 2.4.9.** *Let $G$ be an abelian group on $n$ elements, and let $S \subseteq G$ be a generating set of $G$ such that $0_G \notin S$. Let $p$ be the smallest prime factor of $n$.*

*(1) If $p = 3$, then $c(G,S) \le \sqrt{\frac{3}{(\sqrt{3}-1)e}}\sqrt{n} + 2 \approx 1.2278\sqrt{n} + 2$.*

*(2) If $p \ge 5$, then $c(G,S) \le \frac{2}{\sqrt{e}}\sqrt{n} + 2 \approx 1.2131\sqrt{n} + 2$.*

PROOF. (1) As in the proof of Theorem 2.3.8, we only need to require that $n' \le \frac{n}{3}$ in condition (2) of Lemma 2.2.8. Hence, we may relax the requirement $c \ge \frac{d}{\sqrt{2}}$ from Lemma 2.4.6 to $c \ge \frac{d}{\sqrt{3}}$.
Then, minimizing $d$ with respect to $c \ge \frac{d}{\sqrt{3}}$ and $d \ge \frac{1}{ce} + c$ yields the solution $c = \frac{1}{\sqrt{(\sqrt{3}-1)e}} \approx 0.7089$ and $d = \sqrt{\frac{3}{(\sqrt{3}-1)e}} \approx 1.2278$. The result then follows as in Theorem 2.4.8.

(2) As 2 and 3 do not divide $n$, in condition (2) of Lemma 2.2.8, we only need to require $n' \le \frac{n}{5}$. Hence, we may relax the requirement $c \ge \frac{d}{\sqrt{2}}$ from Lemma 2.3.6 to $c \ge \frac{d}{\sqrt{5}}$. Then, minimizing $d$ with respect to $c \ge \frac{d}{\sqrt{5}}$ and $d \ge \frac{1}{ce} + c$ yields the solution $c = \frac{1}{\sqrt{e}} \approx 0.6065$ and $d = \frac{2}{\sqrt{e}} \approx 1.2131$. The result then follows as in Theorem 2.4.8.

☐

## 2.5. Constructions with cop number in $\Theta(\sqrt{n})$

In this section, we will give constructions for abelian Cayley graphs and digraphs on $n$ vertices with cop number in $\Theta(\sqrt{n})$. If Meyniel's conjecture is true, then for any graph $G$ on $n$ vertices, the greatest possible cop number of $G$ is in $\Theta(\sqrt{n})$. Therefore, for an infinite family $\mathcal{G}$ of graphs, if for each $n \geq 1$, every graph $G \in \mathcal{G}$ on $n$ vertices has a cop number in $\Theta(\sqrt{n})$, then we say that $\mathcal{G}$ is a *Meyniel extremal family.*

We will construct a Meyniel extremal family using undirected abelian Cayley graphs and a Meyniel extremal family using directed abelian Cayley graphs. These families will show that the upper bounds in Theorems 2.3.8 and 2.4.8 are best possible, up to a constant factor. Our constructions will be based on finite fields. We note that in [**36**], Hasiri and Shinkar use similar methods to construct Meyniel extremal families of undirected abelian Cayley graphs, and the largest cop number of a graph on $n$ vertices by their construction is $\sqrt{\frac{n}{5}}$. Our Meyniel extremal family of undirected abelian Cayley graphs will give a sharper lower bound and thus improve the results from [**36**].

In this section, when we consider an abelian group $G$ generated by a set $S$, we will assume that $0_G \in S$, as this will simplify our notation and our arguments. Then, we consider a "non-move" of a cop or robber to be equivalent to playing the move $0_G$. Hence, we will assume that on a given move, each cop or robber chooses a move $s \in S$ and plays $s$, and we will not give "non-moves" special treatment.

We will now define the abelian groups and generating sets used to construct our Meyniel extremal families. Let $p > 3$ be a prime, and let $G$ be the additive group $(\mathbb{Z}/p\mathbb{Z})^2$. Note that $G$ is in fact a field equipped with a multiplication operation. Let $S_1$ and $S_2$ be defined as follows:

$$S_1 = \{(x,x^3) : x \in \mathbb{Z}/p\mathbb{Z}\},$$
$$S_2 = \{(x,x^2) : x \in \mathbb{Z}/p\mathbb{Z}\}.$$

We note that our sets $S_1$ and $S_2$ appear as examples of Sidon subsets for certain finite abelian groups in a paper by Babai and Sós [**5**]. We will see that our proofs that these generating sets give Cayley graphs of high cop number will be similar to the original arguments from [**5**] showing that these sets are Sidon subsets.

It is straightforward to show that $S_1$ and $S_2$ are both generating sets of $G$, seen as a group. We note that $S_1$ is also closed under inverses, while $S_2$ is not closed under inverses in general. Therefore, we consider $\text{Cay}(G,S_1)$ to be an undirected abelian Cayley graph, and we consider $\text{Cay}(G,S_2)$ to be a directed abelian Cayley graph. We note that $|G| = p^2$. The next two theorems show that both $\text{Cay}(G,S_1)$ and $\text{Cay}(G,S_2)$ have a cop number in $\Theta(p)$, demonstrating that our constructions indeed give graphs and digraphs on $n$ vertices with cop number in $\Theta(\sqrt{n})$.

We note that the proofs of the following theorems use key ideas from Proposition 2 and the subsequent discussion of [**30**] and Proposition 2.1 of [**35**], specifically about the number of moves that a single cop can guard. In particular, we could shorten our proofs and refer directly to those results, but we nonetheless present the full proofs for the sake of completeness.

**Theorem 2.5.1.** *Let $G$, $S_1$, and $p$ be as in the construction above. Then the cop number of* $\mathrm{Cay}(G,S_1)$ *is exactly* $\lceil \frac{1}{2}p \rceil = \left\lceil \frac{1}{2}\sqrt{|G|} \right\rceil$.

Proof. We first give a lower bound for the cop number of $\mathrm{Cay}(G,S_1)$. Whenever a cop is able to capture the robber immediately after the robber plays a move $(x,x^3)$, we say that the cop guards the move $(x,x^3)$. We show that a single cop cannot simultaneously guard more than two robber moves. Let $v \in G$ be a vertex occupied by a cop $C$, and let $r \in G$ be the vertex occupied by the robber. If the robber is not yet caught, then $v - r = (a,b)$, for some elements $a$ and $b$ that are not both zero. If $C$ guards a move $(x,x^3) \in S_1$, then there must exist a move $(y, y^3) \in S_1$ by which $C$ can capture the robber in reply to $(x,x^3)$. It then follows that $(x,x^3) - (y,y^3) = (a,b)$. Thus $x$ and $y$ must satisfy

$$x - y = a$$

$$x^3 - y^3 = b.$$

By substitution, we obtain the equation

$$a^3 - 3a^2x + 3ax^2 = b.$$

We see that if $a \neq 0$, then the system of equations has at most two solutions; otherwise, $a = b = 0$. Therefore, for fixed elements $a$ and $b$ not both equal to 0, there exist at most two values $x$ for which a solution to the system of equations exists. Hence $C$ guards at most two robber moves $(x,x^3) \in S_1$.

The robber has a total number of moves equal to $|S_1| = p = \sqrt{|G|}$. If the total number of cops is less than $\frac{1}{2}p$, then the robber will always have some move that is not guarded by any cop. Then by naively moving to an unguarded vertex on each turn, the robber can evade capture forever. Hence the cop number of $\mathrm{Cay}(G,S_1)$ is at least $\frac{1}{2}p = \frac{1}{2}\sqrt{|G|}$. As cop number is an integer, the cop number of $\mathrm{Cay}(G,S_1)$ therefore is at least $\lceil \frac{1}{2}p \rceil$. It follows from Theorem 2.2.2 that the cop number of $\mathrm{Cay}(G,S_1)$ is exactly $\lceil \frac{1}{2}p \rceil$. $\square$

We now show an analoguous result for directed graphs.

**Theorem 2.5.2.** *Let $G$, $S_2$, and $p$ be as in the construction above. Then the cop number of the directed graph* $\mathrm{Cay}(G,S_2)$ *is equal to* $|S_2| = p = \sqrt{|G|}$.

Proof. We first give a lower bound for the cop number of Cay($G$,$S_2$). Whenever a cop is able to capture the robber immediately after the robber plays a move ($x$,$x^2$), we say that the cop guards the move ($x$,$x^2$). We show that a single cop cannot guard more than one robber move. Let $v \in G$ be a vertex occupied by a cop $C$, and let $r \in G$ be the vertex occupied by the robber. If the robber is not yet caught, then $v - r = (a,b)$, for some elements $a$ and $b$ that are not both zero. If $C$ guards a move ($x$,$x^2$), then there must exist a move $(y, y^2)$ by which $C$ can capture the robber in reply to ($x$,$x^2$). It then follows that $(x,x^2) - (y,y^2) = (a,b)$. Thus $x$ and $y$ must satisfy

$$x - y = a$$
$$x^2 - y^2 = b.$$

By substitution, we obtain the equation $a^2 - 2ax = b$, from which we see that whenever $a \neq 0$, $x$ is uniquely determined; otherwise $a = b = 0$. Therefore, for fixed elements $a$ and $b$ not both equal to 0, there exists exactly one value $x$ for which a solution to the system of equations exists. Hence the cop occupying $C$ guards at most one robber move $(x,x^2) \in S_2$.

The robber has a total number of moves equal to $|S_2| = p = \sqrt{|G|}$. If the total number of cops is less than $p$, then the robber will always have some move that is not guarded by any cop. Then by naively moving to an unguarded vertex on each turn, the robber can evade capture forever. Hence the cop number of Cay($G$,$S_2$) is at least $|S_2| = p = \sqrt{|G|}$. It follows from Theorem 2.4.1 that the cop number of Cay($G$,$S_2$) is exactly $p$. □

Our construction in Theorem 2.5.2 implies that if Meyniel's conjecture holds for strongly connected directed graphs, written as $c(\Gamma) \leq c\sqrt{n}$, then the coefficient must respect $c \geq 1$. Although our construction in Theorem 2.5.2 uses a digraph whose order is the square of a prime, by using a common argument based on the density of primes (c.f. [**54**, Corollary 4.2]), we may extend our construction to give a digraph with an order of any large integer $n$ and a cop number in $(1 - o(1))\sqrt{n}$. This will give us a Meyniel extremal family of digraphs. It is shown in [**8, 15, 43, 54**] that there exist graph and digraph families on $n$ vertices with cop number in $\Omega(\sqrt{n})$, but to the authors' knowledge, our multiplicative coefficient of $1 - o(1)$ is the largest of any digraph construction.

**Corollary 2.5.3.** *For $n$ sufficiently large, there exist a strongly connected directed graph on $n$ vertices with cop number at least $\sqrt{n - 2n^{0.7625}} \in (1 - o(1))\sqrt{n}$.*

Proof. We borrow a lemma from number theory which tells us that for $x$ sufficiently large, there exists a prime in the interval $[x - x^{0.525}, x]$ [**9**]. From this lemma it follows that for sufficiently large $x$, there exists a square of a prime in the interval $[x - 2x^{0.7625}, x]$.

For our construction, we let $n$ be sufficiently large, and we choose a prime number $p > 3$ with $p^2 \in [n - 2n^{0.7625}, n]$. We let $G = (\mathbb{Z}/p\mathbb{Z})^2$, and we let $S_2$ be as in Theorem 2.5.2.

We then attach a sufficiently long bidirectional path to one of the vertices of Cay($G$,$S_2$), which increases the number of vertices without changing the cop number. This gives us a strongly connected directed graph on $n$ vertices with cop number equal to $c(G,S_2) = p \geq \sqrt{n - 2n^{0.7625}} \in (1 - o(1))\sqrt{n}$. $\square$

By using a similar approach, the construction in Theorem 2.5.1 can be modified to give a Meyniel extremal family of undirected graphs on $n$ vertices with cop number in $(\frac{1}{2} - o(1))\sqrt{n}$. However, this lower bound is not best possible, as constructions from [**15**] and [**54**] show that there exist undirected graph families in which a graph on $n$ vertices has cop number in $(\frac{\sqrt{2}}{2} - o(1))\sqrt{n}$.

## 2.6. Further directions

We conjecture that the constructions given in Theorems 2.5.1 and 2.5.2 have greatest possible cop number in terms of $n$, up to an additive constant.

**Conjecture 2.6.1.** *The cop number of any undirected Cayley graph on an abelian group of $n$ elements is in $\frac{1}{2}\sqrt{n} + O(1)$.*

**Conjecture 2.6.2.** *The cop number of any directed Cayley graph on an abelian group of $n$ elements is in $\sqrt{n} + O(1)$.*

There are multiple possible avenues of improvement on the proofs of this article. One obvious improvement would be to improve our bounds on the number of robber moves that can be accounted for by one group element. In the explanation behind the choice of $g$, the inequality can be strengthened to be $z_i \geq z_{i-1} + \left\lceil \frac{s(t - z_{i-1})}{n-i} \right\rceil$, as $z_i$ is always an integer and as we can apply the pigeonhole argument only to choose elements which have not previously been chosen. Resolution of this recursion might suggest a better function.

We note that as $g$ and $h$ are defined over integers, the proofs of our upper bounds only depend on the sizes of $G$, $S$ and $T$. Another possible improvement would be to use other group properties of $G$, $S$, and $T$ to get better bounds on the number of robber moves that a single element $k \in G$ can account for, or to better characterize the structure of a quotient $G/\langle k \rangle$ in our inductive strategy.

## Acknowledgements

We thank Ladislav Stacho and Geňa Hahn for bringing this problem to our attention. We also thank Ladislav Stacho, Bojan Mohar, Geňa Hahn, and Ben Seamone for fruitful discussions and encouragement. We furthermore thank Matt Devos for his suggestion to consider Sidon subsets for constructing abelian Cayley graphs with cop number in $\Theta(\sqrt{n})$, Frank Ramamonjisoa for his idea of obtaining improved bounds by considering prime factorizations and for his help in organizing earlier versions of the article, as well as Julien Codsi, Alizée Gagnon and Simon St-Amant for helpful comments. Finally, we thank the referees for their comments, which have greatly improved the presentation of this article.

The authors are partially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds de Recherche du Québec – Nature et technologies (FRQNT).

# Chapitre 3

# Cops and robbers on $2K_2$-free graphs

par

Jérémie Turcotte[1]

([1]) Départment de mathématiques et de statistique, Université de Montréal, Montréal, Canada

Cet article a été soumis au journal Discrete Mathematics.

Résumé. Nous prouvons que le *cop number* de tout graphe $2K_2$-libre graph est au plus 2, résolvant une conjecture récente de Sivaraman et Testa.

**Mots clés :** Policiers-voleur, *Cop number*, Sous-graphes induits exclus, Graphes $2K_2$-libres

Abstract. We prove that the cop number of any $2K_2$-free graph is at most 2, solving a recent conjecture by Sivaraman and Testa.

**Keywords:** Cops and robbers, Cop number, Forbidden induced subgraphs, $2K_2$-free graphs

## 3.1. Introduction

Cops and robbers [**52, 57, 2**] is a turn-based game opposing a group of cops and a robber on some connected graph $G$. The cops' objective is to capture the robber, whereas the latter attempts to escape indefinitely. The possible positions during the game are the vertices of $G$, and when a cop or the robber is on some vertex $u$, its possible moves are staying on $u$ or moving to a vertex adjacent to $u$, that is moving along an edge. On the first turn of the game, starting with the cops, each player picks the vertex where it start the game from. The *cop number* $c(G)$ is the number of cops which is both sufficient and necessary for their victory [**2**]. We say that $G$ is $k$-cop-win if $c(G) = k$ and that $G$ is $k$-cop-lose if $c(G) > k$.

We define the graphs $P_t$, $K_t$ and $K_{t,r}$ as, respectively, the path on $t$ vertices, the complete graph on $t$ vertices and the complete bipartite graph with partitions of size $t$ and $r$. For any graph $G$, we define $rG$ as the graph composed of $r$ disjoint copies of $G$.

It is frequent in graph theory to consider excluding, or forbidding, some substructures in graphs, most notably induced subgraphs, subgraphs or minors. We will say that say a graph $G$ is $H$-free, $H$-subgraph-free or $H$-minor free if $G$ does not contain any induced subgraph, subgraph, or minor, respectively, which is isomorphic to $H$. One may similarly define graphs which exclude a collection of graphs as subgraphs or minors.

There has been a fair amount of research on cops and robbers in this direction. The first major general result of this type is the following.

**Theorem 3.1.1.** [**4**] *If $H$ is a graph, then there exists $M_H < \infty$ such that for any $H$-minor-free graph $G$ we have $c(G) \leq M_H$.*

We assume that $M_H$ is as small as possible; we note that this concept is noted as $\alpha(H)$ in [**4**]. With the existence of such a bound proved, one might also be interested in optimizing this value $M_H$ for specific choices of $H$. For instance, it is also proved in [**4**] that $M_{K_5} = 3$ and that $M_{K_{3,3}} = 3$, hence improving the result from [**2**] that planar graphs have cop number at most 3; Wagner's theorem [**66**] states that the class of planar graphs and the class of $\{K_5, K_{3,3}\}$-minor-free graphs coincide.

Similar results for $H$-subgraph-free and $H$-free graphs were found in [**42**].

**Theorem 3.1.2.** [**42**] *If $H$ is a graph and $S_H$ is such that for any $H$-subgraph-free graph $G$ we have $c(G) \leq S_H$, then $S_H < \infty$ if and only if every connected component of $H$ is a tree with at most 3 vertices of degree at most 1.*

**Theorem 3.1.3.** [**42**] *If $H$ is a graph and $I_H$ is such that for any $H$-free graph $G$ we have $c(G) \leq I_H$, then $I_H < \infty$ if and only if every connected component of $H$ is a path.*

Some families with multiple excluded induced subgraphs are discussed in [**50**].

We will be consider the problem of excluding one graph from being an induced subgraph, as in Theorem 3.1.3. Here again, we want to optimise the value $I_H$ from Theorem 3.1.3. The simplest and most interesting case is that of a single forbidden path as induced subgraph, for which the following bound has been proved.

**Theorem 3.1.4.** [**42**] *If $G$ is a connected $P_t$-free graph* ($t \geq 3$)*, then* $c(G) \leq t-2$.

In other words, we know that $I_{P_t} \leq t-2$. It has been conjectured that this bound can be improved by using one fewer cops.

**Conjecture 3.1.5.** [**63**] *If $G$ is a connected $P_t$-free graph* ($t \geq 5$)*, then* $c(G) \leq t-3$.

This conjecture appears to be fairly difficult. An argument for the case of $P_5$ is likely to generalize to the whole conjecture. It has been suggested by Seamone and Hosseini in private communication that possibly the best approach to proving this conjecture is first proving it for the proper subclass of $2K_2$-free graphs.

**Conjecture 3.1.6.** [**64**] *Let $G$ be a connected $2K_2$-free graph. Then* $c(G) \leq 2$.

Our objective is to prove this conjecture. Only a few properties of hypothetical $2K_2$-free 2-cop-lose graphs are known. For instance, they have diameter 2 and contain induced cycles of length 3, 4 and 5, as well as an induced subgraph isomorphic to the house graph (the complement to $P_5$); see [**47, 64**] for previous work on this conjecture. However, we will not be using these results in our proof.

## 3.2. Traps

We begin with some basic notation. We denote by $N(u)$ the neighbourhood of a vertex $u$ and by $N[u] = N(u) \cup \{u\}$ the closed neighbourhood of $u$. If $S \subseteq V(G)$, then $G-S$ denotes the subgraph of $G$ induced by $V(G) \setminus S$; if $S = \{x\}$, we write $G-x$ for $G-S$.

We can now introduce an important concept which will be central in our proof.

**Definition 3.2.1.** Let $G$ be a graph. A vertex $u \in V(G)$ is a *trap* if there exists $x_1,x_2 \in V(G)$ (not necessarily distinct) such that $x_1,x_2 \neq u$ and $N[u] \subseteq N[x_1] \cup N[x_2]$.

In other words, a vertex $u$ is a trap if we can find two vertices which dominate $u$ and all of its neighbours. We will say $u$ is trapped by $x_1$,$x_2$, or that $x_1$,$x_2$ trap $u$.

The purpose of this definition is that if the robber is on $u$ and the cops are on the vertices trapping $u$, then the robber cannot escape and will lose at the next turn. In fact, a trap is a generalization of the classical definition of a corner (also called an irreducible vertex) in the game with one cop, see [**52**]. We note that this concept coïncides with the concept of a 2-trap in [**67**] from which the terminology is inspired; the term *trapped* is also used in [**53, 67**]

We now define different types of traps.

**Definition 3.2.2.** Let $G$ be a graph. Let $u \in V(G)$ be a trap and $x_1,x_2 \in V(G)$ be a choice of vertices trapping $u$.

We say $u$ is a *type-I trap* if exactly one of $x_1,x_2$ is adjacent to $u$. We say $u$ is a *type-II trap* if both $x_1$ and $x_2$ are adjacent to $u$.

We say $u$ is a *connected trap* if $x_1,x_2$ are adjacent vertices (in particular they are distinct). We will say that $u$ is c-trapped by $x_1$ and $x_2$.

Note that a trap can be both of type-I and type-II, and both connected and not connected, as a vertex may be trapped in multiple ways, but every trap must at least one of type-I or type-II.

## 3.3. Finding connected traps

The structural properties of $2K_2$-free graphs have been studied in various papers, for example in [**25**]. In this section, we prove the existence of connected traps in such graphs.

We start with some well-known remarks about $2K_2$-free graphs, for which we omit the obvious proofs.

**Lemma 3.3.1.** *Let $G$ be a $2K_2$-free graph.*

*(a) Only one connected component of $G$ can contain edges.*
*(b) The diameter of any connected $2K_2$-free graph is at most 3.*
*(c) Any induced subgraph of $G$ is $2K_2$-free.*

The following reformulation of the $2K_2$-free property will be used later to simplify some arguments.

**Lemma 3.3.2.** *Let $G$ be a $2K_2$-free graph. Let $vw \in E(G)$ and $u \in V(G)$ such that $u$ is not a neighbour of $v,w$. Then every neighbour of $u$ is adjacent to $v$ or $w$ (or both).*

Proof. Suppose the contrary, that there exists a neighbour $x$ of $u$, but not of $v,w$. Then, the edges $ux,vw$ form a $2K_2$. □

This lemma also yields a direct proof that 3 cops can catch the robber on connected $2K_2$-free graphs: choose an edge and place a cop on each end of this edge, as noted in [**64**]. By the lemma, the robber, who must choose a starting vertex not adjacent to the cops, cannot move, and a third cop can go catch the robber.

We denote by $C_5$ a cycle of length 5.

**Lemma 3.3.3.** *Let $G$ be a connected $2K_2$-free graph and let $u \in V(G)$. If $G - N[u] \simeq C_5$, then $G$ contains a connected trap.*

PROOF. Denote $a_1,\dots,a_5$ the vertices of $G - N[u]$, such that $a_ia_{i+1} \in E(G)$ (working in modulo 5).

It is easily seen that any vertex $v \in N(u)$ must be adjacent to at least 3 vertices of the 5-cycle $G - N[u]$, by applying Lemma 3.3.2 for each edge $a_ia_{i+1}$.

If $v$ is adjacent to 3 or more consecutive vertices $(a_{i-1},a_i,a_{i+1})$ of $G - N[u]$, then $a_i$ is c-trapped by $u$ and $v$ (all vertices in $G$ are dominated by $u$ or $v$, except possibly for $a_{i+2},a_{i+3}$, to which $a_i$ is not adjacent), and we are done.

Thus, we may now consider that every vertex of $N(u)$ is adjacent to exactly 3 vertices of the five-cycle, only two of which are adjacent: if $v \in N(u)$, then $N(v) \setminus N[u] = \{a_i,a_{i+2},a_{i+3}\}$ for some $1 \leq i \leq 5$.

If $v_1,v_2 \in N(u)$ have the same neighbours in $G - N[u]$, then $v_1$ is c-trapped by $v_2$ and $u$, and we are done.

Hence, we may now consider that every vertex of $N(u)$ has a distinct neighbourhood in $G - N[u]$. We will denote the *possible* vertices of $N(u)$ as follows: $N(u) \subseteq \{b_1,\dots,b_5\}$, such that $b_i$ is adjacent to $a_i$, $a_{i+2}$ and $a_{i+3}$. If $b_i,b_{i+1} \in N(u)$, then $b_ib_{i+1}$ is an edge, as otherwise $b_ia_{i+2},b_{i+1}a_{(i+1)+3}$ would form an induced $2K_2$. This does not exclude that there may be other edges between the $b_i$'s.

Choose a vertex $b_i \in N(u)$. Then, $u$ is c-trapped by $b_i$ and $a_i$. Indeed, $b_i$ is adjacent to $b_{i+1}$ and $b_{i-1}$ (if they are in the graph), $a_i$ is adjacent to $b_{i+2}$ and $b_{i+3}$ (if they are in the graph), and $a_i$ and $b_i$ are adjacent. This concludes the proof. $\square$

We are now ready to prove the desired result.

**Proposition 3.3.4.** *If $G$ is a connected $2K_2$-free graph, then either*

*(1) $G \simeq K_1$;*
*(2) $G \simeq K_2$;*
*(3) $G \simeq C_5$; or*
*(4) $G$ contains a connected trap.*

PROOF. We proceed by induction. If $|V(G)| = 1,2$, this is trivially true. Suppose the statement is true by induction for connected $2K_2$-free graphs $G'$ such that $|V(G')| < |V(G)|$ and that $|V(G)| \geq 3$. Let $u$ be any vertex of $G$. Recall that $G - N[u]$ is $2K_2$-free, by Lemma 3.3.1($c$).

If $G - N[u]$ is empty, then $u$ dominates $G$. As $|V(G)| \geq 3$, the vertex $u$ has at least two (distinct) neighbours $x_1,x_2$. Then, $x_1$ is c-trapped by $u$ and $x_2$.

If $G - N[u]$ contains a connected component which is a single vertex $y$, then $y$ is c-trapped by $u$ and any neighbour of $y$ (which is necessarily in $N(u)$). Otherwise, $G - N[u]$ contains no isolated vertex and by Lemma 3.3.1(a), $G - N[u]$ is connected. Also, $G - N[u]$ contains more than one vertex.

If $G - N[u]$ is an edge $x_1x_2$: If $x_1$ and $x_2$ have a common neighbour $t$ in $N(u)$, then $x_1$ is c-trapped by $t$ and $u$. Otherwise, $x_1$ and $x_2$ have no common neighbour. Denote by $A$ the neighbours of $x_1$ in $N(u)$ and by $B$ the neighbours of $x_2$ in $N(u)$. By Lemma 3.3.2, $N(u) = A \cup B$. Without loss of generality, $|A| \geq |B|$. If $|A| = 1$ and $|B| = 0$, then $G$ is path of length 4, which contains a connected trap. If $|A| = |B| = 1$, then we either have that $G \simeq C_5$ (if the vertex in $A$ and the vertex in $B$ are not adjacent) or $G$ contains a connected trap (if the vertex of $A$ and the vertex of $B$ are adjacent, $u$ is a connected trap). Now consider that $|A| > 1$ and let $a_1$,$a_2 \in A$ be distinct vertices. As $a_1$ and $a_2$ are both adjacent to $x_1$ but not $x_2$, we have that $a_1$ is c-trapped by $a_2$ and $u$.

If $G - N[u]$ contains at least 3 vertices (and is connected): By the inductive hypothesis, $G - N[u]$ is either a $C_5$ or contains a connected trap. If $G - N[u] \simeq C_5$, then Lemma 3.3.3 yields that $G$ contains a connected trap. Otherwise, denote $v$ the vertex of $G - N[u]$ which is a connected trap, and $w_1$,$w_2$ the vertices trapping $v$. We know that $w_1$,$w_2$ dominate $v$ and all neighbours of $v$ in $G - N[u]$. As $w_1w_2 \in E(G)$, they also dominate all vertices in $N(u)$, by Lemma 3.3.2. Hence, $v$ is also a connected trap in $G$. □

## 3.4. A strategy

In this section, we bound the cop number of $2K_2$-free graphs by using traps to restate the problem in terms of the local structure of our graphs, similarly to the equivalence between cop-win and dismantable graphs, see [**52**].

In general, the fact that a graph contains a (or many) traps does not necessarily imply that the cops can bring the game to that position.

For example, it is shown in [**53**] that all planar graphs of order at most 19 contains a trap, but it is still open as to whether 2 cops can win on all planar graphs of order at most 19. Another example is that it is shown in [**67**] that all diameter 2 graphs of order $n$ contains a set of vertices of size at most $\sqrt{n}$ which dominates the neighbourhood of some other vertex (called a $\sqrt{n}$-trap), but it unknown whether the cop number of these graphs is bounded by $\sqrt{n}$ (it is proved to be bounded by $\sqrt{2n}$). In our case, we will show that containing a trap will give us meaningful information.

For the remainder of this section, we will denote by $\hat{G}$ a minimal (relative to the number of vertices) connected $2K_2$-free 2-cop-lose graph. We first need the following lemma.

**Lemma 3.4.1.** *For any $u \in V(\hat{G})$, the induced subgraph $\hat{G} - u$ is connected. Furthermore, the induced subgraph $\hat{G} - N[u]$ is non-empty, connected and contains no isolated vertex.*

Proof. Recall that any induced subgraph of $\hat{G}$ is $2K_2$-free, by Lemma 3.3.1($c$). If $\hat{G} - u$ is disconnected, then by Lemma 3.3.1($a$), there is a vertex $x$ isolated in $\hat{G} - u$. This implies that in $\hat{G}$, the only neighbour of $x$ is $u$. It is easily seen that removing a vertex of degree 1 does not change the cop number of a graph, which contradicts the minimality of $\hat{G}$, as $\hat{G} - x$ would be a connected $2K_2$-free 2-cop-lose graph on fewer vertices.

It is clear $\hat{G} - N[u]$ is non-empty, otherwise a single cop on $u$ would catch the robber instantly, contradicting that $\hat{G} - N[u]$ is 2-cop-lose.

Suppose there exists a vertex $x$ which is isolated in $\hat{G} - N[u]$: $x$ is such that all of its neighbours in $\hat{G}$ are in $N(u)$. As $\hat{G} - x$ is a connected $2K_2$-free graph on fewer vertices than $\hat{G}$, there exists a winning strategy for 2 cops on $\hat{G} - x$.

We define a strategy for 2 cops on $\hat{G}$ using the strategy on $\hat{G} - x$. We say the robber's shadow is on $u$ whenever the robber is actually on $x$, and for all other positions the robber's shadow is on the same vertex as the robber. Now, as $N(x) \subseteq N(u)$, any move the robber makes corresponds to a valid move for the robber's shadow on $\hat{G} - x$. The cops apply the strategy on $\hat{G} - x$ to catch the robber's shadow. At the end of this strategy, if the robber is not caught, then the robber is on $x$ and a cop is on $u$. This cop stays on $u$, and the robber on $x$ cannot move. The other cop may then go capture the robber. This is a well known argument, see Theorem 1 of [**52**] and Theorems 3.1 and 3.2 of [**10**] for more general versions. This contradicts that $\hat{G}$ is 2-cop-lose.

Thus, $\hat{G} - N[u]$ is connected, as it contains no isolated vertex (by Lemma 3.3.1($a$)). □

We are now ready for a lemma which shows we have great power in not only placing the cops, but also the robber.

**Lemma 3.4.2.** *If $u \in V(\hat{G})$ and $vw \in E(\hat{G})$ such that $u$ is not a neighbour of $v$,$w$, then, playing with two cops, there exists a strategy ensuring that the cops are on $v$,$w$ and the robber is on $u$ and cannot move.*

Proof. We first wish to force the robber to move to $u$. By Lemma 3.4.1, $\hat{G} - u$ is connected, and by Lemma 3.3.1($c$) that it is $2K_2$-free. Hence, by the minimality of $\hat{G}$, $\hat{G} - u$ is must have cop number at most 2. As long as the robber is not on $u$, the cops copy the strategy for $\hat{G} - u$ on $\hat{G}$. If the robber never moves to $u$, the robber will eventually be caught: the robber has no choice but to eventually move to $u$. Denote $x_1$ and $x_2$ the positions of the cops at that point, we know that $x_1$,$x_2 \notin N[u]$, as otherwise the cops could capture the robber one turn later, a contradiction as $\hat{G}$ is 2-cop-lose.

We now wish to bring the two cops to the ends of an edge in $\hat{G} - N[u]$, while keeping the robber on $u$. If $x_1 = x_2$, one of the cops moves to a neighbour of $x_1$ in $\hat{G} - N[u]$, which must exist as $\hat{G} - N[u]$ is not be a unique vertex (by Lemma 3.4.1). If $x_1x_2 \in E(\hat{G})$, then they are already in such a position. If $x_1$ and $x_2$ have a common neighbour in $\hat{G} - N[u]$ (recall that Lemma 3.4.1 shows that $\hat{G} - N[u]$ is connected), let us denote it $x$, we move the cop on $x_2$ to $x$. If not, by Lemma 3.4.1, $\hat{G} - N[u]$ is connected and, by Lemma 3.3.1(b), $x_1$ and $x_2$ are at distance 3 in $\hat{G} - N[u]$: there exists $x_1'$,$x_2'$ such that $x_1x_1'x_2'x_2$ is a path contained in $\hat{G} - N[u]$. We move the cop on $x_1$ to $x_1'$ and the cop on $x_2$ to $x_2'$.

Now that the cops are on adjacent vertices, both not in $N(u)$, then by Lemma 3.3.2, the robber cannot move.

We now wish to bring the cops to the edge $vw$, while keeping the robber on $u$. We will do so by never leaving $\hat{G} - N[u]$ and always keeping the cops on adjacent vertices, which guarantees that the robber will never be able to move. Suppose the cops are now on the edge $ab$. Let $P$ be a path completely contained in $\hat{G} - N[u]$ starting with the edge $ab$ and ending with the edge $vw$, which exists as $\hat{G} - N[u]$ is connected. The cops move along $P$ one behind the other. This concludes the proof. ☐

In section 3.2, we defined type-I and type-II traps. Using the strategy we developed in the last lemma, we will be able to exclude these from $\hat{G}$.

**Lemma 3.4.3.** *$\hat{G}$ does not contain a type-I trap.*

Proof. Suppose to the contrary that there exists a type-I trap $u$. We will define a strategy for 2 cops on $\hat{G}$.

Let $x_1$,$x_2$ be the vertices trapping $u$, with $x_1$ adjacent to $u$ and $x_2$ in $\hat{G} - N[u]$. Let $y$ be any neighbour of $x_2$ in $\hat{G} - N[u]$, which exists as $\hat{G} - N[u]$ contains no isolated vertex (by Lemma 3.4.1). Using Lemma 3.4.2, place the cops on $x_2$ and $y$, and the robber on $u$.

If $yx_1$ is an edge, then move the cop on $y$ to $x_1$ and keep the other cop on $x_2$. If $yx_1$ is not an edge, then $x_1x_2$ is an edge by Lemma 3.3.2. Move the cop on $x_2$ to $x_1$ and the cop on $y$ to $x_2$.

In both cases, the robber is now on $u$ with the cops on $x_1$,$x_2$: the robber is caught at the next move. This is a contradiction as $\hat{G}$ is 2-cop-lose. ☐

Before considering the case of type-II traps, we need the following proposition from [**25**]. We prove it here in order for this paper to be self-contained.

**Proposition 3.4.4.** [**25**] *If $G$ is a connected bipartite $2K_2$-free graph, then each colour class of $G$ contains a vertex adjacent to all vertices of the other colour class of $G$.*

Proof. Denote $A$,$B$ the colour classes of $G$. Choose $m \in A$ of maximum degree. Suppose there exists $b \in B$ such that $mb$ is not an edge. As $G$ is connected, there exists $a \in A$ such that $ab$ in an edge. Now, for every neighbour $x \in N(m)$ (necessarily, $x \in B$), we compare edges $ab$ and $mx$: the $2K_2$-free property yields that $ax$ is an edge. Thus, $|N(a)| \geq |N(m)|+1$, as $N(m) \subseteq N(a)$ and $b \in N(a) \setminus N(m)$, which contradicts that $m$ has maximum degree. $\square$

**Lemma 3.4.5.** *If $\hat{G}$ contains a type-II trap, then $\hat{G}$ contains a type-I trap.*

Proof. Let $x_1$,$x_2$ be the vertices trapping a vertex $u$ such that $x_1$ and $x_2$ are both adjacent to $u$. We can suppose $x_1$ and $x_2$ are distinct, as if $N[u] \subseteq N[x_1]$, then simply pick $x_2$ to be any other neighbour of $u$ (which must exist as otherwise $\hat{G}-x_1$ is disconnected, contradicting Lemma 3.4.1).

Suppose $y$ is a neighbour of $x_1$ in $\hat{G}-N[u]$, we wish to prove $y$ is adjacent to $x_2$. Suppose $y$ is not adjacent to $x_2$, then denote by $z$ any neighbour of $y$ in $\hat{G}-N[u]$, which exists as $\hat{G}-N[u]$ contains no isolated vertex (by Lemma 3.4.1). Then, $z$ must adjacent to $x_2$ by Lemma 3.3.2. Playing with 2 cops, place the cops on $y$ and $z$ and the robber on $u$ using Lemma 3.4.2. Then, move the cop on $y$ to $x_1$ and the cop on $z$ to $x_2$. The robber will be caught one turn later, which is a contradiction as $\hat{G}$ is 2-cop-lose. Thus, $y$ must be adjacent to $x_2$.

By applying this reasoning for every neighbour of $x_1$ and of $x_2$ in $\hat{G}-N[u]$, we find that every vertex of $\hat{G}-N[u]$ is either adjacent to both $x_1$ and $x_2$, or to neither. We can thus partition $V(\hat{G}) \setminus N[u]$ into the sets $A = \{v \in V(\hat{G}) \setminus N[u] : vx_1, vx_2 \in E(\hat{G})\}$ and $B = \{v \in V(\hat{G}) \setminus N[u] : vx_1, vx_2 \notin E(\hat{G})\}$.

If there is an edge between 2 vertices in $B$, comparing this edge with $ux_1$ yields an induced $2K_2$, and thus $B$ is a stable set. If there is an edge between two vertices in $A$, then, playing with 2 cops, place the cops on the ends of this edge and the robber on $u$, using Lemma 3.4.2, and then move the cops to $x_1$ and $x_2$, yielding a contradiction as $\hat{G}$ is 2-cop-lose. Thus, $\hat{G}-N[u]$ is a (connected, by Lemma 3.4.1) bipartite graph. Note that $B$ is non-empty as $A$ is a stable set and $\hat{G}-N[u]$ contains no isolated vertex.

By Proposition 3.4.4, there exists a vertex $b$ in $B$ adjacent to every vertex of $A$. Every neighbour of $x_1$ in $N[u]$ is (by definition) either $u$ or adjacent to $u$, and every neighbour of $x_1$ in $\hat{G}-N[u]$ is adjacent to $b$. Furthermore, $x_1b \notin E(\hat{G})$. Thus, $x_1$ is a type-I trap, trapped by $u$ and $b$. $\square$

We are now ready to prove Conjecture 3.1.5.

**Theorem 3.4.6.** *If $G$ is a connected $2K_2$-free graph, then $c(G) \leq 2$.*

Proof. Let $\hat{G}$ be a minimal counter-example. Lemmas 3.4.3 and 3.4.5 imply that $\hat{G}$ does not contain any trap, hence does not contain any connected trap. Thus, by Proposition 3.3.4, $\hat{G}$ is isomorphic to either $K_1$, $K_2$ or $C_5$, all of which have cop number at most 2. $\square$

The more general question of the cop number of $rK_2$-free graphs ($r \geq 2$) is raised in [**64**]. One notices that $2r-1$ cops can win, as noted in [**64**], by a proof similar to proving 3 cops can win on $2K_2$-free graphs. Having improved by 1 the bound on the cop number of $2K_2$-free graphs, we can also improve by 1 the bound on the cop number of $rK_2$-free graphs with an analogous argument.

**Corollary 3.4.7.** *If $G$ is a connected $rK_2$-free graph, $r \geq 2$, then $c(G) \leq 2r-2$.*

We also note that the same idea allows us to modify theorem 4 of [**47**] by removing the condition that at least one index is at least 3 if at least two of the indices are 2.

## 3.5. Further directions

It remains to be seen if it is possible to further improve the bound on the cop number of $rK_2$-free ($r > 2$) graphs or if this bound is tight. It would also be interesting to see if the approach used to prove Theorem 3.4.6 can be used to improve the bound on the cop number of $P_5$-free graphs, and even possibly to prove Conjecture 3.1.5.

## Acknowledgments

I would thank Seyyed Aliasghar Hosseini and Ben Seamone for introducing me to this problem and for their helpful comments and suggestions, including the idea of adding Corollary 3.4.7. I also thank Seyyed Aliasghar Hosseini for helping start the redaction of previous versions. I am very grateful to Samuel Yvon for helpful discussions, which have inspired me greatly. I thank to Geňa Hahn and Simon St-Amant for their comments on the paper. Finally, I would like to express my gratitude to Alizée Gagnon for noticing an oversight in a previous version of this proof.

# Chapitre 4

# 4-cop-win graphs have at least 19 vertices

par

Jérémie Turcotte[1] et Samuel Yvon[2]

([1]) Départment de mathématiques et de statistique, Université de Montréal, Montréal, Canada

([2]) Département d'informatique et de recherche opérationnelle, Université de Montréal, Montréal, Canada

Cet article a été soumis au journal Discrete Applied Mathematics.

Tous les coauteurs ont contribué de façon importante à l'article.

Résumé. Nous montrons que le *cop number* de tout graphe sur 18 sommets ou moins est au plus 3. Ceci résout une question posée par Andreae en 1986, ainsi que par Baird et al. en 2011. Nous trouvons aussi tous les graphes 3-policiers-gagnants sur 11 sommets, réduisons les graphes possiblement 4-policiers-gangnats sur 19 sommets et faisons progresser la recherche de l'ordre minimum des graphes planaires 3-policiers-gagnants.

**Mots clés :** Policiers-voleur, *Cop number*, Problèmes extrémaux, Construction de graphes, Preuve assistée par l'ordinateur

Abstract. We show that the cop number of any graph on 18 or fewer vertices is at most 3. This answers a question posed by Andreae in 1986, as well as by Baird et al. in 2011. We also find all 3-cop-win graphs on 11 vertices, narrow down the possible 4-cop-win graphs on 19 vertices and make some progress on finding the minimum order of 3-cop-win planar graphs.

**Keywords:** Cops and robbers, Cop number, Extremal problems, Graph construction, Computer-assisted proof

## 4.1. Introduction

The game of cops and robbers was first defined by Quilliot in [**57**] and Nowakowski and Winkler in [**52**]. Playing on a connected, undirected and finite graph, the cops try to catch a robber. The cops and the robber alternate turns. On the first turn, each cop selects a starting vertex followed by the robber. At each subsequent turn, each player may either stay put or move to an adjacent vertex. If at any point the robber and one of the cops share a vertex, the cops win. The robber wins if it has a strategy ensuring it is never caught by the cops. At all times, the positions of the cops and of the robber are known by all. Furthermore, the cops may coordinate their strategies, and are allowed to share vertices.

For a connected graph $G$, we denote by $c(G)$ the minimal number of cops which can always catch the robber on $G$. Introduced by Aigner and Fromme in [**2**], $c(G)$ is called the *cop number* of $G$. If $c(G) = k$, we say $G$ is $k$-cop-win.

The cop number has been the main focus of most articles on cops and robbers, but other parameters, such as the capture time, are also studied. See [**14**] for a quick overview of this field or [**18**] for a more in-depth introduction. A multitude of variants of this game have been considered in recent years, but in this paper we only study the classical version.

While there has been a significant amount of research on the cop number of graphs, often on specific classes of graphs, there are still surprisingly many elementary open questions. We consider here the problem of finding the minimum order of $k$-cop-win graphs, more specifically for $k = 4$. This question was first posed by Andreae in [**4**]. It is also raised, seemingly independently, by Baird et al. in [**7**].

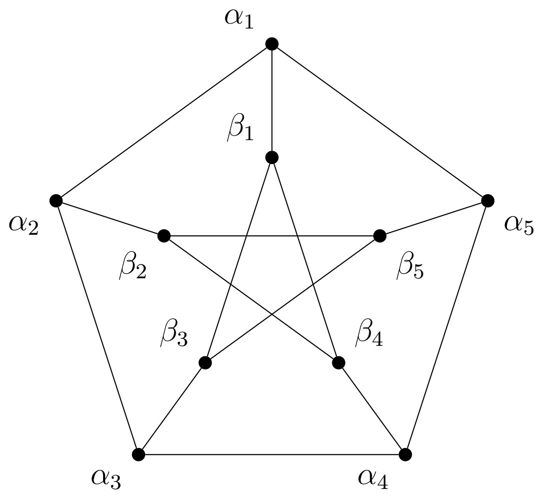
The case of $k = 3$ has already been solved. Andreae claims without proof in [**4**] that the Petersen graph (see Figure 4.1) is the unique smallest 3-cop-win graph. This statement is proved in [**7**].

We denote by $V(G)$ and $E(G)$, respectively, the set of vertices and of edges of $G$. We denote by $M_k$ the minimum order of $k$-cop-win graphs; formally, $M_k = \min\{|V(G)| : G \text{ connected graph}, c(G) = k\}$. Interestingly, Hosseini proved in [**38**] that $M_k < M_{k+1}$, confirming the intuition that if one scans all graphs by increasing order, one cannot find a $(k+1)$-cop-win graph before finding a $k$-cop-win graph.

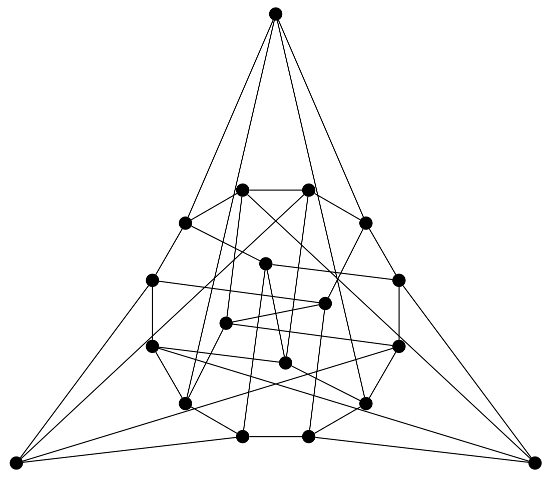
The problem of finding the minimum order of 4-cop-win graphs has also received some interest. Hosseini proved in [**39**] that $M_4 \geq 16$, and that such a minimal graph is 3-connected, provided it does not contain a vertex of degree 2. The problem is also referenced in [**14**].

It is suggested in [**4, 7**] that the value of $M_4$ might be 19. Indeed, the smallest known 4-cop-win graph is the Robertson graph (see Figure 4.1). This graph was first discovered by Robertson in [**60**] as the smallest 4-regular graph with girth 5. A $(d,g)$-cage is a regular graph of degree $d$ and girth $g$ of minimum order. For instance, the Petersen graph is the unique (3,5)-cage and the Robertson graph is the unique (4,5)-cage.

It has been proved in [**2**] that graphs with girth at least 5 have a cop number of a least their minimum degree. This result has since been generalized in [**29**], and recently in [**22**]. One easily deduces that the cop number of the Robertson graph is therefore at least 4. It is easily seen in the figure that placing a cop on each of the three exterior vertices only leaves 4 unprotected vertices, which form independent edges. A last cop may then easily capture the robber, thus the Robertson graph is 4-cop-win (this argument appears in [**11**]). It is suggested in [**7**] that the smallest $d$-cop-win graphs might be the $(d,5)$-cages for $d \geq 3$.



**(a)** The Petersen graph



**(b)** The Robertson graph[1]

**Fig. 4.1.** Some small $(d,5)$-cage graphs

The main result of this article is to confirm that $M_4 = 19$. Although we are not able to prove a complete uniqueness result, we narrow down the possible 4-cop-win graphs on 19 vertices.

[1]Computer-generated drawing [**41**].

Although our proof is not directly based on those in [**7**] and [**39**], there are certainly some common elements. In particular, we also break down the problem by maximum degree and find properties of potential 4-cop-win graphs by constructing explicit strategies.

While we are able to obtain many interesting results formally, this article makes extensive use of computational methods to verify the remaining cases. All of the code and data produced in the writing of this article is available online at [**65**]. This includes not only the final results, but the graphs we generate in the intermediate algorithms, precise counts of the number of graphs we generate at every step in these algorithms, and the time required for almost all computations. All of the computations are split up in small parts to facilitate verification. At various points in this article, we will also discuss possible improvements and alternative computational approaches.

## 4.2. Notation and previous results

In this section, we introduce most of the notation used in the article. We also cite previously known results that will be useful.

When considering a graph $G$, we will respectively denote by $n(G)$, $d_G(u)$, $\delta(G)$ and $\Delta(G)$ the number of vertices of $G$, the degree of a vertex $u$ in $G$, the minimum degree of $G$ and the maximum degree of $G$. If $u$ is a vertex of $G$, $N_G(u)$ will denote the (open) neighbourhood of $u$ and $N_G[u] = N_G(u) \cup \{u\}$ will denote the closed neighbourhood of $u$. For these symbols, we will usually omit the $G$ when the choice of graph is easily deduced.

For $S \subseteq V(G)$, $S^c$ will denote the complement (in $V(G)$) of $S$, $\langle S \rangle$ will denote the subgraph of $G$ induced by $S$, and $G - S$ will denote $\langle S^c \rangle$. When $S = \{x\}$, we will use the notation $G - x$ instead of $G - S$. Similarly, if $H$ is a subgraph of $G$, then $G - H$ will denote $G - V(H)$.

We will use the symbol $\simeq$ to denote graph isomorphism.

Denote by $\mathcal{P}_0$ the Petersen graph, as seen in Figure 4.1. As $\mathcal{P}_0$ is 3-regular with girth 5, we get that $c(\mathcal{P}_0) \geq 3$, and as it contains a dominating set of size 3, we know that $c(G) = 3$. As stated in the introduction, the following theorem was stated by Andreae in [**4**] and proved by Baird et al., first by computer verification and then formally.

**Theorem 4.2.1.** [**4, 7**] *Let $G$ be a connected graph.*

*(1) If $n \leq 9$, then $c(G) \leq 2$.*

*(2) If $n = 10$, then $c(G) \leq 2$, unless $G$ is the Petersen graph.*

*In particular, $M_3 = 10$.*

The proof of the previous theorem makes use of the following lemma, which will also be useful to us.

**Lemma 4.2.2.** [**7**] *Let $G$ be a connected graph. If $\Delta \geq n-5$, then $c(G) \leq 2$.*

A simple, visual proof of this lemma is available in [**11**]. We now define a useful concept, which has been used many times to study the game of cops and robbers. The following is based on [**10**].

**Definition 4.2.3.** Let $G$ be a graph. If $H$ is an induced subgraph of $G$, we say $H$ is a *retract* of $G$ if there exists a mapping $f : V(G) \to V(H)$ such that:

(1) If $xy \in E(G)$, then $f(x)f(y) \in E(H)$ or $f(x) = f(y)$.
(2) $f|_{V(H)} : V(H) \to V(H)$ is the identity mapping.

Such a mapping $f$ is called a *retraction*.

This definition formalizes the intuitive idea that $G$ can be "folded" onto $H$, where each edge must either be sent onto an edge or onto a vertex. Those familiar with graph homomorphisms will notice that condition (1) states that $f$ is a homomorphism from $G$ to $H$, if we consider $H$ to be reflexive (that is, if we add a loop at each vertex of $H$). This reflexivity is a consequence of allowing the cops and the robber to stay on a vertex at their turn, implying a loop on each vertex. The concept of retracts has been central in the study of the game of cops and robbers, appearing also [**52**].

If $G$ is disconnected, denote $G_1, \ldots, G_t$ the connected components of $G$. By extension, we may define the cop number of a disconnected graph by $c(G) = \max_{1 \leq i \leq t} c(G_i)$.

These definitions allow us to state the following result of Berarduci and Intriglia, which we will use many times to reduce the number of cases we need to consider.

**Theorem 4.2.4.** [**10**] *If $G$ is a connected graph and $H$ is a retract of $G$, then*

$$c(H) \leq c(G) \leq \max\{c(H), c(G-H)+1\}.$$

A specific case of this theorem is the following, which is a reformulation of a corollary in [**10**], that will often be easier to use.

**Corollary 4.2.5.** *If $G$ is a connected graph, $u$ is a vertex of $G$ and $K$ is a union of some connected components of $G - N[u]$, then*

$$c(G-K) \leq c(G) \leq \max\{c(G-K), c(K)+1\}.$$

*In particular, if $c(K) \leq k-1$, then $c(G) \leq k$ if and only if $c(G-K) \leq k$.*

Proof. It is easy to verify that $f : V(G) \to V(G-K)$ defined by

$$f(x) = \begin{cases} u & \text{if } x \in V(K) \\ x & \text{otherwise} \end{cases}$$

is a retraction. It is only left to apply Theorem 4.2.4 to $H = G - K$. □

One trivial consequence of this corollary is that if the cop number of every component of $G - N[u]$ is at most $k-1$, then $c(G) \le k$. One can also see this directly by leaving a fixed cop on $u$ and playing with $k-1$ cops on the connected component of $G - N[u]$ in which the robber is located.

We then easily get the following result, which is implicit in [**39**].

**Corollary 4.2.6.** *If $G$ is a connected graph and $\Delta > n - 11$, then $c(G) \le 3$.*

Proof. If $\Delta > n - 11$ and $u$ is a vertex of maximum degree, then $|V(G - N[u])| < 10$. By Theorem 4.2.1, every connected component of $G - N[u]$ has cop number at most 2. The last remark yields the result. □

Finally, we recall a well known concept in the study of the game of cops and robbers.

**Definition 4.2.7.** Let $x$,$u$ be distinct vertices of $G$. If $N(x) \subseteq N[u]$, we say $x$ is *cornered* by $u$ or that $x$ is a *corner*.

We note that this is a slight variation on the classical notion of a corner (or irreducible vertex), as it normally requires $ux$ to be an edge, see [**52**]. We may now get the following well-known result as a further simplification of Corollary 4.2.5.

**Corollary 4.2.8.** *Let $G$ be a connected graph and $x$ be a corner of $G$. If $c(G-x) \ge 2$, then $c(G) = c(G-x)$. If $c(G-x) = 1$, then $c(G) \in \{1,2\}$.*

Proof. If $x$ is cornered by $u$, notice that $x$ is isolated in $G - N[u]$. Applying Corollary 4.2.5 with $K = \{x\}$ yields the result. □

## 4.3. Computational results for small 3-cop-win graphs

In this section, we find some 3-cop-win graphs on at most 14 vertices respecting some degree conditions. These results will be useful in the following sections.

We will do this by computing the cop number of every graph of the desired orders and degrees. Graph generation in this section and in Section 4.5 is done using the `geng` function provided with the nauty/Traces package (version 26r12) [**51**].

The algorithm to compute the cop number is similar to that proposed, in particular, in [**16, 26, 59**], which we have implemented for cop numbers 1,2,3 in the Julia language

[**12, 23**]. For a given $k$ (which will be between 1 and 3 in our case), the algorithm determines whether $c(G) \leq k$ or $c(G) > k$.

To test the validity of our implementation, we have compared the results for the cop number of connected graphs up to 10 vertices to those in [**7**]. Following a small discrepancy between the counts, our tallies of cop-win graphs were also verified to be correct by implementing a dismantling algorithm [**52**] and by comparing with the implementation at [**1**]. To test our code for higher cop numbers, it was also run on some cage graphs which we know 3 cops lose. Based on the results of these tests, we are confident in the correctness of our implementation.

We first define a variant of the Petersen graph.

**Definition 4.3.1.** We say a connected graph $G$ is a *cornered Petersen graph* if $G$ contains a corner $x$ such that $G - x \simeq \mathcal{P}_0$. There are 6 such graphs up to isomorphism. We denote them $\mathcal{P}_i$, $i = 1, \ldots, 6$, as seen in Figure 4.2.

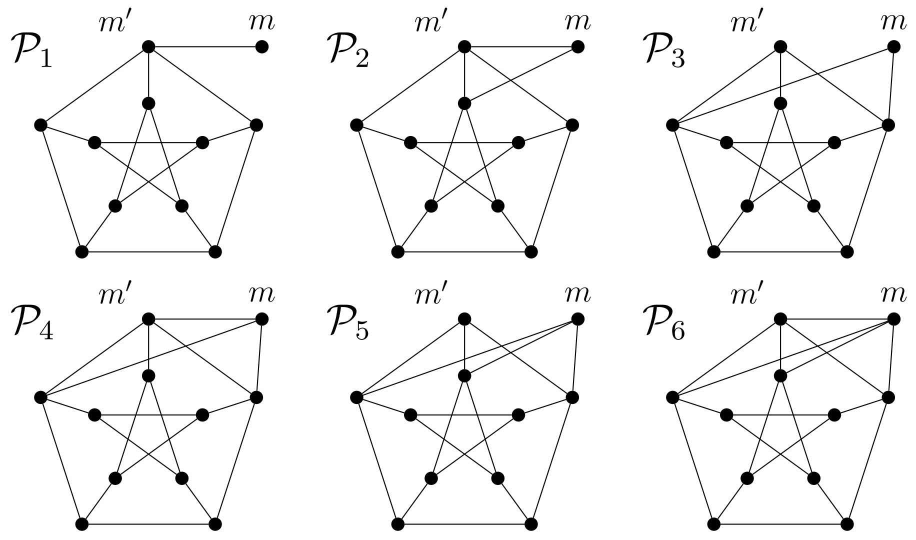


**Fig. 4.2.** The cornered Petersen graphs

We now solve a question raised in [**7**], classifying the 3-cop-win graphs on 11 vertices, albeit computationally.

**Proposition 4.3.2.** *If $G$ is a connected graph such that $n = 11$, then $c(G) = 3$ if and only if $G \simeq \mathcal{P}_i$ for some $1 \leq i \leq 6$. Otherwise, $c(G) \leq 2$.*

Proof. Firstly, it is clear by Theorem 4.2.1 and Corollary 4.2.8 that the cornered Petersen graphs are 3-cop-win.

We would like to show that these graphs are the only graphs on 11 vertices with cop number 3, and that all other graphs have cop number at most 2.

By Lemma 4.2.2, we may only consider graphs such that $\Delta \leq n-6=5$. We generate all graphs on 11 vertices such that $\Delta \leq 5$ and classify each graph according to its cop number. The results are presented in Table 4.1 (the counts are up to isomorphism). The 6 graphs found are the graphs $\mathcal{P}_i$ for $i=1,\ldots,6$, which concludes the proof. □

This is an interesting phenomenon: the 3-cop-win graphs on 11 vertices are all retracts of the unique 3-cop-win graph on 10 vertices. This behaviour does not occur for the 2-cop-win graphs: the minimum 2-cop-win graph is the 4-cycle, on which the 5-cycle does not retract. Although we will not have any answer for this question in this article, it would be interesting to know whether in general (even for 4-cop-win graphs only), the $k$-cop-win graphs on $M_k+1$ vertices can be retracted on $k$-cop-win graph(s) on $M_k$ vertices.

In Section 4.6, we will also need the following lemma.

**Lemma 4.3.3.** *There exist*

- 80 *connected graphs* $G$ *on* 12 *vertices with* $\Delta \leq 4$*,*
- 173 *connected graphs* $G$ *on* 12 *vertices with* $\Delta \leq 5$*,*
- 1105 *connected graphs* $G$ *on* 13 *vertices with* $\Delta \leq 4$*, and*
- 16523 *connected graphs* $G$ *on* 14 *vertices with* $\Delta \leq 4$

*such that* $c(G)=3$*. All other connected graphs considered with these orders and maximum degrees are such that* $c(G) \leq 2$*.*

Proof. Firstly, all graphs on at most 14 vertices have cop number at most 3. For cases where $\Delta \geq 4$, this is a direct consequence of Corollary 4.2.6. For $\Delta = 2$, the graph is either a path or a cycle. For $\Delta = 3$, see the results of Table 4.4. This is also a direct consequence of knowing that $M_4 \geq 16$, see [**39**].

We generate, up to isomorphism, all connected graphs on 12 vertices such that $\Delta \leq 5$ and on 13 and 14 vertices such that $\Delta \leq 4$. We classify these graphs according to their cop number. Afterwards, we also count which of the graphs on 12 vertices with $\Delta \leq 5$ and $c(G) \geq 3$ are such that $\Delta \leq 4$. The results are in Table 4.1. □

| | | | Cop number | | |
|---|---|---|---|---|---|
| $n$ | Degree bounds | Number of graphs | 1 | 2 | $\geq 3$ |
| 11 | $\Delta \leq 5$ | 21503340 | 69310 | 21434024 | 6 |
| 12 | $\Delta \leq 4$ | - | - | - | 80 |
| 12 | $\Delta \leq 5$ | 471142472 | 295377 | 470846922 | 173 |
| 13 | $\Delta \leq 4$ | 68531618 | 73876 | 68456637 | 1105 |
| 14 | $\Delta \leq 4$ | 748592936 | 247022 | 748329391 | 16523 |

**Table 4.1.** Cop number breakdown for connected graphs on 11-14 vertices with some degree restrictions

While the counts are presented to summarize the results, the precise 3-cop-win graphs are the focus of our attention as we use them in the following sections.

To achieve these results, we exhaustively computed the cop number of every connected graph that satisfied our maximum degree constraints. Since we proceeded by exhaustion, the run time of these computations is somewhat long due to the high number of graphs, especially in the case of $\Delta = 14$. We note that a more clever approach might yield faster calculation time.

The first and most obvious improvement would be to only look at graphs with a minimum degree of at least 2, which can reduce the number of graphs to consider by up to around 50%. If we already know the 3-cop-win graphs on one fewer vertices, we can then just consider all possible ways to attach an extra vertex of degree 1 to those graphs. Using this method, we can get all connected 3-cop-win graphs of a given order.

However, this method is still an exhaustive search. A more clever approach would be to consider every 2-cop-win graph $G'$ on $n - \Delta - 1$ vertices, add a vertex $u$ with $\Delta$ neighbours and consider each way of adding edges between $N(u)$ and $G'$ (up to isomorphism), then checking which of these graphs are 3-cop-win. We would recommend the interested reader try this approach.

An more refined approach of this would be to use the algorithm of Section 4.6 to build candidate 3-cop-win graphs, by merging 2-cop-win graphs on fewer vertices. We will see later that although this method can reduce significantly the computation time, in practice it requires some effort to make sure all the possible cases are considered. For the size of graphs we are considering, this may not be necessary.

## 4.4. Graphs with high maximum degree

In this section, we consider the cop number of graphs $G$ such that $\Delta = n - 11$ or $\Delta = n - 12$. We start by investigating some properties of the game of cops and robbers on the Petersen graph ($\mathcal{P}_0$) and its variants, the cornered Petersen graphs ($\mathcal{P}_i, 1 \leq i \leq 6$). Many of the arguments in this section are extremely simple once visualized. For this reason, we have provided many figures representing visualizations of the situation in some of the proofs. Of course, we cannot provide figures for every case, so we encourage the reader draw out the graphs while reading the proof, especially regarding player movements.

By considering the Petersen graph as the Kneser graph $KG_{5,2}$ [**6**], one easily gets the following well-known result (although maybe not with this precise formulation), an illustration of the fact that the Petersen graph is highly transitive.

**Definition 4.4.1.** We say a set of 3 vertices $\{x,y,z\}$ is a *strong stable set* if it is a stable set and if $N(x) \cap N(y) \cap N(z) = \emptyset$.

**Lemma 4.4.2.**

*(a) If $\{x,y,z\}$ and $\{x',y',z'\}$ are strong stable sets of $\mathcal{P}_0$, then there exists an automorphism $\phi_1$ of $\mathcal{P}_0$ such that $\phi_1(x) = x'$, $\phi_1(y) = y'$ and $\phi_1(z) = z'$.*

*(b) If $ab$ and $a'b'$ are two edges of $\mathcal{P}_0$, then there exists an automorphism $\phi_2$ of $\mathcal{P}_0$ such that $\phi_2(a) = a'$, $\phi_2(b) = b'$. This property is known as being arc-transitive.*

We use the labels $m$, $m'$ on the graphs $\mathcal{P}_i$ for $i = 1,\ldots,6$, as shown in Figure 4.2. In particular, for each of these graphs, $\mathcal{P}_i - m \simeq \mathcal{P}_0$. We also see that $m'$ always corners $m$, which will be very useful. We also note that as $m,m' \notin V(\mathcal{P}_0)$, we can say that $\mathcal{P}_0 - m = \mathcal{P}_0 - m' = \mathcal{P}_0$.

As stated in Theorem 4.2.1, we know that $c(\mathcal{P}_0) = 3$. In the next two lemmas, we show that although two cops do not have a winning strategy, they have a lot of power as to which positions can be reached. These lemmas would be very easy to establish computationally, but we consider that formalizing the strategies is worthwhile.

**Lemma 4.4.3.** *If $\{x,y,z\}$ is a strong stable set of $\mathcal{P}_i - m$, then there exists a strategy for 2 cops on $\mathcal{P}_i$ to reach the following situation.*

*(1) The robber is on $x$, except possibly in the case $x = m'$ and $i \in \{5,6\}$, where the robber is either on $m'$ or $m$.*

*(2) The cops are on $y$ and $z$.*

*(3) It is the cops' turn.*

Proof. Let us first consider the case of $\mathcal{P}_0$. Consider the labelling of $\mathcal{P}_0$ with $\alpha_i$ and $\beta_i$ as shown in Figure 4.1. For any $j,j'$, observe that if two cops are on $\beta_j, \beta_{j+1}$ (working in modulo 5), they can directly move to any pair $\beta_{j'},\beta_{j'+1}$.

Without loss of generality, we may consider that $x = \alpha_1, y = \beta_2, z = \beta_3$, as for all other strong stable sets we can apply the automorphism of Lemma 4.4.2(a). For some $k$, we start the game with two cops on $\beta_k$ and $\beta_{k+1}$ (modulo 5). Notice that if the robber is on $\alpha_j$, moving the cops to $\beta_j$ and $\beta_{j+1}$ forces the robber to move to $\alpha_{j-1}$. By repeating this strategy, the cops can essentially make the robber turn in circles on the outer 5-cycle of $\mathcal{P}_0$. At the end of every cops' turn (except the first), the robber is on $\alpha_j$ and the cops are on $\beta_j$ and $\beta_{j+1}$, for some $j$. The cops repeat until the robber is on $\alpha_1$ (unless if we are on the first term in which case we do a full cycle around the graph); it is now the cops' turn and the game is in the desired situation. Observe that this strategy works for any initial choice of $k$. This will be useful later, as for any vertex $w \in \mathcal{P}_0$, we may choose an initial position such that one of the cops is in $N[w]$. We call this the chasing strategy for the Petersen graph. An example is illustrated in Figure 4.3. Even though this might be a very simple idea, this strategy is critical for the rest of this section as it enables more complicated strategies.

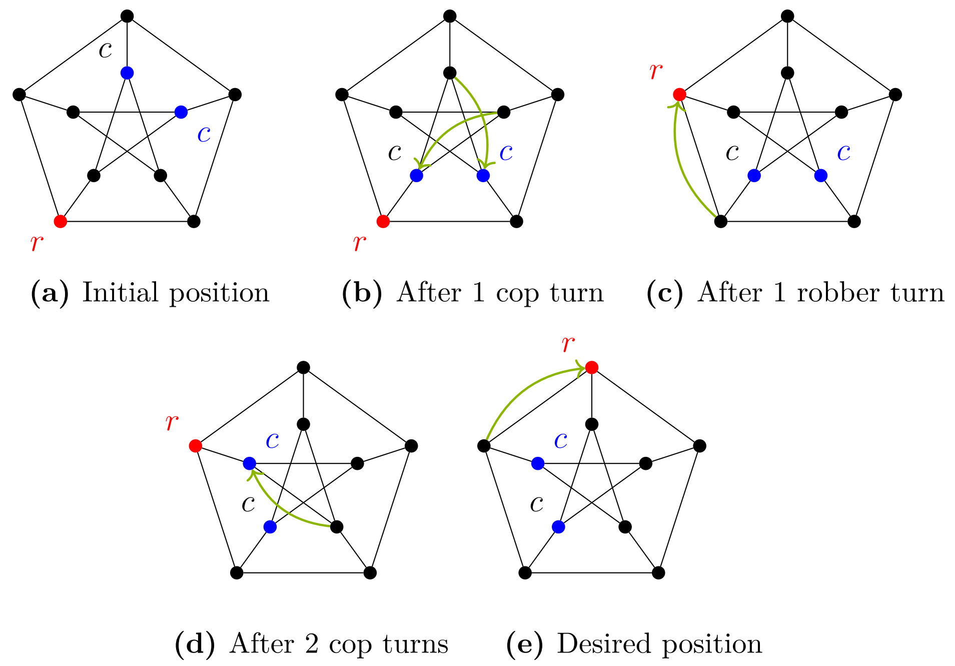


**Fig. 4.3.** Typical application of the chasing strategy on the Petersen graph.

We now consider the cases of $\mathcal{P}_5$ and $\mathcal{P}_6$. In both cases, observe that $m$ and $m'$ are completely indistinguishable: $N(m) = N(m')$. It is then easily seen that the strategy for 2 cops on $\mathcal{P}_5$ or $\mathcal{P}_6$ will be the same as the strategy developed above for $\mathcal{P}_0$, except that the robber may choose to go to either $m$ or $m'$. We apply the strategy for $\mathcal{P}_0$ by considering the robber to be on $m'$ whenever it is actually on $m$. This is essentially a simplified version of the well-known argument used to prove, in particular, Theorem 4.2.4.

Finally, we consider the cases $\mathcal{P}_i$, $i \in \{1, \ldots, 4\}$. Our goal is to apply the strategy of $\mathcal{P}_0$ developed above, with only slight modifications. Using that strategy, we choose initial positions for the cops in $\mathcal{P}_i - m$ such that one of the cops is in $N[m']$ (it is described above why this is possible). If the robber chooses $m$ as an initial position, this cop may then move to $m'$. As $m'$ corners $m$, the robber cannot move without being captured. The other cop may then, within a few turns, capture the robber. Thus, the robber will choose an initial vertex in $\mathcal{P}_i - m$. Now, as long as the robber is not on $m$, copy the strategy for $\mathcal{P}_0$. Suppose that, at some point, the robber moves to $m$. In the strategy above, the robber is adjacent to a cop before every of its turns. Thus, this cop can move to a vertex adjacent to $m$. One easily verifies that in all graphs $\mathcal{P}_i$ for $i \in \{1, \ldots, 4\}$, if one cop is adjacent to $m$, there is at most one other escape route $t$ for the robber. As $\mathcal{P}_i - m \simeq \mathcal{P}_0$ has diameter 2, the other cop can move to block this escape route by moving to some vertex in $N[t]$. Thus, while applying this strategy, the robber will never move to $m$. Hence, the strategy copied from $\mathcal{P}_0$ yields the desired final position. $\square$

By weakening the condition that it is the cops' turn at the end of the strategy, we can get more freedom as to where we can place the cops, enabling more strategies.

**Lemma 4.4.4.** *If $x$,$y$,$z$ are any three distinct vertices of $\mathcal{P}_i - m$, then there exists a strategy for 2 cops on $\mathcal{P}_i$ to reach the following situation.*

*(1) The robber is on $x$, except possibly in the case $x = m'$ and $i \in \{5,6\}$, where the robber is either on $m'$ or $m$.*
*(2) The cops are on $y$ and $z$.*
*(3) It is the robber's turn.*

Proof. Without loss of generality, we show the statement for $\mathcal{P}_0$. For this lemma, generalizing to the cornered Petersen graphs is immediate.

We first consider the case where $xz \in E(\mathcal{P}_0)$. We will enumerate the main cases and conclude by symmetry for the others. We may assume that $x = \alpha_1$ and $z = \beta_1$ (using the labelling from Figure 4.1), all other possibilities can be solved using the automorphisms of Lemma 4.4.2 (b).

We apply Lemma 4.4.3 to place the robber on vertex $\alpha_1$ and the cops on the vertices specified in Table 4.2 (always forming a strong stable set), and then specify the additional move required to place the cops in the desired final position.

| Final position for cops ($x$,$y$) | Position after applying Lemma 4.4.3 | Movements |
|---|---|---|
| $\alpha_2$,$\beta_1$ | $\beta_2, \beta_3$ | $\beta_2 \to \alpha_2, \beta_3 \to \beta_1$ |
| $\beta_2$,$\beta_1$ | $\beta_2, \beta_3$ | $\beta_2 \to \beta_2, \beta_3 \to \beta_1$ |
| $\beta_3$,$\beta_1$ | $\beta_4, \beta_5$ | $\beta_4 \to \beta_1, \beta_5 \to \beta_3$ |
| $\alpha_3$,$\beta_1$ | $\alpha_3, \beta_4$ | $\alpha_3 \to \alpha_3, \beta_4 \to \beta_1$ |

**Table 4.2.** Strategy on $\mathcal{P}_0$ to bring the robber to $\alpha_1$ with the cops in the desired final position, where at least one cop will be adjacent to robber.

It is easily seen that all other choices of $y$ are analogous by reflection of the graph relative to the vertical axis.

We use a similar approach for the case where $xz \notin E$. We may suppose without loss of generality that $x = \alpha_1$ and $z = \beta_2$: it is easily verified that any two non-adjacent vertices can be expanded into a strong stable set, then apply Lemma 4.4.2 (a). To further reduce the number of cases, we can also assume that $xy \notin E$ (if $xy \in E$, switching the roles of $y$ and $z$ brings us back to the previous case), as we can see in Table 4.3.

$\square$

| Final position for cops ($x$,$y$) | Position after applying Lemma 4.4.3 | Movements |
|---|---|---|
| $\alpha_3$,$\beta_2$ | $\alpha_3, \beta_4$ | $\alpha_3 \to \alpha_3, \beta_4 \to \beta_2$ |
| $\alpha_4$,$\beta_2$ | $\alpha_3, \beta_4$ | $\alpha_3 \to \alpha_4, \beta_4 \to \beta_2$ |
| $\beta_3$,$\beta_2$ | $\beta_3$,$\beta_2$ | $\beta_3 \to \beta_3, \beta_2 \to \beta_2$ |
| $\beta_4$,$\beta_2$ | $\beta_4, \beta_5$ | $\beta_4 \to \beta_4, \beta_5 \to \beta_2$ |
| $\beta_5$,$\beta_2$ | $\beta_4, \beta_5$ | $\beta_4 \to \beta_2, \beta_5 \to \beta_5$ |

**Table 4.3.** Strategy on $\mathcal{P}_0$ to bring the robber to $\alpha_1$ with the cops in the desired final position, where neither cop will be adjacent to robber.

In the next lemmas, we will consider consider graphs with the following properties, with the goal of eventually showing that these do not exist. We state these properties now to avoid repetition.

**Hypothesis 4.4.5.** *Let $G$ be a connected graph such that $c(G) > 3$ and $u \in V(G)$ such that $G - N[u] \simeq \mathcal{P}_i$, for some $0 \leq i \leq 6$.*

In the cases of $1 \leq i \leq 6$, we may in particular consider that $m$,$m' \in V(G)$ by fixing the isomorphism. In the cases of $i = 5$,$6$, as the labels $m$ and $m'$ can be switched, we will always suppose $m'$ to be the vertex of the two which has the greatest degree in $G$ (if both have the same degree, then we choose arbitrarily). To simplify notation, we will denote $B_u = V(G - N[u] - m)$. It is easily seen that in all cases $\langle B_u \rangle \simeq \mathcal{P}_0$. In other words, $B_u$ is the (a) set of vertices inducing a Petersen graph.

The approach will be to build up a number of structural properties of $G$ by showing that otherwise there exists a winning strategy for 3 cops, yielding a contradiction. We start by proving that all vertices in $B_u$ have a neighbour in $N(u)$.

**Lemma 4.4.6.** *Consider Hypothesis 4.4.5. For all $x \in B_u$, $|N(x) \cap N(u)| \geq 1$.*

Proof. Let $x \in B_u$. Suppose that $|N(x) \cap N(u)| = 0$ and that there exists a neighbour $v$ of $x$ in $B_u$ such that $|N(v) \cap N(u)| \geq 1$. If $x$ is adjacent to $m$ (in particular, $m \in V(G)$, meaning that $1 \leq i \leq 6$) but $x \neq m'$, we also suppose $v \neq m'$ (this additional hypothesis will be useful later). We note that in the case with $x = m'$, then by our choice of $m'$ we know that $m$ also has no neighbours in $N(u)$.

We show this situation yields a winning strategy for 3 cops.

Let $y$,$z$ be the other neighbours of $x$ in $B_u$ and let $w \in N(v) \cap N(u)$. The situation is portrayed in Figure 4.4. We start by placing a cop on $u$, which will only move if the neighbour enters $N[u]$. This is commonly referred to as a stationary cop. As long as this cop stays on $u$, the robber is stuck in $G - N[u] \simeq \mathcal{P}_i$. Thus, the two other cops may apply the strategy from Lemma 4.4.4 on $G - N[u]$ to place the robber on $x$, a cop on $y$ and a cop on $z$. In the special case where $i \in \{5,6\}$ and $x = m'$, the robber might actually be on $m$.

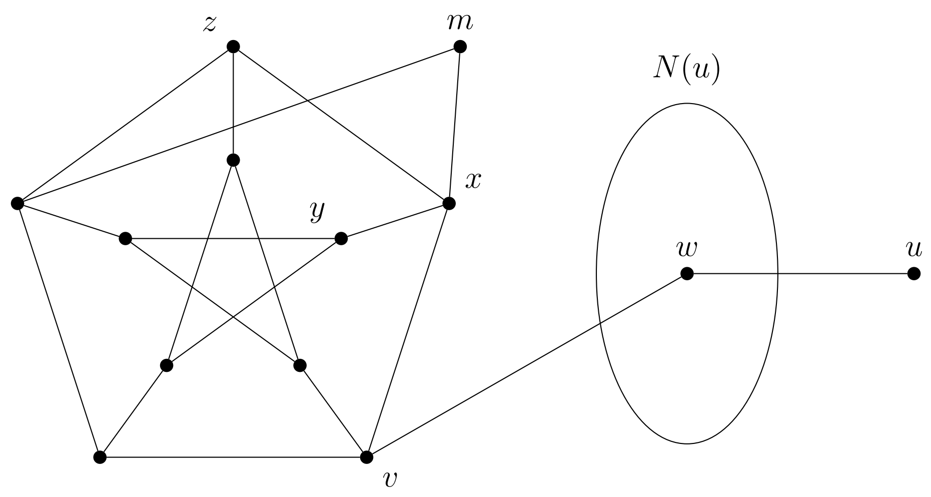


**Fig. 4.4.** Example situation during the proof of Lemma 4.4.6. Unused or unknown vertices and edges are omitted.

During the last turn of this strategy, the cop on $u$ moves to $w$. It is now the robber's turn. In all cases (except one considered below), all of the robber's neighbours in $B_u$ are protected by cops, there is a cop adjacent to the robber, and the robber has no neighbour in $N(u)$.

This is not necessarily obvious in the special case with $i = 5$,6 and $x$ is adjacent to $m$ and thus the robber can be on $m$. As noted above, in this case we know that neither $m$ nor $m'$ has neighbours in $N(u)$. We have supposed that $v \neq m'$, so we must have that $m'$ is one of $x$,$y$,$z$. If $x = m'$, then $m$ is indistinguishable from $m'$ and the statement is trivial. If $m'$ is $y$ or $z$, then the cop on that vertex does indeed cover $N[m] = N[m']$. The only case in which the robber does not get immediately caught is when $m'$ and $m$ are not adjacent (as when $i = 5$), the cop on $m'$ is not adjacent to the robber, but does prevent the robber on $m$ from moving, and can be caught shortly thereafter.

The robber is caught unless it can move to an unprotected vertex outside $B_u$ (necessarily $m$, and necessarily not in the special case we have just discussed), which only happens if the robber is on $x$, if $x$ is adjacent to $m$, and if $m$ is adjacent to neither $y$ or $z$. We may suppose this is the case.

If $x = m'$, the cop on $w$ moves back to $u$ and the cop on $y$ moves to $m'$: as $m'$ covers all neighbours of $m$ in $B_u$ and $u$ covers all neighbours in $N(u)$, the robber is trapped and will be caught one turn later.

Now, suppose that $x \neq m'$, but that $x$ is adjacent to $m$. In particular, $x$ is also adjacent to $m'$. Recall that, in this case, we supposed that $v \neq m'$. Thus, $m'$ is either $y$ or $z$. The cop on $m'$ stays and the cop on $w$ moves back to $u$, trapping the robber on $m$. The last cop may then capture the robber within a few turns.

In all cases, a contradiction is reached with the hypothesis that $c(G) > 3$.

In other words, as soon as we know that a vertex of $B_u$ (other than $m'$) has a neighbour in $N(u)$, we can say the same for its 3 neighbours in $B_u$ (and then their neighbours, etc.). Thus, in order to prove the lemma, it suffices to show that there exists at least one vertex of $B_u \setminus \{m'\}$ that has a neighbour in $N(u)$ (because $\langle B_u \rangle - m'$ is necessarily connected).

Suppose the contrary: no vertex of $B_u \setminus \{m'\}$ has a neighbour in $N(u)$. If we are in the case of $G - N[u] \simeq \mathcal{P}_0$, then $G$ would be disconnected (as $B_u \setminus \{m'\} = B_u$), which is a contradiction. In the remaining cases, we explicit a winning strategy for 3 cops. Place a stationary cop on a vertex $t \in N[m] \cap N[m']$ (one easily verifies that in all cases this set is non-empty), and place another on $u$ (the third cop can be placed anywhere initially). The robber must choose an initial position in $B_u$. As any exit from $B_u$ will go through $m$ or $m'$ (by our hypothesis), the stationary cop guarantees that the robber will never leave $B_u$. The two other cops then have a winning strategy on $B_u \setminus N[t]$, which contains at most 7 vertices. $\square$

The idea of applying the chasing strategy of $G - N[u]$ while we leave a cop on $u$ is one we will use frequently. As long as there is a cop on $u$, it is as we were playing on $G - N[u]$. The idea of moving the cop from $u$ during the last move of this strategy does not affect it, as it happens after the last robber move which is part of that strategy.

We now characterize the intersection of neighbourhoods of vertices in $N(u)$ with $B_u$.

**Lemma 4.4.7.** *Consider Hypothesis 4.4.5. If $w \in N(u)$, then $N(w) \cap B_u$ does not contain a subset $\{a,b,c\}$ of distinct vertices such that :*

*(1) $ab \notin E(G)$;*
*(2) $c \notin N(a) \cap N(b)$ ($c$ is not the common neighbour of $a$ and $b$ in $\langle B_u \rangle$);*
*(3) $c \notin N(x)$ for $x \in N(a) \cap N(b) \cap B_u$ ($c$ is not adjacent to the common neighbour of $a$ and $b$ in $B_u$).*

Proof. Suppose that $N(w) \cap B_u$ does contains a subset $\{a,b,c\}$ respecting these conditions. We explicit a winning strategy for 3 cops on $G$, which will lead to a contradiction. We denote by $x$ the common neighbour of $a$ and $b$ in $B_u$ and by $d$ the neighbour of $x$ in $B_u$ that is neither $a$ or $b$.

Let $z$ be a vertex of $B_u$ such that $\{x,c,z\}$ is a strong stable set of $\langle B_u \rangle$ (it is easily seen that any stable set of size 2 in the Petersen graph can be expanded into a strong stable set). The situation is portrayed in Figure 4.5.

We place a cop on $u$ at the start of the game, and then use Lemma 4.4.3 to place the other cops on $c$ and $z$, and the robber on $x$ (or if $x = m'$ and $i \in \{5,6\}$ possibly on $m$).

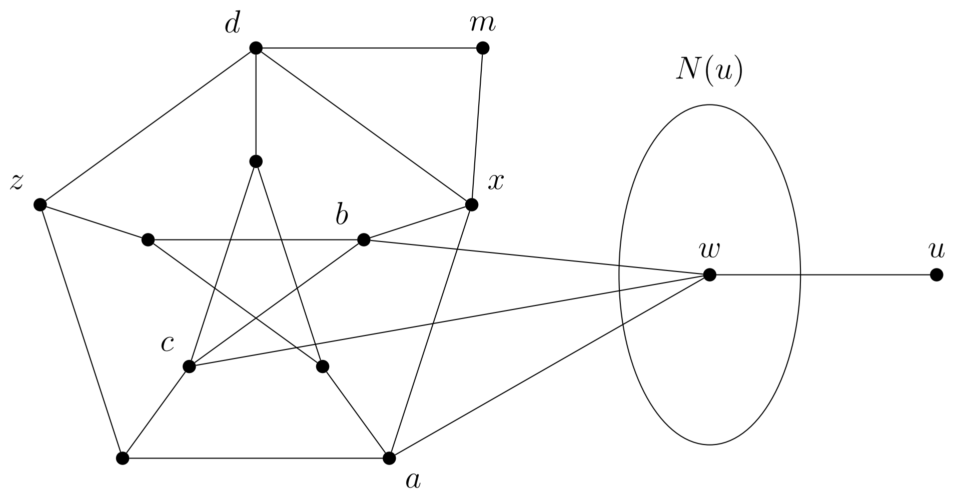


**Fig. 4.5.** Example situation during the proof of Lemma 4.4.7. Unused or unknown vertices and edges are omitted.

It is now the cops' turn. The cop on $c$ moves to $w$, the cop on $z$ moves to either $d$ or a neighbour of $d$ (this is possible because the Petersen graph has diameter 2), and the cop on $u$ stays still. All neighbours of $x$ in $N(u)$ are covered by the cop on $u$, $a$ and $b$ covered by the cop on $w$, and $d$ is covered by the 3rd cop, which is either on $d$ or on a neighbour of $d$. Hence, after the robber's move it must either be on $x$ or on $m$ (if $m$ exists, that is $1 \leq i \leq 6$). We note that in the special case with $x = m'$ and $i \in \{5,6\}$ the robber might have already been on $m$, but that the same argument as above shows that the robber could not have move outside of either $x = m'$ or $m$.

If the robber is still on $x$, but cannot be immediately captured, the cop which is adjacent to $d$ moves to $d$. Now, the robber cannot stay put without being captured. Hence, for the rest of the proof we can assume the robber moves to $m$ or was already on $m$.

If $m'$ is $a$ or $b$, the cop on $w$ moves to $m'$. If $m' = d$, the cop that is on adjacent to $d$ or on $d$ moves to (or stays on) $d$. In both cases, there is now a cop on $m'$, which, together with the cop on $u$, guarantees that the robber is now stuck on $m$. The third cop may capture the robber within a few turns.

If $m' = x$, then, by definition, $N(m) \cap B_u \subseteq \{a,b,d,x\}$. As previously, move a cop to $d$ (if it is not already there). The pair of cops on $d$ and $w$ cover this set, hence the robber cannot move. At the next cops' turn, the cop on $d$ moves to $m'$, and at the following turn capture the robber.

In all cases, there is a contradiction as $c(G) > 3$. □

**Lemma 4.4.8.** *Consider Hypothesis 4.4.5. If $w \in N(u)$, then there exists a vertex of $B_u$ dominating $N(w) \cap B_u$.*

PROOF. If $|N(w) \cap B_u| \leq 2$, the result is trivial, as $\mathrm{diam}(\langle B_u \rangle) = 2$.

If $|N(w) \cap B_u| \geq 3$, suppose the statement is false. As $\langle B_u \rangle$ does not contain a triangle, not all vertices of $N(w) \cap B_u$ can be pairwise adjacent: we can choose $a,b \in N(w) \cap B_u$ such that $a,b$ are not adjacent. Denote $x$ the common neighbour of $a,b$ in $B_u$. By our previous supposition, $x$ does not dominate $N(w) \cap B_u$, thus we can choose $c \in N(w) \cap B_u$ not in $N[x]$. The subset $\{a,b,c\}$ then contradicts Lemma 4.4.7. □

In particular, every vertex of $N(u)$ can have at most 4 neighbours in $B_u$ because $\langle B_u \rangle$ is 3-regular. We also note that in some of the cases there is a unique choice for this dominating vertex, in particular when $N(w) \cap B_u$ has 3 or 4 vertices. We are now ready to strengthen Lemma 4.4.6.

**Lemma 4.4.9.** *Consider Hypothesis 4.4.5. For all $x \in B_u$, the following holds.*

*(1) If $x \notin N[m']$, then $|N(x) \cap N(u)| \geq 3$.*
*(2) If $x \in N[m']$, then $|N(x) \cap N(u)| \geq 2$.*

PROOF. (1) Suppose the contrary: there exists $x \in B_u \setminus N[m']$ such that $|N(x) \cap N(u)| \in \{1,2\}$ (by Lemma 4.4.6, $|N(x) \cap N(u)| \geq 1$). We explicit a winning strategy for 3 cops on $G$. Denote by $w_1,w_2$ the neighbours of $x$ in $N(u)$ (if there is only one neighbour, set $w_1 = w_2$) and by $y_1,y_2,y_3$ the neighbours of $x$ in $B_u$.

By Lemma 4.4.8, there exists a vertex of $B_u$ dominating the neighbourhood of $w_1$ in $B_u$. As $x$ is in this neighbourhood, we know this dominating vertex (there might be more than one possible choice) is in $\{y_1,y_2,y_3,x\}$. This is also true for $w_2$. Thus, we can pick at most 2 elements of $\{y_1,y_2,y_3,x\}$ that dominate all neighbours of $w_1,w_2$ in $B_u$.

Without loss of generality (by symmetry of $y_1,y_2,y_3$ in the Petersen graph), we assume the 2 elements can be picked in $\{y_1,y_2,x\}$. By Lemma 4.4.6, $y_3$ must have a neighbour $t$ in $N(u)$. The situation is portrayed in Figure 4.6.

We place one cop on $u$. We use Lemma 4.4.4 to place the robber on $x$ and the two other cops on $y_1, y_2$. During the last move of this strategy, the cop on $u$ moves to $t$. It is now the robber's turn.

The robber on $x$ cannot move to a neighbour inside of $B_u$ (there are cops on $y_1$ and $y_2$, and $y_3$ is covered by the cop on $t$) and there are cops adjacent to the robber. As $x \notin N[m']$, we know that $m \notin N(x)$. Thus, the robber has no choice but to move to either $w_1$ or $w_2$. Without loss of generality, let us say the robber moves to $w_1$.

Denote by $a$ the vertex dominating the neighbours of $w_1$ in $B_u$. We recall that $a$ is either $y_1$, $y_2$ or $x$. We now move the cop on $t$ back to $u$. Of the two cops on $y_1$ and $y_2$, one must be able to move to $a$, and does so. If $m' \in B_u$ (that is, if we are not in the case of $\mathcal{P}_0$), the third cop moves to either $m'$ or a neighbour of $m'$. After this move,

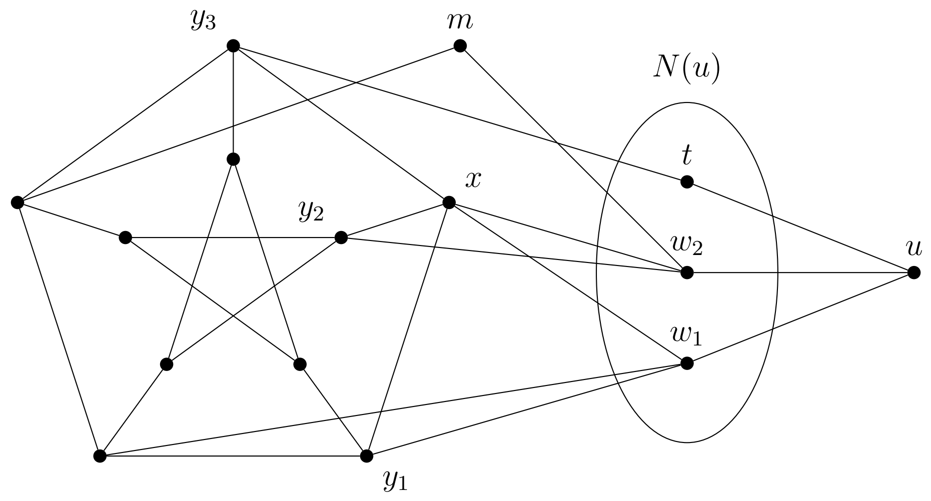


**Fig. 4.6.** Example situation during the proof of Lemma 4.4.9 (1). Unused or unknown vertices and edges are omitted.

all escapes in $N(u)$ are covered by the cop on $u$, all escapes in $B_u$ are covered by the cop on $a$, and the robber cannot stay still as $u$ is adjacent to $w_1$. Thus, the robber is caught one move later unless the robber can move to $m$. In this case, the third cop can now move to $m'$ and trap the robber. Leaving the cops on $u$ and $m'$ fixed, the cop on $a$ can then go capture the robber. This is a contradiction as $c(G) > 3$.

(2) As the proof will be very similar to the previous case, we outline the main differences. If $x = m'$ and $i \in \{5{,}6\}$, suppose that $m'$ and $m$ each have at most one neighbour each in $N(u)$. Then, consider $w_1$ and $w_2$ these vertices and apply the same strategy as above. Even though the robber will have chosen to go to either $m'$ or $m$, both cases for its subsequent move will be covered using the strategy above. We note that the robber will not be able to stay on either $m'$ or $m$, as both are adjacent to $y_1$,$y_2$. Thus, either $m$ or $m'$ must have 2 or more neighbours in $N(u)$. As we have selected $m'$ to have the greatest degree of the two, the statement follows for this case.

Consider now that $x \in N[m']$ but $x \neq m'$ or $i \notin \{5{,}6\}$ (as we covered that case above). Our goal is to prove that $x$ cannot have a unique neighbour in $N(u)$. Suppose the contrary, we denote by $w_1$ this neighbour. As in the above strategy, one cop's role will be to cover the vertex dominating the neighbourhood of $w_1$ in $B_u$, or if this is $x$, then to be on an adjacent vertex. We will also want a cop to be on $m'$, or on a neighbour of $m'$ if $x = m'$. In the notation of the original case, this could informally be seen as considering $w_2$ to be $m$. If the robber moves to $w_1$, we follow a similar strategy to above. If the robber moves to $m$, then one of the cops will move to $m'$. Recalling that another robber will return to $u$, the last cop will be able to go capture the robber.

□

We are now ready to prove the desired results.

**Proposition 4.4.10.** *If $G$ is a connected graph such that $\Delta \in \{n-12, n-11\}$ and $n \leq 18$, then $c(G) \leq 3$.*

Proof.

(1) We consider $\Delta = n - 11$. Let $u$ be a vertex of maximum degree. We know that $|V(G - N[u])| = 10$.
If $G - N[u]$ is disconnected, every one of its connected components has cop number at most 2, as no connected component can contain at least 10 vertices. Applying Corollary 4.2.5 yields the desired result. Otherwise, $G - N[u]$ must be connected. Suppose that $c(G) > 3$. Then, $c(G - N[u]) > 2$, and by Theorem 4.2.1, $G - N[u]$ must be isomorphic to $\mathcal{P}_0$.
Then, $G$ and $u$ satisfies the conditions of Hypothesis 4.4.5.
By Lemma 4.4.9, every vertex of $B_u$ has at least 3 neighbours in $N(u)$. As $B_u$ has 10 vertices, there are at least 30 edges between $N(u)$ and $B_u$.
As $n \leq 18$, we have that $\Delta \leq 7$. By Lemma 4.4.8, every vertex of $N(u)$ has at most 4 neighbours in $B_u$. Thus, there are at most $4\Delta \leq 28$ edges between $N(u)$ and $B_u$.
This is a contradiction, as we have claimed there are at least 30 but at most 28 edges between $N(u)$ and $B_u$. Thus, $c(G) \leq 3$.

(2) We consider $\Delta = n - 12$. Let $u$ be a vertex of maximum degree. We know that $|V(G - N[u])| = 11$.
Let us first consider the case where $G - N[u]$ is disconnected. If every connected component has cop number at most 2, then, as in the previous case, we are done. By Theorem 4.2.1, the only other case is if one component is isomorphic to $\mathcal{P}_0$ and the other is an isolated vertex $x$. By applying Corollary 4.2.5, $c(G) \leq 3$ if and only if $c(G-x) \leq 3$. As $G-x$ satisfies the conditions of the previous case of this proposition, we conclude that $c(G - x) \leq 3$.
We may now consider that $G - N[u]$ is connected. By Proposition 4.3.2, $G - N[u] \simeq \mathcal{P}_i$, for some $1 \leq i \leq 6$. Suppose that that $c(G) > 3$. Then, $G$ and $u$ satisfies the condition of Hypothesis 4.4.5.
By Lemma 4.4.9, every vertex of $B_u$ has at least 2 neighbours in $N(u)$, and each vertex not in $N[m']$ (of which there are at least 6) has at least 3 neighbours in $N(u)$. In total, there are at least 26 edges between $N(u)$ and $B_u$.
By Lemma 4.4.8, each vertex of $N(u)$ has at most 4 neighbours in $B_u$. As $n \leq 18$, we have that $\Delta \leq 6$. Thus, there are at most $4\Delta \leq 24$ edges between $N(u)$ and $B_u$.

This is a contradiction, as we have claimed there are at least 26 but at most 24 edges between $N(u)$ and $B_u$.

□

These results will be used to prove that $M_4 = 19$, but we would also like to reduce the number of possible 4-cop-win graphs on 19 vertices. This will be possible with more work, but we first need the following definition.

**Definition 4.4.11.** Consider Hypothesis 4.4.5. Let $w \in N(u)$ such that $|B_u \cap N(w)| = 4$. The vertex $x$ of $B_u$ dominating $B_u \cap N(w)$ will be called the *projection* of $w$. If $x$ is the projection of $k$ vertices of $N(u)$, we will call $p(x) = k$ the *projection multiplicity* of $x$.

By Lemma 4.4.8, this is well defined and the projection of a vertex is unique.

**Observation 4.4.12.** *If $x \in B_u$, $|N(x) \cap N(u)| \geq \sum_{y \in N[x] \cap B(u)} P(y)$. In particular, if $y \in B_u$ has projection multiplicity $k$, then each vertex in $N(y) \cap B_u$ has degree at least $k$.*

Proof. We recall that when a vertex $y$ has projection multiplicity $k$, this means that $k$ vertices of $N(u)$ have for neighbours in $B_u$ exactly $N[y] \cap B_u$, giving each vertex in this set at least $k$ neighbours in $N(u)$.

Noting that the projection of a vertex is unique, we see that the neighbours $x$ inherits from each projection on it or on its neighbours in $B_u$ are pairwise distinct. The lower bound follows immediately by summing the projective multiplicity for each vertex in $N[x]$. □

We now see an interesting property of projections.

**Lemma 4.4.13.** *Consider Hypothesis 4.4.5. Let $x \in B_u \setminus N[m']$.*

*(1) If $|N(x) \cap N(u)| = 3$, then $p(x) = 0$.*
*(2) More generally, $p(x) \leq |N(x) \cap N(u)| - 2$.*

Proof.

(1) Suppose the contrary, we explicit a winning strategy for 3 cops.
Suppose that $x$ is the projection of a vertex $w$ of $N(u)$: $w$ is adjacent to $x$ and to each neighbour of $x$ in $B_u$.
As $|N(x) \cap N(u)| = 3$, $x$ has two other neighbours in $N(u)$, which we will denote by $t_1$ and $t_2$. If $t_1$ has a neighbour in $B_u$ other than $x$, choose one and denote it $r_1$. If not, then choose $r_1$ to be any neighbour of $x$ in $B_u$. We choose $r_2$ similarly. The situation is portrayed in Figure 4.7.
We start by placing one cop on $u$. Using Lemma 4.4.4 (recall that $x \neq m'$, which avoids the exceptional case), we place the robber on $x$ and the two other cops on $r_1$ and $r_2$. During the last move of that strategy, move the cop from $u$ to $w$. It is now the robber's turn.

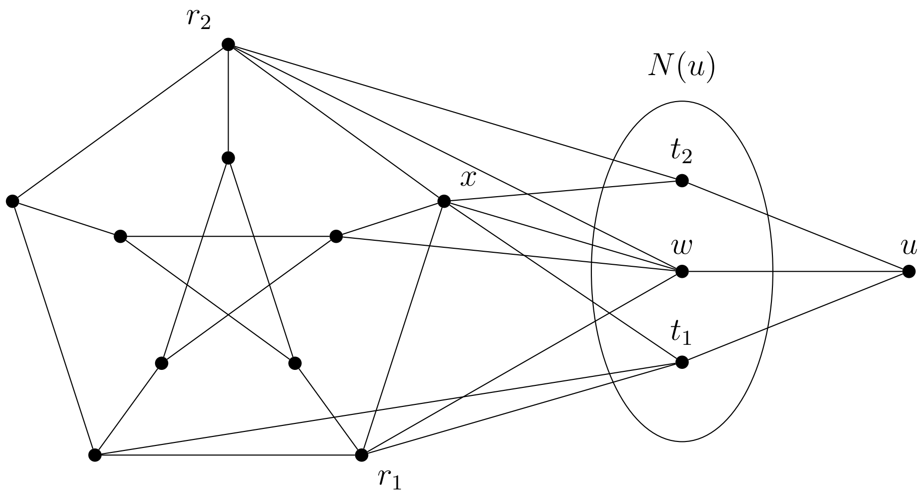


**Fig. 4.7.** Example situation during the proof of Lemma 4.4.13 (1). Unused or unknown vertices and edges are omitted.

As there is a cop on $w$, the robber cannot stay in $B_u$. As $x \notin N[m']$, $x$ is not adjacent to $m$. If $t_1$ had a neighbour in $B_u$ other than $x$, then the cop on $r_1$ blocks the robber from moving to $t_1$, and similarly for $t_2$.

Thus, the only scenario in which the robber does not get captured immediately after moving is if (without loss of generality),= $t_1$ only has one neighbour in $B_u$, and the robber moves to $t_1$. In this case, the cop on $r_1$ is adjacent to $x$. The cop on $w$ moves back to to $u$, and the cop on $r_1$ moves to $x$. The third cop (on $r_2$) moves to $m'$ or a neighbour of $m'$. The robber is caught one turn later, as $x$ dominates the neighbourhood of $w$ in $B_u$ and $u$ dominates $N(u)$, unless it can move to $m$. In this case, the third cop can move to $m'$ and trap the robber. The cop on $x$ can capture the robber within a few turns. This contradicts that $c(G) > 3$.

(2) The strategy is similar to the previous case. Suppose to the contrary that $x$ has projection multiplicity at least $|N(x) \cap N(u)| - 1$. Then, there is at most 1 neighbour of $x$ in $N(u)$ which does not project onto $x$. Choose $t_1$ to be this vertex (if there is any) and select the corresponding $r_1$ as above. Choose $r_2$ to be any other neighbour of $x$ in $B_u$: $r_2$ covers all vertices projecting onto $x$. The rest of the strategy is identical.

□

We are now ready for the desired result.

**Proposition 4.4.14.** *If $G$ is a connected graph such that $n = 19$ and $\Delta \in \{7,8\}$, then $c(G) \leq 3$.*

Proof. Suppose $c(G) > 3$. Let $u$ be a vertex of maximal degree in $G$.

(1) We consider $\Delta = 8$. Recall the arguments of the proof of Proposition 4.4.10. In particular, we can consider that $G - N[u] \simeq \mathcal{P}_0$.
There are at most $4\Delta = 32$ edges between $N(u)$ and $B_u$, by Lemma 4.4.8. By Lemma 4.4.9, each vertex in $B_u$ has at least 3 neighbours in $N(u)$: there are at least 30 edges between $B_u$ and $N(u)$. Thus, there are at most 2 extra edges. By extra edges, we mean that there are edges which, if removed, would leave each vertex in $B_u$ with exactly the lower bound number of neighbours in $B_u$, as specified in Lemma 4.4.9. Then, there are at least 8 vertices in $B_u$ incident to exactly 3 such edges.
Furthermore, if there are fewer than 6 vertices of $N(u)$ that each have exactly 4 neighbours in $B_u$, then there cannot be at least 30 edges between $N(u)$ and $B_u$. Thus, $\sum_{x \in B_u} p(x) \geq 6$.
Recall that Lemma 4.4.13 states that no vertex in $N(u)$ with 3 neighbours in $B_u = B_u \setminus N[m']$ can be a projection. If all vertices of $B_u$ have exactly 3 neighbours in $N(u)$, this is a direct contradiction.
Otherwise, there are at most 2 vertices which can have non-zero projective multiplicity, that is the vertices with 4 or 5 neighbours. Denote them by $a_1$,$a_2$ (if there is only one vertex, $a_1 = a_2$). Then, $p(a_1) + p(a_2) \geq 6$ (if $a_1 = a_2$ then simply $p(a_1) \geq 6$). Let $x \in N[a_1] \cap N[a_2] \cap B_u$ (which exists as $\mathcal{P}_0$ has diameter 2). As $x$ is adjacent to all projections, $x$ must be adjacent at least 6 vertices of $N(u)$ (Observation 4.4.12). As $x$ also has 3 neighbours in $B_u$, the degree of $x$ is at least 9, which is a contradiction.

(2) We consider $\Delta = 7$. Recall the arguments of the proof of Proposition 4.4.10. In particular, we can say that $G - N[u] \simeq \mathcal{P}_i$, for some $1 \leq i \leq 6$.
There are at most $4\Delta = 28$ edges between $N(u)$ and $B_u$, by Lemma 4.4.8. By Lemma 4.4.9, each vertex in $B_u \setminus N[m']$ has at least 3 neighbours in $N(u)$ and each vertex in $B_u \cap N[m']$ has at least 2 neighbours in $N(u)$: in total, there are at least 26 edges between $B_u$ and $N(u)$.
Using the same argument as above, depending on the number of edges between $B_u$ and $N(u)$, we can find between 5 and 7 vertices in $N(u)$ which have 4 neighbours each in $B_u$, and thus the total projection multiplicity of $B_u$ is as follows.
(a) 26 edges: $\sum_{x \in B_u} p(x) = 5$;
(b) 27 edges: $\sum_{x \in B_u} p(x) = 6$;
(c) 28 edges: $\sum_{x \in B_u} p(x) = 7$.
Recall that Lemma 4.4.13 states that no vertex in $B_u \setminus N[m']$ with 3 neighbours in $N(u)$ can be a projection. Also, if $x \in B_u \setminus N[m']$, $p(x) \leq |N(x) \cap N(u)| - 2$: if $x$ has 4 neighbours in $N(u)$ it can be the projection of at most 2 vertices.
If all vertices in $B_u \setminus N[m']$ have exactly 3 neighbours in $N(u)$, then this implies all projections will be vertices in $N[m']$: at least 5 vertices project on $m'$ or on a neighbour. Thus, $m'$ will have at least 5 neighbours in $N(u)$. As $m'$ also has at least

3 neighbours in $B_u$, $d(m') \geq 8$, which is impossible as $\Delta = 7$. This situation includes the case in which there are exactly 26 edges between $B_u$ and $N(u)$.
Suppose there is exactly one vertex $x$ of $B_u \setminus N[m']$ with exactly 4 neighbours in $N(u)$, with all others having exactly 3. This vertex will have projection multiplicity at most 2, so the total projection multiplicity of vertices of $N[m']$ is at least 4. Thus, $m'$ will have at least 4 neighbours in $N(u)$, which is impossible as this would imply there are 29 edges between $B_u$ and $N(u)$ ($x$ has 4, the other 5 vertices in $B_u \setminus N[m']$ have 3, $m'$ has at least 4, and each of the 3 vertices of $B_u \cap N(m')$ has at least 2).
Suppose now there are 2 vertices $x_1$,$x_2$ of $B_u \setminus N[m']$ with 4 neighbours in $N(u)$. These two additional edges bring the total to 28. Thus, the total projection multiplicity is at least 7. There are at most 2 vertices projecting onto $x_1$ and 2 vertices projecting on $x_2$. Thus, at least 3 vertices project onto vertices in $N[m']$. This a contradiction, as $m'$ must have exactly 2 neighbours in $N(u)$, otherwise there would be more than 28 edges between $B_u$ and $N(u)$. Considering that with $\Delta = 7$, no vertex of $B_u$ can have 5 or more neighbours in $N(u)$, there are no cases left.

In all possible cases, a contradiction was found. Thus, $c(G) \leq 3$. □

## 4.5. Graphs with maximum degree 3

In this section, we consider the cop number of graphs with maximum degree 3. We start with the main result of this section.

**Proposition 4.5.1.** *If $G$ is a connected graph such that $\Delta \leq 3$ and $n \leq 20$, then $c(G) \leq 3$.*

Proof. We first prove the statement for $\delta \geq 2$. For $10 \leq n \leq 20$, we generate all graphs such that $\delta \geq 2$ and $\Delta \leq 3$. We then classify each graph according to its cop number. We present the results in Table 4.4, which shows that no such graph with cop number at least 4 exists. We have also extracted the 3-cop-win graphs.

We now considers graphs which contain vertices of degree 1. We know that removing a vertex of degree 1 from a graph does not change the cop number nor the fact that it is connected (as the vertex of degree 1 is cornered by its neighbour). We successively remove vertices of degree 1 from the graph. We eventually either get to a graph such that $\delta \geq 2$ and $n \geq 10$ (in which case the above results can now be applied) or we eventually get to a graph of order at most 9 (in which case we apply Theorem 4.2.1). □

Notwithstanding the slight improvement of considering $\delta \geq 2$, the approach here is clearly far from optimal. The algorithm described in the following section is an example of a possibly better strategy. However, as we will see, this algorithm would not be the most efficient for maximum degree 3 : to compute potential 4-cop-win graphs on 19 vertices,

|  |  | Cop number |  |  |  |
|---|---|---|---|---|---|
| $n$ | $G : \delta \geq 2, \Delta \leq 3$ | 1 | 2 | 3 | $\geq 4$ |
| 10 | 458 | 7 | 450 | 1 | 0 |
| 11 | 1353 | 12 | 1341 | 0 | 0 |
| 12 | 4566 | 21 | 4543 | 2 | 0 |
| 13 | 15530 | 35 | 15495 | 0 | 0 |
| 14 | 56973 | 63 | 56901 | 9 | 0 |
| 15 | 214763 | 114 | 214642 | 7 | 0 |
| 16 | 848895 | 211 | 848622 | 62 | 0 |
| 17 | 3454642 | 388 | 3454093 | 161 | 0 |
| 18 | 14542574 | 735 | 14540858 | 981 | 0 |
| 19 | 62871075 | 1389 | 62865352 | 4334 | 0 |
| 20 | 279175376 | 2664 | 279147564 | 25148 | 0 |

**Table 4.4.** Cop number breakdown for connected subcubic graphs.

one would still need to compute subcubic 3-cop-win graphs on 15 vertices. A potentially more interesting algorithm for building possible 4-cop-win subcubic graphs would consist in building graphs around long shortest paths (see [**2**, Lemma 4], which describes how a cop can protect a shortest path) by adding the desired number of other vertices and considering all possible ways to add edges. Nonetheless, our exhaustive testing approach is not without its advantages, as we can use it to gain further knowledge on the cop number of small graphs.

In fact, Hosseini, Mohar and Gonzalez Hermosillo de la Maza [**40**] have recently showed that studying the cop number for graphs with $\Delta \leq 3$ is of interest for the study of the cop number at large. In this regard, we consider that getting a distribution of the cop-number of small subcubic graphs might be interesting, even if it is somewhat skewed by adding the condition $\delta \geq 2$. Our computations show that not only there are no 4-cop-win subcubic graphs on at most 20 vertices, but that subcubic 3-cop-win graphs are overwhelmingly rare for these orders.

The exhaustive search approach also gives us progress on a related problem. Arguably the most well-known result on the game of cops and robbers is Aigner and Fromme's proof that the cop number of any planar graph is at most 3, see [**2**]. This yields the analogous question of finding the minimum order of 3-cop-win planar graphs, and an enumeration of such graphs. The smallest known planar 3-cop-win graph is the dodecahedral graph, see Figure 4.8. It is easy to see that this graph requires 3 cops, as it has girth 5 and is 3-regular. It has been asked whether the dodecahedral graph is the unique smallest 3-cop-win planar graph, first in [**4**], as well as in [**17**].

There are some partial results for this problem. In [**39**], Hosseini proved that a minimal 3-cop-win planar graph must be 2-connected. Furthermore, Pisantechakool and Tan have shown in [**53**] that any planar graph on 19 or fewer vertices must contain a winning position

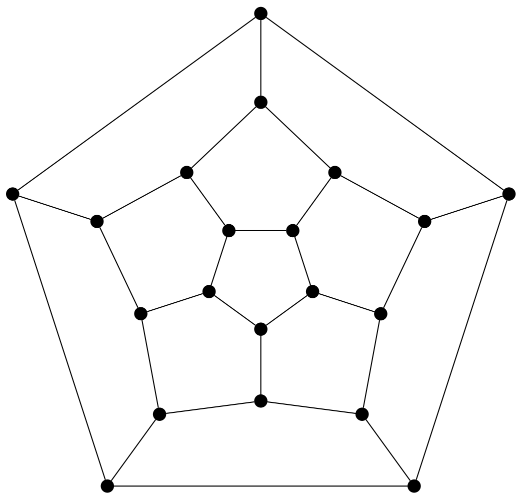


**Fig. 4.8.** The dodecahedral graph[2]

for 2 cops, although it has not been proved that the cops can bring the game to this winning state. Using the computations in the proof of Proposition 4.5.1, we are able to get more evidence supporting the conjecture.

**Corollary 4.5.2.** *If $G$ is a connected planar graph such that $\Delta \leq 3$ and $n \leq 20$, then $c(G) \leq 2$, unless $G$ is the dodecahedral graph.*

Proof. We simply test the 3-cop-win graphs found in the proof of Proposition 4.5.1 for planarity [**41**]. The only graph which was planar was the dodecahedral graph. □

## 4.6. Remaining cases

In this section, we consider the few remaining cases needed to prove that $M_4 = 19$, and also work towards reducing the possible 4-cop-win graphs on 19 vertices. More precisely, we consider graphs such that $n = 17$ with $\Delta = 4$, $n = 18$ with $\Delta = 4,5$, and $n = 19$ with $\Delta = 4$.

As in Section 4.4, our main tool will be knowing that if a graph $G$ is 4-cop-win, then for each vertex $u$, $c(G - N[u]) \geq 3$. We know there are relatively few such graphs. Since we will be attempting to construct minimal 4-cop-win graphs, we know that $c(G - N[u]) < 4$ and so $c(G - N[u]) = 3$. In the cases of $\Delta = n - 11$ or $\Delta = n - 12$, these graphs were only the Petersen and cornered Petersen graphs. As these were very few and very similar, we were able to build structural properties that allowed us to show that the graphs were not 4-cop-win. As they had somewhat a large maximum degree, a computational approach would have been difficult due to the fact that there are too many possible edges we need to consider.

[2]Computer-generated drawing [**41**].

For the cases we will now consider, a computational approach is possible, while a formal approach would be difficult, although certainly not impossible given a large amount of time. Most graphs found in Lemma 4.3.3 contain the Petersen graph as an induced subgraph, so modifying the strategy to take these vertices into account would most likely be possible. But, just as we saw, adding even a single vertex yields significant complications for the proof. Furthermore, some of the graphs do not contain the Petersen graph as an induced subgraph, and would need to be considered separately. As a final blow, this proof method would not scale very well, as the more vertices we add the further away from Petersen graphs we stray. For these reasons, we have mostly investigated the computational approach.

Our goal is to build graphs which are possibly 4-cop-win: graphs $G$ for which we cannot say that $c(G) \leq 3$ simply by looking at $G - N[u]$ for the vertices $u$ of maximum degree. Throughout, we will call these graphs *candidate 4-cop-win graphs*.

The simplest idea, which we have briefly discussed in Section 4.3, would be simply to consider a 3-cop-win graph $G'$ on 12 or 13 vertices, add a vertex $u$ of chosen maximum degree and its neighbourhood, and then look at every possible ways of joining $N(u)$ to $G'$ by respecting the maximum degree condition. Even by reducing the number of cases by isomorphism, the number of graphs to consider is massive, especially in the case $\Delta = 5$. We must be a tad smarter. We present the Merging Algorithm as a way to generate candidate 4-cop-win graphs, which we then test using a standard cop-number algorithm.

We briefly introduce some notation. In general, when considering a graph $G$ and a vertex $u$, the degree of $u$ will always refer to the degree of $G$ in $u$. If we want to discuss the degree of $u$ in some induced subgraph $H$, we will refer to it as the $H$-degree of $u$. In general, if we say there exists a vertex of $H$-degree $r$, we are also implicitly stating that this vertex is in $H$.

### 4.6.1. Presentation of the Merging Algorithm

4.6.1.1. Quick Overview. Our approach to build candidate 4-cop-win graphs will be the following. Let $v_1$ and $v_2$ be non-adjacent vertices, which we will in general choose to be a pair with the highest possible degree.

Then, knowing the computational results of Section 4.3, we are able to determine every possible option for $G_1 = G - N[v_2]$ and $G_2 = G - N[v_1]$. We denote by $L_1$ and $L_2$ the sets of 3-cop-win graphs in which $G_1$ and $G_2$ are respectively chosen from.

We want to determine every possible graph $G$, with maximum degree $\Delta$, which can be formed with this structure. We will call the process the Merging Algorithm, which we will now describe.

4.6.1.2. Input of the Algorithm. Integers $n$,$D_1$,$D_2 = \Delta$ and sets of isomorphism classes of graphs $L_1$ and $L_2$, such that

(1) the graphs in $L_1$ and $L_2$ are 3-cop-win and have maximum degree at most $\Delta$,
(2) the graphs in $L_1$ have $n - D_2 - 1$ vertices and the graphs in $L_2$ have $n - D_1 - 1$ vertices.

4.6.1.3. Output of the Algorithm. The algorithm returns all connected graphs $G$ on $n$ vertices and maximum degree exactly $\Delta$ which contain a pair of non-adjacent vertices $v_1$ and $v_2$, with the following 4 properties. Denote $G_1 = G - N[v_2]$ and $G_2 = G - N[v_1]$. Then,

(1) $v_1$ and $v_2$ have degree respectively $D_1$ and $D_2$,
(2) $G_1 \in L_1$ and $G_2 \in L_2$, and
(3) for all other vertex $u$ of degree $\Delta$, $G - N[u] \in L_1$, and
(4) if $D_1 < D_2$, then the set of vertices of $G$ of maximum degree forms a clique and $v_1$ and $v_2$ have at least 1 common neighbour.

Isomorphic graphs may be omitted from the results, as we are not interested in the precise labellings of the graphs.

4.6.1.4. Phase 1 of the Algorithm. We first choose some $G_1$ and $G_2$ from $L_1$ and $L_2$ respectively, we will repeat the rest of the algorithm for each possible choice of $G_1$ and $G_2$. We also choose strictly positive integers $d_1$ and $d_2$ such that $D_2 - d_2 = D_1 - d_1$, $d_1 \leq D_1$ and $d_2 \leq D_2$, we will also consider every possible choice.

We then consider every possible choice of $v_1 \in V(G_1)$ and $v_2 \in V(G_2)$ such that $v_1$ has $G_1$-degree $d_1$ and $v_2$ has $G_2$-degree $d_2$ (we can of course choose $v_1$ and $v_2$ up to automorphism in $G_1$ and in $G_2$). For each choice of vertices, consider every possible way of identifying $G_1 - N[v_1]$ and $G_2 - N[v_2]$, by computing every isomorphism between these graphs. If there are none, this branch of the algorithm simply doesn't yield a graph. Using this identification, we may then merge the graphs by union, keeping the closed neighbourhoods of $v_1$ and $v_2$ distinct.

If this process has created vertices of degree greater than $\Delta$, we throw out the graph, as the rest of the algorithm can only raise the degree again, yielding graphs we do not want to consider.

We now add vertices which are not in $V(G_1) \cup V(G_2)$. The only vertices which are neither in $V(G_1)$ and $V(G_2)$ are those which are to be adjacent to both $v_1$ and $v_2$. Thus, we add $D_2 - d_2 = D_1 - d_1$ common neighbours to $v_1$ and $v_2$, which ensures that the degrees $D_1$ and $D_2$ are respected. All vertices of $G$ are now in the graph, but there possibly exists some missing edges, which we will add in the second phase. The result of the current phase is called a *base graph*. Illustrated in Figure 4.9 is such a base graph.

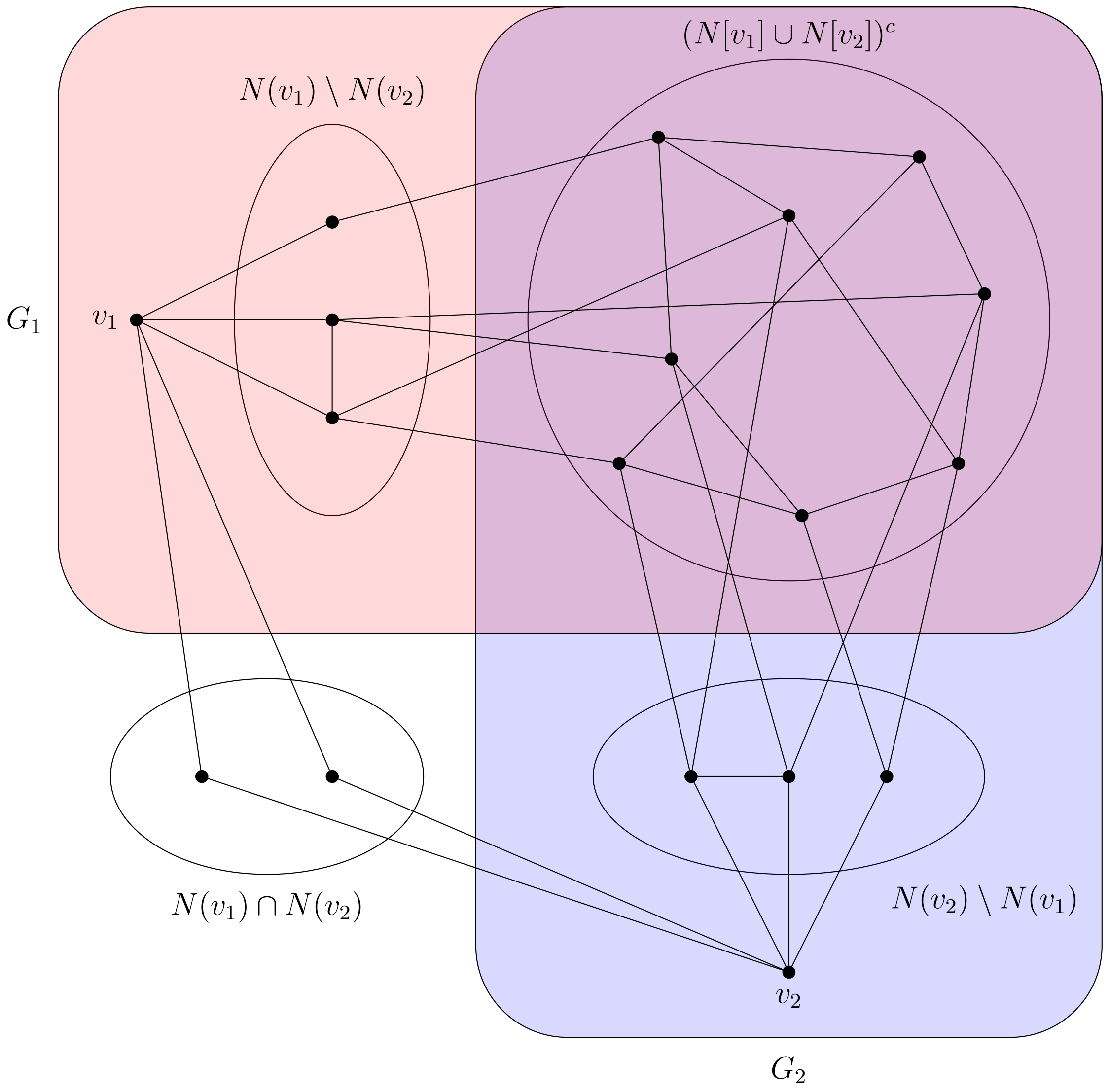


**Fig. 4.9.** Example of Phase 1 of the Merging Algorithm. Here, the base graph was generated using parameters $n = 18$, $D_1 = D_2 = \Delta = 5$ and $d_1 = d_2 = 3$.

It is easily seen in Figure 4.9 that the construction implicitly partitions the vertices into six sets : $\{v_1\}$,$N(v_1) \setminus N(v_2)$,$V(G) \setminus (N[v_1] \cup N[v_2])$,$N(v_1) \cap N(v_2)$,$N(v_2) \setminus N(v_1)$,$\{v_2\}$. If two graphs $G$,$G'$ are generated with the same properties (same choices of $G_1$,$G_2$,$v_1$,$v_2$ but by choosing a different identification), we may be able to reduce the number of cases to consider: if $\phi$ is an isomorphism between $G$ and $G'$ such that $\phi(S) = S$ for each $S$ being one of these 6 sets (we call this a strong isomorphism), we can consider to these graphs to be duplicates: each base graph, once the algorithm is over, will be transformed into the same candidate 4-cop-win graphs (again, up to isomorphism).

4.6.1.5. Phase 2 of the Merging Algorithm. The goal of this phase is to complete the 4-cop-win candidates graphs. As the base graph contain all required vertices, we now need to add the missing edges. We know that we do not want to add any edge such that both ends are in $G_1$ or both ends in $G_2$ as these are chosen to be induced subgraphs of $G$. Furthermore, we have already created all incident edges to either $v_1$ or $v_2$, by giving them the desired number of neighbours. Thus, we only need to consider adding edges which are either

(1) between $N(v_1) \cap N(v_2)$ and $\{v_1,v_2\}^c$, including edges with both ends in $N(v_1) \cap N(v_2)$, or
(2) between $N(v_2) \setminus N(v_1)$ and $N(v_1) \setminus N(v_2)$.

We proceed by considering every vertex (first those in $N(v_1) \cap N(v_2)$, then those in $N(v_2) \setminus N(v_1)$) and creating a new graph for every possible subset of new edges. Of course, we only consider subsets such that the degree will be at most $\Delta$. We repeat this step on the new graphs for the next vertex.

We are also able to reduce some cases by isomorphism in this case. As above, we consider two graphs to be equivalent if at any step in the process the two graphs can be related by some isomorphism which has the property that the final graphs they will generate will be identical, up to isomorphism. The additional consideration here is that we must distinguish vertices for which we have already considered adding extra edges and those for which this remains to be done. This additional piece of information is crucial to ensure we are indeed considering every possible set of additional edges. Thus, the condition will be that the isomorphism $\phi$ not only preserves the six sets as above, but also identifies the vertices in $N(v_2)$ for which we have not yet run the second part of the algorithm with other vertices with the same property. As this procedure is often lengthy, we only apply this improvement on lists of graphs of reasonable length.

After considering every possible way of adding edges, we can throw out all graphs $G$ such that $G - N[u]$ is not a 3-cop-win graph for every vertex $u$ of maximum degree (by construction, we do not need to verify this for $v_1$ and $v_2$). We can also also remove isomorphic graphs. We note that as we have split up the computations in many pieces, we only compare for isomorphism graphs which were generated with the same choice of $G_1$.

4.6.1.6. Specific cases. As one can deduce from the parameters and output section, the algorithm can be divided in two main cases.

In the first case, $D_1 = D_2$. In other words, the resulting graphs contain non-adjacent vertices $v_1$ and $v_2$ of maximum degree. In general, we apply the merging for every possible way (up to automorphism) of choosing vertices $v_1$ in $G_1$ and $v_2$ in $G_2$. Applying this naively may yield multiple isomorphic graphs, as different choices of $v_1$ in $G_1$ can yield isomorphic graphs. We can tweak the algorithm to partially avoid this problem.

For each graph $G_1$ in $L_1$, we first define a total ordering on its vertices as follows. We list the vertices by decreasing degree, where the choice of order on vertices of same degree is arbitrary except that vertices which are equivalent by automorphism are consecutive in this order (or we could define the ordering on the classes of vertices up to automorphism). Then, when considering some choice of $v_1$, we will add for the remaining of the algorithm the restriction that vertices greater than $v_1$ in this order do not have maximum degree in $G$. We will do the computations by decreasing $v_1$. In essence, the graphs $G$ where the vertices which are greater in this order have maximum degree in $G$ will already have been considered in the algorithm.

A particular case of the above is when $G_1$ already contains multiple vertices of degree $\Delta$. When choosing any vertex $v_1$ other than the vertex $u$ which is maximal in the chosen order, no graph will be generated: $u$ would have degree $\Delta$ in $G$, which we have excluded. For this reason, when $G_1$ contains a vertex of degree $\Delta$ we not even try and simply do the merging algorithm for one choice of $v_1$ only, that is we only consider $d_1 = D_1 = \Delta$.

It is not directly obvious that this simplification is compatible with the one we described earlier, which was considering strongly isomorphic partially-constructed graphs equivalent. It suffices to see that automorphically equivalent vertices of $G_1$ have the same restriction on their degree: either both are allowed to have degree $\Delta$ in $G$ or neither is. Indeed, this property is preserved when considering the strong isomorphism $\phi$ between two of the partially constructed graph : as $\phi$ preserves in particular $\{v_1\}$, $N(v_1)\backslash N(v_2)$ and $V(G)\backslash(N[v_1]\cup N[v_2])$, we know that $\phi$ restricted to the vertices of $G_1$ is an isomorphism of $G_1$.

In the second case of the algorithm, $D_1 < D_2$. We no longer apply the improvements to the Merging Algorithm we described in the previous case, but we still apply some minor modifications to the base algorithm to fulfill the condition (4) of Section 4.6.1.3. Observe that at any point in the algorithm, if the graph contains non-adjacent vertices both of degree $\Delta$, we can throw out this graph, since the vertices will also be non-adjacent in the final graphs. We also only test for choices of $d_1$ such that $d_1 < D_1$ (and thus $d_2 < D_2$) to only build graphs where $v_1$ and $v_2$ have common neighbours.

4.6.1.7. Validity of the Algorithm. Considering the algorithm itself is relatively straightforward, we do not present a complete proof of the validity of the algorithm. We however present a few key points towards a formal proof.

Consider a graph $G$ respecting the conditions described in the Section 4.6.1.3. Choose $v_2$ to be any vertex of degree $D_2$ in $G$ such that $G - N[v_2]$ contains at least one vertex respecting the conditions for $v_1$ in Section 4.6.1.3. Denote $S$ this non-empty set of possible choices for $v_1$. For some choice of $v_1$, we set $G_1 = G - N[v_2]$, $G_2 = G - N[v_1]$, $D_1 = d(v_1)$, $D_2 = d(v_2)$, $d_1 = d_{G_1}(v_1)$ and $d_2 = d_{G_2}(v_2)$.

In the case $D_1 < D_2$, we choose $v_1$ to be any vertex of $S$. If $D_1 = D_2$, then choose $v_1$ to be maximal in $S$ relative to the order on the vertices of $G - N[v_2]$ as described in the previous section.

It is easy to verify that

$$\begin{aligned}
d_2 - d_1 &= d_{G_2}(v_2) - d_{G_1}(v_1) \\
&= |N_{G-N[v_1]}(v_2)| - |N_{G-N[v_2]}(v_1)| \\
&= |N(v_2) \setminus (N[v_1] \cap N(v_2))| - |N(v_1) \setminus (N(v_1) \cap N[v_2])| \\
&= |N(v_2) \setminus (N(v_1) \cap N(v_2))| - |N(v_1) \setminus (N(v_1) \cap N(v_2))| \\
&= (|N(v_2)| - |N(v_1) \cap N(v_2)|) - (|N(v_1)| - |N(v_1) \cap N(v_2)|) \\
&= |N(v_2)| - |N(v_1)| \\
&= D_2 - D_1.
\end{aligned}$$

Thus, the pair of degrees $d_1$ and $d_2$ is indeed considered in the Merging Algorithm. It is then easy to see that the first part of the algorithm has considered this case. In particular, for the case $D_1 = D_2$, choosing $v_1$ as maximal in $S$ implies that all vertices of $G - N[v_2]$ which are greater in this order do not have maximal degree in $G$, which is consistent with the simplification that we implemented.

Then, in the second part of the algorithm we consider adding every possible edge not totally contained in $G_1$ or $G_2$ (or at least, up to isomorphism), while still respecting some degree conditions (which we have just seen to be consistent). We thus see that $G$ has indeed been constructed by the Merging Algorithm.

## 4.6.2. Results

We will now use this algorithm to build all possible 4-cop-win graphs. We will use some additional heuristics in some cases to reduce the number of cases to consider, which we will explain in detail in the proof of the following proposition. Our implementation of the algorithm is done in the Wolfram language [**41**].

**Proposition 4.6.1.** *Let $G$ be a connected graph such that either*

*(1) $n = 17$ and $\Delta = 4$,*
*(2) $n = 18$ and $\Delta \in \{4,5\}$, or*

*(3) $n = 19$ and $\Delta = 4$.*

*If every proper induced connected subgraph $H$ of $G$ respects $c(H) \leq 3$, then $c(G) \leq 3$, unless $G$ is the Robertson graph.*

Proof. Let $u$ be any vertex of $G$. We know that $G - N[u]$ has at most $n - 2 \leq 17$ vertices. If $G - N[u]$ is disconnected, it must contain at least one component $K$ which has at most 8 vertices. Then, Theorem 4.2.1 implies that $c(K) \leq 2$. Furthermore, our hypothesis implies that $c(G-K) \leq 3$, as $G-K$ is necessarily a connected induced subgraph of $G$. By Corollary 4.2.5, we get that $c(G) \leq \max\{c(G-K),c(K)+1\} \leq 3$. Thus, for the remainder of the proof, we assume that for every vertex $u$, $G - N[u]$ is connected.

Likewise, we can assume that $G$ does not contain a corner $x$. Indeed, $G - x$ is necessarily connected and has cop number at most 3, therefore Corollary 4.2.8 would then imply that $G$ also has cop number at most 3.

We may also assume that $c(G - N[u]) = 3$: if $c(G - N[u]) \leq 2$, placing a stationary cop on $u$ implies that $c(G) \leq 3$.

Before going further, we define a property $P$ with the usual definition: a property $P$ is a function from a set to a Boolean value. For instance, if $C_3$ is the graph property of being 3-cop-win, then $C_3(\mathcal{P}_0)$ is whether the Petersen graph is 3-cop-win (which is true).

With this language, we can bring together the last assumptions. We define property $M$ as follows : $G$ is a graph respecting the hypotheses of the proposition and such that $G - N[u]$ is a connected 3-cop-win graph for every vertex $u$ of $G$ and such that $G$ does not contain a corner. By the previous discussion, it suffices to show the proposition for graphs respecting $M$.

We now define property $P_1$. A graph $G$ is said to have property $P_1$ if $G$ contains two non-adjacent vertices of degree $\Delta$. We use the Merging Algorithm to generate all graphs $G$ such that $M(G)$ that respect property $P_1$, and then compute their cop numbers. More precisely, we choose $n$ and $\Delta$ according to the case we are consider, $D_1 = D_2$, and $L_1 = L_2$ to be the set of 3-cop-win graphs on $n - \Delta - 1$ vertices with maximum degree at most $\Delta$, as computed in Lemma 4.3.3. We note that the Merging Algorithm computes a somewhat larger class of graphs than we want. In particular, the Merging Algorithm does not exclude graphs which contain corners and in its last step only tests $G - N[u]$ for vertices of maximum degree.

The summary results are presented in Table 4.5. For more detail, we also split up the graphs relative to the various possible maximum degrees of $G_1$, although we of course always merge with all of the possible graphs $G_2$, not only the $G_2$ with the same maximum degree. We note that there are no 3-cop-win graphs with maximum degree 3 on 13 vertices (which can also be seen in Table 4.4), and that the 3-cop-win graphs with maximum degree 3 on 12 and 14 vertices are 3-regular (and thus the only possible value of $d_1$ is 3).

| | | | | | | | Cop number | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\Delta$ | $n$ | $\Delta_1$ | $G_1$ | $d_1$ | Base graphs | Final graphs | 1 | 2 | 3 | $\geq 4$ |
| 4 | 17 | 4 | 78 | 4 | 123 | 0 | 0 | 0 | 0 | 0 |
| | | 3 | 2 | 3 | 10 | 0 | 0 | 0 | 0 | 0 |
| | 18 | 4 | 1105 | 4 | 1668 | 0 | 0 | 0 | 0 | 0 |
| | 19 | 4 | 16514 | 4 | 33785 | 3 | 0 | 0 | 0 | 3 |
| | | 3 | 9 | 3 | 911 | 0 | 0 | 0 | 0 | 0 |
| 5 | 18 | 5 | 93 | 5 | 14232 | 24416 | 0 | 5484 | 18932 | 0 |
| | | 4 | 78 | 4 | 10062 | 39318 | 0 | 7410 | 31908 | 0 |
| | | | | 3 | 534 | 18645 | 0 | 3455 | 15190 | 0 |
| | | | | 2 | 111 | 24238 | 0 | 1494 | 22744 | 0 |
| | | | | 1 | 88 | 698809 | 0 | 82882 | 615927 | 0 |
| | | 3 | 2 | 3 | 22 | 12778 | 0 | 4960 | 7818 | 0 |

**Table 4.5.** Results of the first wave of computations using the Merging Algorithm. It presents the counts for the graphs built with the property that they contain 2 non-adjacent vertices of maximum degree. In particular, $d_1 = d_2$ and $\Delta = D_1 = D_2$. Furthermore, $G_1$ is chosen with maximum degree $\Delta_1$.

We note that the 4-cop-win graphs found on 19 vertices are actually all copies of the Robertson graph, which can be see in Figure 4.1. In fact, the 3 copies correspond to 3 different choices of $G_1$ which can yield the Robertson graph.

It is also interesting to note that for all cases with $\Delta = 4$, not only is the Robertson graph the only 4-cop-win graph, but there are no other candidate 4-cop-win graphs. It would appear that when merging, too many vertices of high degree are created: either a vertex of degree 5 or more is created (in which case the graph is immediately thrown out) or there are "too many" vertices of degree 4, such that there is always some $u$ of maximum degree for which $G - N[u]$ not 3-cop-win.

With these results, we will then only consider graphs which do not have property $P_1$. In other words, the graphs left to consider are those such that the set of vertices of maximum degree of $G$ forms a clique. This is a very restrictive property, and will be very useful.

Note that graphs $G$ such that $M(G)$ and $\Delta = 4$ respect property $P_1$: let $u$ be a vertex of maximum degree in $G$. Consider $G' = G - N[u]$. If $G'$ contains a vertex of degree 4, $P_1(G)$ is satisfied. Otherwise, we must have $\Delta(G) = 3$. If $G'$ is not 3-regular, it is at most 2 cop-win (by the results mentioned above) and therefore $G$ is not a 4-cop-win candidate. Therefore, any vertex in $G'$ that was adjacent to a vertex of $N(u)$ is also of degree 4 in $G$ and not adjacent to $u$. Thus, $P_1(G)$ is verified. We can therefore suppose $\Delta(G) = 5$. Furthermore, since $P_1(G)$ is false, we can assume that if there exists two vertices of maximum degree, they must be adjacent (as they must form a clique: otherwise, $P_1$ is satisfied).

We now define property $P_2$. We say a graph $G$ has property $P_2$ if $G$ contains two non-adjacent vertices $v_1$ and $v_2$ such that $v_1$ has degree either 3 or 4, $v_2$ has degree 5, $v_1$ and $v_2$ have a common neighbour, and $G - N[v_1]$ has maximum degree at most 4. Then, we compute the graphs $G$ such that $M(G)$ and $P_2(G)$, but not $P_1(G)$. Precisely, we set $n = 18$, $D_2 = \Delta = 5$, $D_1$ to either 3 or 4, $L_1$ to be the 3-cop-win graphs on 12 vertices with maximum degree at most 4 (if we choose $G_1$ with maximum degree 5, then the generated graphs automatically respect property $P_1$), and $L_2$ to be the 3-cop-win graphs on respectively either 14 or 13 vertices with maximum degree at most 4. We have computed these lists $L_1$ and $L_2$ in Lemma 4.3.3. The results of this computation are presented in Table 4.6.

We note that as the number of possible vertices of maximum degree is generally smaller than before, there are fewer graphs thrown out because for some $u$ of maximum degree $G - N[u]$ is not a 3-cop-win graph. Furthermore, we note that as the graphs on 14 vertices with maximum degree 3 are 3-regular, when choosing any of these graphs as $G_1$ it is impossible for $d_1$ to be anything other than 3.

| | | | | | | | Cop number | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $D_1$ | $G_2$ | $\Delta_1$ | $G_1$ | $d_1$ | Base graphs | Final graphs | 1 | 2 | 3 | $\geq 4$ |
| 4 | 1105 | 4 | 78 | 3 | 993 | 41872 | 0 | 9299 | 32573 | 0 |
| | | | | 2 | 504 | 70224 | 0 | 4278 | 65946 | 0 |
| | | | | 1 | 1138 | 3350712 | 0 | 417144 | 2933568 | 0 |
| | | 3 | 2 | 3 | 153 | 41006 | 0 | 15440 | 25566 | 0 |
| 3 | 16523 | 4 | 78 | 2 | 2419 | 83509 | 0 | 4187 | 79322 | 0 |
| | | | | 1 | 10582 | 6293171 | 0 | 786173 | 5506998 | 0 |

**Table 4.6.** Results of the second wave of computations with the Merging Algorithm. It presents the counts for the graphs $G$ built with the property that $G - N[v_1]$ has maximum degree 4, $v_1$ and $v_2$ always have a common neighbour (in particular $d_1 < D_1$) and the vertices of maximum degree form a clique. Here, we always have $n = 18$ and $D_2 = \Delta = 5$, and $G_1$ is chosen with maximum degree $\Delta_1$.

We see that none of the graphs are 4-cop-win. We claim that all graphs $M(G)$ implies that either $P_1(G)$ or $P_2(G)$.

Let $G$ be as graph such that $M(G)$. If $P_1(G)$; we are done. Let us then consider that $P_1(G)$ is false and show that $P_2(G)$ must be true. As discussed earlier, we may only consider the case where $\Delta = 5$.

We first suppose that $G$ contains a unique vertex of degree 5 and show that either $P_2(G)$ holds or there is a contradiction. Let $v_2$ be such a vertex. Then, for any choice of $v_1$ in $N[v_2]^c$, if $v_1$ and $v_2$ have a common neighbour, $G - N[v_1]$ has maximum degree at most 4, as the only vertex of degree 5 (which is $v_2$) has lost one of its neighbours. Thus, if such a $v_1$ has degree either 3 or 4 in $G$, we know that $G$ has property $P_2$. In other words, no vertex of $G_1$-degree 2 or 3 has a neighbour in $N(v_2)$, and no vertex of $G_1$-degree 1 has more than 1 neighbour in $N(v_2)$. Indeed, any of these cases gives a vertex $v_1$ degree 3 or 4 with a common neighbour with $v_2$. Therefore, only vertices of $G_1$-degree 1 can "receive" an edge from $N(v_2)$, and even then they can only receive 1 each. As $G$ does not contain a corner, each vertex in $N(v_2)$ must have at least 1 neighbour in $N[v_2]^c$. Thus, there are at least 5 edges between $N(v_2)$ and the vertices of $N[v_2]^c$. By the previous argument, there must then be at least 5 vertices of degree 1 in $G_1$, to be able to receive these edges. This is impossible: if $G_1$ contains at least 5 vertices of degree 1, removing them (which does not change the cop number of $G_1$), yields a graph with 7 vertices, which has cop number at most 2. In fact, $G_1$ cannot have more than 2 vertices of degree 1. We reached the desired contradiction.

Thus, we may assume that $N(v_2)$ contains at least one other vertex of degree 5. We recall that these vertices of degree 5 must form a clique as $G$ does not respect property $P_1$. Suppose $v_1$ is a vertex of $N[v_2]^c$ of degree either 3 or 4 that has a neighbour of degree 5. Then, as the vertices of degree 5 are all pairwise adjacent, removing $N[v_1]$ removes at least 1 neighbour from each vertex of degree 5: $G - N[v_1]$ necessarily has maximum degree at most 4. Thus, graphs with this property have property $P_2$.

Let us make sure such a choice of $v_1$ exists. Using a similar argument as above, the vertices of degree 5 can only have neighbours in $N[v_2]^c$ which have $G_1$-degree 1, and when this happens the receiving vertices can have no other neighbours in $N(v_2)$, otherwise $G$ automatically respects property $P_2$. In particular, we have that $N(v_2)$ contains either 1 or 2 vertices of degree 5.

We first consider the case with 2 such vertices, let us denote them $x_1$,$x_2$. Then, $x_1$ and $x_2$ each have exactly one neighbour in $N[v_2]^c$ (which are different). Thus, having degree 5, both $x_1$ and $x_2$ must have 3 neighbours in $N(v_2)$ (including each other). In particular, $x_1$ and $x_2$ must have a common neighbour in $N(v_2)$, denote it $y$. We know that $y$ must have a neighbour in $N[v_2]^c$ (otherwise $y$ is cornered by $v_2$), which will be our choice of $v_1$. As there are no other unaccounted vertices of $G_1$-degree 1 (these are adjacent to the degree 5 vertex and nothing else), $v_1$ necessarily has degree either 3 or 4. We can then see that $G - N[v_1]$ has maximum degree at most 4: removing $N[v_1]$ removes $y$, which is a common neighbour to $v_2$, $x_1$ and $x_2$. Thus, in this case, the graphs have property $P_2$.

We can similarly consider the case with exactly 1 vertex $x$ of degree 5 in $N(v_2)$. Then, $x$ has at most 2 neighbours in $N[v_2]^c$. Thus, $x$ has at least 2 neighbours $y_1$,$y_2$ in $N(v_2)$. We have already seen that they cannot be adjacent to the neighbours of $x$ in $N[v_2]^c$. If one of them is adjacent to a vertex of $G_1$-degree 2 or 3, then this will be a valid choice for $v_1$, as

$N[v_1]$ then contains a vertex adjacent to both vertices of degree 5. The only remaining case is if there is a vertex of $G_1$-degree 1 in $G_1$ to which $x$ is not adjacent, and both $y_1$,$y_2$ are adjacent. In this case, choose $v_1$ to be this vertex. It has degree at least 3, and removing it removes neighbours of all vertices of degree 5. In all cases, the graph has property $P_2$.

Thus, our claim is verified: all graphs $G$ that satisfy $M(G)$ respect either $P_1(G)$ or $P_2(G)$. We have computer the cop number of graphs such that $M(G)$ and $P_1(G)$, and graphs such that $M(G)$, $P_2(G)$ but not $P_1(G)$: we have computed the cop number of all graphs such that $M(G)$. This proves the current proposition. □

### 4.6.3. Possible improvements

This is only one of many possible computational approaches to solving the problem. We now discuss a few improvements and alternatives that the interested reader may want to apply.

Our approach was based on merging 3-cop-win graphs by looking at non-adjacent vertices $v_1$,$v_2$. It is easy to see that one could instead choose $v_1$ and $v_2$ to be adjacent. Even if the construction would be somewhat different, the ideas are similar. In particular, after proving that $G$ does not contain non-adjacent vertices of maximum degree, we could have proved that $G$ does not contain any adjacent vertices of maximum degree, instead of considering $v_1$ of smaller degree. This would then leave only the case with a single vertex of maximum degree to be treated. With some additional heuristics or with a simplification of the methods we used, this could be dealt with more specifically.

We decided against this approach for few reasons. Although our approach required us to compute more 3-cop-win graphs than otherwise, it allowed us to implement only one Merging Algorithm. Furthermore, computing the 3-cop-win graphs on 14 vertices with maximum degree (at most) 4 allowed us to simultaneously handle on the case on $n = 18$ with $d(v_1) = 3$, and build the candidate 4-cop-win graphs on 19 vertices with maximum degree 4.

Another method would be to not only merge graphs relative to pairs of vertices, but varying sizes of subsets. This approach would certainly reduce the number of intermediate graphs generated by the algorithm: instead of pruning out graphs after adding edges, we could build up a larger part of the graph. The difficulty lies in implementing this approach. In particular, we must keep track of which pairs of vertices do not have an edge because both vertices are in one of the $G_i$ but this edge is not present in that graph, or whether such an edge could be considered later.

Although at the expense of some computation time, we have chosen not to implement these improvements in order to keep the code as simple as possible. Indeed, the simplicity of the code reduces the chances of it being erroneous, as well as making it easier to verify. As the proof is completely dependent on the results of the algorithm, we felt this compromise was justified.

A last idea essentially combines the processes of generating the graphs and testing their cop number. Let $G$ be a connected graph and $e$ be some edge of $G$. In general, it is unclear whether removing $e$ will help the robber or help the cops, as this depends on many other factors. If we consider a slightly modified ruleset so that the robber can use the edge $e$ but not the cops, we might achieve some results. Denote $c'$ the cop number of this modified game. With these rules, $c(G) \leq c'(G)$, as the new edge can only benefit the robber. Furthermore, $c'(G-e) \leq c'(G)$ because, removing $e$ can only help the cops, as they were not allowed to use it anyways. Thus, both $c(G)$ and $c(G-e)$ are bounded above by $c'(G)$. If we modify the algorithm which calculates the cop number to take into account the robber-only edge $e$ (a fairly easy modification), we could then determine simultaneously whether both $G$ and $G-e$ have cop number at most 3. This generalizes to larger subsets of edges. Hence, in theory, we can reduce by a significant amount the number of cases to consider by not distinguishing the distinguishing $G$ and $G-e$; do this for many edges and the number of graphs to consider could decrease exponentially. It is not clear how many such "special edges" we can take in $G$ before the cop numbers diverge. We leave implementing and studying this approach as a problem. Modifying slightly the rules of the game to study the cop number has been done many times before. For instance, cop-only edges are studied in [**30**] and allowing the cops to teleport in [**46**].

## 4.7. Main results

We are now ready to prove the desired results.

**Theorem 4.7.1.** *If $G$ is a connected graph such that $n \leq 18$, then $c(G) \leq 3$.*

Proof. This is a direct consequence of Corollary 4.2.6 and Propositions 4.4.10, 4.5.1 and 4.6.1. □

Considering there exists a known 4-cop-win graph on 19 vertices, the Robertson graph, we get the following corollary.

**Corollary 4.7.2.** $M_4 = 19$

We also want to narrow down the possible 4-cop-win graphs on 19 vertices.

**Theorem 4.7.3.** *Let $G$ be a connected graph such that $n = 19$. If $\Delta \leq 3$ or $\Delta \geq 7$, then $c(G) \leq 3$. If $\Delta = 4$, then $c(G) \leq 3$, unless $G$ is the Robertson graph.*

Proof. This is a direct consequence of Corollary 4.2.6 and Propositions 4.4.14, 4.5.1 and 4.6.1. □

We leave filling the missing cases in this theorem as a conjecture.

**Conjecture 4.7.4.** *There does not exist a connected graph $G$ such that $n = 19$, $\Delta \in \{5,6\}$ and $c(G) = 4$.*

This would then show that the Robertson graph is the unique 4-cop-win graph on 19 vertices. With a better implementation of the algorithm, in some low overhead programming language such as C, and with a few good ideas, this problem seems within reach. On the other hand, finding $M_5$ with the methods used in this article is clearly unfeasible.

It is asked in [**7**] whether the minimum $d$-cop-win graphs are $(d,5)$-cage graphs for every $d$. Although we now have further evidence pointing towards this conjecture, any general proof of this statement is still beyond our grasp.

## Acknowledgments

We would like to thank Seyyed Aliasghar Hosseini for introducing us to this problem and for many helpful discussions. We would also like to thank Ben Seamone and Geňa Hahn for their support over the course of this project.

We also acknowledge that most computations were done on the Université de Montréal - Département de mathématiques et de statistique computer network.

We also wish to thank the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds de Recherche du Québec - Nature et technologies (FRQNT) for their financial support.

# Références bibliographiques

[1] Anton AFANASSIEV : Cop-Number, 2017. GitHub, *https://github.com/Jabbath/Cop-Number*. 81

[2] Martin AIGNER et Michael FROMME : A game of cops and robbers. *Discrete Applied Mathematics*, 8(1):1 – 12, 1984. 27, 28, 30, 34, 38, 39, 65, 66, 76, 77, 98

[3] Thomas ANDREAE : Note on a pursuit game played on graphs. *Discrete Applied Mathematics*, 9(2):111–115, octobre 1984. 35

[4] Thomas ANDREAE : On a pursuit game played on graphs for which a minor is excluded. *Journal of Combinatorial Theory, Series B*, 41(1):37–47, août 1986. 66, 76, 77, 78, 98

[5] László BABAI et Vera T. SÓS : Sidon Sets in Groups and Induced Subgraphs of Cayley Graphs. *European Journal of Combinatorics*, 6(2):101 – 114, 1985. 59

[6] John BAEZ : Petersen Graph, juillet 2015. *https://blogs.ams.org/visualinsight/2015/07/01/petersen-graph/*. 83

[7] William BAIRD, Andrew BEVERIDGE, Anthony BONATO, Paolo CODENOTTI, Aaron MAURER, John MCCAULEY et Silviya VALEVA : On the minimum order of k-cop-win graphs. *Contributions to Discrete Mathematics*, 9:70–84, 2014. 76, 77, 78, 79, 81, 112

[8] William BAIRD et Anthony BONATO : Meyniel's conjecture on the cop number: A survey. *Journal of Combinatorics*, 3, 2013. 30, 31, 36, 39, 61

[9] Roger C. BAKER, Glyn HARMAN et János PINTZ : The Difference Between Consecutive Primes, II. *Proceedings of the London Mathematical Society. Third Series*, 83, 2001. 61

[10] Alessandro BERARDUCCI et Benedetto INTRIGILA : On the Cop Number of a Graph. *Advances in Applied Mathematics*, 14(4):389 – 403, 1993. 29, 71, 79

[11] Andrew BEVERIDGE : The Petersen graph is the smallest 3-cop-win graph, mai 2012. GRAScan Workshop, *https://math.ryerson.ca/ abonato/GRASCan/AndrewBeveridge.pdf*. 77, 79

[12] Jeff BEZANSON, Alan EDELMAN, Stefan KARPINSKI et Viral B SHAH : Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. 80

[13] Béla BOLLOBÁS, Gábor KUN et Imre LEADER : Cops and robbers in a random graph. *Journal of Combinatorial Theory, Series B*, 103(2):226–236, mars 2013. 36

[14] Anthony BONATO : Conjectures on Cops and Robbers. *In Graph Theory: Favorite Conjectures and Open Problems - 1*, pages 31–42. Springer International Publishing, Cham, 2016. 76, 77

[15] Anthony BONATO et Andrea BURGESS : Cops and robbers on graphs based on designs. *Journal of Combinatorial Designs*, 21(9):404–418, 2013. 36, 61, 62

[16] Anthony BONATO, Ehsan CHINIFOROOSHAN et Paweł PRAŁAT : Cops and Robbers from a distance. *Theoretical Computer Science*, 411(43):3834 – 3844, 2010. 80, 117

[17] Anthony BONATO et Bojan MOHAR : Topological directions in Cops and Robbers. *Journal of Combinatorics*, 11, 2017. 30, 98

[18] Anthony BONATO et Richard NOWAKOWSKI : *The Game of Cops and Robbers on Graphs*. 2011. 76

[19] John Adrian BONDY et U. S. R. MURTY : *Graph Theory with Applications*. American Elsevier Publishing Company, 1976. 19

[20] Nathan BOWLER, Joshua ERDE, Florian LEHNER et Max PITZ : Bounding the cop number of a graph by its genus. novembre 2019. arXiv: 1911.01758. 30

[21] Peter BRADSHAW : A proof of the Meyniel conjecture for abelian Cayley graphs. *Discrete Mathematics*, 2019. 33, 39, 41, 42, 45, 49, 50, 54

[22] Peter BRADSHAW, Seyyed Aliasghar HOSSEINI, Bojan MOHAR et Ladislav STACHO : On the cop number of graphs of high girth. 2020. arXiv: 2005.10849. 35, 77

[23] Seth BROMBERGER, James FAIRBANKS et Other CONTRIBUTORS : JuliaGraphs/LightGraphs.jl: an optimized graphs package for the Julia programming language, 2017. 80

[24] Ehsan CHINIFOROOSHAN : A better bound for the cop number of general graphs. *Journal of Graph Theory*, 58(1):45–48, mai 2008. 32

[25] Fan-Rong King CHUNG, András GYÁRFÁS, Zsolt TUZA et William T. TROTTER : The maximum number of edges in 2K2-free graphs of bounded degree. *Discrete Mathematics*, 81(2):129 – 135, 1990. 68, 72

[26] Nancy E. CLARKE et Gary MACGILLIVRAY : Characterizations of k-copwin graphs. *Discrete Mathematics*, 312(8):1421 – 1425, 2012. 29, 80, 117

[27] Reinhart DIESTEL : *Graph Theory*. Numéro 173 de Graduate Texts in Mathematics. Springer, 2017. 19, 23

[28] David S. DUMMIT et Richard M. FOOTE : *Abstract algebra*. John Wiley and Sons, 2004. 24, 26

[29] Peter FRANKL : Cops and robbers in graphs with large girth and Cayley graphs. *Discrete Applied Mathematics*, 17(3):301 – 305, 1987. 31, 35, 39, 77

[30] Peter FRANKL : On a pursuit game on Cayley graphs. *Combinatorica*, 7(1):67–70, mars 1987. 39, 41, 42, 45, 47, 48, 60, 111

[31] Alan FRIEZE, Michael KRIVELEVICH et Po-Shen LOH : Variations on Cops and Robbers. *Journal of Graph Theory*, 69(4):383–402, février 2011. 32, 39

[32] Tomáš GAVENČIAK, Przemysław GORDINOWICZ, Vít JELÓNEK, Pavel KLAVÍK et Jan KRATOCHVÍL : Cops and Robbers on intersection graphs. *European Journal of Combinatorics*, 72:45 – 69, 2018. 39

[33] Sebastian Gonzalez Hermosillo de la MAZA, Seyyed Aliasghar HOSSEINI, Fiachra KNOX, Bojan MOHAR et Bruce REED : Cops and robbers on oriented toroidal grids. *arXiv:1904.10113*, avril 2019. 39

[34] Geňa HAHN : Définitions IFT3545/MAT6490, 2020. *http://www-labs.iro.umontreal.ca/~hahn/IFT3545/definitions.pdf*. 19

[35] Yahya Ould HAMIDOUNE : On a Pursuit Game on Cayley Digraphs. *European Journal of Combinatorics*, 8(3):289 – 295, 1987. 39, 41, 42, 55, 60

[36] Fatemeh HASIRI et Igor SHINKAR : Meyniel Extremal Families of Abelian Cayley Graphs. septembre 2019. arXiv: 1909.03027. 36, 59

[37] Seyyed Aliasghar HOSSEINI : *Game of Cops and Robbers on Eulerian Digraphs*. PhD Thesis, Simon Fraser University, 2018. 39

[38] Seyyed Aliasghar HOSSEINI : A note on k-cop-win graphs. *Discrete Mathematics*, 341(4):1136 – 1137, 2018. 76

[39] Seyyed Aliasghar HOSSEINI et Bojan MOHAR : Game of cops and robbers in oriented quotients of the integer grid. *Discrete Mathematics*, 341(2):439 – 450, 2018. 39, 77, 78, 80, 82, 98

[40] Seyyed Aliasghar HOSSEINI, Bojan MOHAR et Sebastian Gonzalez Hermosillo de la MAZA : Meyniel's conjecture on graphs of bounded degree. décembre 2019. arXiv: 1912.06957. 98

[41] Wolfram Research Inc. : Mathematica. 47, 77, 99, 105

[42] Gwenaël Joret, Marcin Kaminski et Dirk Oliver Theis : The Cops and Robber game on graphs with forbidden (induced) subgraphs. *Contributions to Discrete Mathematics*, 5, 2008. 39, 66, 67

[43] Devvrit Khatri, Natasha Komarov, Aaron Krim-Yee, Nithish Kumar, Ben Seamone, Virgélot Virgile et AnQi Xu : A study of cops and robbers in oriented graphs. *arXiv:1811.06155*, janvier 2019. arXiv: 1811.06155. 39, 61

[44] László Kozma : *Useful inequalities.* 2020. *http://www.lkozma.net/inequalities_cheat_sheet/ineq.pdf.* 52

[45] Ji Chang Kuang : *Changyong budengshi.* Hunan Jiaoyu Chubanshe, Changsha, fourth édition, 2010. 52

[46] Florian Lehner : On the cop number of toroidal graphs. 2019. arXiv: 1904.07946. 30, 31, 111

[47] Mingrui Liu : The Cop Number of Graphs with Forbidden Induced Subgraphs. 2019. arXiv: 1908.11478. 67, 74

[48] Linyuan Lu et Xing Peng : On Meyniel's conjecture of the cop number. *Journal of Graph Theory*, 71(2):192–205, 2012. 32, 33, 39

[49] Masood Masjoody : *Cops and Robbers on Geometric Graphs and Graphs with a Set of Forbidden Subgraphs.* PhD Thesis, Simon Fraser University, 2019. 39

[50] Masood Masjoody et Ladislav Stacho : Cops and robbers on graphs with a set of forbidden induced subgraphs. *Theoretical Computer Science*, page 9, 2020. 66

[51] Brendan D. McKay et Adolfo Piperno : Practical graph isomorphism, II. *Journal of Symbolic Computation*, 60:94 – 112, 2014. 80

[52] Richard Nowakowski et Peter Winkler : Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2):235 – 239, 1983. 27, 28, 29, 38, 65, 67, 70, 71, 76, 79, 80, 81

[53] Photchchara Pisantechakool et Xuehou Tan : On the Conjecture of the Smallest 3-Cop-Win Planar Graph. *In* T.V. Gopal, Gerhard Jäger et Silvia Steila, éditeurs : *Theory and Applications of Models of Computation*, pages 499–514, Cham, 2017. Springer International Publishing. 67, 70, 98

[54] Paweł Prałat : When does a random graph have constant cop number? *The Australasian Journal of Combinatorics*, 46:285–296, 2010. 36, 61, 62

[55] Paweł Prałat et Nicholas Wormald : Meyniel's conjecture holds for random graphs. *Random Structures & Algorithms*, 48(2):396–421, mars 2016. 33

[56] Paweł Prałat et Nicholas Wormald : Meyniel's conjecture holds for random d-regular graphs. *Random Structures & Algorithms*, 55(3):719–741, octobre 2019. 33

[57] Alain Quilliot : *Problèmes de jeux, de point fixe, de connectivité et de représentation sur des graphes, des ensembles ordonnés et des hypergraphes.* PhD Thesis, Université de Paris VI, 1978. 27, 38, 65, 76

[58] Alain Quilliot : A short note about pursuit games played on a graph with a given genus. *Journal of Combinatorial Theory, Series B*, 38(1):89 – 92, 1985. 30, 39

[59] Joël Rickert : *Cops and Robbers, Bachelor Thesis.* ETH Zürich, 2017. 80, 117

[60] Neil Robertson : The smallest graph of girth 5 and valency 4. *Bulletin of the American Mathematical Society*, 70(6):824–825, 1964. 77

[61] Bernd S. W. Schröder : The Copnumber of a Graph is Bounded by [3/2 genus (G)] + 3. *In Categorical Perspectives*, pages 243–263. Birkhäuser Boston, Boston, MA, 2001. 30, 31, 39

[62] Alex Scott et Benny Sudakov : A Bound for the Cops and Robbers Problem. *SIAM Journal on Discrete Mathematics*, 25(3):1438–1442, janvier 2011. 32

[63] Vaidy Sivaraman : An application of the Gyárfás path argument. *Discrete Mathematics*, 342(8):2306 – 2307, 2019. 67

[64] Vaidy Sivaraman et Stephen Testa : Cop number of 2K2-free graphs. 2019. arXiv: 1903.11484. 67, 68, 74

[65] Jeremie Turcotte et Samuel Yvon : Code and data complement - 4-cop-win graphs have at least 19 vertices, 2020. *https://www.jeremieturcotte.com/research/min4cops/index.html*. 78

[66] Klaus Wagner : Über eine Eigenschaft der ebenen Komplexe. *Mathematische Annalen*, 114(1):570–590, décembre 1937. 66

[67] Zsolt Adam Wagner : Cops and Robbers on diameter two graphs. *Discrete Mathematics*, 338(3):107 – 109, 2015. 33, 34, 39, 54, 67, 70

[68] Wikipedia : *Moore graph — Wikipedia, The Free Encyclopedia*. 2020. *https://en.wikipedia.org/wiki/Moore_graph*. 31

# Annexe A

# Code utilisé

Dans cette annexe, nous présentons dans leur intégrité les scripts utilisés dans le troisième article. Ils sont aussi disponibles au https://github.com/tjeremie/Cops-and-robbers.

Notons que certains de ceux-ci incluent des exemples de code afin d'importer ou d'exporter les données et devraient être modifiés dépendamment de leur utilisation.

## A.1. Algorithme de calcul du *cop number*

L'algorithme de calcul du *cop number* (jusqu'à 3) implémenté ici est basé sur celui présenté par Bonato et Chiniforooshan dans [**16**], ainsi que sur [**26**] et [**59**].

```
#=
    Algorithm for testing cop numbers 1,2,3
    By Jérémie Turcotte

    Algorithm inspired by those suggested, for example, in https://math.ryerson.ca/~abonato/papers/
        distcops_bcp030109.pdf, https://www.sciencedirect.com/science/article/pii/S0012365X12000064
        and https://pub.tik.ee.ethz.ch/students/2016-HS/BA-2016-20.pdf.

    Also includes an example of how to scan various files of graphs, in the g6 format, for which to
        breakdown the number of graphs of each cop number.

    As codes for the various cop numbers are very similar, we only comment the code for cop number 2.
=#


using LightGraphs, GraphIO, Base

# Returns true if the line i in mat (of size n^2) is all true
function convertToBool1(mat,i)
```

```julia
    return !(false in mat[i,:])
end

# Returns true if the line ij in mat (of size n^3) is all true
function convertToBool2(mat,i,j)
    return !(false in mat[i,j,:])
end

# Returns true if the line ijl in mat (of size n^4) is all true
function convertToBool3(mat,i,j,l)
    return !(false in mat[i,j,l,:])
end

# Returns true if c(G)=1, false if c(G)>1
function oneCopWin(g)
    n=nv(g)
    configs=falses(n,n)
    queued=trues(n)

    for i in 1:n
        for k in 1:n
            templist=vcat(neighbors(g,k),k)
            if i in templist
                configs[i,k]=true
            end
        end
    end

    for i in 1:n
        if (convertToBool1(configs,i))
            return true
        end
    end

    bigchange=true

    while bigchange
        bigchange=false

        for i in 1:n
            if queued[i]
                queued[i]=false
                for ip in vcat(neighbors(g,i),i)

                changed=false

                for kp in 1:n
                    if !configs[ip,kp]
                        temp=true

                        for k in vcat(neighbors(g,kp),kp)
                            if !configs[i,k]
                                temp=false
                                break
                            end
                        end

                        if temp
                            configs[ip,kp]=true
                            changed=true

                            if (convertToBool1(configs,ip))
                                return true
                            end
                        end
                    end
                end

                if changed
                    queued[ip]=true
                    bigchange=true
                end
            end
        end
    end
end

    return false
end

# Returns true if c(G)<=2, false if c(G)>2
function twoCopWin(g)
    n=nv(g) # number of vertices in g

    configs=falses(n,n,n)
    #=
        At any given moment in the code, configs[i,j,k] will be true if we know that there is a strategy
            to win if there are cops on i and j and a robber on k (and it is cops' turn).
    Will be false if either we do not know yet, or there is no winning strategy in that position (
        which is why we start by initializing the array to all false ).
    The idea of the algorithm is to progressively fill up the matrix by finding new winning
        strategies , going backwards starting from the final turn.
=#

queued=trues(n,n) # Will represent the positions (i,j) that think we should verify soon. We start by
    deeming all positions interesting .

# If either i or j is in the neighbourhood of k, then with cops on i and j, a robber on k will
    immediately be caught.
for i in 1:n
    for j in 1:i
        for k in 1:n
            templist=vcat(neighbors(g,k),k)
            if i in templist || j in templist
                configs[i,j,k]=true
                configs[j,i,k]=true # To not repeat the same calculations , we only chose j between 1
                    and i : the order of (i,j) does not matter.
            end
        end
    end
end

# If there exists a choice of starting positions (i,j) for the cops such that for any choice of
    robber position k there is a winning strategy , then 2 cops can win.
# Checking this now amounts to verifying if there is a dominating set of size 2 in the graph.
for i in 1:n
    for j in 1:i
        if (convertToBool2(configs,i,j))
            return true
        end
    end
end

bigchange=true

while bigchange # If the matrix did not see any change in the last iteration , then there will not be
    any further change at any time: 2 cops cannot win.
    bigchange=false

    for i in 1:n
        for j in 1:i
```

```
            if queued[i,j] # Suppose we consider (i,j) to be an interesting position: either because
                we are in the start of the algorithm or because for some k mat[i,j,k] changed value
                recently, and we want to see if this impacts neighbouring positions.

                queued[i,j]=false

                for ip in vcat(neighbors(g,i),i), jp in vcat(neighbors(g,j),j) # We choose (ip,jp)
                    such that cops can move to (i,j) in 1 turn.

                    changed=false

                    for kp in 1:n # We consider every possible position for the robber

                        if !configs[ip,jp,kp] # We only need to consider those kp such that we don't
                            already know gives a winning position
                            temp=true

                            for k in vcat(neighbors(g,kp),kp) # We consider the vertices k that the
                                robber can move to from kp
                                if !configs[i,j,k] # If there is a vertex k where the robber can go
                                    such that the move (ip,jp)->(i,j) does not yield a winning
                                    position, we cannot say anything new.
                                    temp=false
                                    break
                                end
                            end

                            if temp # Otherwise, if the move (ip,jp)->(i,j) gives a winning position
                                whatever the robber does (for every choice of k), we know that (ip,
                                jp,kp) is a winning position.
                                configs[ip,jp,kp]=true
                                configs[jp,ip,kp]=true

                                if (convertToBool2(configs,ip,jp)) # We verify if this gives us a
                                    winning starting position (ip,jp), if so we are done.
                                    return true
                                end

                                changed=true # Indicates that at least one triple (ip,jp,kp) has
                                    changed value.
                            end
                        end
                    end
```

```
                        if changed # If at least one triple (ip,jp,kp) changed value, then we deem that
                            (ip,jp) might be interesting : this means we may "be ready" to find winning
                            strategies for neighbour positions .
                            queued[ip, jp]=true
                            queued[jp, ip]=true
                            bigchange=true
                        end
                    end
                end
            end
        end
    end

    return false
end

# Returns true if c(G)<=3, false if c(G)>3
function threeCopWin(g)
    n=nv(g)
    configs=falses(n,n,n,n)
    queued=trues(n,n,n)

    for i in 1:n
        for j in 1:i
            for l in 1 : j
                for k in 1:n
                    templist=vcat(neighbors(g,k),k)
                    if i in templist || j in templist || l in templist
                        configs[i,j,l,k]=true
                        configs[j,i,l,k]=true
                        configs[i,l,j,k]=true
                        configs[j,l,i,k]=true
                        configs[l,i,j,k]=true
                        configs[l,j,i,k]=true
                    end
                end
            end
        end
    end

    for i in 1:n
        for j in 1:i
            for k in 1:j
                if (convertToBool3(configs,i,j,k))
                    return true
                end
            end
        end
    end

    bigchange=true

    while bigchange
        bigchange=false

        for i in 1:n
            for j in 1:i
                for l in 1:j
                    if queued[i,j,l]
                        queued[i,j,l]=false
                        for ip in vcat(neighbors(g,i),i), jp in vcat(neighbors(g,j),j), lp in vcat(
                            neighbors(g,l),l)

                            changed=false

                            for kp in 1:n
                                if !configs[ip,jp,lp,kp]
                                    temp=true

                                    for k in vcat(neighbors(g,kp),kp)
                                        if !configs[i,j,l,k]
                                            temp=false
                                            break
                                        end
                                    end

                                    if temp
                                        configs[ip,jp,lp,kp]=true
                                        configs[jp,ip,lp,kp]=true
                                        configs[ip,lp,jp,kp]=true
                                        configs[jp,lp,ip,kp]=true
                                        configs[lp,ip,jp,kp]=true
                                        configs[lp,jp,ip,kp]=true

                                        changed=true

                                        if (convertToBool3(configs,ip,jp,lp))
                                            return true
                                        end
                                    end
                                end
                            end

                            if changed
                                queued[ip, jp, lp]=true
                                queued[jp, ip, lp]=true
                                queued[ip, lp, jp]=true
                                queued[jp, lp, ip]=true
                                queued[lp, ip, jp]=true
                                queued[lp, jp, ip]=true
                                bigchange=true
                            end
                        end
                    end
                end
            end
        end
    end

    return false
end

# Example of function to load a file of graphs in the graph6 format
function loadList(i)
    return collect(values(loadgraphs(string("/n20d2D3/graphs_20_2_3_",i,"_20000.g6"), GraphIO.Graph6
        .Graph6Format())));
end

#=
    Example of function to manage reading each file and breaking down the cop numbers

    Will print the breakdown of the cop numbers of the graphs and will save the graphs needing 3 and 4
        cops in files

    This example supports multithreading, see https://julialang.org/blog/2019/07/multithreading/ to
        change the number of threads.
    Experimentally, speedup of almost factor 2 between 1 and 2 threads but does not improve much after 4
         threads.
=#

function fctThreaded(part)
    totalonecopcount=0
```

```
totaltwocopcount=0
totalthreecopcount=0
totalfourcopcount=0

d3=Dict{AbstractString,AbstractGraph}()
d4=Dict{AbstractString,AbstractGraph}()

start=(part-1)*400
stop=part*400-1

for i in start:stop

    localonecopcount=Threads.Atomic{Int}(0)
    localtwocopcount=Threads.Atomic{Int}(0)
    localthreecopcount=Threads.Atomic{Int}(0)
    localfourcopcount=Threads.Atomic{Int}(0)

    liste=loadList(i)

    Threads.@threads for g in liste
        if oneCopWin(g)
            Threads.atomic_add!(localonecopcount,1)
        elseif twoCopWin(g)
            Threads.atomic_add!(localtwocopcount,1)
        elseif threeCopWin(g)
            Threads.atomic_add!(localthreecopcount,1)
            d3[string(i,"--",part,"--",Threads.threadid(),"--",localthreecopcount[])]=g
        else
            Threads.atomic_add!(localfourcopcount,1)
            d4[string(i,"--",part,"--",Threads.threadid(),"--",localfourcopcount[])]=g
        end
    end

    totalonecopcount=totalonecopcount+localonecopcount[]
    totaltwocopcount=totaltwocopcount+localtwocopcount[]
    totalthreecopcount=totalthreecopcount+localthreecopcount[]
    totalfourcopcount=totalfourcopcount+localfourcopcount[]


    println(string(i," ",localonecopcount[]," ",localtwocopcount[]," ",localthreecopcount[]," ",
        localfourcopcount[]," "))
    flush(stdout)
end

```

```
    savegraph( string ("./n14d1D4_3cops_part",part,".g6"), d3, GraphIO.Graph6.Graph6Format())
    savegraph( string ("./n14d1D4_4cops_part",part,".g6"), d4, GraphIO.Graph6.Graph6Format())

     println ( string ("Total : ", totalonecopcount," ",totaltwocopcount," ",totalthreecopcount ," ",
         totalfourcopcount ," "))
end

# To start the program from the command line
@time fctThreaded(parse(Int64, ARGS[1]))
```

## A.2. Algorithme de vérification de graphes démantelables

Nous présentons ici l'algorithme qui vérifie si un graphe est policier-gagnant en se basant sur l'équivalence entre les graphes policier-gagnants et démantelables, voir Théorème 1.3.6.

```
(* ::Package:: *)

(*
Algorithm for determining cop−win graphs
By J\[EAcute]r\[EAcute]mie Turcotte

Uses the equivalence between cop−win and dismantlable graphs, see https://www.sciencedirect.com/science
    /article/pii/0012365X83901607.
*)


(* Returns the closed neighbourhood of v in g *)
closedNeighbourhood[g_,v_]:=Append[AdjacencyList[g,v],v]


(* Attemps to find a corner in the graph g *)
corner[g_]:=Module[{i,j},
  Catch[
    Do[
      If [i!=j,
        If [SubsetQ[closedNeighbourhood[g,i], closedNeighbourhood[g,j ]],
          Throw[{True,j}] (* if the neighbourhood of j is a subset of the neighbourhood of i, then j is
              a corner/ irreducible vertex *)
        ]
      ];
      ,{ i , VertexList [g]},{ j , VertexList [g ]}]; (* we test for all pairs of vertices (i,j), the if above
          makes sure that these are separate vertices *)
      Throw[{False,0}] (* if no corner is found *)
  ]
]


(* Connected graphs up to order 3 are cop−win *)
copWin[g_]:=True/;VertexCount[g]<=3


(* Tests if the graph g (which we suppose is connected) is cop−win *)
copWin[g_]:=Module[{val=corner[g],containCorner,corner},

  containCorner=val [[1]]; corner=val [[2]];

  If [containCorner,
    copWin[Subgraph[g,DeleteCases[VertexList[g],corner ]]], (* if g contains a corner u, we remove it and
          verify if g−u is cop−win (note that if g is connected, so is g−u) *)
    False
  ]
]


(* Example of how to import graph files *)
importData[i_]:=Flatten[{Import["/n10/graphs_10_1_10_"<>ToString[i]<>"_1000.g6","graph6"]}] (* the
      path is absolute *)


(* Example of code to manage reading each file and counting down the number of cop−win graphs *)
counter=0;

Do[
  tempList=importData[i];
  tempNumber=Count[Map[copWin,tempList],True];
  Print[ToString[i]<>" "<>ToString[tempNumber]];
  counter=counter+tempNumber;
  ,{ i ,0,999}
]

Print[counter ];
```

## A.3. Implémentation du *Merging Algorithm*

Nous présentons ici le code implémentant le *Merging Algorithm* décrit dans la Section 4.6. Celui-ci devrait être vu comme partie intégrante du troisième article.

### A.3.1. Phase 1

```
(* ::Package:: *)

(* ::Input:: *)
(**)


(* ::Subtitle:: *)
(*Generating small 4-cop-win candidate graphs *)
(*Part 1/2 - Generating the base graphs*)
(*For Finding the minimum order of 4-cop-win graphs*)
(*By J\[EAcute]r\[EAcute]mie Turcotte and Samuel Yvon*)


(* ::Subtitle:: *)
(*Usage*)
(* Option 1 : Run in Mathematica by going to end of file and choosing which computation to run.*)
(* Option 2 : Run in a shell with wolframscript : "wolframscript -script 4copcandidates-part1.wl x x x
   x x x", where x are the desired parameters of createGraphs.*)


(* Specify here the path to get the lists of 3-cop-win graphs, by default fetches the results online *)
importPath="https://www.jeremieturcotte.com/research/min4cops/data/smallgraphs/results/3copwingraphs/";


(* ::Subtitle:: *)
(*Code:*)


(* Debugging function to print a graph with labels *)

printLabel[g_]:=Graph[g,VertexLabels->"Name"]


(* Some functions on neighbourhoods *)

(* The neighbourhood of v in g. *)
```

```
openNeighbourhood[g_,v_]:=AdjacencyList[g,v]
(* The closed neighbourhood (includes v) of v in g. *)
closedNeighbourhood[g_,v_]:=Prepend[AdjacencyList[g,v],v]
(* The set of vertices of g−N[v]. *)
noNeighbourhood[g_,v_]:=Complement[VertexList[g],closedNeighbourhood[g,v]]


(* Functions to create formal edges *)

(* Returns a list of edges between v and the vertices of list . *)
convertToEdge[v_,list_]:=UndirectedEdge[v,#]&/@list
(* Returns a list of edges between the vertices in list and v1 and with v2. *)
doubleConvertToEdge[v1_,v2_,list_]:=Join[convertToEdge[v1, list ], convertToEdge[v2, list ]]


(* Functions to reduce vertices to consider by automorphism *)


(* Given a list of automorphisms (encoded as associations) of some graph, returns all
   vertices equivalent to v through one of these automorphisms. *)
automorphismsImage[automorphismList_,v_]:=Union[Map[#[v]&,automorphismList]]

(* Given a graph g and a vertex v, returns all vertices equivalent to v through some automorphism of g.
    *)
automorphicEquivalentVertices [g_,v_]:=automorphismsImage[FindGraphIsomorphism[g,g,All],v]

(* Given a graph g and a list of vertices of g, removes from this list vertices which are equivalent
    through some automorphism. *)
reduceByAutomorphism[g_,list_]:=DeleteDuplicates[ list ,MemberQ[automorphicEquivalentVertices[g
    ,#1],#2]&]

(* Given a graph g, returns a list of vertices of chosen degree, all of which are not equivalent by
    automorphism.
   This function is currently with memory to save computation time. *)
reducedVertexChoices[g_,degree_]:=(reducedVertexChoices[g,degree]=reduceByAutomorphism[g,
    selectDegreeVertices[g,degree ]])


(* Functions relating to the maximum authorized possible degree of vertices *)

(* Given maxDeg, which is the maximum authorized degree we consider, and a list of vertices whose degree
     must be strictly smaller than maxDeg,
   returns the maximum possible degree of v ( either maxDeg or maxDeg−1). *)
```

```
realDegreeBound[v_,lowerDegreeVerticesList_, maxDeg_]:=maxDeg−Boole[MemberQ[
    lowerDegreeVerticesList,v]]

(* Verifies if the maximum degree of g is at most maxDeg and all vertices in lowerDegreeVerticesList
    have strictly smaller degree.
  If true, returns g, otherwise returns Nothing (an element which vanishes in any list). *)
 degreesFilter [g_, lowerDegreeVerticesList_ , maxDeg_]:=If[AllTrue[VertexList [g], VertexDegree[g,#]<=
    realDegreeBound[#,lowerDegreeVerticesList,maxDeg] &],g,Nothing]


(* Functions to reduce graphs to consider by isomorphism *)


(* Given an isomorphism iso (encoded as an association) between two graphs on the same set vertices and
    a list of sets of vertices,
  returns true if the image (through iso) of every such set of vertices is itself. *)
fixedPointsIsomorphism [iso_, list_]:=AllTrue[ list ,Sort[#/.iso]==Sort[#]&]

(* Given two graphs and a list of sets of vertices, returns true if there exists an isomorphism between
    g1 and g2 such that the image
  of every set of vertices is itself (which is what we call a "strong isomorphism"). *)
strongIsomGraphs[g1_,g2_,list_]:=AnyTrue[FindGraphIsomorphism[g1,g2,All],fixedPointsIsomorphism[#, list
    ]&]


(* Functions used in the labelling of new vertices *)


(* The list of vertices that will need to be added as common neighbors of v1 and v2 when merging the
    graphs. *)
verticesToAdd[g2_,v2_,maxDeg_,nbInteriorVertices_]:= nbInteriorVertices +Range[maxDeg−VertexDegree[g2,
    v2]]

(* The list of labels we will give to the current neighbours of v2 when merging the graphs. *)
alreadyNeighbours[g2_,v2_,maxDeg_,nbTotalVertices_]:=nbTotalVertices−Range[VertexDegree[g2,v2]]

(* The six classes of vertices in the merged graph g, as described in the proof. *)
graphSections[g1_,v1_,g2_,v2_,maxDeg_,nbTotalVertices_]:={{v1},openNeighbourhood[g1,v1],
    noNeighbourhood[g1,v1],verticesToAdd[g2,v2,maxDeg,VertexCount[g1]],alreadyNeighbours[g2,v2,maxDeg,
    nbTotalVertices],{nbTotalVertices}}


(* Functions to merge graphs g1 and g2 relative to a choice of v1 and v2 *)

(* Deletes in list the graphs not respecting the authorized degrees and removes graphs which are
    strongly isomorphic.
   It is important to note that it would technically also be necessary to verify that this isomorphism
      is compatible with
   the list of degrees which must have degree strictly smaller than maxDeg. But as we use our strong
      isomorphisms to,
   in particular, fix g1, the restriction of the isomorphism will also be an automorphism of g1,
   and as equivalent vertices have the same degree bounds, this is valid.
 *)
removeIsoAndClean[list_,g1_,v1_,g2_,v2_,lowerDegreeVerticesList_,maxDeg_,nbTotalVertices_]:=
    DeleteDuplicates[degreesFilter[#, lowerDegreeVerticesList ,maxDeg]&/@list,strongIsomGraphs[#1,#2,
    graphSections[g1,v1,g2,v2,maxDeg,nbTotalVertices]]&]

(* Lists all isomorphisms (encoded as associations) from g2−N[v2] to g1−N[v1]. *)
subgraphIsomorphisms[g1_,v1_,g2_,v2_]:=FindGraphIsomorphism[Subgraph[g2,noNeighbourhood[g2,v2]],
    Subgraph[g1,noNeighbourhood[g1,v1]],All]

(* Merge graphs g1 and g2 with the following rules. Keep the numbering of g1, relabel v2 and it's
    neighbourhood
   respectively with labels nbVertices and nbVertices−1 to nbVertices−d_g2(v2). Relabel g2−N[v2] using
      the isomorphism iso
   (which is an isomorphism between g2−N[v2] and g1−N[v1]) to be able to merge with g1. If d_g2(v2)<
      maxDeg, add common neighbors
   to v1 and v2 to bring v2 to maximum degree.
 *)
mergeGraphsWithSpecificIso[g1_,v1_,g2_Graph,v2_,iso_,maxDeg_,nbTotalVertices_]:=EdgeAdd[GraphUnion[
    g1,VertexReplace[g2,Join[Normal[iso],
             Table[closedNeighbourhood[g2,v2][[ i]]−>nbTotalVertices−i+1,{i,VertexDegree[g2,v2 ]+1}]]]],
                  doubleConvertToEdge[v1,nbTotalVertices,verticesToAdd[g2,v2,maxDeg,VertexCount[g1]]]]

(* Merge graphs g1 and g2 as above, by considering every possible way of merging g1−N[v1] and g2−N[v2
    ].
   Also adds to every graph relevant information for the next part of the algorithm (which is adding
      possible missing edges).
 *)
mergeGraphs[g1_,v1_,g2_,v2_,lowerDegreeVerticesList_,maxDeg_,nbTotalVertices_]:={#,graphSections[g1,v1
    ,g2,v2,maxDeg,nbTotalVertices],
          lowerDegreeVerticesList }&/@removeIsoAndClean[mergeGraphsWithSpecificIso[g1,v1,g2,v2,#,
              maxDeg,nbTotalVertices]&/@subgraphIsomorphisms[g1,v1,g2,v2],g1,v1,g2,v2,
              lowerDegreeVerticesList,maxDeg,nbTotalVertices]


(* Functions to choose vertices depending on degree *)

(* Returns all vertices of degree deg in g. *)
selectDegreeVertices [g_,deg_]:=Select[VertexList[g], VertexDegree[g,#]==deg&]

(* Returns all vertices of degree greater than deg in g.*)
selectUpperDegreeVertices [g_,deg_]:=Select[VertexList[g], VertexDegree[g,#]>deg&]


(* Function to clean and load the list of graphs we are going to merge *)

(* Returns a cleaned version of list : sorts the graphs by decreasing maximum degree and all graphs in
      canonical form. *)
cleanGraphList [ list_ ]:=Sort[CanonicalGraph/@list, Max[VertexDegree[#1]]>=Max[VertexDegree[#2]]&]

(* Loads the list of 3-cop-win connected graphs on nbVertices vertices such that c(G)=3 with Delta\[
      LessEqual]maxDeg. *)
loadList [nbVertices_, maxDeg_]:=cleanGraphList[Import[importPath<>"n"<>ToString[nbVertices]<>"d1D"
      <>ToString[maxDeg]<>"_3cops.g6","graph6"]]


(* :: Text:: *)
(*Main function*)
(*  Parameters*)
(*    nbTotalVertices: The total number of vertices in the graphs we will create.*)
(*    v1degree: The degree of the vertices v1 we will choose in g1.*)
(*    g1MaximumDegree: The maximum degree of the graphs g1 we will choose.*)
(*    maxDeg: The maximum degree of the graphs we will create.*)
(*    testAll : If True, graphs will be created for each possible choice of v1 in g1, otherwise will be
      done for one choice of v1.*)
(*    v2DegreeGreater: The degree of v2 will be set to v1Degree+v2DegreeGreater. If v2DegreeGreater>0,
      we suppose that g-N[v1] has maximum degree 4. Only works if nbTotalVertices=18 vertices and maxDeg
      =5.*)
(*  Optional parameters (otherwise res=mod=1)*)
(*    res: The part of the computation to do, between 1 and mod.*)
(*    mod: The number of parts to split the computation in.*)
(**)
(*  Output*)
(*    Exports to file a list where each item is itself a list of length 3:*)
(*      The first item is the created base graph.*)
(*      The second item is the breakdown into the 6 types of vertices of the graph.*)
(*      The third item is the list of vertices which must have maximum degree strictly smaller than
      maxDeg, this is useful to reduce the number of cases in part 2 of the algorithm.*)
(*    Also creates a file which summarizes the computation. On each line there is a list of length 3:
      the index of g1 in the (cleaned-up and reordered) list, the number of graph created from this choice
       of g1 and the time required for this computation.*)
```

```


createGraphs[nbTotalVertices_,v1degree_,g1MaximumDegree_,maxDeg_,testAll_,v2DegreeGreater_,res_,
    mod_]:=
Block[{graphList1,graphList2, start ,end,g1,v1, lowerDegreeVerticesList ,reducedVerticesToConsider, results ,
    v2degree,totalTime, outputFile ,temp},
  (* File that will contain an outline of the results of the computation.*)
  outputFile="basegraphs_"<>ToString[nbTotalVertices]<>"_"<>ToString[v1degree]<>"_"<>ToString[
      g1MaximumDegree]<>"_"<>ToString[maxDeg]<>"_"<>ToString[testAll]<>"_"<>ToString[
      v2DegreeGreater]<>"_results"<>If[mod>1,"_part"<>ToString[res],""]<>".txt";

  totalTime=AbsoluteTiming[
    (* We load the list of graphs in which we pick g1. *)
    graphList1=loadList[ nbTotalVertices-maxDeg-1,maxDeg];

    (* This will be the degree of v2 in g2. *)
    v2degree=v1degree+v2DegreeGreater;

    (* We load the list of graphs in which we pick g2. *)
    graphList2=If[v2DegreeGreater>0,loadList[12+v2DegreeGreater,4],graphList1 ];

    (* We select the start and the end indices of all graphs with maximum degree exactly
        g1MaximumDegree in graphList. *)
    start = FirstPosition [graphList1 ,_?(Max[VertexDegree[#]]==g1MaximumDegree&)][[1]];
    end=Length[graphList1]+1-FirstPosition[Reverse[graphList1],_?(Max[VertexDegree[#]]==
        g1MaximumDegree&)][[1]];

     results =Flatten[
       Table[
         If [Mod[i-res,mod]==0,
           (* For each possible graph g1 with maximum degree exactly g1MaximumDegree, we will also
               reduce by automorphism the possible choices of v1. *)
           g1=graphList1[[i ]];
           reducedVerticesToConsider=reducedVertexChoices[g1,v1degree];

           temp=AbsoluteTiming[Flatten[Table[
               (* We choose a vertex v1. *)
               v1=reducedVerticesToConsider[[j ]];

               (* All vertices which either come before v1 in reducedVerticesToConsider or which have
                   higher degree than v1 are considered to already having been verified , so have degree
                   strictly smaller than maxDeg. *)
```

```
                    lowerDegreeVerticesList =If[v2DegreeGreater>0,Range[12],Union[Flatten[Table[
                        automorphicEquivalentVertices[g1,reducedVerticesToConsider[[k ]]],{ k ,1, j −1}]],
                        selectUpperDegreeVertices [g1,v1degree ]]];

                    (* For some choice of g2,v2, we compute the merged list. *)
                    mergeGraphs[g1,v1,g2,v2, lowerDegreeVerticesList ,maxDeg,nbTotalVertices]

                  (* In the case where v1 and v2 have the same degree, w only need to consider the graphs g2
                      which come after g1 in the list . We choose v2 up to automorphism. *)
                 ,{ j ,1, If [ testAll ,Length[reducedVerticesToConsider],Boole[Length[reducedVerticesToConsider
                      ]>0]]},{g2,graphList2},{v2,reducedVertexChoices[g2,v2degree]}
               ],3]];

             PutAppend[{i,Length[temp[[2]]],temp[[1]]}, outputFile ];

             temp [[2]]

             , Nothing
           ]

           ,{ i , start ,end} (* The index of g1 in the list . *)
         ]
        ,1 (* We flatten 4 layers as there are 4 levels in our table. *)
      ];

      Export["basegraphs_"<>ToString[nbTotalVertices]<>"_"<>ToString[v1degree]<>"_"<>ToString[
          g1MaximumDegree]<>"_"<>ToString[maxDeg]<>"_"<>ToString[testAll]<>"_"<>ToString[
          v2DegreeGreater]<>If[mod>1,"_part"<>ToString[res],""]<>".mx",results]
    ][[1]];

  PutAppend[{Total, Length[results],totalTime},outputFile];
]
createGraphs[nbTotalVertices_,v1degree_,g1MaximumDegree_,maxDeg_,testAll_,v2DegreeGreater_]:=
      createGraphs[nbTotalVertices,v1degree,g1MaximumDegree,maxDeg,testAll,v2DegreeGreater,1,1] (* Version
       of the function with only one part. *)


createGraphs@@(ToExpression/@$ScriptCommandLine[[2;;]]) (* For calls from a shell . Otherwise, call
      createGraphs with the desired parameters. *)
```

### A.3.2. Phase 2

```
(* ::Package:: *)
(* :: Title :: *)
(*Generating small 4−cop−win candidate graphs*)
(*Part 2/2 − Filling in the graphs with possible edges*)
(*For Finding the minimum order of 4−cop−win graphs*)
(*By J\[EAcute]r\[EAcute]mie Turcotte and Samuel Yvon*)


(* :: Text:: *)
(*Usage*)
(* Option 1 : Run in Mathematica by going to end of file and choosing which computation to run.*)
(* Option 2 : Run in a shell with wolframscript : "wolframscript −script 4copcandidates−part1.wl x x
    x x x x x x", where x are the desired parameters of fillGraphs .*)


(* Specify here the path to get the required files for the computation, by default fetches the results
    online *)

(* The path to the 3−cop−win graphs on fewer vertices, same as in the part 1 of the algorithm. . *)
importPathSmallGraphs="https://www.jeremieturcotte.com/research/min4cops/data/smallgraphs/results/3
    copwingraphs/";

(* The path to the results of part 1 of the algorithm. *)
importPathPart1Results="https://www.jeremieturcotte.com/research/min4cops/data/remainingcases/
    part1results/graphs/";


(* :: Subtitle :: *)
(*Code:*)


(* Debugging function to print a graph with labels *)

 printLabel [g_]:=Graph[g,VertexLabels−>"Name"]


(* Some functions on neighbourhoods *)

(* The neighbourhood of v in g. *)
openNeighbourhood[g_,v_]:=AdjacencyList[g,v]

(* The closed neighbourhood (includes v) of v in g. *)
closedNeighbourhood[g_,v_]:=Prepend[AdjacencyList[g,v],v]

```

```
(* The set of vertices of g−N[v]. *)
noNeighbourhood[g_,v_]:=Complement[VertexList[g],closedNeighbourhood[g,v]]


(* Functions to create and add edges *)

(* Returns a list of edges between v and the vertices of list . *)
convertToEdge[v_,list_]:=UndirectedEdge[v,#]&/@list

(* Returns the graph g with the added edges of list . This is a substitute for EdgeAdd,
   as EdgeAdd seems to have some memory leak (as of version 12.1.0.0) . *)
edgeadd[g_,list_]:=Graph[Join[EdgeList[g], list ]]


(* Functions on the maximum authorized degrees for vertices *)

(* Given maxDeg, which is the maximum authorized degree we consider, and a list of vertices whose degree
     must be strictly smaller than maxDeg,
   returns the maximum possible degree of v ( either maxDeg or maxDeg−1). *)
realDegreeBound[v_,lowerDegreeVerticesList_,maxDeg_]:=maxDeg−Boole[MemberQ[
    lowerDegreeVerticesList,v]]

(* Given a graph g and the maximum authorized degree information, returns true if v still has capacity
    for new neighbour(s).*)
 viableVertices [g_,v_,lowerDegreeVerticesList_,maxDeg_]:=VertexDegree[g,v]<realDegreeBound[v,
    lowerDegreeVerticesList,maxDeg]


(* Functions to prune out graphs for which the vertices of degree maxDeg do not form a clique (only
    applied for graphs with v2DegreeGreater>0 *)

(* Selects the valid graphs in list . *)
specialCleanup [ list_ ,v2DegreeGreater_]:=If[v2DegreeGreater>0,Select[list ,graphHubIsClique], list ]

(* Returns True if the vertices of maximum degree of g forms a clique. *)
graphHubIsClique[g_]:=CompleteGraphQ[Subgraph[g,GraphHub[g]]]


(* Functions to add possible missing edges *)

(* Given a graph g and a start vertex v, returns the list of possible sets of edges between v and the
   vertices of possibleEndVertices that can be added while respecting the degree conditions . *)
```

```
newEdgePossibilities [g_,v_,possibleEndVertices_, lowerDegreeVerticesList_ , maxDeg_]:=Subsets[Select[
    possibleEndVertices, viableVertices [g,#, lowerDegreeVerticesList , maxDeg]&],realDegreeBound[v,
    lowerDegreeVerticesList , maxDeg]−VertexDegree[g,v]]

(* Given a graph g and a start vertex v, generates the graphs for each possible sets of edges to add. *)
newGraphPossibilities [g_Graph,v_,possibleEndVertices_, lowerDegreeVerticesList_ , maxDeg_,
    v2DegreeGreater_]:=specialCleanup[edgeadd[g,convertToEdge[v,#]]&/@newEdgePossibilities[g,v,
    possibleEndVertices, lowerDegreeVerticesList , maxDeg],v2DegreeGreater]

(* Applies the previous function to each graph in list , and brings the resulting list down to one level .
     *)
newGraphPossibilities [ list_List , v_, possibleEndVertices_, lowerDegreeVerticesList_ , maxDeg_,
    v2DegreeGreater_]:=Flatten[newGraphPossibilities[#,v,possibleEndVertices, lowerDegreeVerticesList ,
    maxDeg,v2DegreeGreater]&/@list]

(* Applies the j−th iteration of the edge−adding procedure : adds edges incident to the j−th neighbour
    of v2.
   The possible neighbours change depending on if it is a common neighbour of v1 and v2 or not. *)
iteration [j_, tempList_, lowerDegreeVerticesList_ , partition_ , maxDeg_,v2DegreeGreater_]:=
  If [ j<=Length[partition [[4]]],
     Flatten[ newGraphPossibilities [#, partition [[4, j ]], Join[ partition [[2]], partition [[3]], partition [[4, j
          +1;;]], partition [[5]]], lowerDegreeVerticesList , maxDeg,v2DegreeGreater]&/@tempList],
     Flatten[ newGraphPossibilities [#, partition [[5, j−Length[partition [[4]]]]], partition [[2]],
          lowerDegreeVerticesList , maxDeg,v2DegreeGreater]&/@tempList]
  ];


(* Functions to select graphs which have the proper structure *)

(* Returns True if g−N[v] is isomorphic to a graph in list . We consider all graphs in list are already
    in canonical form. *)
validSubgraph[g_,v_, list_]:=MemberQ[list,CanonicalGraph[Subgraph[g,noNeighbourhood[g,v]]]]

(* Returns true if g has the proper form. Does the previous test for every vertex of maximum degree,
    except for the vertices in the list ignoreVertices ,
   for which we assume this is true (in order not to test what we already know is true). *)
validGraph [g_, list_ , ignoreVertices_ ]:=AllTrue [Complement[GraphHub[g],ignoreVertices],validSubgraph[g,#,
    list]&]


(* Functions to reduce graphs to consider by isomorphism *)

(* Given an isomorphism iso (encoded as an association ) between two graphs on the same set vertices and
    a list of sets of vertices ,
    returns true if the image (through iso) of every such set of vertices is itself . *)
fixedPointsIsomorphism[iso_Association, list_List]:=AllTrue[ list ,Sort[#/.iso]==Sort[#]&]

(* Given two graphs and a list of sets of vertices , returns true if there exists an isomorphism between
    g1 and g2 such that the
   image of every set of vertices is itself (which is what we call a "strong isomorphism"). *)
strongIsomGraphs[g1_Graph,g2_Graph,list_List]:=AnyTrue[FindGraphIsomorphism[g1,g2,All],
    fixedPointsIsomorphism[#, list]&]


(* Function to clean and load the list of graphs we are going to merge *)

(* Returns a cleaned version of list : sorts the graphs by decreasing maximum degree and all graphs in
    canonical form. *)
cleanGraphList[list_]:=Sort[CanonicalGraph/@list,Max[VertexDegree[#1]]>=Max[VertexDegree[#2]]&]

(* Loads the list of 3-cop-win connected graphs on nbVertices vertices such that c(G)=3 with Delta\[
    LessEqual]maxDeg. *)
loadList[nbVertices_,maxDeg_]:=cleanGraphList[Import[importPathSmallGraphs<>"n"<>ToString[
    nbVertices]<>"d1D"<>ToString[maxDeg]<>"_3cops.g6","graph6"]]


(* ::Text:: *)
(*Main function*)
(*  Parameters*)
(*     First 6 parameters : The same as in the first 6 parameters of part 1 of the algorithm, will be
    used to load the appropriate list .*)
(*  Optional parameters (otherwise res,mod=1) *)
(*     res: The residue class to compute, between 1 and mod.*)
(*     mod: The number of classes in which we split the computation.*)
(*     *)
(*  Output*)
(*     Exports to file a list of candidate 4-cop-win graphs. One file will be generated for each graph
    produced in part 1 of the algorithm, the graphs are in g6 format.*)
(*     Also generates a file which summarizes the computation. On each line there is a list with a
    variable number of elements: the first element is the index of the base graph, followed by the
    number of graphs in each part of the algorithm, the last element is the time    required for this
    computation.*)


fillGraphs[nbTotalVertices_,v1degree_,g1MaximumDegree_,maxDeg_,testAll_,v2DegreeGreater_,res_,mod_
    ]:=
Block[{graphList,baseGraphs,g, partition , lowerDegreeVerticesList ,tempList,tempList2,tempList3,
    outputGraphs,iterationCount ,graphCounts,timing, outputFile , counterList ,totalTime,
    totalGraphsGenerated,parameterToFileName},
  (* To convert parameters to string format. *)
  parameterToFileName=ToString[nbTotalVertices]<>"_"<>ToString[v1degree]<>"_"<>ToString[
      g1MaximumDegree]<>"_"<>ToString[maxDeg]<>"_"<>ToString[testAll]<>"_"<>ToString[
      v2DegreeGreater];

  (* If it does not already exist , we create a directory in which we create the results files . *)
  Quiet[CreateDirectory["finalgraphs_"<>parameterToFileName], CreateDirectory::filex];

  (* We load the 3-cop-win graphs, same as in part 1 of the algorithm.*)
  graphList=loadList[ nbTotalVertices -maxDeg-1,maxDeg];

  (* We start by loading the results of the first part of the algorithm. *)
  baseGraphs=Import[importPathPart1Results<>"basegraphs_"<>parameterToFileName<>".mx"];

  totalGraphsGenerated=0; (* Will contain the total number of graphs which we have outputed to file . *)

  outputFile="./"<>"finalgraphs_"<>parameterToFileName<>"/finalgraphs_"<>parameterToFileName<>
      If[mod>1,"_part"<>ToString[res],""]<>"_results.txt";
  totalTime=AbsoluteTiming[
  Do[
    timing=AbsoluteTiming[
      (* We load a specific base graph. *)
      {g, partition , lowerDegreeVerticesList }=baseGraphs[[i ]];

      (* Will collect the number of graphs after each step in the algorithm, for debugging and
          verification purposes. *)
      graphCounts={i};

      (* Will contain the graphs after each iteration . *)
      tempList={g};

      (* Will count the number of iterations of the edge adding procedure. *)
      iterationCount=0;

      Do[
        tempList=iteration [ j ,tempList, lowerDegreeVerticesList , partition ,maxDeg,v2DegreeGreater]; (*
            Look at possible edges to add to the j-th neighbour of v2. *)

        AppendTo[graphCounts,Length[tempList]];

        (*
        We remove graphs which are strongly isomorphic, in the sense that they will give the same
            graphs later in the algorithm. This can save a significant amount of time and memory.
        As this procedure is itself very lengthy when tempList is large, we only apply it for the 2
            first iterations of the edge adding procedure.

        We want to see if there exists an isomorphism such that the " classes " of vertices are
            unchanged.
        At this point in the algorithm, the types are the same as when generating the base graphs,
            except that we must remember which vertices have already been considered.

        As this procedure is itself very lengthy, we only apply it if there are fewer than 40000
            graphs in the list . We estimate that for anything more than this it is not worth it .
      *)
      If [Length[tempList]<40000,
        tempList=If[j<=Length[partition [[4]]],
          DeleteDuplicates [tempList,strongIsomGraphs[#1,#2,{partition [[1]], partition [[2]], partition
              [[3]], partition [[4,1;; j ]], partition [[4, j +1;;]], partition [[5]], partition [[6]]}]&]
          ,
          DeleteDuplicates [tempList,strongIsomGraphs[#1,#2,{partition [[1]], partition [[2]], partition
              [[3]], partition [[4]], partition [[5,1;; j−Length[partition [[4]]]]], partition [[5, j−Length[
            partition [[4]]]+1;;]], partition [[6]]}]&]
        ];
      ];

      AppendTo[graphCounts,Length[tempList]];

      ,{ j ,1,2}
    ];

    counterList=ConstantArray[0,maxDeg−2]; (* Will contain the number of graphs after each of the next
        few iterations .*)

    (* To save memory, we split up the next few iterations . We do it separately for each graph
        resulting of the 2 first iterations . *)
    outputGraphs=Flatten[Reap[
      Do[
        tempList2={g2};

        Do[
          tempList2=iteration [ j ,tempList2, lowerDegreeVerticesList , partition ,maxDeg,v2DegreeGreater];
          counterList [[ j−2]]+=Length[tempList2];

          ,{ j ,3, maxDeg−1}
        ];
```

```

            (* For the last iteration , to save memory, we do not need to save the graphs for later
                 iterations . We only save the graphs which we consider possible candidate 4−cop−win
                graphs. *)
            Do[
              tempList3=newGraphPossibilities[g3, partition [[5,−1]], partition [[2]], lowerDegreeVerticesList ,
                   maxDeg,v2DegreeGreater];
              counterList [[−1]]+=Length[tempList3];

              (* The viable candidate graphs are those such that for each vertex u of maximum degree, g−
                   N[u] is in graphList (and further down in the list without loss of generality ) . *)
              Sow[CanonicalGraph/@Select[tempList3,validGraph[#,graphList,{ partition [[1,1]], partition
                    [[6,1]]}]&]];

              tempList3={};

              ,{g3,tempList2}
            ];

            tempList2={};
            ,{g2,tempList}
          ]
        ][[2]]];

      (* We merge the counts. *)
      graphCounts=Join[graphCounts,counterList];

      (* We append the number of graphs we output. *)
      AppendTo[graphCounts,Length[outputGraphs]];

      (* We now remove all isomorphic graphs. *)
      outputGraphs=DeleteDuplicates[outputGraphs];
      AppendTo[graphCounts,Length[outputGraphs]];

      totalGraphsGenerated+=Length[outputGraphs];
        ][[1]];

      AppendTo[graphCounts,timing];

      (* We export the results . *)
      PutAppend[graphCounts,outputFile];
      Export["./"<>"finalgraphs_"<>parameterToFileName<>"/finalgraphs_"<>parameterToFileName<>"
           _"<>ToString[i]<>".g6",outputGraphs,"Graph6"];

      , {i , res , Length[baseGraphs],mod} (* We do the computation for each choice of graph (in the proper
           modulo class) from part 1 of the algorithm . *)
  ];
   ][[1]];

  PutAppend[{Total,totalGraphsGenerated,totalTime},outputFile];
]
 fillGraphs [nbTotalVertices_,v1degree_,g1MaximumDegree_,maxDeg_,testAll_,v2DegreeGreater_]:=
      fillGraphs[nbTotalVertices,v1degree,g1MaximumDegree,maxDeg,testAll,v2DegreeGreater,1,1] (* Version of
      the function with only one part. *)


fillGraphs@@(ToExpression/@$ScriptCommandLine[[2;;]]) (* For calls from a shell . Otherwise, call
       fillGraphs with the desired parameters. *)
```