Classical and Parameterized Complexity of Cliques and Games

By

Allan Edward Jolicoeur Scott
B.Sc., University of Victoria, 2002

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER of SCIENCE

in the Department of Computer Science

Abstract

We investigate several variants of the venerable clique problem, all of which turn out to be NP-complete, and most of which are also $W[1]$-hard. One, called PROFIT CLIQUE, has an interesting relationship with the existing problems PROFIT COVER and INDEPENDENT SET. Another variant, called EDGE-PROFIT CLIQUE, turns out to be a generalization of BALANCED COMPLETE BIPARTITE SUBGRAPH, a problem which has yet to be classified in the context of parameterized complexity.

Further, we explore some variant rules for the COPS & ROBBERS problem inspired by the board game Scotland Yard, and show that the problem is PSPACE-complete when many cops are given a limited number of turns to catch a single robber who must disclose partial information about his or her location. In addition, we provide a general FPT result for a specific class of games and puzzles.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to thank my friends who filled the role of "opponent" to further my game research, my colleagues who pointed me in the right directions when I was just getting started, and my family for their support. Finally, I would like to thank my fellow PITA members, especially my supervisor Ulrike Stege, for all their feedback and countless pieces of good advice.

Chapter 1

Introduction

This thesis is composed of two parts. The first part explores variants of CLIQUE.

The CLIQUE problem appears to have been first outlined in [LP49] - an attempt to define what social scientists refer to as a "sociometric clique". It turns out that while a complete subgraph has many uses, it is actually ill-suited to this original task for which it was named [Luc50, Mok79, WF94]. In addition, theoretical computer scientists have shown that CLIQUE itself is very difficult to solve [DF95a, Kar72, Has99]. In Part I of this thesis, we search for a solution to both these deficiencies - a broader, more tractable version of CLIQUE.

This search coincides with results by Stege and van Rooij [SvRHH02, SvR04], who found *profit* versions of VERTEX COVER and DOMINATING SET that admitted faster algorithms and accepted broader solution sets than the original problems. We employ this same tactic with CLIQUE to create

several new variants.

In Chapter three, we explore PROFIT CLIQUE, which is closely related to two previously investigated profit problems, PROFIT COVER and PROFIT INDEPENDENCE. Then in Chapter four we study EDGE-PROFIT CLIQUE, which turns out to be a generalization of BALANCED COMPLETE BIPARTITE SUBGRAPH. We finish the part in Chapter five with a couple more interesting clique variants, and then a summary in Chapter six.

In the second part, we turn our attention to games.

Games are a very rich source of complexity. In fact, in the setting of classical complexity virtually all "interesting" games turn out to be hard for PSPACE or even EXP, while their parameterized counterparts are usually hard for $AW[*]$ or XP. In effect, most games are "harder" than NP and $W[1]$-complete problems.

Chapter seven provides a brief introduction to the field of algorithmic combinatorial game theory, and concludes with a general parameterized result for a specific class of games. In Chapter eight, we move on to the complexity of a variant of the cops and robbers problem based on the board game Scotland Yard. We show this variant to be PSPACE-complete and $W[2]$-hard. Finally, we summarize the part in Chapter nine.

For convenience, Appendix A contains the definitions for every problem discussed, along with complexity results and references.

Chapter 2

Background

2.1 NP-completeness

This thesis focuses on the computational complexity of many different problems. While we will quickly skim the absolute basics here, those who are interested and have not done so already should read [GJ79], as much of this summary was derived from it.

Two of the most basic classes in computational complexity are P and NP. A decision problem is in P if there is a polynomial-time algorithm that solves it. A problem is in NP if there is is a nondeterministic polynomial-time algorithm that solves it. Clearly, any problem in P is also in NP, so P $\subseteq$ NP.

One of the great unanswered questions of complexity theory is whether P $\subset$ NP or P = NP. There is no proof either way, though there is "convincing evidence" of the former, which will be explained shortly. First, we need more concepts.

The first concept is that of a *polynomial-time reduction.* We say there is a polynomial time reduction from problem X to problem Y (or X reduces to Y) if there is an algorithm which, in polynomial time, transforms any instance of X into an instance of Y so that we can retrieve a solution for that X-instance from the solution for the transformed Y-instance. If there is a polynomial-time reduction from X to Y, then the following must hold: if Y $\in$ P then X $\in$ P, and conversely if X $\notin$ P then Y $\notin$ P.

In [Coo72] Cook established the class of NP-complete problems by proving that every problem in NP can be reduced in polynomial time to the problem SATISFIABILITY, implying that if any problem in NP is not in P, then SATISFIABILITY is not in P. This made SATISFIABILITY the first NP-complete problem.

Now that the class of NP-complete problems is established, the usual method of proving a problem to be NP-complete is to show that it is both in NP and NP-hard, while the usual method of proving a problem to be NP-hard is to show there is a polynomial-time reduction from a known NP-complete problem to it.

So now for the "convincing evidence" alluded to above: since every NP-complete problem reduces to every other NP-complete problem in polynomial time, the entire class would be subsumed by the class P if even a single NP-complete problem was shown to be in P. The "convincing evidence" is the fact that P has not yet subsumed NP despite the vast number of people who have worked at solving these NP-complete problems.

2.2 Parameterized Complexity

Showing a problem to be NP-hard is only the beginning. An NP-hardness proof is useful in that it offers convincing evidence that there is no polynomial time algorithm to solve that problem, but it does not offer any more insight into how we might solve the problem efficiently.

Enter fixed-parameter tractability and the class of problems called FPT, first established by Downey and Fellows in [DF92] and [DF95b] (for a complete treatment of the subject, see [DF98]). If a problem has one or more parameters (other than the size of the input) that can be fixed to result in a running time that is no worse than exponential in the size of the fixed parameters, and no worse than polynomial in the size of the input itself, then we say the problem belongs to the class FPT for those parameters. This is because it will behave in a polynomial manner when those parameters remain small. Specifically, a decision problem of size x is said to be *in FPT* for parameter k if there is an algorithm to determine the problem which has a running time with an upper bound of $f(k)|x^c|$, where c is a constant.

Even if a problem is NP-complete, with fixed-parameter tractability there is still hope of finding a fast algorithm for solving the problem. VERTEX COVER, for example, is NP-complete, but it is also fixed-parameter tractable for parameter k and as such there have been very fast algorithms found to solve it.

Of course, parameterized tractability comes with its own parameterized version of intractability, and in this case it comes in the form of the W-

hierarchy. Downey and Fellows define the W-heirarchy with their normalization theorem, which states that for all $t \geq 1$, WEIGHTED t-NORMALIZED SATISFIABILITY is complete for $W[t]$ [DF92].

If a problem is shown to be $W[t]$-hard for parameter k (where t is a natural number), it is considered "strong evidence" that the problem is not in FPT for parameter k in precisely the same way NP-hardness is considered strong evidence a problem is not in P.

The standard method of proving parameterized hardness is with a parameterized reduction, which is in essence a classical reduction except in that we accept any transformation that takes fixed-parameter tractable time and which keeps the parameters independent of the input size.

Part I

Profit Variants of Clique

Chapter 3

Profit Clique

The concept of a clique has a long tradition in social science [Alb73, LP49, Luc50], where it refers simply to a densely connected group of individuals: Consider a graph where every vertex denotes an individual and every edge denotes the presence of a "social relation". Then, informally, we could say that a subset of individuals is considered to form a social clique if the subgraph induced by that subset is sufficiently dense.

One of the first pieces of work in this field, [LP49], explored using complete subgraphs as sociometric cliques, but it is now widely recognized that this definition has two significant drawbacks: First, it is too restrictive and does not fit the intuitive idea of a social clique [Luc50, Mok79, WF94]. From a social science perspective, subgraphs that are almost complete—or otherwise very dense—are also considered to reflect social cliques. Second, the problem of determining whether a given graph contains a complete subgraph of a given size remains decidedly hard. Computer scientists have shown that

the CLIQUE problem is NP-complete [Kar72], $W[1]$-complete for parameter k [DF95a], and even hard to approximate [Has99].

This research was motivated by both drawbacks of the original definition of clique. The approach is parallel to that of social scientists: we relax the original definition of clique to include less-than-complete subgraphs [Luc50, Mok79, WF94].

Given the successes of [SvRHH02] and [SvR04], there seemed to be reason to think that some profit versions of CLIQUE might prove more tractable. Though this was not the case for the profit relaxations explored here, the problems remain and display interesting relationships with other problems.

3.1 Terminology

Let $G = (V, E)$ be a graph and $V' \subseteq V$. We use the three following shorthand terms to present many of our problem definitions:

$$\begin{aligned} E_{\text{IN}}(G, V') &= \{uv \in E : u, v \in V'\} \\ E_{\text{OUT}}(G, V') &= \{u, v \in V' : u \neq v \wedge uv \notin E\} \\ E_{\text{ADJ}}(G, V') &= \{uv \in E : u \in V' \vee v \in V'\} \end{aligned}$$

The graph $G(V') = (V', E_{\text{IN}}(G, V'))$ is the subgraph of $G = (V, E)$ induced by $V' \subseteq V$. The degree of a vertex v in a graph $G = (V, E)$ is denoted by $\deg_G(v)$.

Note that a subset $V' \subseteq V$ is an independent set in G if and only if $E_{\text{IN}}(G, V') = \emptyset$, V' is a clique in G if and only if $E_{\text{OUT}}(G, V') = \emptyset$, and V' is

a vertex cover for G if and only if $E_{\text{ADJ}}(G, V') = E$.

Let $\overline{G} = (V, E_{\text{OUT}}(G, V))$ denote the *complement* of G. Also, let $G(V')$ denote the subgraph of G induced by V'. That is, $G(V') = (V', E_{IN}(G, V'))$.

In a bipartite graph $G = (V, E)$, with $V_1 \cup V_2 = V$, $V_1 \cap V_2 = \emptyset$, and $E \subseteq V_1 \times V_2$, we refer to V_1 and V_2 as the *partite sets* of G.

3.2 Complexity of Profit Clique

Our first variant of CLIQUE is based on an intuitive profit formula in the same vein as PROFIT COVER[1] [SvRHH02]; it promotes having more vertices and penalizes missing edges.

> PROFIT CLIQUE
>
> *Input*: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.
>
> *Question*: Does G contain a *profit clique* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{PCl}(G, V') = |V'| - |E_{OUT}(G, V')|$?

We show that there is a strong relationship between CLIQUE and PROFIT CLIQUE in the following lemma.

Lemma 1. *A graph $G = (V, E)$ contains a profit clique of profit p if and only if G contains a clique of size p.*

Proof. ($\Rightarrow$): Consider each $v \in V'$. If v is independent from (does not have edges to) any other vertices in V', then we can remove v from V' and the

[1]The problem definition of PROFIT COVER is reproduced in the appendix.

profit will not decrease ($|V'|$ decreases by one, while $|E_{OUT}(G, V')|$ decreases by at least one). By the time this process is finished, E_{OUT} will be 0 and the profit will not have decreased, so we will have a clique of at least size p.

($\Leftarrow$): If there is a clique of size p then there is a profit clique of profit p. The profit of a clique of size p is $|V| - |E_{OUT}(G, V')| = p - 0 = p$. Hence, there must be a profit clique of profit at least p. □

Hence we have that there is a profit clique of profit p in G if and only if there is a clique of size p in G. This leads us to the following corollary.

Corollary 1. PROFIT CLIQUE *is NP-complete and* $W[1]$*-complete for parameter* p.

Proof. Membership in NP is trivial, as we can test a potential solution in polynomial time. Membership in $W[1]$ is shown by using Lemma 1 as a reduction from PROFIT CLIQUE to CLIQUE.

Hardness for both NP and $W[1]$ comes by using Lemma 1 in the other direction as a reduction from CLIQUE to PROFIT CLIQUE. □

3.3 Relationships with Other Profit Problems

We can show that PROFIT CLIQUE is directly related to the problem PROFIT INDEPENDENCE, first introduced in [vR03].

PROFIT INDEPENDENCE

Input: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.

Question: Does G contain a *profit independence* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{PI}(G, V') = |V'| - |E_{IN}(G, V')|$?

Lemma 2. *Solving* PROFIT INDEPENDENCE *is equivalent to solving* PROFIT CLIQUE *on the complement graph and vice versa.*

Proof. This follows directly from the fact that $E_{IN}(G, V') = E_{OUT}(\bar{G}, V')$. □

Since we can transform a graph into its complement in polynomial time, Lemma 2 gives us a polynomial-time, parameterized reduction from PROFIT CLIQUE to PROFIT INDEPENDENCE and back again.

We can also demonstrate a relationship with the problem PROFIT COVER, first presented in [SvRHH02].

PROFIT COVER

Input: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.

Question: Does G contain a *profit cover* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{PC}(G, V') = |E_{ADJ}(G, V')| - |V'|$?

First, we observe a relationship between $E_{IN}(G, V')$ and $E_{ADJ}(G, V - V')$:

$$\begin{aligned} & E_{IN}(G, V') \\ = \; & \{(u, v) \in E : u \in V' \wedge v \in V'\} \\ = \; & E - \{(u, v) \in E : u \in V - V' \vee v \in V - V'\} \\ = \; & E - E_{ADJ}(G, V - V') \end{aligned}$$

Now we can see the relationship between PROFIT INDEPENDENCE and PROFIT COVER:

$$\begin{aligned} p &\leq |V'| - |E_{IN}(G, V')| \\ p &\leq (|V| - |V - V'|) - (|E| - |E_{ADJ}(G, V - V')|) \\ p - |V| + |E| &\leq |E_{ADJ}(G, V - V')| - |V - V'| \end{aligned}$$

Corollary 2. *A graph G contains a profit independence of profit p if and only if graph G contains a profit cover of profit $p - |V| + |E|$.*

Not surprisingly, the relationships between these profit problems mirror those of their parent problems.

Chapter 4

Edge-Profit Clique

In this chapter, we consider a new variation of the clique concept:

> EDGE-PROFIT CLIQUE
>
> *Input*: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.
>
> *Question*: Does G contain an *edge-profit clique* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{EPCl}(G, V') = |E_{IN}(G, V')| - |E_{OUT}(G, V')|$?

EDGE-PROFIT CLIQUE is a fairly intuitive generalization of the CLIQUE problem in that it will accept only graphs which are fairly dense. In fact, the higher our parameter (p), the more dense a graph we require as a solution.

We prove that EDGE-PROFIT CLIQUE is NP-hard with a reduction from the known NP-complete problem BALANCED COMPLETE BIPARTITE SUBGRAPH [GJ79].

BALANCED COMPLETE BIPARTITE SUBGRAPH

Input: Bipartite graph $G = (V_1 \cup V_2, E)$, $k \in \mathbb{Z}^+$.

Question: Does there exist a balanced complete bipartite subgraph of size k in G (i.e. are there two subsets $A \subseteq V_1$ and $B \subseteq V_2$ such that $|A| = |B| = k$ and $u \in A$, $v \in B$ implies that $(u, v) \in E$)?

It is important to note that BALANCED COMPLETE BIPARTITE SUBGRAPH takes only bipartite graphs as input.

4.1 Complexity of Edge-Profit Clique

Definition 1. *Given a graph $G = (V, E)$, an edge-profit clique $V' \subseteq V$ in G is said to be* minimal *if there does not exist $v \in V'$ such that profit$(G, V' - v) \geq$ profit(G, V').*

That is, a minimal edge-profit clique is one in which the removal of any single vertex would reduce the profit.

It should be observed that this definition does not preclude the possibility of attaining higher profit by removing several vertices. An example of this situation occurs in the prism of a four-clique, as is shown in Figure 4.1.

Lemma 3. *Every edge-profit clique contains within it a minimal edge-profit clique of the same or greater profit.*

Proof. If an edge-profit clique is not minimal, we can remove a vertex without loss of profit. Continue removing vertices like this until we can not remove

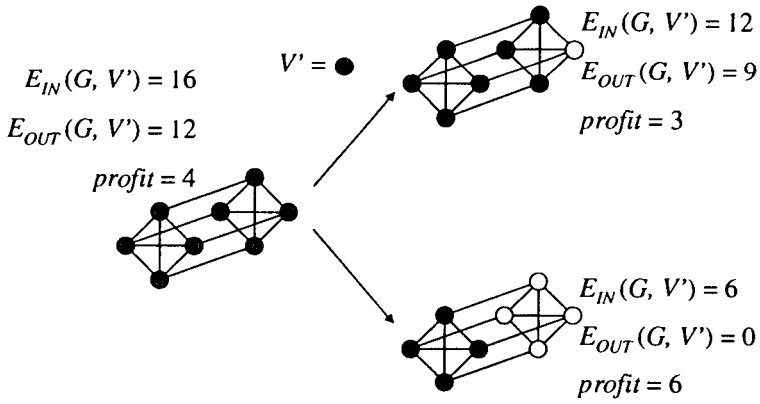

Figure 4.1: Edge-profit cliques in the prism of a four-clique

a single vertex without reducing the profit. Since we never lose profit, the resulting subgraph must be a minimal edge-profit clique with at least as much profit as the original. □

In the degenerate case that occurs when an edge-profit clique is an independent set, the minimal edge-profit clique it contains is the empty set.

Lemma 4. *Given a graph $G = (V, E)$ and a minimal edge-profit clique $V' \subseteq V$, every vertex $v \in V'$ must have induced degree $deg(G(V'), v) \geq \left\lceil \frac{|V'|}{2} \right\rceil$.*

Proof. For any vertex $v \in V'$:

$$\begin{aligned} &\text{profit}\,(G, V' - v) - \text{profit}\,(G, V') \\ &= (|E_{\text{IN}}\,(G, V' - v)| - |E_{\text{IN}}\,(G, V')|) - (|E_{\text{OUT}}\,(V' - v)| - |E_{\text{OUT}}\,(G, V')|) \\ &= \left(-\text{deg}_{G(V')}\,(v)\right) - \left(-\left(|V'| - \text{deg}_{G(V')}\,(v) - 1\right)\right) \end{aligned}$$

For the removal of v to result in decreased profit, we must have that

$\deg_{G(V')}(v) > |V'| - \deg_{G(V')}(v) - 1$ which simplifies to $\deg_{G(V')}(v) > \frac{|V'|-1}{2}$.

□

Intuitively, every vertex must have edges to more than half the other vertices in the edge-profit clique or we would be able to remove it without reducing the profit.

Lemma 5. *Let $G = (V, E)$ be a graph. If $V' \subseteq V$ is a minimal edge-profit clique, then the diameter of $G(V')$ is at most 2.*

In other words, any two non-adjacent vertices of V' have a common neighbour in V'.

Proof. Suppose we have a graph $G = (V, E)$ and a minimal edge-profit clique $V' \subseteq V$ which has two non-adjacent vertices $u \in V'$ and $v \in V'$ with no common neighbour. Then $|V'| \geq 1 + deg_{G(V')}(u) + 1 + deg_{G(V')}(v)$.

However, since V' is a minimal edge-profit clique, we have by Lemma 4 that:

$$|V'| \geq 1 + deg_{G(V')}(u) + 1 + deg_{G(V')}(v) \geq 2 + \left\lceil \frac{|V'|}{2} \right\rceil + \left\lceil \frac{|V'|}{2} \right\rceil \geq 2 + |V'|$$

Contradiction. □

Lemma 6. *Given a graph $G = (V, E)$ and a minimal edge-profit clique $V' \subseteq V$, if $G(V')$ is bipartite then it is also complete.*

Proof. The shortest path between two non-adjacent vertices in opposite partite sets of a bipartite graph is at least 3, violating Lemma 5. □

Lemma 7. *Given a graph $G = (V, E)$ and a minimal edge-profit clique $V' \subseteq V$, if $G(V')$ is bipartite then it is also balanced.*

Proof. If the number of vertices from each partite set of a bipartite minimal edge-profit clique is m and n, then a vertex in the size-m partite set can have degree at most n. By Lemma 4, $n \geq \lceil \frac{m+n}{2} \rceil$. But if $m > n$, then $\lceil \frac{m+n}{2} \rceil > \lceil \frac{n+n}{2} \rceil = n$. Contradiction. □

Lemma 8. *If $G = (V_1 \cup V_2, E)$ is a balanced complete bipartite graph, then profit$(G, V_1 \cup V_2) = |V_1|$.*

Proof. In a size-k balanced complete bipartite graph there are k^2 edges. There are $\frac{k(k-1)}{2}$ missing edges in each partite set. $|E_{IN}(G, V')| - |E_{OUT}(G, V')| = k^2 - \frac{2k(k-1)}{2} = k$. □

Theorem 1. *If G is a bipartite graph, then there is a balanced complete bipartite subgraph of size k in G if and only if there is an edge-profit clique of profit k in G.*

Proof. ($\Rightarrow$): By Lemma 8, a balanced complete bipartite subgraph of size k will yield a profit of k, so there is an edge-profit clique of profit k.

($\Leftarrow$): By Lemma 3, an edge-profit clique of profit k must contain a minimal edge-profit clique of profit at least k. By Lemmas 6 and 7, this minimal edge-profit clique is a balanced complete bipartite subgraph with a profit of at least k. By Lemma 8, the size of this subgraph is at least k. □

Corollary 3. EDGE-PROFIT CLIQUE *is NP-complete.*

Proof. From Theorem 1, he have a reduction from BALANCED COMPLETE BIPARTITE SUBGRAPH to EDGE-PROFIT CLIQUE. This proves NP-hardness, and membership is trivial. □

4.1.1 Parameterized Complexity of Edge-Profit Clique

The reduction yielded by Theorem 1 is trivially a parameterized once, since both the graph and parameter go unchanged. From this we can state that if EDGE-PROFIT CLIQUE is in FPT, then so is BALANCED COMPLETE SUBGRAPH. Conversely, if BALANCED COMPLETE BIPARTITE SUBGRAPH is W[1]-hard, then so is EDGE-PROFIT CLIQUE. However, at this time the parameterized complexity of both remains unknown.

Chapter 5

More Clique Problems

In this chapter we consider two more variants of the clique problem. VIABLE CLIQUE finds subgraphs that are clique-like in the sense that they are dense - in particular they contain only vertices with "high" degree and $|E_{IN}|$ is larger than $|E_{OUT}|$. The other problem, FIXED-SIZE EDGE-PROFIT CLIQUE, adds an additional constraint to EDGE-PROFIT CLIQUE.

Both these problems are shown to be both NP-complete and $W[1]$-hard by the same technique: we transform the problem to an instance on the complement graph and then we show that the property we are looking for is hereditary, thus allowing us to use the following results from [Yan78] and [KR00]:

Theorem 2 ([Yan78]). *If Π is a graph property that has all the following properties:*

- *Hereditary - if graph G has the property, then all induced subgraphs of*

G will have it.

- *Nontrivial - true for a single node and is not satisfied by all the graphs in a given input domain.*
- *Interesting - there is no upper bound on the order of a graph that can satisfy* Π.

then the node-deletion problem for Π *is NP-complete.*

Theorem 3 ([KR00]). *Let* Π *be a hereditary property that includes all independent sets but not all cliques (or vice versa). Then the problem* $P(G, k, \Pi)$ *is* $W[1]$*-hard.*

5.1 Viable Clique

VIABLE CLIQUE asks the question, "can we find a size-k subset of the given graph which is edge-profit minimal?"

VIABLE CLIQUE
Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V|$.
Question: Does G contain a *viable clique* of size k, i.e. a subset $V' \subseteq V, |V'| \geq k$ such that $\forall v \in V', \deg_{G(V')}(v) \geq \frac{k}{2}$?

We prove that VIABLE CLIQUE is NP-complete and $W[1]$-hard for parameter k by introducing VIABLE INDEPENDENCE, showing it is equivalent to VIABLE CLIQUE on the complement graph, and then proving hardness for VIABLE INDEPENDENCE.

VIABLE INDEPENDENCE

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V|$

Question: Does G contain a *viable independence* of size k, i.e. a subset $V' \subseteq V, |V'| \geq k$ such that $\forall v \in V', \deg_{G(V')}(v) < \frac{k}{2}$

Lemma 9. *A graph G contains a viable clique of size k if and only if $\bar{G}$ contains a viable independence of size k.*

Proof. We observe that $\deg_{G(V)}(v) \geq |V'| - \frac{k}{2} \Leftrightarrow \deg_{\overline{G}(V)}(v) < \frac{k}{2}$. Hence, if V' is a viable clique of size k in G then V' is a viable independence of size k in $\bar{G}$, and vice versa. □

Corollary 4. VIABLE INDEPENDENCE *and* VIABLE CLIQUE *are NP-complete.*

Proof. Clearly, the degree bound of VIABLE INDEPENDENCE is hereditary, non-trivial, and interesting. Thus we can conclude by Theorem 2 that VIABLE INDEPENDENCE is NP-complete. This makes VIABLE CLIQUE NP-complete by reduction from VIABLE INDEPENDENCE via Lemma 9. □

In addition, since the degree bound is satisfied by every vertex in any independent set, but not by any vertex in a sufficiently large clique, we conclude by Theorem 3:

Corollary 5. *p*-VIABLE INDEPENDENCE *and p*-VIABLE CLIQUE *are $W[1]$-hard.*

5.2 Fixed-Size Edge-Profit Clique

FIXED-SIZE EDGE-PROFIT CLIQUE

Input: A graph $G = (V, E), k \in \mathbb{Z}^+ : k \leq |V|, p \in \mathbb{Z}^+$.

Question: Does G contain a profit clique of size k and profit p, i.e. a subset $V' \subseteq V, |V'| = k$ such that $p \leq \text{profit}_{\text{EPCl}}(G, V') = E_{\text{IN}}(G, V') - E_{\text{OUT}}(G, V')$?

This fixed-size variant of EDGE-PROFIT CLIQUE is interesting from a parameterized point of view because it gives an additional parameter to work with - the size of the profit clique.

As before, we prove hardness from the equivalent problem on the complement graph.

FIXED-SIZE EDGE-PROFIT INDEPENDENCE

Input: A graph $G = (V, E), k \in \mathbb{Z}^+ : k \leq |V|, p \in \mathbb{Z}^+$

Question: Does G contain a profit independence of size k and profit p, i.e. a subset $V' \subseteq V, |V'| = k$ such that $p \leq \text{profit}_{\text{EPI}}(G, V') = E_{\text{OUT}}(G, V') - E_{\text{IN}}(G, V')$?

Lemma 10. *A graph G contains a size-k profit clique of profit p if and only if $\bar{G}$ contains a size-k profit independence of profit p.*

Proof. Suppose we have a subset of vertices V' from a graph G. Then,

because $E_{\text{IN}}(G, V') = E_{\text{OUT}}(\overline{G}, V')$:

$$\begin{aligned} profit_{\text{FSEPCI}}(G, V') &= E_{IN}(G, V') - E_{OUT}(G, V') \\ E_{IN}(G, V') - E_{OUT}(G, V') &= E_{OUT}(\overline{G}, V') - E_{IN}(\overline{G}, V') \\ E_{OUT}(\overline{G}, V') - E_{IN}(\overline{G}, V') &= profit_{\text{FSEPI}}(G, V') \end{aligned}$$

Therefore, if V' is a size-k profit clique of profit p in G it must also be a size-k profit independence of profit p in $\bar{G}$ and vice versa. □

Theorem 4. FIXED-SIZE EDGE-PROFIT INDEPENDENCE *and* FIXED-SIZE EDGE-PROFIT CLIQUE *are NP-complete.*

Proof. We take the profit condition for FIXED-SIZE EDGE-PROFIT INDEPENDENCE, substitute k for V', and then perform some algebraic manipulation to get:

$$|E_{\text{IN}}(G, V')| \leq \frac{k(k-1) - 2p}{4}$$

Clearly this property is hereditary, nontrivial, and interesting. Thus, by Theorem 2 we conclude that FIXED-SIZE EDGE-PROFIT INDEPENDENCE is NP-complete and, by Lemma 10, that FIXED-SIZE EDGE-PROFIT CLIQUE is also NP-complete. □

Theorem 5. (k,p)-FIXED-SIZE EDGE-PROFIT INDEPENDENCE *and* (k,p)-FIXED-SIZE EDGE-PROFIT CLIQUE *are* $W[1]$*-hard.*

Proof. Since our hereditary property is an upper limit on the number of edges, it's clear that it holds for all independent sets but not for all cliques. Hence, by Theorem 3, our claim is proved. □

Chapter 6

Summary and Open Questions

The complexity results that have been achieved in this part are summarized in Table 6.

We have been unable to find the parameterized complexity of EDGE-PROFIT CLIQUE and BALANCED COMPLETE BIPARTITE SUBGRAPH, though we have shown a link between them. It is left as an open question.

Barring an unlikely FPT result for EDGE-PROFIT CLIQUE, we did not

Problem	**Classical**	**Parameterized**
PROFIT CLIQUE	NP-complete	$W[1]$-complete
EDGE-PROFIT CLIQUE	NP-complete	-
VIABLE CLIQUE	NP-complete	$W[1]$-hard
FIXED-SIZE		
EDGE-PROFIT CLIQUE	NP-complete	$W[1]$-hard

Table 6.1: Complexity results proved in Part I

achieve our original goal of finding a profit variant of clique which was in FPT. Finding one remains an open problem.

Finally, we pose one more open question that arises from the relationship between EDGE-PROFIT CLIQUE and BALANCED COMPLETE BIPARTITE SUBGRAPH: can we extend EDGE-PROFIT CLIQUE to generalize BALANCED COMPLETE MULTIPARTITE SUBGRAPH?

BALANCED COMPLETE MULTIPARTITE SUBGRAPH

Input: $G = (V, E)$, where V is partitioned into $V_1, V_2, \ldots, V_\ell$ and there is no edge $e \in E$ such that $e \in V_i \times V_i$ for all $i : 1 \leq i \leq \ell$, $k \in \mathbb{Z}^+$

Question: Do there exist $V_1' \subseteq V_1$, $V_2' \subseteq V_2$, ..., $V_\ell' \subseteq V_\ell$ with $|V_1'| = |V_2'| = \cdots = |V_\ell'| = k$?

Part II

Games and Puzzles

Chapter 7

Games in General

7.1 Introduction

Many games give the appearance of being complex simply by virtue of how many different positions can occur. Any human hoping to win simply by considering all possible future moves will quickly get bogged down with a combinatorial explosion of positions. Even the most practised players can only look ahead a few moves.

It turns out that this complexity is often a key component to making a game "fun". Any game which is not provably hard usually affords some "trick" to winning, and once the trick is discovered the players need only to follow the algorithm that assures a good result. This makes the game uninteresting.

In this part, we refer to both games and puzzles. While puzzles can sometimes be viewed as simply being one-player games (or vice-versa - games

as competitive puzzles), the questions we ask in either case are significantly different.

In the case of games (consisting of two or more players), we ask whether there is a strategy such that one player can always force a win. That is, we ask if it is possible for this player to win no matter what his opponents do. Such a strategy is called a *winning strategy.* In the case of puzzles, the question we ask is simply whether there is a solution to the puzzle.

We could also form optimization problems for games and puzzles as follows: In a game, we ask what is the fewest number of moves required for player one to force a win. In a puzzle, we ask what is the fewest number of moves required to achieve a solution. We could also form a decision version from this optimization problem and ask whether player one can force a win within t moves. This corresponds to the *short* variants of games studied by Downey and Fellows in [DF98]. We also adopt this convention and apply it to puzzles as well, where the question is simply whether a solution can be achieved within t moves.

These questions can make very little sense in the wrong context. For example, there is no winning strategy for poker, blackjack, or any other game of chance. If you are dealt a poor hand and your opponent calls, you lose. Most research has concentrated on asking these questions strictly in a deterministic environment. Examples include chess, checkers, and go. Furthermore, these settings are most often perfect-information in nature. That is, both sides are always fully informed of the game state - there is no information hidden from either. In this part, all the games we consider are

deterministic and the only game we consider that is not perfect-information is Scotland Yard. The question is whether there exists a forced win for player one.

7.2 Game (and Puzzle) Trees

The most basic tool available for solving games is the *game tree.* Figure 7.1 gives an example of a game tree for tic-tac-toe. At the root node of the figure, Player I (X) and Player II (O) have both moved three times and it is Player I's turn again.

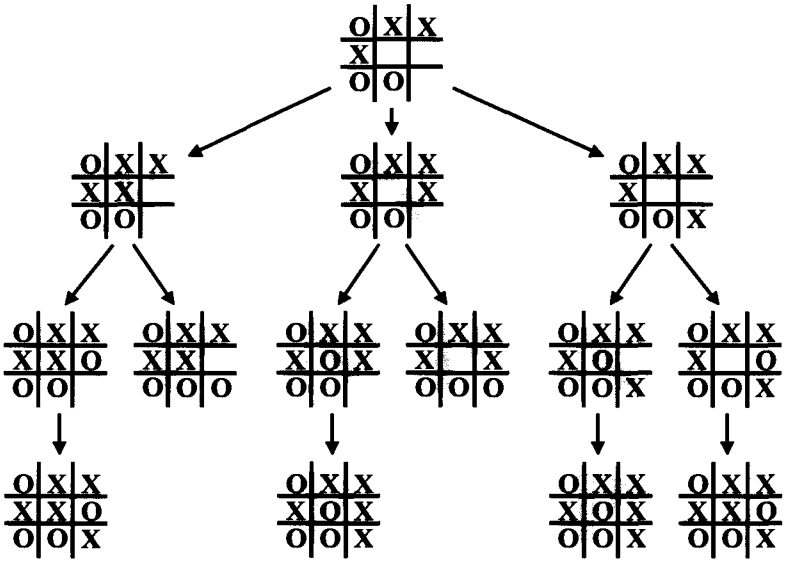

Figure 7.1: A partial game tree for Tic-Tac-Toe

A game tree is an acyclic directed graph in which nodes represent game positions and edges represent possible moves. The game tree of a two-player partisan[1] game will also be a bipartite graph. *Terminal nodes* (nodes with an out-degree of zero) indicate endgame positions. They can be of three types: won (the player whose turn it is has won), lost (the player whose turn it is has lost), and tied (no moves are possible, and no player has won or lost). Most of the time, as a feature of the rules, we will have only won or lost positions defined for any given game, but not both. Chess, for example, has no won positions since the only victory condition is a checkmate and a player can not legally move himself into one.

Given the game tree for an alternating two-player game, we can determine the winner of the game by performing a P-N labelling. First, won nodes are labelled N and lost nodes are labelled P. Then, recursively, we label the parent of every P node with an N. Every node whose children are all labelled N becomes a P node. When this labelling is complete, we can determine the nature of any position by its label: under best play, unlabelled nodes produce ties, N nodes produce victories for the player whose turn it is, and P nodes produce losses.

Puzzles can also be solved with the game tree concept. Instead of a P-N labelling, we simply perform a breadth-first search from the starting position to find the shortest path to a solution. We will refer to such a tree as a *puzzle*

[1] A *partisan* game is one in which each player's set of possible moves is disjoint. This is usually achieved by giving players control over disjoint sets of game pieces. For example: in Chess, Player I controls the white pieces, while Player II controls the black.

tree[2].

Solving these deterministic games is the domain of combinatorial game theory. Algorithmic combinatorial game theory [Dem01] is a much younger field in which the focus is finding the complexity of solving these games. For the most part, the work in this document pertaining to games falls clearly within the domain of this second field.

7.3 Classical Complexity

While in the first part of the thesis we could limit our discussion of classical complexity to the theory of NP-completeness, in this part we must introduce another complexity class: PSPACE.

Originally, PSPACE was the class of problems that could be solved by a deterministic Turing machine in polynomial space, while the class NPSPACE was the class of problems that could be solved by a non-deterministic Turing machine in polynomial space. However, it was shown that PSPACE = NPSPACE by Savitch [Sav70].

The class NP is a subset of PSPACE, since NP is clearly a subset of NPSPACE, and NPSPACE = PSPACE. Whether or not NP is a proper subset of PSPACE remains unknown, as with P and NP. Still, PSPACE-

[2]The difference between solving a puzzle tree and solving a game tree provides an intuitive illustration of the difference between a nondeterministic Turing machine and an alternating Turing machine. The puzzle tree needs only a single successful path from start state to solution state, while the game tree requires an entire subtree that takes into account all the possible actions of the player's opponent(s).

hard problems can be considered theoretically "more complex" than NP-hard problems since if it were proven that P = NP, it could still be the case that P $\neq$ PSPACE, while a P = PSPACE proof would immediately imply that P = NP.

Like NP, the set of PSPACE-complete problems was established with a single logic problem in [SM73]. In this case, the problem is QUANTIFIED BOOLEAN FORMULA.

> QUANTIFIED BOOLEAN FORMULA (QBF)
> *Input*: Set $U = u_1, u_2, \ldots, u_n$ of variables, well-formed quantified Boolean formula $F = (Q_1 u_1)(Q_2 u_2) \cdots (Q_n u_n) E$, where E is a Boolean expression and each Q_i is either $\forall$ or $\exists$.
> *Question*: Is F true?

QUANTIFIED BOOLEAN FORMULA is a generalization of SATISFIABILITY since an instance of SATISFIABILITY is also an instance of QUANTIFIED BOOLEAN FORMULA where $\forall i, 1 \leq i \leq n : Q_i = \exists$.

Again, polynomial-time reductions are the tool of choice to prove PSPACE-hardness [GJ79].

In [FG87], and other places, game trees are used to develop a PSPACE-membership proof for a class of games.

In general, puzzles tend to be NP-complete while games are usually PSPACE-complete. There is, however, one notable exception that comes in the form of a class of sliding block puzzles that produce PSPACE-completeness results. Sokoban [Cul98], Rush Hour [FB01], and Lunar Lockout [HLH03]

are examples of this.

7.4 Parameterized Complexity of Games and Puzzles

Fixed-parameter tractability has yet to be applied to many games or puzzles. The author is aware only of two instances prior to this work: the first being the exploration of several games by Downey and Fellows in [DF98] (SHORT GENERALIZED GEOGRAPHY, SHORT NODE KAYLES, PEBBLE GAME, PEG GAME, and CAT AND MOUSE), and the second being the work done on the Rush Hour puzzle in [FHN+03].

Part of the reason for this may be that the results in [DF98] are not very promising in terms of finding FPT results. Downey and Fellows showed that SHORT GENERALIZED GEOGRAPHY is $AW[*]$-complete for its natural parameter, and speculate that this may be the "natural home" for short variants of PSPACE-complete games. $AW[*]$ is another class that contains problems which are plausibly not in FPT, defined as the union of all classes $AW[t]$ for $t \geq 1$. Each individual class $AW[t]$ is defined in turn by the problem QBFSAT$_t$. Both $AW[1]$ and $AW[2]$ have been proved to be equivalent to $AW[*]$, so the heirarchy has collapsed and we are left with $AW[*]$ [DFR97, ADF93].

Downey and Fellows also showed that PEBBLE GAME, PEG GAME, and CAT AND MOUSE are XP-complete for various parameters. XP appears to

be the parameterized counterpart to EXP, the class of problems which can be solved in exponential time (and is provably a proper superset of P). As its parameterized counterpart, XP is the class of problems which provably is a proper superset of FPT.

However, there clearly are cases where games are fixed-parameter tractable. Examples of this are presented below.

Theorem 6. *Short variants of games and puzzles which have fixed branching vectors (that is, the number of choices at any point in the game is bounded from above by either a fixed parameter or a constant) and can be tested for win/loss conditions in FPT time are in FPT for parameter t, the number of turns.*

Proof. Since we need only find solutions within the next t turns, we need not consider positions in the game (or puzzle) tree beyond depth t. If we have a fixed branching vector b, then this means we need consider no more than b^t positions of the game (or puzzle) tree. The amount of work we do at each node is FPT. □

While that may sound quite restricted, it turns out that virtually all games have winning/losing conditions which can be derived in polynomial time. As such, there are situations where this theorem applies, two of which we show in the following corollary.

Corollary 6. SHORT PLANAR BIPARTITE GENERALIZED GEOGRAPHY *and* SHORT $N^2 - 1$ PUZZLE *are in FPT for parameter t(see Appendix A for definitions).*

Proof. PLANAR BIPARTITE GENERALIZED GEOGRAPHY is a game played on a directed graph which has maximum out-degree 2, and thus a fixed branching vector of 2. As for the $N^2 - 1$ puzzle, it has a maximum fixed branching vector of 4: there is always one position open to slide a tile into, and there cannot be more than 4 tiles adjacent to slide into it. In both cases, checking for won/lost conditions are trivially polynomial. □

While PLANAR BIPARTITE GENERALIZED GEOGRAPHY is not a game many would want to play in practice, many board games (including go, checkers, othello, and hex) have been proved PSPACE-hard by reduction from it, so the parameterized complexity of the short variants of these games is still very much an open problem.

The $N^2 - 1$ puzzle, which is a generalization of the 15-puzzle, is much more popular and has been well-studied [Sto79, RW90, Arc99]. However, until now there was no known parameterized complexity result.

Chapter 8

Scotland Yard

The pursuit-evasion problem (also called cops and robbers) has been fairly well-studied and has many variants. The common elements are that this pursuit takes place on a graph, with the cops and robbers occupying the vertices. If ever a cop occupies the same vertex as a robber, the robber is caught. Depending on the variant, the graph can be directed or undirected, the robber(s) may or may not be visible, the cops may move along edges or by "helicopter" (i.e., to any vertex they choose), and the robber(s) may move along a fixed or infinite number of edges each turn. Possibly the most famous variant is one in which the robber is invisible to the cops and moves with infinite speed. This variant was shown to be equivalent to determining the treewidth of the graph in [ST93]. A summary of classical complexity results for many variants is available in [GR95].

The board game Scotland Yard is clearly inspired by this general cops and robbers problem, though it calls the robber "Mister X" and the cops are

promoted to detectives. It also introduces many new elements:

- The edges of the graph are coloured and all players must pay to travel along an edge with a ticket of the corresponding colour. Each cop (detective) is allocated a block of tickets at the beginning, while the robber (Mister X) collects all tickets spent by the cops, effectively giving him an infinite supply.
- The cops learn the colour of the ticket the robber spent each turn.
- The robber is invisible most of the time, but periodically is forced to reveal his location to the cops.
- The robber possesses a number of black tickets which can be used to travel along any edge, including some the cops may not have tickets for.
- The robber is given a couple 2x coupons which, when spent, allow him to move twice in one turn.

In this chapter, we examine the effects of some of these changes on the complexity of the pursuit-evasion problem.

8.1 A Simplified Version of Scotland Yard

We will define a Cops & Robbers variant using only some of the rules of the board game Scotland Yard. We have omitted some of the rules which would

require more "housekeeping" (such as the ticket quotas and 2x coupons) simply to streamline the proof. We will discuss which elements of Scotland Yard are actually necessary to the PSPACE-hardness proof later.

SIMPLE SCOTLAND YARD (SSY)

Input:

- A graph $G = (V, E_0 \cup E_1 \cup \ldots \cup E_C)$
- A set $X \subseteq V$ of possible starting positions for the robber ($X \neq \emptyset$).
- An actual starting position $x \in X$ for the robber.
- $D \subseteq V$ positions where the cops start.
- a, the number of turns until the robber's next appearance.
- b, the number of turns between the robber's appearances.
- $t \in \mathbb{Z}^+$ the number of turns in the game.

Rules:

- The robber begins at x, while a cop is placed on every node in D.
- First the robber then the cops take alternate turns.
- On his turn, the robber moves to an adjacent and unoccupied vertex. During their turn, all the cops move simultaneously to adjacent vertices.

- Every turn, it is revealed which set the edge the robber travelled along belongs to
- The robber's position is revealed to the cops only after he moves a times and every b turns thereafter. Initially, the cops know only that the robber began at some position in X.
- Only the robber may use the edges in E_0, while the rest of the edges can be used by both him and the cops.
- If a cop enters the node currently occupied by the robber, the robber is captured and Scotland Yard wins.
- If the robber is unable to move because all adjacent nodes are occupied by cops, he must surrender himself and again Scotland Yard wins.
- If Scotland Yard moves t times without capturing the robber, the game ends and the robber wins.

Question: In the game described above, is there a winning strategy for a) the cops (SSY-C) or b) the robber (SSY-R)?

Because this is not a game of perfect information the problem is significantly different depending on whether you ask for a winning strategy for the cops or the robber. In a game where there are no draws we could normally state that one side must have a winning strategy, but in this case it's quite possible that the winner will be determined simply by luck.

Consider the following situation, an example of which is shown in Figure 8.1: the robber may be forced to enter a node adjacent to a cop while he is invisible. If the cops are lucky, one might move into that location. However, since they will move lucky, and

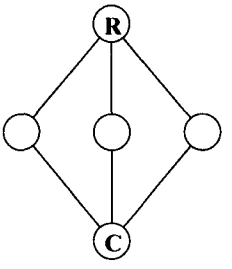

Figure 8.1: A Scotland Yard position that can be won only by luck

8.2 Simplified Scotland Yard is in PSPACE

Theorem 7. SSY-R *is in PSPACE.*

Proof. Our strategy in this proof is to provide an algorithm that determines whether the robber has a winning strategy, and then show that it executes in polynomial space. To do this we will create two functions: `robber-R`, which will handle traversing the positions of the game tree from which the robber moves, and `cop-R`, which will traverse the positions of the game tree from which a cop moves.

The `robber-R` function simply tries moving the robber to every single possible adjacent node, and returns success if any of those nodes eventually led to an escape route. Meanwhile, `cop-R` tries moving each cop to each adjacent node and returns failure (success for the cops) if any of those moves eventually leads to the capture of the robber. Combined, these two algorithms simply perform a traversal and labelling of the game tree.

We can not generate and store the entire game tree in memory, as it is not polynomial in size (and if it were, the problem would trivially be in P). Instead, we must generate the game tree as we go. Also, instead of moving the cops simultaneously we move them in order (this has no consequence on the game, though it does increase the depth of the tree to $t(|D| + 1)$).

The inputs to our algorithm include:

- `G` - The graph on which the game is being played, as defined for SSY.
- `x` - The robber's current position.
- `D[1..d]` - An array of positions for each detective.
- `t` - The number of turns remaining in the game.
- `n` - The number of the cop whose turn it is.

We will also use a helper function, `adj(E, v)`, which returns the set of vertices which share an edge in `E` with the vertex `v`.

The only output produced is a yes or no, corresponding to whether the robber has a winning strategy.

```
Algorithm robber-R(G, x, D[1..d], t)
   if t = 0 then return 'YES'
   for each v ∈ adj(E0 ∪...∪ Ec, x) do
      s ← cop-R(G, v, D[1..d], 1, t)
      if s = 'YES' then return s
   end
   return 'NO'
end

Algorithm cop-R(G, x, D[1..d], n, t)
   if n > d return robber-R(G, x, D[1..d], t - 1)
   u ← D[n]
   for each v ∈ adj(E1 ∪ ...∪ Ec, u) do
      if v = x return 'NO'
      D[n] ← v
      s ← cop-R(G, x, D[1..d], n + 1, t)
      D[n] ← u
      if s = 'NO' return s
   end
   return 'YES'
end
```

Notice that we simply ignore what kind of edge the robber used and whether he appeared. This is safe to do because the algorithm will try every

possible move for the cops regardless of what information they receive.

These two functions run in polynomial space because each function call individually executes in constant space (each call needs only a handful of local variables), and no more than $t(d+2)$ recursive function calls will be placed on the call stack at one time. Hence, the whole problem is solved in polynomial space. □

Theorem 8. SSY-C *is in PSPACE.*

Proof. We prove this using the same basic approach as we did with SSY-R.

Algorithm `robber-C` no longer tracks an actual position for the robber, just the set of positions he might possibly occupy. When it is time for the robber to appear, the function tries having the robber appear at every available location. While this is not what happens in a real game, the cops do not have a winning strategy unless they can handle every possible move by the robber, so it is equivalent.

```
Algorithm robber-C(G, X, D[1..d], t, a, b)
   X ← X \ D
   if X = ∅ then return 'YES'
   if t = 0 then return 'NO'
   Z ← ∅
   if a > 0 then
      for i = 0 to C do
         Y = ∅
         for each x ∈ X do
```

```
            Y ← Z ∪ adj(E_i, x)
         end
         s = cop-C(G, Y, D[1..d], 1, t-1, a-1, b)
         if s = 'No' then return s
      end
      return 'Yes'
   else
      for each x ∈ X do
         for each y ∈ adj(E_0 ∪...∪ E_C, x)
            s ← cop-C(G, {y}, D[1..d], 1, t-1, b-1, b)
            if s = 'No' then return s
         end
      end
      return 'Yes'
   end
end
```

The cop-C function is largely the same as cop-R. The cops still just try every single move available to them.

```
Algorithm cop-C(G, X, D[1..d], n, t, a, b)
   if n > d return robber-C(G, X, D[1..d], t - 1, a, b)
   u ← D[n]
   for each v ∈ adj(E_1 ∪ ...∪ E_C, u) do
      D[n] ← v
```

```
        s ← cop-C(G, X, D[1..d], n + 1, t, a, b)
        D[n] ← u
        if s = 'YES' return s
    end
    return 'NO'
end
```

Again, these functions run in polynomial space, so SSY-R is in PSPACE.

□

8.2.1 Variants

While we defined SSY to use a very specific subset of the variant rules introduced by the Scotland Yard board game, the problem would remain in PSPACE for many other subsets of those rules. For example: we can remove the rule that requires the robber to appear by changing "`a` $>$ `0`" to "`true`" in `robber-C`. Similarly, we can handle 2x coupons by adding a parameter to count them and a branch in the robber function to try using them. We could even handle the limited tickets rule by tracking tickets for each individual cop.

Ultimately, the only variant rule we absolutely need for our PSPACE-membership proof is the rule that restricts the depth of the game tree: the t-turn termination rule. This keeps the game tree from growing too deep to be stored in polynomial space. Whether or not PSPACE-membership can

still be proved without this rule is an open question.

8.3 Simplified Scotland Yard is PSPACE-complete

Theorem 9. *SSY-R and SSY-C are PSPACE-hard.*

Proof. Having already shown both problems to be in PSPACE in the last section, we now show hardness by reducing directly from QUANTIFIED BOOLEAN FORMULA.

The goal of our reduction is to create a game graph on which there will be a winning strategy for the robber if and only if the corresponding QBF formula is satisfiable. (We also get SSY-C because for the kinds of graphs our reduction builds, if one side does not have a winning strategy, the other must. Hence, there is a winning strategy for the cops if and only if there is no satisfying assignment for the corresponding QBF instance, as we show in Claim 3.)

This reduction is similar to the reduction from QBF to Generalized Geography presented by Shaefer [Sch78]. Like Shaefer's reduction, we transform the QBF instance to be in a specific form: there must be an even number of quantifiers, which must alternate between universal and existential (starting with universal). This can be accomplished by adding at most n dummy variables. Then we use logical transformations to put the expression into conjunctive normal form.

There are three components the reduction relies on: a "trunk", a "selector", and some "runways" (Figure 8.5 shows a complete example). Play

in the trunk, in conjunction with the runways, will determine the values we assign to the variables of our QBF instance. Then, in the selector, the cops will be able to force the robber into a clause. From there, the robber can escape (and win the game) if and only if he has satisfied at least one of the literals contained within the clause. The idea is that if there is no satisfying assignment for the QBF instance, then the robber will not have a winning strategy, and vice versa.

To handle the variable assignment, we build the "trunk" - two side-by-side paths with crossover edges. (See Figure 8.2 for an example.) As shown in the figure, the robber starts near the top of the trunk as do two "chaser" cops. The function of these chaser cops is to force the robber to always move "down" the trunk. The height of this trunk from top to bottom is (2+number of quantifiers).

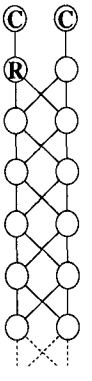

Figure 8.2: The "trunk" gadget

For each variable, we have a forked path which we call a "runway". (See Figure 8.3 for an example.) A cop begins at the beginning of this runway, and the variable is set either true or false, depending on which side of the fork he follows (the variable is set to be the path he does not follow). The length of these runways is exactly such that the cop can reach the end of it on the last turn. That is, there are $t + 1$ nodes from the beginning of the runway to either end.

The robber appears after his first turn and every second turn thereafter. The appearances coincide with the locations on the trunk where he chooses settings for the existentially-qualified variables. At these locations, there are extra edges leading off either path of the trunk: one to the first vertex of the "true" fork of the corresponding variable's "runway", and one to the first vertex of the "false" fork. The cop on this "runway" must move to block the robber's access to it or face certain defeat: the runway is long enough that the robber could run out the rest of his turns fleeing down it while the cops give chase.

The cops get to choose the universally-qualified variables, and so there are to edges to the trunk in the corresponding runways.

At the end of the trunk, we have the "selector" - two sets of vertices, each exactly as large as the number of clauses. (See Figure 8.4 for an example.) This is where the cops get to force the robber into a specific clause. There are robber-only edges from both end vertices of the trunk to every vertex in the first layer of the selector, giving the robber an escape from the chaser detectives in the trunk. There is then an edge connecting every vertex in

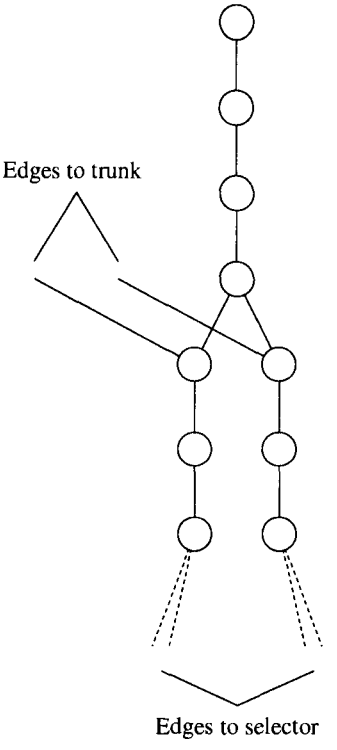

Figure 8.3: A "runway" gadget

the first layer of the selector to every vertex in the second layer, making it a complete bipartite c, c graph where c is the number of clauses. There are $c - 1$ cops in the selector, and they start play in the first layer.

The end of each fork of a runway has robber-only edges connecting it to to each vertex in the second layer of the selector that represents a clause which contained that literal. These edges serve as the robber's escape route. However, the unsatisfied literals will have cops coming down them, so the robber will be captured if he enters them. He will be safe only if he enters a satisfied literal.

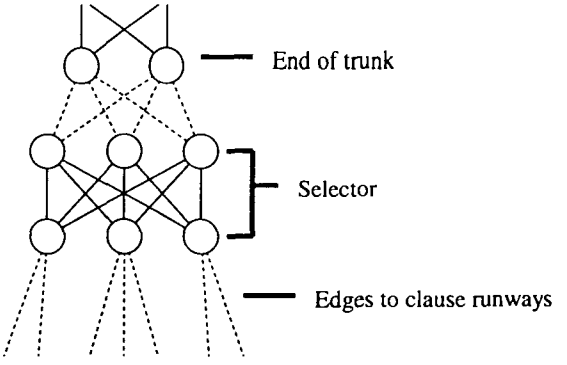

Figure 8.4: The "selector" gadget

There is one additional cop that follows a path such that he appears in the first layer of the selector (with the choice of any position he wants) the turn after the robber moves to the second layer, plugging up the hole to prevent the robber from seeking safety in the first layer.

The number of rounds the robber must survive for is then (number of qualifiers +3).

An example of a game created by this reduction is shown in Figure 8.5. The specific game shown is a no-instance, since the formula is not satisfiable (to assure the robber's capture, the cops choose the same value for Y as the robber does for X).

Since each component is polynomially-sized, and there are a polynomial number of components, this construction would take polynomial time.

Under optimal play, the chaser cops will force the robber down the trunk while the runway cops block any escape into the runways. At the bottom of the trunk the robber enters the selector, and the cops there leave him only

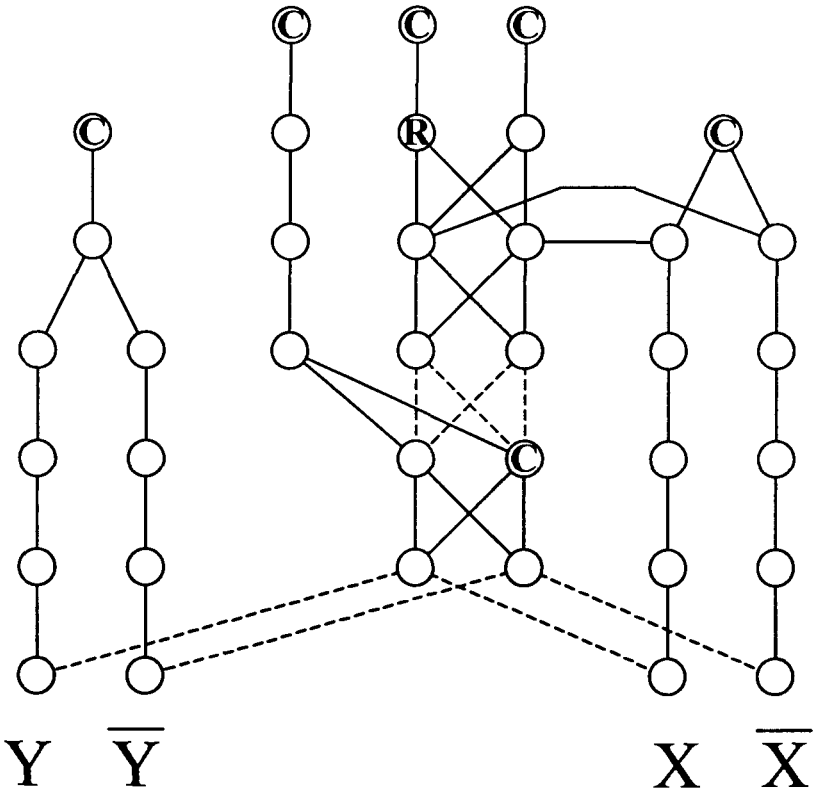

Figure 8.5: A SSY game representing the formula $\exists X \forall Y (X \vee Y) \wedge (\overline{X} \vee \overline{Y})$

one free node in the second layer to enter - corresponding to a clause. Thus, the robber can win only if there is one free runway for him to move into at the very end - a satisfied literal for that clause.

Of course, we must show that other lines of play will not generate an unexpected result. To this end, the next part is structured into claims.

Claim 1. *The chaser cops have no better strategy than to chase the robber down the trunk.*

Proof. If the chaser cops cannot win by chasing the robber down the trunk their only other options are to go back up the trunk or into the runways, but neither will prevent the robber from executing a winning strategy. A cop from the trunk entering a runway will be unable to reach the end before the game ends. A cop which goes back up the trunk accomplishes nothing. □

Claim 2. *The runway cops have no better strategy than to rush to the end of their runway, blocking the robber on the way if necessary.*

Proof. We already noted that the cops lose if the robber is allowed to enter a runway. Entering the trunk is also a poor choice, since it's impossible for the runway cops to catch the robber in the trunk. □

Claim 3. *If there is no solution for the QBF instance, then the cops have a winning strategy and the robber cannot win.*

Proof. If there is no solution to the corresponding QBF instance, then the cops can choose values for the variable to create an unsatisfied clause no matter what values the robber picks for his variables. Then the chaser cops can force the robber down the trunk and into the selector, where the cops there force him into a vertex representing an unsatisfied clause. Since the clause is unsatisfied, there will be a cop coming down the side of the runway the robber is in to capture him. □

Claim 4. *If there is a solution for the QBF instance, then the robber has a winning strategy.*

Proof. To escape the cops, all the robber needs to do is produce a satisfying assignment. Then, when he is forced into a clause by the cops, he will be able to choose a safe runway to escape to. This is guaranteed because in the satisfying assignment there is a satisfied literal for each clause, and if a literal is satisfied then the corresponding runway will be clear. □

These last two claims establish the correctness of the reduction. □

With claims 3 and 4 we have that there must be a winning strategy for one side or the other (either an instance of QBF is satisfiable or it isn't), so as mentioned before we can also use this as a SSY-C reduction.

8.3.1 Variants

We can accommodate most of the variant rules of Scotland Yard within this reduction. Tickets are largely irrelevant, since we only have two edge types: normal edges and robber-only edges. So we can accommodate the ticket rule simply by distributing t normal tickets to each detective, and 2 robber-only tickets to the robber.

The robber-only edges are critical to this reduction and cannot be removed. Without them, the detectives would simply follow the robber out of the selector, virtually ensuring capture.

The periodic appearances are also very important, but can be substituted for by colouring the edges in the trunk and reporting the edges travelled by the robber. However, in this case we can not use the variant rule which allows the robber to use a black ticket to travel along any edge. This would

ruin the reduction with regards to SSY-C, since the robber could then try to fool the cops by forcing them to guess which fork of a runway he's trying to escape into. The exact complexity of these variants for which the reduction does not work remains open.

8.4 Parameterized Complexity

In this section, we will provide a parameterized hardness result for SSY.

Theorem 10. *SSY is W[2]-hard.*

Proof. Let $(G = (V, E), k)$ be an instance of k-DOMINATING SET. We will build an instance of SSY from this instance. Formally:

$$
\begin{aligned}
G' &= (V', E_0 \cup E_1) \\
V' &= V \cup D \cup X \\
|D| &= k \\
E_0 &= \emptyset \\
E_1 &= E \cup \{\forall v \in V, u \in D \cup X : (v, u)\} \cup \{(x_1, x_2)\} \\
C &= 1 \\
X &= \{x_1, x_2\} \\
a &= 2 \\
b &= 2 \\
t &= 2
\end{aligned}
$$

In words: first we create k nodes, which we will call D, plus two more nodes which we will call X. We connect each of these new nodes to every node in the original graph. We place a cop in every node of D. We add one more edge to connect the two nodes of X, and place the robber in one of them. The robber uses his first move to move to the other node in X (not wanting to risk getting caught by accident when the cops move out), then the cops move into position on their turn.

If the cops leave any nodes undominated, the robber moves to one of those on his final turn and then the cops will be unable to catch him on their final turn. Otherwise, the robber must appear adjacent to one of the cops and on his final turn that cop will move in and capture the robber.

Since no aspect of the input size is introduced to the parameter ($|D| = k$), this is a parameterized reduction. Then, because DOMINATING SET is $W[2]$-complete for k, SSY is $W[2]$-hard for $|D|$. □

In fact, SSY is still $W[2]$-hard even if we fix $|D|$, $|E_0|$, C, $|X|$, a, b, and t because those elements are fixed in our reduction. Hence, they will not be in FPT unless P = NP = PSPACE.

8.5 Final Comments

There are certain aspects of the Scotland Yard board game which are not taken into account here: in the original Scotland Yard board game, each edge set E_i is planar and connected. The edge sets also have a hierarchy where a

vertex is only incident to edges of a given set if it is also incident to edges of the previous set ($\forall i > 1 : \forall v \in V : \exists (v, u) \in E_i \rightarrow \exists (v, w) \in E_{i-1}$). It seems unlikely this would have a significant effect on the classical complexity of the problem, but it may affect the parameterized complexity.

Finally, the game can be solved in a probabilistic sense (that is, we can determine the chance of either side winning) by the use of payoff matrices. In this context, we usually assume both players are moving simultaneously (except that the robber cannot enter nodes cops are leaving from). Only when the robber appears do we have reason to do otherwise, and in this case we simply assert that he loses if he appears adjacent to a cop.

Chapter 9

Summary and Open Questions

Specific complexity results achieved in this part are summarized in the following table:

Game/Puzzle	**Classical**	**Parameterized**
SHORT (N^2-1) PUZZLE	-	FPT
SHORT PLANAR BIPARTITE GENERALIZED GEOGRAPHY	-	FPT
SIMPLE SCOTLAND YARD (-R and -C)	PSPACE-complete	$W[2]$-hard

Table 9.1: Complexity results proven in Part II

9.1 Open Questions

While the algorithms we presented to solve Scotland Yard took many variant rules into account, most of them were unnecessary to show membership in PSPACE. In fact, only the termination rule was necessary, since it guarantees a game tree of polynomial height. The hardness reduction, on the other hand, requires either appearances by the robber or reporting of the edge sets he travelled across, and robber-only edges. Is it possible to modify our reduction (or provide an entirely new one) to eliminate the need for one or both of these elements? That is, is the problem still PSPACE-complete even if one or both of these elements are removed?

Also, while we showed SSY to be $W[2]$-hard, we did not show membership. What parameterized complexity class does SSY belong to?

Finally, as we mentioned before, SHORT PLANAR BIPARTITE GENERALIZED GEOGRAPHY is in FPT for parameter t. What is the parameterized complexity of the short variants of all the board games (Othello [IK94], Go [LS78] and Checkers [FGJ+78] with polynomial termination, etc.) that were shown to be PSPACE-complete by reduction from PLANAR BIPARTITE GENERALIZED GEOGRAPHY? Can we preserve the parameter and show that these games are in FPT too?

Bibliography

[ADF93] K. A. Abrahamson, Rodney G. Downey, and Michael R. Fellows. Fixed-parameter intractability II. In *Lecture Notes in Computer Science*, pages 374–385. Springer-Verlag, 1993. STACS 93.

[Alb73] R. D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:113–126, 1973.

[Arc99] Aaron F. Archer. A modern treatment of the 15 puzzle. *American Mathematical Monthly*, 106(9):793–799, 1999.

[Coo72] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proc. 3rd Ann. ACM Symposium on Theory of Computing*, pages 151–158. Association for Computing Machinery, 1972.

[Cul98] Joseph Culberson. Sokoban is PSPACE-complete. In *Proceedings of the International Conference on Fun with Algorithms*, pages 65–76, June 1998.

[Dem01] Erik D. Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In JiríSgall, Ale Pultr, and Petr Kolman, editors, *Proceedings of the 26th Symposium on Mathematical Foundations in Computer Science (MFCS 2001)*, number 2136 in LNCS, pages 18–32. Springer-Verlag, August 2001.

[DF92] Rodney G. Downey and Micheal R. Fellows. Fixed-parameter tractability and completeness. *Congressus Numerantium*, 87:161–187, 1992.

[DF95a] Rodney G. Downey and Michael R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. *Theoretical Computer Science*, 141:109–131, 1995.

[DF95b] Rodney G. Downey and Micheal R. Fellows. Fixed-parameter tractability and completeness I: Basic results. *SIAM Journal of Comput.*, 24:873–921, 1995.

[DF98] Rodney G. Downey and Michael R. Fellows. *Parameterized Complexity.* Springer, 1998.

[DFR97] Rodney G. Downey, Michael R. Fellows, and K. Regan. Parameterized circuit complexity and the W-heirarchy. *Theoretical Computer Science*, 189, 1997.

[FB01] Gary W. Flake and Eric B. Baum. Rush hour is PSPACE-complete, or why you should gener-

ously tip parking lot attendants. Manuscript, 2001. www.neci.nj.nec.com/homepages/flake/rushhour.ps.

[FG87] Aviezri S. Fraenkel and Elisheva Goldschmidt. PSPACE-hardness of some combinatorial games. *Journal of Combinatorial Theory*, 46:21–38, 1987.

[FGJ+78] Aviezri S. Fraenkel, Michael R. Garey, David S. Johnson, Thomas J. Shaefer, and Y. Yesha. The complexity of checkers on an $N \times N$ board - preliminary report. In *Proc. 19th Ann. Symp. of Foundationd of Computer Science*, pages 55–64. IEEE Computer Society, 1978.

[FHN+03] Henning Fernau, Torben Hagerup, Naomi Nishimura, Prabhakar Ragde, and Klaus Reinhardt. On the parameterized complexity of the generalized rush hour puzzle. In *Proc. 15th Canad. Conf. Comput. Geom.*, pages 6–9, 2003.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

[GR95] Arthur S. Goldstein and Edward M. Reingold. The complexity of pursuit on a graph. *Theoretical Computer Science*, 143:93–112, 1995.

[Has99] Johan Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182(1):105–142, 1999.

[HLH03] Jeffrey R. Hartline and Ran Libeskind-Hadas. The computational complexity of motion planning. *SIAM Review*, 45(3):543–557, 2003.

[IK94] Shigeki Iwata and Takumi Kasai. The othello game on an $n \times n$ board is PSPACE-complete. *Theoretical Computer Science*, 123:329–340, 1994.

[Joh85] David S. Johnson. The NP-completeness column: An ongoing guide. *Journal of Algorithms*, 6(2):291–305, 1985. Column 15: Uniqueness.

[Kar72] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[KR00] Subhash Khot and Venkatesh Raman. Parameterized complexity of finding subgraphs with hereditary properties. In *LNCS 1858*, pages 137–147. 2000. COCOON 2000.

[LP49] R. Duncan Luce and Albert D. Perry. A method of matrix analysis of group structure. *Psykometrika*, 14:95–116, 1949.

[LS78] David Lichtenstein and Michael Sipser. Go is Pspace hard. *SIAM Journal of Comput.*, pages 48–54, 1978.

[Luc50] R. Duncan Luce. Connectivity and generalized cliques in sociometric group structure. *Psychometrika*, 15:169–190, 1950.

[Mok79] R. Mokken. Cliques, clubs and clans. *Quality and Quantity*, 13:161–173, 1979.

[RW90] Daniel Ratner and Manfred Warmuth. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10:111–137, 1990.

[Sav70] J. W. Savitch. Relationship between deterministic and nondeterministic tape complexities. *Journal of Computer and System Sciences*, 4:177–192, 1970.

[Sch78] Thomas J. Schaefer. On the completity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16:185–225, 1978.

[SM73] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time. In *Proc. 5th Ann. ACM Symp. on Theory of Computing*, pages 1–9. Association for Computing Machinery, 1973.

[ST93] P. D. Seymour and Robin Thomas. Graph searching, and a min-max theorem for tree-width. *J. Combin. Theory*, 58:22–33, 1993.

[Sto79] W. E. Story. Note on the '15' puzzle. *American Mathematical Monthly*, 2:399–404, 1879.

[SvR04] Ulrike Stege and Iris van Rooij. Distance domination: A refined hierarchy and parameterized complexity results. Manuscript, 2004.

[SvRHH02] Ulrike Stege, Iris van Rooij, Alex Hertel, and Phillip Hertel. An $O(pn + 1.151^p)$-algorithm for p-profit cover and its practical implications for vertex cover. In Prosenjit Bose and Pat Martin, editors, *Proceedings of the 13th International Symposium on Algorithms and Computation (ISAAC 2002)*, number 2518 in LNCS, pages 249–261, 2002.

[vR03] Iris van Rooij. *Tractable Cognition: Complexity Theory in Cognitive Psychology.* PhD thesis, University of Victoria, Canada, 2003.

[WF94] S. Wasserman and K. Faust. *Social network analysis: methods and applications.* Cambridge: Cambridge University Press, 1994.

[Yan78] Mihalis Yannakakis. Node-and edge-deletion NP-complete problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing*, pages 253–264. Association for Computing Machinery, 1978.

Appendix A

Problem Definitions

BALANCED COMPLETE BIPARTITE SUBGRAPH

Input: Bipartite graph $G = (V_1 \cup V_2, E)$, $k \in \mathbb{Z}^+$.

Question: Does there exist a balanced complete bipartite subgraph of size k in G (i.e. are there two subsets $A \subseteq V_1$ and $B \subseteq V_2$ such that $|A| = |B| = k$ and $u \in A$, $v \in B$ implies that $(u, v) \in E$)?

References: NP-complete [Joh85].

BALANCED COMPLETE MULTIPARTITE SUBGRAPH

Input: $G = (V, E)$, where V is partitioned into $V_1, V_2, \ldots, V_\ell$ and there is no edge $e \in E$ such that $e \in V_i \times V_i$ for all $i : 1 \leq i \leq \ell$, $k \in \mathbb{Z}^+$

Question: Do there exist $V_1' \subseteq V_1$, $V_2' \subseteq V_2$, ..., $V_\ell' \subseteq V_\ell$ with $|V_1'| = |V_2'| = \cdots = |V_\ell'| = k$?

CLIQUE

Input: Graph $G = (V, E)$, $k \leq |V|$.

Question: Does G contain a clique of size k, i.e. a subset $V' \subseteq V$ with $|V'| \geq k$ such that every two vertices in V' are joined by an edge in E?

References: NP-complete [Kar72]. $W[1]$-complete for parameter k [DF95a].

EDGE-PROFIT CLIQUE

Input: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.

Question: Does G contain an *edge-profit clique* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{EPCl}(G, V') = |E_{IN}(G, V')| - |E_{OUT}(G, V')|$?

References: NP-complete [Corollary 3, p. 18].

FIXED-SIZE EDGE-PROFIT CLIQUE

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V|$, $p \in \mathbb{Z}^+$

Question: Does G contain a subset $V' \subseteq V, |V'| = k$ such that $p \leq \text{profit}_{\text{EPCl}}(G, V') = E_{\text{IN}}(G, V') - E_{\text{OUT}}(G, V')$?

References: NP-complete [Theorem 4, p. 24] and $W[1]$-hard for parameters k and p [Theorem 5, p. 24].

FIXED-SIZE EDGE-PROFIT INDEPENDENCE

Input: A graph $G = (V, E), k \in \mathbb{Z}^+ : k \leq |V|, p \in \mathbb{Z}^+$

Question: Does G contain a subset $V' \subseteq V, |V'| = k$ such that $p \leq \text{profit}_{\text{EPI}}(G, V') = E_{\text{OUT}}(G, V') - E_{\text{IN}}(G, V')$?

References: NP-complete [Theorem 4, p. 24] and $W[1]$-hard for parameters k and p [Theorem 5, p. 24].

INDEPENDENT SET

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V'|$.

Problem: Does G contain an *independent set* $V' \subseteq V$ such that $|V'| = k$ and there is no edge between any two vertices in V'?

References: NP-complete [GJ79] and $W[1]$-complete [DF95a].

PROFIT CLIQUE

Input: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.

Question: Does G contain a *profit clique* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{PCl}(G, V') = |V'| - |E_{OUT}(G, V')|$?

References: NP-complete [Corollary 1, p. 11] and $W[1]$-hard for parameter p [Corollary 1, p. 11].

PROFIT COVER

Input: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.

Question: Does G contain a *profit cover* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{PC}(G, V') = |E_{ADJ}(G, V')| - |V'|$?

References: NP-complete and FPT for parameter p [SvRHH02].

PROFIT INDEPENDENCE

Input: Graph $G = (V, E)$, $p \in \mathbb{Z}^+$.

Question: Does G contain a *profit independence* of profit p, i.e. a subset $V' \subseteq V$ such that $p \leq \text{profit}_{PI}(G, V') = |V'| - |E_{IN}(G, V')|$?

References: NP-complete and $W[1]$-hard for parameter p [vR03].

QBFSAT$_t$

INPUT: An integer r, a sequence s_1, ..., s_r of pairwise disjoint sets of boolean variables, a boolean formula X involving variables $s_1 \cup ... \cup s_r$

which consists of t alternating layers of conjunctions and disjunctions with negations applied only to variables, integer s $k_1, ..., k_r$.

QUESTION: Is it the case that there exists a size k_1 subset t_1 of s_1 such that for every size k_2 subset t_2 of s_2, there exists a size k_3 subset t_3 of s_3 such that ... (alternating quantifiers) such that, when the variables in $t_1 \cup ... \cup t_r$ are made true and all other variables are made false, formula X is true?

REFERENCE: $AW[t]$-complete for parameters r, k_1, ..., k_r. [DF98]

QUANTIFIED BOOLEAN FORMULA

Input: Set $U = u_1, u_2, \ldots, u_n$ of variables, well-formed quantified Boolean formula $F = (Q_1 u_1)(Q_2 u_2) \cdots (Q_n u_n)E$, where E is a Boolean expression and each Q_i is either $\forall$ or $\exists$.

Question: Is F true?

References: PSPACE-complete [SM73].

SATISFIABILITY

Input: A set U of variables, collection C of clauses over U.

Question: Is there a satisfying truth assignment for C?

Reference: NP-complete [Coo72].

SHORT $N^2 - 1$ PUZZLE

Input: A source permutation on $\{\emptyset, 1, 2, \ldots, n^2 - 1\}$, a target permutation on $\{\emptyset, 1, 2, \ldots, n^2 - 1\}$, t $\in \mathbb{Z}^+$.

Question: Imagine that the source permutation is laid out in a n by n grid. If a move consists of swapping $\emptyset$ with the contents of any adjacent grid cell,

can we transform the source permutation into the target permutation in $\leq t$ moves?

Reference: NP-complete [RW90], and in FPT for parameter t [Corollary 6, p. 35]. If we remove the input t and simply ask whether it is possible to transform the source into the target the problem can be solved in polynomial time [Sto79, Arc99].

SHORT PLANAR BIPARTITE GENERALIZED GEOGRAPHY

Input: A planar, directed, bipartite graph $G = (V, E)$ with maximum in and out degrees of 2 and a maximum total degree of 3, a specified vertex v_0, and $t \in \mathbb{Z}^+$.

Question: Does player 1 have a forced win within t turns in the following game played on G? Players alternate choosing a new edge from E. The first edge chosen must have its tail at V_0 and each subsequently chosen edge must have its tail at the vertex that was the head of the previous edge. The first player unable to choose such a new edge loses.

Reference: FPT for parameter t [Corollary 6, p. 35]. If we remove parameter t and ask simply whether first player has a forced win, the problem is PSPACE-complete [LS78].

SIMPLE SCOTLAND YARD

Input:

- A graph $G = (V, E_0 \cup E_1 \cup \ldots \cup E_C)$
- a set of possible starting positions for the robber $X \subseteq V$

- an actual starting position for the robber $x \in X$
- $D \subseteq V$ positions for cops to start
- the number of turns until the robber's next appearance a
- the number of turns between the robber's appearances b
- $t \in \mathbb{Z}^+$ the number of turns in the game

Rules:

- the robber begins at x, while cops are placed on every node in D.
- First the robber then the cops take alternate turns.
- the robber moves to an adjacent and unoccupied vertex on his turn, and all the cops move to adjacent vertices on Scotland Yard's.
- Every turn, it is revealed which set the edge the robber travelled along belongs to
- his position is only revealed to the cops after he moves a times and every b turns thereafter
- Only the robber may use the edges in E_0, but the rest of the edges can be used by both him and the cops.
- If a cop enters the same node as the robber, he captures him and Scotland Yard wins.

- If the robber is unable to move because all adjacent nodes are occupied by cops, he must surrender himself and again Scotland Yard wins.
- If Scotland Yard moves t times without capturing the robber, the game ends and the robber wins.

References: PSPACE-complete [Theorem 9, p. 47] and $W[2]$-hard for many parameters [Theorem 10, p. 55].

VERTEX COVER

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V|$

Question: Does G contain a subset $V' \subseteq V, |V'| \leq k$ such that every edge in E is adjacent to at least once vertex in V'?

References: NP-complete [Kar72] and FPT for parameter k [DF92].

VIABLE CLIQUE

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V|$

Question: Does G contain a subset $V' \subseteq V, |V'| \geq k$ such that $\forall v \in V'$, $\deg_{G(V')}(v) \geq \frac{k}{2}$

References: NP-complete [Corollary 4, p. 22] and $W[1]$-hard for parameter k [Corollary 5, p. 22].

VIABLE INDEPENDENCE

Input: A graph $G = (V, E)$, $k \in \mathbb{Z}^+ : k \leq |V|$

Question: Does G contain a subset $V' \subseteq V, |V'| \geq k$ such that $\forall v \in V'$, $\deg_{G(V')}(v) < \frac{k}{2}$

References: NP-complete [Corollary 4, p. 22] and $W[1]$-hard for parameter k [Corollary 5, p. 22].

WEIGHTED t-NORMALIZED SATISFIABILITY

Input: A t-normalized boolean expression X, a positive integer k.

Question: Does X have a satisfying truth assignment where exactly k of the variables occurring in X are set to true (also called a satisfying assignment of weight k)?

Note: A propositional formula is considered t-normalized it is in the form products-of-sums-of-products... of literals with t alternations.

Reference: $W[t]$-complete for parameter k. [cite]