5-2011

# Partition a 3-colorable graph into a small bipartite subgraph and a large independent set

Qing Wang
*University of Texas-Pan American*

# PARTITION A 3-COLORABLE GRAPH INTO A SMALL BIPARTITE SUBGRAPH AND A LARGE INDEPENDENT SET

A Thesis

by

QING WANG

Submitted to the Graduate School of the
University of Texas-Pan American
In partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Computer Science

# PARTITION A 3-COLORABLE GRAPH INTO A SMALL BIPARTITE SUBGRAPH AND A LARGE INDEPENDENT SET

A Thesis
by
QING WANG

COMMITTEE MEMBERS

Dr. Yang Liu
Chair of Committee

Dr. Zhixiang Chen
Committee Member

Dr. Bin Fu
Committee Member

Dr. Robert Schweller
Committee Member

May 2011

ABSTRACT

Wang, Qing, Partition A 3-Colorable Graph Into A Small Bipartite Subgraph And A Large Independent Set. Master of Science (MS), May, 2011, 35 pp., 4 tables, 8 figures, references, 15 titles.

Exact algorithms have made a little progress for the 3-coloring problem: improved from $1.4422^n$ to $1.3289^n$ since 1976. The best exact algorithm for the 3-coloring problem is by Beigel and Eppstein, and its analysis is very complicated. We study the parameterized 3-coloring problem: partitioning a 3-colorable graph into a bipartite subgraph and an independent set. Taking the size of the bipartite subgraph as the parameter k, we propose the first parameter algorithm of complexity $O(1.713^k n^{O(1)})$. Our algorithm can solve the 3-coloring problem faster than the best exact algorithm for graphs with $k \leq 0.527n$ where n is the graph size. Our study of the parameterized 3-coloring problem brings new insight on studies of the 3-coloring problem. Experiments show that the parameterized algorithm is faster than the exact algorithm for graphs of small parameter k. Moreover, the running time of parameterized algorithm is not much related to edge density, while the running time of exact algorithm increases dramatically as edge density increases.

# DEDICATION

The completion of my graduate studies would not have been possible without the love and support of my family. My mother, Hongping Du, my father, Jianmin Wang, my grandmother, Fengying Wang, my grandfather, Kejian Wang, wholeheartedly inspired, motivated and supported me by all means to accomplish this degree. Thank you for your love and patience.

ACKNOWLEDGMENTS

I will always be grateful to Dr. Yang Liu, chair of my thesis committee, for all his mentoring and advice. From research design, and data processing, to manuscript editing, he encouraged me to complete this process through his infinite patience and guidance. My thanks go to my thesis committee members: Dr. Zhixiang Chen, Dr. Bin Fu and Dr. Robert Schweller. Their advice, input, and comments on my thesis helped to ensure the quality of my intellectual work.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

# CHAPTER I

# INTRODUCTION

In graph theory, graph coloring is a special case of graph labeling, it is an assignment of labels traditionally called "colors" to elements of a graph subject to certain constraints. In its simplest form, it is a way of coloring the vertices of a graph such that no two adjacent vertices share the same color; this is called a vertex coloring. Similarly, an edge coloring assigns a color to each edge so that no two adjacent edges share the same color, and a face coloring of a planar graph assigns a color to each face or region so that no two faces that share a boundary have the same color.

## Problem Description

Graph coloring has been studied as an algorithmic problem since the early 1970s, the chromatic number problem is one of Karp's 21 NP-complete problems from 1972, and at approximately the same time various exponential-time algorithms were developed based on backtracking and on the deletion-contraction recurrence of Zykov. One of the major applications of graph coloring, register allocation in compilers was introduced in 1981[6].

Given an undirected graph $G = (V, E)$, coloring each vertex $v \in V$ with one of three colors so that no two vertices connected by an edge $e \in E$ are colored with the same color is known as

the Graph 3-Coloring Problem. Several variations exist, like finding the least number of colors that is needed to color the graph, or finding the largest subgraph in $G$ that can be colored with the given number of colors. All of these problems are known to be NP-complete, so it is unlikely that a polynomial time algorithm exists that solves any of these problems.

## Previous work

The best algorithm for the problem is of time complexity $O(2^n n^{O(1)})$, and requires exponential space $O(2^n n)$ [3]. When polynomial space complexity is desired, the best algorithm for this problem has time complexity $O(2.2461^n)$ [3]. The 3-coloring problem is a special case of the chromatic number problem. In the 3-coloring problem, we are asked to determine whether the chromatic number of graphs is 3 or not. Table 1 shows the history of exact algorithms for the 3-coloring problem. Meanwhile, approximation solutions with $O(n^{3/14})$ colors can be found in polynomial time for graphs of chromatic number 3 [4]. For 3-coloring, we know of several relevant references. Lawler is primarily concerned with the general chromatic number [9], but he also gives the following very simple algorithm for 3-coloring: for each maximal independent set, test whether the complement is bipartite. The maximal independent sets can be listed with polynomial delay, and there are at most $3^{n/3}$such sets, so this algorithm takes time $O(1.4422^n)$. Schiermeyer gives a complicated algorithm for solving 3-colorability in time $O(1.415^n)$ [15], based on the following idea: if there is one vertex $v$ of degree $n-1$ then the graph is 3-colorable iff $G-v$ is bipartite, and the problem is easily solved. Otherwise, Schiermeyer performs certain reductions involving maximal independent sets that attempt to increase the degree of $G$ while partitioning the problem into subproblems, at least one of which will remain solvable. Beigel and Eppstein gives a faster algorithms are known for 3-colorability and 4-

colorability, which can be decided in time $O(1.3289^n)$ [13], they consider worst case time bounds for several NP-complete problems, based on a constraint satisfaction (CSP) formulation of these problems: $(a, b)$ -CSP instances consist of a set of variables, each with up to a possible values, and constraints disallowing certain b-tuples of variable values; a problem is solved by assigning values to all variables satisfying all constraints, or by showing that no such assignment exist. 3-SAT is equivalent to $(2,3)$-CSP while 3-coloring and various related problems are special cases of $(3,2)$-CSP; there is also a natural duality transformation from $(a, b)$-CSP to $(b, a)$-CSP.

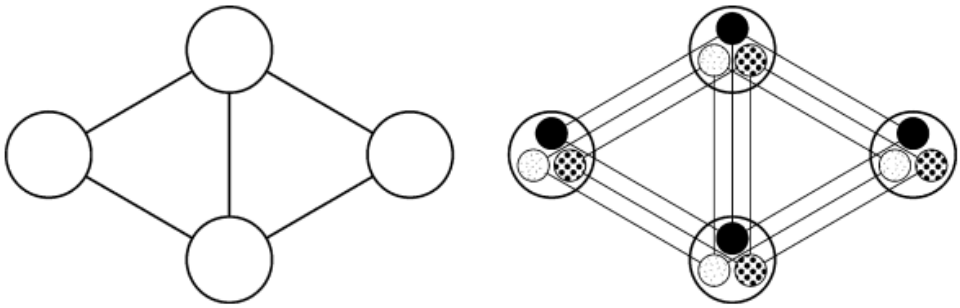

Figure 1: Example 3-coloring instance and translation into a (3, 2)-CSP instance.

| Authors | Complexity | Year |
|---|---|---|
| Lawler | $1.4422^n$ | 1976 [11] |
| Schiermeyer | $1.415^n$ | 1994 [15] |
| Beigel and Eppstein | $1.3446^n$ | 1995 [1] |
| Beigel and Eppstein | $1.3289^n$ | 2005 [2] |

Table 1: History of Exact Algorithms for 3-coloring

# CHAPTER II

# FIXED-PARAMETER ALGORITHM IN 3 COLORING PROBLEM

Classic complexity theory indicates that a large number of natural combinatorial problems are inherently hard to solve algorithmically.

The 3-coloring problem can be viewed from another perspective: partitioning the vertices of a graph into three disjoint independent sets, if such a partition exists. This perspective leads to our study of parameterized complexity of the 3-coloring problem when we take the total number of vertices in two independent sets as a parameter.

Parameterized 3-coloring is that given a graph $G$, can the vertices of $G$ be partitioned into a bipartite subgraph of at most $k$ vertices and an independent set? Find such a partition if it exists, or report 'NO' otherwise.

## Fixed-parameter tractable algorithm

Parameterized complexity and algorithms have developed rapidly during the last three decades. Since the fundamental work of Downey and Fellows, parameterized complexity theory introduced numerous innovative ideas in algorithmic design and offered insightful results in almost all disciplines of theoretical computer science.

According to the common belief that P ≠ NP, NP-complete, or otherwise NP-hard, problems require time that is exponential in input size. Therefore, if the input size is large, it is unfeasible

to find solutions to those problems. In real world, applications of NP-complete may have some small parameters which can be used to find solutions efficiently. Some problems with certain parameter fixed can be solved by algorithms that are exponential only in the size of the fixed parameter tractable algorithm. A parameterized problem that allows for such a fixed-parameter tractable algorithm is said to be a fixed-parameter tractable problem and belongs to the class FPT. It seems a good supplement of the theory of NP-completeness. The problem in FPT can be solved efficiently for small values of the fixed parameters. For example, the vertex cover problem is in FPT. This problem is that given a graph $G$, to find k number of vertices in $G$ such that every edge of $G$ is incident to at least one of those vertices. It is a NP-complete problem which has been applied in many areas such as network optimization and bioinformatics.

An exhaustive search algorithm can solve the problem in time $2^{O(k)}n^{O(1)}$. Vertex cover is therefore a fixed-parameter tractable problem, and there may only need a vertex cover of a few vertices in some applications. These parameters can be used to define parameterized problems, the case in the $k$-vertex cover problem, where the input consists of a graph $G$ and a positive integer k as a parameter, and asks whether G has a vertex cover with at most $k$ vertices. The input to a parameterized problem $L$ is defined as a pair $(n, k)$, where n is the size of the input and $k$ is the parameter. Often, parameterized algorithms find solutions to problem instances in polynomial time in terms of the size of the input. The problem $L$ is said to be fixed-parameter tractable (FPT) if there exists an algorithm that correctly decides whether an input $(n, k)$ is a yes-instance, or not, in time $f(k)n^{\alpha}$(or $f(k) + n^{\alpha}$), where α is a constant, and $f$ is an arbitrary function independent of n. If a parameterized problem is fixed parameter tractable, it is said to be in the class FPT. For those applications of small vertex cover (i.e. k is small), we can solve the problem efficiently. After many researches, many fixed parameterized algorithm for this problem

have been developed. A well known algorithm for this problem has a running time $O(1.286^k + kn)$ in [9].

However, some problems are not believed to be in FPT. An example is deciding whether an n-vertex graph contains an independent set of cardinality k or not. The complement of a maximum independent set is the set of vertices not belonging to the independent set, forms a minimum vertex cover, which is a fixed-parameter tractable problem. There is an algorithm which can solve the independent set of cardinality k with an upper bound of $O(n^{0.792k})$ in [13]. So far no algorithm with a running time of the form $f(k)n^{O(1)}$ is known.

Unlike classical complexity theory, which focuses on whether a problem is hard or not, parameterized complexity theory, introduced by Downey and Fellows, accepts that a problem is hard and asks the question "What makes the problem computationally difficult?". Downey and Fellows claim parameters arise naturally in many computational problems.

Fixed-parameter tractable algorithms (FPT-algorithms) are helpful in solving real world problems that are in general NP-Hard, but where most instances of interest have small parameter values. This is the case for many practical problems such as multiple sequence alignment in computational biochemistry, known to be equivalent to the vertex cover problem, which has an FPT-algorithm with running time O($k$n + $1.271^k k^2$).

Next we introduce two important concepts: P-coloring and Bipartite-independent partition, which will be used in discussions of our algorithm.

## P-Coloring

Let $G = (V, E)$ be a simple graph without multiple edges between a pair of vertices. A $p\text{-}coloring$ of graph $G$ is an coloring of vertices with $p$ colors such that each vertex is colored

with exactly one color, and no two adjacent vertices are colored with the same color. A graph $G$ is $p\text{-}colorable$ if $G$ there is a $p\text{-}coloring$ of $G$. Let $V_1$ and $V_2$ be vertex subsets in graph $G$ such that $V_1 \subseteq V_2$, $(C_1, C_2)$ be a $2\text{-}coloring$ of $V_1$, and $(C_1', C_2')$ be a $2\text{-}coloring$ of $V_2$. Then $(C_1', C_2')$ is $compatible$ to $(C_1, C_2)$ if $C_1 \subseteq C_1'$ and $C_2 \subseteq C_2'$.

**Bipartite – independent partition**

Let $S_B$ and $S_I$ be two disjoint vertex subsets in graph $G$. A partition $[S_B, S_I]$ is a $bipartite\text{-}independent\ partition$ if the induced subgraph by $S_B$ is $2\text{-}colorable$ and the induced subgraph by $S_I$ is an independent set. If $V = S_B + S_I$, then a bipartite-independent partition $[S_B, S_I = V - S_B]$ is a complete bipartite-independent partition. A bipartite-independent partition $[S'_B, S'_I]$ extends another bipartite-independent partition $[S_B, S_I]$ if $S_B \subseteq S'_B$, $S_I \subseteq S'_I$. Given a bipartite-independent partition $[S_B, S_I]$, $[S'_B, S'_I]$ k-extends $[S_B, S_I]$ if $|S'_B| \leq |S_B|$ + k.

**Candidate Set**

Let $P = x_1 \cdots x_p$ be a simple path induced by vertices $x_1, \cdots, x_p$, i.e., $x_i = x_j$ for $1 \leq i < j \leq p$. The candidate set of P is $\{x_2, x_4, \cdots, x_p\}$ if p is even, or $\{x_2, x_4, \cdots, x_{p-1}\}$ if p is odd. Let $C = x_1 \cdots x_p x_1$ be a simple cycle induced by vertices $x_1, \cdots, x_p$, i.e., $x_i = x_j$ for $1 \leq i < j \leq p$. The candidate set of cycle C is $\{x_2, x_4, \cdots, x_p\}$ if p is even, or $\{x_2, x_4, \cdots, x_{p-1}\} \cup \{x_p\}$ if p is odd.

**Upper bound analysis**

Let $f(k)$ be the maximum number of leaves of the search tree if the parameter is at most k(let $f(k) = 1$ for $k \leq 0$).

$$f(k) = f(k-1) + f(k-3)$$

There is a standard technique for bounding such functions asymptotically. We assume that $x_0$ is a solution of this equation. We prove by induction:

$$x_0^k \leq x_0^{k-1} + x_0^{k-3}$$

$$x_0^3 - x_0^2 - 1 \leq 0$$

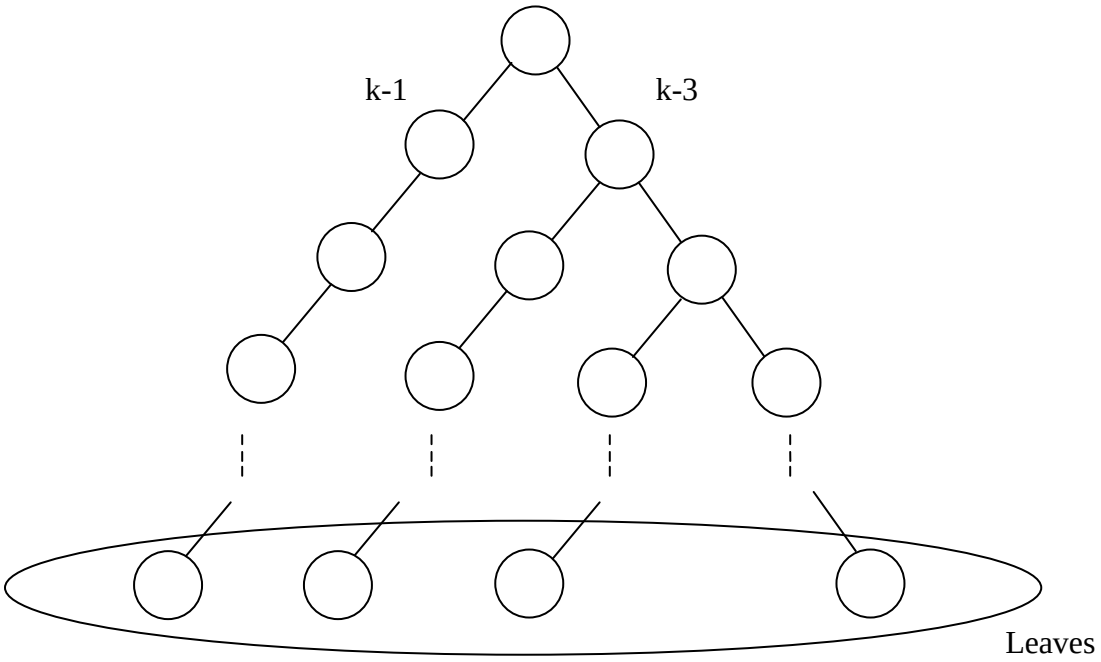


Figure 2: (1,3) Search Tree

We need to find the roots of the characteristic equation $x_0^3 - x_0^2 - 1 = 0$. $x_0$ = 1.4656 is the solution. Now we check if the $x_0$ is the best solution for this equation:

$$f(k) = x^{-1} + x^{-3} - 1$$

$$f'(x) = -x^{-2} + (-3)x^{-4}$$

When the $x$ increase, the $f(k)$ decrease.

We try $x_1 \geq x_0$, see if $x_1$ it's good solution or not:

$$x_1^{k-1} + x_1^{k-3} \leq x_1^k$$

Because $x_1 \geq x_0$

$$x_1^{k-1} + x_1^{k-3} \leq x_1^k \leq 0$$

So, $x_1^k$ is an upper bound of the equation, but it is not as tight as possible.

We try $x_2 < x_0$, see if $x_2$ it's good solution or not:

$$x_2^{k-1} + x_2^{k-3} \leq x_2^k$$

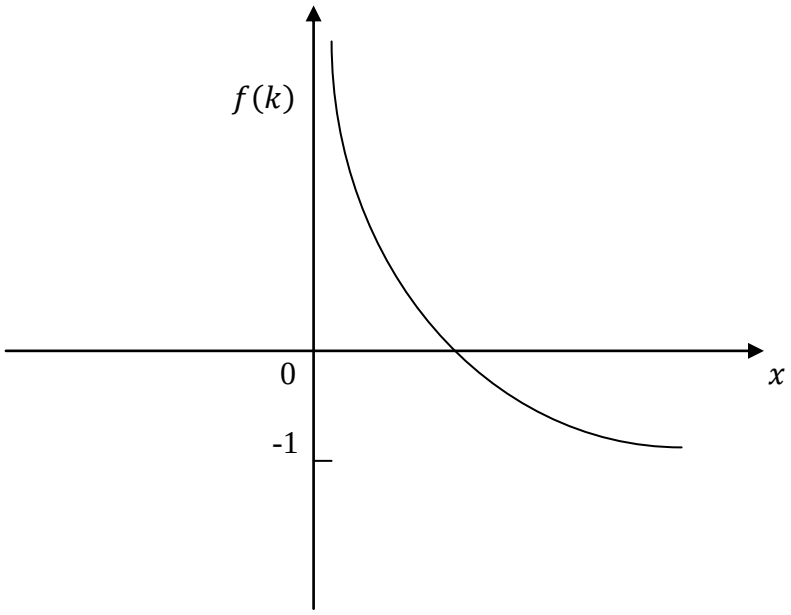


Figure 3: Coordinate graphs of $f(k) = x^{-1} + x^{-3} - 1$

Because $x_2 < x_0$

$$x_2^{k-1} + x_2^{k-3} \leq x_2^k > 0$$

So, at this point, we don't know $x_2^k$ is an upper bound of the equation or not.

## Union of Disjoint Paths/Cycles

Let $[S_B , S_I]$ be a bipartite-independent partition of graph G. This section will show two properties for the case when subgraph induced by $V - S_B - S_I$ is a union of disjoint paths and cycles.

Let $CS_1, \cdots, CS_q$ be the candidate sets of those paths/cycles. In this subsection, we assume that any vertex $x \in V - S_B - S_I$ has no neighbors in $S_I$. The first property is that at least $\sum_{i=1}^{q}|CS_i|$ vertices from the disjoint paths/cycles should be put into any complete bipartite-independent partition which k-extends $[S_B , S_I]$.

**Lemma 1.** *Let* $[S'_B , V - S'_B]$ *be a complete bipartite-independent partition. If* $[S'_B , V - S'_B]$ *k-extends* $[S_B , S_I]$*, then* $V - S'_B$ *contains at least* $|CS_i|$ *vertices from the paths/cycles corresponding to candidate set* $CS_i$ *, i.e.,* $k \geq \sum_{i=1}^{q}|CS_i|$.

*Proof.* We number the paths/cycles corresponding to the candidate set $CS_i$ by the index $i$ of the candidate set. We first prove that for each path/cycle $i$, $S'_B$ must contain at least $|CS_i|$ vertices from path/cycle $i$. There are two cases: $CS_i$ is from either a path or a cycle.

If $CS_i$ is from a path $i$, let path $i$ be $x_1x_2 \cdots x_p$ which is induced by vertices $x_1 , x_2 , \cdots, x_p$ . By definition of candidate sets, $|CS_i| = \left|\{x_2 , x_4 , \cdots , x_p \}\right| = \frac{p}{2}$ if p is even, $|CS_i| = \left|\{x_2 , x_4 , \cdots , x_{p-1}\}\right| = \left\lfloor\frac{p}{2}\right\rfloor$ if p is odd. For both cases, $|CS_i| = \left\lfloor\frac{p}{2}\right\rfloor$ . Moreover, there are at least $\left\lfloor\frac{p}{2}\right\rfloor$ disjoint edges in path $i$: $x_1x_2, x_3x_4, \cdots, x_{p-1}x_p$ . Then each of those disjoint edges can have at most one vertex in $-S'_B$ , since $[S'_B , V - S'_B]$ is a complete bipartite-independent partition and $V - S'_B$ must be an independent set by the definition of bipartite-independent partition. Therefore, $S'_B$ contains at least one vertex from each of those $\left\lfloor\frac{p}{2}\right\rfloor$ disjoint edges. It follows that $S'_B$ contains at least $\left\lfloor\frac{p}{2}\right\rfloor = |CS_i|$ vertices from path.

If $CS_i$ is from a cycle $i$, let cycle $i$ be $x_1x_2 \cdots x_px_1$ which is induced by vertices $x_1 , x_2 , \cdots, x_p$ . When p is even, we have that $|CS_i| = \left|\{x_2 , x_4 , \cdots , x_p \}\right| = \frac{p}{2}$ , In the cycle $i$ there are $\frac{p}{2}$ disjoint edges: $x_1x_2, x_3x_4, \cdots, x_{p-1}x_p$ . By the same arguments above for path $i$, $S'_B$ contains at

least one vertex from each of those disjoint edges, and thus contains at least $\frac{p}{2} = |CS_i|$ vertices from cycle $i$. Next we show that when p is odd, $S'_B$ also contain at least $|CS_i|$ vertices from cycle $i$.

When p is odd, by definition we have that $|CS_i| = \left|\{x_2, x_4, \cdots, x_{p-1}\} \cup \{x_p\}\right| = \frac{p+1}{2}$ for cycle $i$. In this cycle of $p$ vertices ($p$ is odd), there are $\frac{p-1}{2}$ disjoint edges: $x_1x_2, \cdots, x_{p-2}x_{p-1}$. By the same arguments for path $i$, $S'_B$ contains at least one vertex from each of those $\frac{p-1}{2}$ disjoint edges. If $S'_B$ contains both vertices of one of those $\frac{p-1}{2}$ disjoint edges, $S'_B$ contains at least $\frac{p-1}{2} + 1 = |CS_i|$ vertices from cycle $i$. Otherwise, $S'_B$ contains exactly one vertex from each of those $\frac{p-1}{2}$ disjoint edges, and then $V - S'_B$ contains exactly one vertex from each of those disjoint edges. We have two cases:

*Case 1*: $S'_B$ contains $x_1$. Since both $S'_B$ and $V - S'_B$ contains exactly one vertex from each of those $\frac{p-1}{2}$ disjoint edges, $V - S'_B$ must contains $x_2$. It follows that $S'_B$ must contain $x_3$ since $V - S'_B$ is an independent set, and then $V - S'_B$ must contains $x_4$. Repeat this, we will have that $S'_B$ contains $x_1, x_3, \cdots, x_{p-2}$ and $V - S'_B$ contains $x_2, x_4, \cdots, x_{p-1}$. Since (1) $x_{p-1}$ is in $V - S'_B$, (2) $V - S'_B$ is an independent set, and (3) there is an edge $x_{p-1}x_p$ in the cycle, $S'_B$ must contains $x_p$. Therefore, $S'_B$ contains $\frac{p-1}{2} + 1 = \frac{p+1}{2} = |CS_i|$ vertices: $\{x_1, x_3, \cdots, x_{p-2}\} \cup \{x_{p1}\}$.

*Case 2*: $S'_B$ does not contains $x_1$. Then $V - S'_B$ contains $x_1$. Since $V - S'_B$ is an independent set and there is an edge $x_1x_p$ in cycle $i$, $S'_B$ must contains $x_p$. Besides $x_p$, $S'_B$ contains $\frac{p-1}{2}$ vertices from those $\frac{p-1}{2}$ disjoint edges ($x_1x_2, \cdots, x_{p-2}x_{p-1}$), by our assumption that

$S'_B$ contains exactly one vertex from each of those disjoint edge. Therefore, $S'_B$ contains at least $\frac{p-1}{2}+1 = \frac{p+1}{2} = |CS_i|$ vertices from cycle *i*.

We have shown that $S'_B$ contains at least $|CS_i|$ vertices from path/cycle *i*. It follows that $S'_B$ contains at least $\sum_{i=1}^{q}|CS_i|$ vertices from $V - S_B - S_I$, since $G[V - S_B - S_I]$ is a union of disjoint paths/cycles. Note that $S_B$ and $V - S_B - S_I$ are disjoint and $CS_i$ are from $V - S_B - S_I$. It follows that $|S'_B| \geq |S_B| + \sum_{i=1}^{q}|CS_i|$ . Moreover, because $[S'_B, V - S'_B]$ k-extends $[S_B, S_I]$, we have that $|S'_B| \leq |S_B| + k$ . Therefore, $|S_B| + k \geq |S'_B| \geq |S_B| + \sum_{i=1}^{q}|CS_i|$ , i.e., $k \geq \sum_{i=1}^{q}|CS_i|$ . This completes our proof.

The second property is that when a complete 2-coloring $(C_1, C_2)$ of $S_B$ is given, we can determine in polynomial time whether there is a complete bipartite-independent partition $[S'_B, V - S'_B]$ which k-extends $[S_B, S_I]$ and $S'_B$ has a complete 2-coloring compatible to $(C_1, C_2)$. If such a complete bipartite-independent partition exists, we construct a complete 2-coloring $(C'_1, C'_2)$ of $S'_B$ compatible to $(C_1, C_2)$.

**Lemma 2.** *Let* $(C_1, C_2)$ *be a complete 2-coloring of* $S_B$. *We can find a complete bipartite-independent partition* $[S'_B, V - S'_B]$ *k-extending* $[S_B, S_I]$ *such that* $S'_B$ *has a complete 2-coloring compatible to C if there exists such one, or report 'NO' otherwise. This can be done in polynomial time.*

*Proof.* Figure 4 gives the algorithm to find the desired bipartite-independent partition if such one exists, or return 'NO' otherwise. It is obvious that the algorithm can terminate in polynomial time, since each step takes polynomial time.

**Algorithm-1**($G$, $k$, [$S_B$ , $S_I$], ($C_1$ , $C_2$ ))
INPUT: a graph $G$, a parameter $k$, a bipartite-independent partition [$S_B$ , $S_I$ ], and a complete 2-coloring ($C_1$ , $C_2$) of $S_B$ .
OUTPUT: either a complete bipartite-independent partition [$S'_B$ , $V - S'_B$ ] which k-extends [$S_B$ , $S_I$ ] such that $S'_B$ has a complete 2-coloring compatible to ($C_1$ , $C_2$ ), or 'NO' otherwise.

1. **if** there is an edge $xy$ where $x, y \in V - S_B - S_I$ such that both x and y have a neighbor in $C_1$ and another neighbor in $C_2$
    return 'NO';
2. **foreach** $x \in G - S_B - S_I$ which has neighbors in both $C_1$ and $C_2$ , let $Y \subseteq V - S_B - S_I$ be neighbors of $x$, $Y_1 \subseteq Y$ be vertices which have neighbors in $C_1$ , $Y_2 \subseteq Y$ be vertices which have neighbors in $C_2$
    put $x$ into $S_I$ and $Y$ into $S_B$ , $Y_1$ into $C_2$ , $Y_2$ into $C_1$ , and $k = k - |Y|$;
3. **if** $k < \sum_{path/cycle\ i} |CS_i|$
    return 'NO';
4. **if** there is a cycle $i$ of odd number vertices such that for each edge $xy$ in the cycle, both $x$ and $y$ have neighbors in $C_1$ ($C_2$ )
    returns 'NO';
5. **foreach** path/cycle $i$
    for a cycle of $p$ vertices where $p$ is odd, W.L.O.G, assume that $x_{p-1}x_p$ is an edge where $x_{p-1}$ and $x_p$ have no neighbors in the same color set (either $C_1$ or $C_2$ );
    put the candidate set $CS_i$ for the path/cycle $i$ into $S_B$ and other vertices $W_i$ of path/cycle $i$ into $S_I$ ;
return [$S_B$ , $S_I$ ];

Figure 4: Algorithm 1

When a vertex $w$ has a neighbor in $C_1$ and a neighbor in $C_2$ , then $w$ must be in $V - S'_B$ , since $S_B + w$ has no 2-colorings compatible to ($C_1$ , $C_2$ ). Then for Step 1, both $x$ and $y$ must be in $V - S'_B$, which is also impossible since $V - S'_B$ should be an independent set. Therefore, 'NO' is returned correctly at Step 1.

In Step 2, vertex $x$ should be in $V - S'_B$ since $x$ has neighbors in both $C_1$ and $C_2$ . Then neighbors $Y$ of $x$ should be in $S'_B$ for any complete bipartite-independent partition [$S'_B$ , $V - S'_B$] which k-extends [$S_B$ , $S_I$ ]. Thus it is safe to put x into $S_I$ and $Y$ into $S_B$ . Moreover, in any complete 2-coloring ($C'_1$, $C'_2$ ) of $S'_B$ compatible to ($C_1$ , $C_2$ ), $Y_1 \subseteq C_2$ since vertices of $Y_1$ have

neighbors in $C_1$ . Similarly, $Y_2 \subseteq C_1$ . Finally, we need to reduce $k$ to $k - |Y|$, since now we need to find a complete bipartite-independent partitions $(k - |Y|)$-extending $[S_B , S_I]$ after Step 2. This concludes that Step 2 is correct. By our assumption that $G[V - S_B - S_I]$ is a union of paths/cycles. When $k < \sum_{path/cycle\ i} |CS_i|$ , there are no complete bipartite-independent partitions k-extending $[S_B , S_I]$ by lemma 1. Therefore, Step 3 returns 'NO' correctly.

For a cycle $i$ of $p$ vertices where $p$ is odd, $|CS_i| = \frac{P+1}{2}$. Let $W$ be those vertices of cycle $i$ which are also in $S'_B$ where $[S'_B , V - S'_B]$ is a complete bipartite-independent partition k-extending $[S_B , S_I]$. Then by lemma 1, $|W| \geq |CS_i| = \frac{P+1}{2}$. This implies that at least two vertices *x* and *y* of *W* should be an edge *xy* of cycle *i*. However, when the condition of Step 4 is true, both *x* and *y* have neighbors in the same color set: either in $C_1$ or $C_2$ . It contradicts that $W + S_B \subseteq S'_B$ is 2-colorable. Therefore, when the condition of Step 4 is true, no complete bipartite-independent partitions k-extending $[S_B , S_I]$ exists, and thus 'NO' is returned correctly.

To simplify discussions on the correctness of Step 5, let $[S'_B , S'_I]$ denotes the $[S_B , S_I]$ returned at Step 5, and $[S_B , S_I]$ refers to the partition after Step 4. Then $S'_B = S_B + \sum_{path/cycle\ i} CS_i$ and $S'_I = S_I + \sum_{path/cycle\ i} W_i = V - S'_B$ .

By definition of $CS_i$ , $W_i$ in Step 5 is an independent set. Recall our assumption on $V - S_B - S_I$ as input: any vertex in $V - S_B - S_I$ has no neighbor in $S_I$ (the assumption is made right before this subsection). Note that this assumption is still valid before Step 5. So $S'_I = S_I + \sum_{path/cycle\ i} W_i$ is still an independent set, since $W_i \subseteq V - S_B - S_I$ . Moreover, $k \geq \sum_{path/cycle\ i} |CS_i|$ after Step 3. It follows that $|S'_B| = |S_B| + \sum_{path/cycle\ i} |CS_i| \leq |S_B| + k$ . Therefore, we can conclude that $[S'_B , S'_I = V - S'_B]$ is a complete bipartite-independent partition k-extending $[S_B , S_I]$ such that $S'_B$ has a complete 2-coloring compatible to $(C_1 , C_2)$

and thus Step 5 is correct, once we show that $S'_B$ has a complete 2-coloring ($C'_1$, $C'_2$ ) compatible to ($C_1$ , $C_2$ ).

To prove that, we first prove the following claim:

*Claim*: any vertex $y \in S_B$ such that *y* is not in $C_1 + C_2$ , y has at most one neighbor in $V - S_I$ .

Note that before Step 2 all vertices of $S_B$ are in $C_1 + C_2$ , since ($C_1$ , $C_2$) is a complete 2-coloring of $S_B$ before Step 2. So $y$ must be put into $S_B$ during Step 2, which implies that a neighbor $x \in V - S_B - S_I$ of $y$ is put into $S_I$ during Step 2. Recall that $G[V - S_B - S_I]$ is a union of disjoint paths/cycles before Step 2. It follows that y can have at most two neighbors in $V - S_B - S_I$ before Step 2. Moreover, $y$ has no neighbors in $S_B$ before Step 2. Otherwise, $y$ should be in either $C_1$ or $C_2$ after Step 2. Since (1) $y$ has no neighbors in $S_B$ before Step 2, (2) $y$ have at most two neighbors in $V - S_B - S_I$ before Step 2, and (3) one neighbor $x \in V - S_B - S_I$ of $y$ is put into $S_I$ during Step 2, it follows that y has at most one neighbor in $V - S_I$ after Step 2, which conclude the proof of the Claim.

Now we continue our proof of that $S'_B$ has a complete 2-coloring ($C'_1$, $C'_2$ ) compatible to ($C_1$ , $C_2$ ). Note that $S'_B = S_B + \sum_{path/cycle\ i} CS_i$ according to Step 5. Moreover, $S_B$ may contains vertices other than those in ($C_1 + C_2$) after Step 2. Let $Y_i$ be those vertices in $S_B - (C_1 + C_2)$ which have neighbors in $CS_i$ , and $Z$ be those vertices in $S_B - (C_1 + C_2) - \sum_{path/cycle\ i} Y_i$ in Step 5. It is obvious that any vertex $x \in \sum_{path/cycle\ i} Y_i + Z$ has no neighbors in $C_1 + C_2$ . Otherwise, $x$ can be put into $C_1 + C_2$ . By our definitions, it is also obvious that $S_B = (C_1 + C_2) + \sum_{path/cycle\ i} Y_i + Z$, and thus $S'_B = (C_1 + C_2) + \sum_{path/cycle\ i} Y_i + \sum_{path/cycle\ i} CS_i$ + Z.

To show that $S'_B = (C_1 + C_2) + \sum_{path/cycle\ i} Y_i + \sum_{path/cycle\ i} CS_i +$ Z has a complete 2-coloring compatible to ($C_1$ , $C_2$ ), we first show that $(C_1 + C_2) + \sum_{path/cycle\ i} CS_i$ has a 2-coloring ($C_1^1$, $C_2^1$ ) compatible to ($C_1$ , $C_2$ ). Initially $C_1^1 = C_1$ and $C_2^1 = C_2$ . Note that after Step 2,

any vertex $x \in V - S_B - S_I$ has at neighbors in at most one of $C_1$ and $C_2$ . Otherwise, it should be processed in Step 2. Let *x* be a vertex of $CS_i$, then *x* has neighbours in at most one of $C_1^1 = C_1$ and $C_2^1 = C_2$, since $CS_i$ is a subset of $V - S_B - S_I$ . We put *x* into $C_1^1$ if it has a neighbor in $C_2$ , or into $C_2^1$ otherwise. Note that path/cycle $i$ $CS_i$ is an independent set, and then it is safe to put *x* into $C_2^1$ when *x* has no neighbors in $C_1$ . Therefore, these operations find a 2-coloring ($C_1^1$, $C_2^1$ ) of ($C_1$ + $C_2$) + $\sum_{path/cycle\ i}\ CS_i$ which is compatible to ($C_1$ , $C_2$ ).

Next We show that ($C_1$ + $C_2$) + $\sum_{path/cycle\ i}\ CS_i$ + $\sum_{path/cycle\ i}\ Y_i$ has a complete 2-coloring ($C_1^2$, $C_2^2$ ) compatible to ($C_1^1$, $C_2^1$) of vertices $C_1^1$ + $C_2^1$ = ($C_1$ + $C_2$) + $\sum_{path/cycle\ i}\ CS_i$ . Initially $C_1^2$ = $C_1^1$ and $C_2^2$ = $C_2^1$ . Recall that all vertices in $Y_i \subseteq S_B$ have no neighbors in $C_1$ + $C_2$ . Then by our Claim, each vertex *y* ∈ $Y_i$ has exactly one neighbor in $CS_i$ , and then no neighbors in $C_1$ + $C_2$ before Step 5. Since each vertex of $CS_i$ is in $C_1^1$ or $C_2^1$ by our processing above, vertex *y* ∈ $Y_i$ can be put into $C_1^2$ ($C_2^2$) if its unique neighbor in $CS_i$ is in $C_2^1$ ($C_1^1$ ). Therefore, all vertices in ( $C_1$ + $C_2$ ) + $\sum_{path/cycle\ i}\ CS_i$ + $\sum_{path/cycle\ i}\ Y_i$ has a complete 2-coloring which is compatible to the ($C_1^1$, $C_2^1$).

Finally, we show that $S_B'$ = ($C_1$ + $C_2$) + $\sum_{path/cycle\ i}\ CS_i$ + $\sum_{path/cycle\ i}\ Y_i$ + $Z$ has a complete 2-coloring ( $C_1^3$ , $C_2^3$ ) compatible to ( $C_1^2$ , $C_2^2$ ) of those vertices in ( $C_1$ + $C_2$ ) + $\sum_{path/cycle\ i}\ CS_i$ + $\sum_{path/cycle\ i}\ Y_i$ = $S_B' - Z$. Initially $C_1^3$ = $C_1^2$ and $C_2^3$ = $C_2^2$ . Recall that vertices of $Z$ have no neighbors in ($C_1$ + $C_2$) + $\sum_{path/cycle\ i}\ CS_i$ by definition, and have no neighbors in $\sum_{path/cycle\ i}\ Y_i$ since by definition $Y_i$ has only neighbors in $CS_i$ . It follows that $Z$ has no neighbors in $S_B' - Z$. That is, any vertex $z \in Z$ can have neighbors only in $Z$ or $V - S_B' = S_I'$. Recall again that $Z$ has no neighbors in $C_1$ + $C_2$. Then by our Claim, any vertex $z \in Z$ has at most one neighbor in $V - S_I$ , thus has at most one neighbors in $Z$ and no neighbors in ($C_1$ + $C_2$) +

$\sum_{path/cycle\ i}\ CS_i + \sum_{path/cycle\ i}\ Y_i$, since $z$ can have neighbors only in $Z + S'_I$ . This implies that the graph induced by $Z$ is a set of disconnected edges and isolated vertices, thus $Z$ is 2-colorable. Let ($C''_1$, $C''_2$ ) be a complete 2-coloring of $Z$. Then ($C^3_1 = C^2_1 + C''_1$ , $C^3_2 = C^2_2 + C''_2$ ) is a complete 2-coloring ($C''_1$, $C''_2$ ) of $S'_B$ which is compatible to ($C^2_1$, $C^2_2$). By transitivity, ($C^3_1$, $C^3_2$) is compatible to ($C_1$ , $C_2$). This completes our proof that Step 5 is correct, and then concludes our proof of this lemma.

**Main Algorithm**

First, we present our main algorithm in Figure 5. Next we show the algorithm is correct.

**Lemma 3.** Algorithm Param-3-Coloring( $G, k, [S_B, S_I]$ ) either finds a complete bipartite-independent partition $[S'_B, V - S'_B]$ which k-extends $[S_B, S_I]$ if such a bipartite-independent partition exists, or reports 'NO' otherwise.

*Proof.* Step 1 deals with the cases when the solution can be easily determined. First, if $k < 0$, there is no bipartite-independent partition which $k$-extends $[S_B, S_I]$. Thus 'NO' is returned correctly. After this, $k \geq 0$. If $[S_B, V - S_B]$ is a bipartite-independent partition, then $[S_B, V - S_B]$ is a complete bipartite- independent partition $k$-extending $[S_B, S_I]$ since $k \geq 0$, and thus $[S_B, V - S_B]$ is returned correctly. After this, $[S_B, V - S_B]$ is not a complete bipartite-independent partition. Since $[S_B, V - S_B]$ is the only complete bipartite-independent partition which could possibly 0-extend $[S_B, S_I]$, any complete bipartite-independent extending $[S_B, S_I]$ must $i$-extend $[S_B, S_I]$ for some $i > 0$. Therefore, 'NO' should be returned when k = 0. In conclusion, step 1 correctly finds the solution.

After step 1, $k > 0$ and $S'_B$ must contain at least a vertex from $V - S_B - S_I$ for any complete bipartite-independent partition $k$-extending$[S_B, S_I]$. So $V - S_B - S_I$ is not empty. If a vertex $x \in V - S_B - S_I$ has a neighbor in $S_I$, then for any complete bipartite-independent

Algorithm Param-3-Coloring(G, k, [ $S_B$ , $S_I$ ])
Input: a graph G, a parameter k, and a bipartite-independent partition [ $S_B$ , $S_I$ ].
Output: either a complete bipartite-independent partition [$S'_B$ ,$V - S'_B$ ] which k-extends [ $S_B$ , $S_I$ ], or 'NO' otherwise.

1. if k < 0 return 'NO';
   if [$S_B$ ,$V - S_B$ ] is a bipartite-independent partition
      return [$S_B$ ,$V - S_B$ ];
   if k = 0 return 'NO';
2. if x ∈ $V - S_B - S_I$ has a neighbor in $S_I$
   if $S_B$ + x is 2-colorable return Param-3-coloring(G, k − 1, $S_B$ + x, $S_I$ );
      return 'NO';
3. if x ∈ $V - S_B - S_I$ in $S_B$ has two neighbours y, z in $S_B$ where yz is an edge in G, or has no neighbors
in $V - S_B - S_I$
      return Param-3-coloring(G, k, $S_B$ , $S_I$ + x);
4. if there is a vertex x ∈ $V - S_B - S_I$ which has three neighbors in $V - S_B - S_I$
   Let N (x) be the neighbors of x in $V - S_B - S_I$ ;
4.1 if $S_B$ + x is 2-colorable
      [$S'_B$ ,$V - S'_B$ ] =Param-3-coloring(G, k − 1, $S_B$ + x, $S_I$ );
      if [$S'_B$ ,$V - S'_B$ ]! ='NO'
         return [$S'_B$ ,$V - S'_B$ ];
4.2 if $S_B$ + N (x) is 2-colorable
      return Param-3-coloring(G, k − |N (x)|, $S_B$ + N (x), $S_I$ + x);
4.3 return 'NO';
Let $CS_1$, ⋯ , $CS_P$ be the candidate sets of paths/cycles induced by vertices in $V - S_B - S_I$ ;
5. if $\sum_{i=1}^{p}|CS_i|$ > k
   return 'NO';
6. if $k \leq S_B$
6.1 for each enumeration of $W_1 \subseteq CS_1$ , ⋯, $W_p \subseteq CS_P$ such that [$S_B$ +$\sum_{i=1}^{p} W_i$ , $S_I$ + $\sum_{i=1}^{p}(CS_i - Wi)$] is a bipartite-independent partition
      Let T ⊆ ($V - S_B - S_I - \sum_{i=1}^{p} CS_p$) be vertices having neighbors in $S_I$ + $\sum_{i=1}^{p}(CS_i - W_i)$
      if [$S'_B$ = SB + $\sum_{i=1}^{p} W_i$ + T, $V - S'_B$ ] is a complete bipartite-independent partition and $|S'_B| \leq |S_B| + k$
         return [$S'_B$ ,$V - S'_B$ ];
6.2 return 'NO';
7. for each 2-coloring C of $S_B$
   if there is a complete bipartite-independent partition [$S'_B$ ,$V - S'_B$ ] which k-extends [ $S_B$ , $S_I$ ] such that there is a 2-coloring of $S'_B$ compatible to C
      return [$S'_B$ ,$V - S'_B$ ];
8. return 'NO';

Figure 5: The main algorithm

partition $[S'_B , V - S'_B ]$ $k$-extending $[ S_B , S_I ]$, $x$ must be in $S'_B$ and not in $V - S'_B$ , since $V - S'_B$ should be an independent set and $S_I \subseteq V - S'_B$ . If $S_B + x$ is 2-colorable, $[S_B + x, S_I]$ is a bipartite-independent partition, and we only need to look for a complete bipartite-independent set $(k - 1)$-extending $[S_B + x , S_I]$. Otherwise, 'NO' should be returned. Hence step 2 is correct.

After step 2, any vertex $x \in V - S_B - S_I$ has no neighbors in $S_I$, and then $[S_B , S_I + x]$ is a bipartite-independent partition. If $x$ has two neighbors $y, z$ in $S_B$ such that $yz$ is an edge, then for any complete bipartite-independent partition $[S'_B , V - S'_B ]$ $k$-extending $[ S_B , S_I ]$, $x$ must be in $V - S'_B$ , since $S'_B$ should be 2-colorable. Thus we only need to search for a complete bipartite-independent set $k$-extending $[S_B , S_I + x]$. For this case, step 3 is correct. Now we consider the case when $x$ has no neighbors in $V - S_B - S_I$. Note that any complete bipartite-independent partition $k$-extending $[S_B , S_I + x]$ also $k$-extends $[ S_B , S_I ]$. On the other hand, given a complete bipartite-independent partition $[S'_B , V - S'_B ]$ $k$-extending $[ S_B , S_I ]$ where $x \in S'_B$, $[S'_B - x , V - S'_B + x]$ is also a complete bipartite-independent partition $k$-extending $[ S_B , S_I ]$, since $x$ has no neighbors in $V - S_B - S_I$. Therefore, there is a complete bipartite-independent partition $k$-extending $[ S_B , S_I ]$ if and only if there is one $k$-extending $[S_B , S_I + x]$. We conclude that Step 3 is still correct for this case.

For any complete bipartite-independent partition $[S'_B , V - S'_B ]$ k-extending$[ S_B , S_I ]$, x is in either $S'_B$ or $V - S'_B$. If $x$ is in $S'_B$ for a partition $[S'_B , V - S'_B ]$, then $S_B + x$ is 2-colorable, and step 4.1 should correctly find one which $(k - 1)$-extends $[S_B + x, S_I]$. If Step 4.1 does not return anything, then for any complete bipartite-independent partition $[S'_B , V - S'_B ]$, $x$ should be in $V - S'_B$, and thus $N(x)$ should be in $S_B$ . This is possible only when $S_I + x$ an independent set, and $S_B + N(x)$ is is 2-colorable. Note that $S_I + x$ is indeed an independent set, since $x$ has no neighbors in $S_I$ after Step 3. Therefore, when $S_B + N(x)$ is 2-colorable, Step 4.2 correctly

returns a complete bipartite-independent partition $[S'_B, V - S'_B]$ which $(k - |N(x)|)$-extends $[S_B + N(x), S_I + x]$ and also $k$-extends $[S_B, S_I]$ if there exists one, or returns 'NO' if it does not find any one. When $S_B + N(x)$ is not 2-colorable, 'NO' is returned correctly at Step 4.3, since by our arguments above, there is no complete bipartite-independent partition k-extending $[S_B, S_I]$.

After Step 4, any vertex $x \in V - S_B - S_I$ has at most two neighbors in $V - S_B - S_I$, no neighbors in $S_I$, and no two neighbors in $S_B$ which are neighbors of each other. Now $G[V - S_B - S_I]$ is a union of disjoint paths/cycles. According to lemma 1, for any complete bipartite-independent partition $[S'_B, V - S'_B]$ $k$-extending$[S_B, S_I]$, we have that $k \geq \sum_{i=1}^{q} |CS_i|$. Step 5 returns 'NO' correctly for the case $k < \sum_{i=1}^{q} |CS_i|$.

To prove that Step 6 is correct, we first show that there is a complete bipartite-independent partition k-extending $[S_B, S_I]$ if and only if Step 6.1 finds one. First, the partition $[S'_B, V - S'_B]$returned by Step 6.1 is a complete bipartite-independent partition k-extending $[S_B, S_I]$, since it is a complete bipartite-independent partition, $|S'_B| \leq |S_B| + k$, $S_B \subseteq S'_B$, and $S_I \subseteq V - S'_B$. On the other hand, if there is a complete bipartite-independent partition $[S'_B, V - S'_B]$k-extending $[S_B, S_I]$, Step 6.1 can find a complete bipartite-independent partition$[S'_B, V - S'_B]$ for some $W_i \subseteq CS_i$ where $1 \leq i \leq p$. Let $W'_i = S'_B \cap CS_i$ for $1 \leq i \leq p$. Then $CS_i - W'_i$ is in $V - S'_B$. So $[S'_B, V - S'_B]$ extends $[S_B + \sum_{i=1}^{p} W'_i, S_I + \sum_{i=1}^{p}(CS_i - W'_i)]$ which is of course a bipartite-independent partition. By our notation, T are those vertices having neighbors in $S_I + \sum_{i=1}^{p}(CS_i - W'_i)$. So T must be in $S'_B$, and then $S_B + \sum_{i=1}^{p} W'_i + T \subseteq S'_B$. It follows that $[S'_B, V - S'_B]$ $[S_B + \sum_{i=1}^{p} W'_i + T, S_I + \sum_{i=1}^{p}(CS_i - W'_i)]$. Note that vertices in $S'_B - (S_B + \sum_{i=1}^{p} W'_i + T)$ can only be from $T' = V - S_B - S_I - \sum_{i=1}^{p} CS_i - T$. Let $T'' = S'_B - (S_B + \sum_{i=1}^{p} W'_i + T)$. Note that $S'_B - T'' = S_B + \sum_{i=1}^{p} W'_i + T$. We complete our proof that Step 6.1 can find a complete

bipartite-independent partition k-extending [ $S_B$ , $S_I$ ], once we show that [$S'_B - T''$, $V - S'_B + T''$] is a complete bipartite-partition k-extending [ $S_B$ , $S_I$ ].

It is easy to see that $|S'_B - T''| \leq |S'_B| \leq |S_B| + k$, $V = (S'_B - T'') + (V - S'_B + T'')$, $S_B \subseteq S'_B - T''$, $S_I \subset V - S'_B + T''$, and $S'_B - T''$ is 2-colorable,. So we only need to show that $V - S'_B + T''$ is an independent set. Since

$$V - S_B - S_I = \sum_{i=1}^{p} CS_i + T + T', \text{ and}$$

$$S'_B = S_B + \sum_{i=1}^{p} W'_i + T + T''.$$

We have that

$$V - S'_B + T'' = S_I + \sum_{i=1}^{p} (CS_i - W'_i) + T'.$$

Now we only need to show that $S_I + \sum_{i=1}^{p} (CS_i - W'_i) + T'$ is an independent set.

By definition of T and $T'$, we have that $T \cap T' = \emptyset$. Since T are vertices of $V - S_B - S_I - \sum_{i=1}^{p} CS_i$ which have neighbors in $S_I + \sum_{i=1}^{p} (CS_i - W'_i)$, all vertices in $T'$ have no neighbors in $S_I + \sum_{i=1}^{p} (CS_i - W'_i)$. It follows that all vertices of $T''$ have no neighbors in $S_I + \sum_{i=1}^{p} (CS_i - W'_i)$, since $T'' \subseteq T'$ by definition. Moreover, by definitions of candidate sets and $T'$ , $T' \subseteq V - S_B - S_I - \sum_{i=1}^{p} CS_i$ is an independent set. Finally, $S_I + \sum_{i=1}^{p} (CS_i - W'_i) \subseteq V - S'_B$ is an independent set, since [$S'_B$ , $V - S'_B$ ] is a complete bipartite-independent partition. Now we conclude that $T'' + (S_I + \sum_{i=1}^{p} (CS_i - W'_i))$ is indeed an independent set, which also complete our proof on Step 6.1.

If for all possible $W_i \subseteq CS_i$, step 6.1 can not return a complete bipartite-independent partition, then there is no complete bipartite-independent partition k-extending [ $S_B$ , $S_I$ ] . Therefore, Step 6.2 returns 'NO' correctly. In conclusion, Step 6 is correct.

The partition [$S'_B$ , $V - S'_B$ ] returned at step 7 is correct, since it is a complete bipartite-independent partition k-extending [ $S_B$ , $S_I$ ] . On the other hand, when there is a complete

bipartite-independent partition $[S'_B, V - S'_B]$ k-extending $[S_B, S_I]$, there is a 2-coloring $C'$ of $S'_B$. Let C be the 2-coloring $C'$ restricted to vertices of $S_B$. By lemma 2, Step 7 should be able to find a complete bipartite-independent partition k-extending $[S_B, S_I]$. This concludes the correctness of step 7.

By the arguments above, Step 7 finds a complete bipartite-independent partition k-extending $[S_B, S_I]$ if and only there exists such one. Therefore, 'NO' should be returned, if Step 7 cannot find any complete bipartite-independent partition k-extending $[S_B, S_I]$ for each possible 2-coloring of $S_B$. So step 8 is correct. This completes the correctness proof of our main algorithm.

**The time complexity of algorithm Param-3-Coloring ($G, k, [S_B, S_I]$)**

Lemma 4 Algorithm Param-3-Coloring $(G, k, [S_B, S_I])$ terminates in time

$$O\left(1.466^k 2^{\frac{k+|S_B|}{2}} n^{O(1)}\right) \ where\ k \geq 0.$$

*Proof.* We use bounded-search tree analysis. A branch-and-bound procedure requires two tools. The first one is a splitting procedure that, given a set S of candidates, returns two or more smaller sets $S_1, S_2, \cdots$ whose union covers S. Note that the minimum of $f(x)$ over S is $min\{V_1, V_2, \cdots\}$, where each $v_i$ is the minimum of $f(x)$ within $S_i$. This step is called branching, since its recursive application defines a tree structure whose nodes are the subsets of S. Another tool is a procedure that computes upper and lower bounds for the minimum value of $f(x)$ within a given subset S. This step is called bounding. Note that each step may directly return, or decrease the number of vertices in $- S_B - S_I$, or decrease k. Algorithm Param-3-Coloring $(G, k, [S_B, S_I])$ terminates after at most $|V - S_B - S_I|$ + k recursive calls. Each step except Step 6.1 and 7 can be done in polynomial time. However, we can regard Step 6.1 as a branch for $2^{\sum_{i=1}^{q} CS_i}$ combination of

subsets of $CS_i$, and Step 7 as a branch for $2^{|S_B|}$ 2-coloring of $S_B$. For each combination of subset of $CS_i$ , Step 6.1 can be done in polynomial time. For each 2-coloring of $S_B$ , Step 7 can be done in polynomial time by lemma 2. Now we conclude that algorithm Param-3-Coloring $(G, k, [S_B, S_I])$ will terminates after at most $|V - S_B - S_I|$ + k recursive calls, and each step takes polynomial time.

Next we prove that $3 \times 1.466^k 1.365^{\frac{k+|S_B|}{2}}$ is the upper bound on the number of branches by mathematical induction. Only Step4, 6 and 7 have branches.

For Step 4, any complete bipartite-independent partition $[S'_B, V - S'_B]$ k-extending$[S_B, S_I]$, x is in either $S'_B$ or $V - S'_B$. If $x$ is in $S'_B$ for a partition $[S'_B, V - S'_B]$, then $S_B + x$ is 2-colorable, and step 4.1 should correctly find one which $(k-1)$-extends $[S_B + x, S_I]$. If Step 4.1 does not return anything, then for any complete bipartite-independent partition $[S'_B, V - S'_B]$, $x$ should be in $V - S'_B$, and thus $N(x)$ should be in $S_B$ . We have recursive equation:

$$f(k, |S_B|) = f(k-1, |S_B| + 1) + f(k - |N(x)|, |S_B| + |N(x)|) \; where \; |N(x)| \geq 3$$

For Step 6, any complete bipartite-independent partition $[S'_B, V - S'_B]$ $k$-extending$[S_B, S_I]$, we have that $k \geq \sum_{i=1}^{q} |CS_i|$. We have recursive equation:

$$f(k \leq |S_B|, |S_B|) = 2^{\sum_{i=1}^{q} |CS_i|} \leq 2^k$$

For Step 7, we have recursive equation by lemma 2:

$$f(k > |S_B|, |S_B|) = 2^{|S_B|}$$

When $k \leq 0$, the algorithm returns directly at step 1. So we have $f(k \leq 0, |S_B|) = 1$. We only need to show that the upper bound is correct when $k > 0$. The upper bound is correct for Step 4, since

The upper bound is correct for Step 4, since

$$f(1 \leq k < |N(x)|, |S_B|) \leq f(k-1, |S_B| + 1) + f(k - |N(x)|, |S_B| + |N(x)|)$$

Because $1 \leq k < |N(x)|$ , so $k - |N(x)| < 0$, we have $f(k \leq 0, |S_B|) = 1$

$$f(1 \leq k < |N(x)|, |S_B|) \leq 3 \times 1.466^{k-1} 1.365^{\frac{k+|S_B|}{2}} + 1$$

$$\leq 3 \times 1.466^{k} 1.365^{\frac{k+|S_B|}{2}}$$

$$f(k > |N(x)|, |S_B|) \leq f(k-1, |S_B|+1) + f(k-|N(x)|, |S_B|+|N(x)|)$$

$$\leq 3 \times 1.466^{k-1} 1.365^{\frac{k+|S_B|}{2}} + 3 \times 1.466^{k-|N(x)|} 1.365^{\frac{k+|S_B|}{2}}$$

$$\leq 3 \times 1.466^{k} 1.365^{\frac{k+|S_B|}{2}}$$

The upper bound is correct for Step 6 and 7, since

$$f(k \leq |S_B|, |S_B|) = 2^{\sum_{i=1}^{q} |CS_i|} \leq 2^k \leq 3 \times 1.466^{k} 1.365^{\frac{k+|S_B|}{2}}$$

To make sure $2^k \leq 3 \times 1.466^{k} 1.365^{\frac{k+|S_B|}{2}}$, first get $2^k \leq 1.466^{k} 1.365^{\frac{k+|S_B|}{2}}$

$$f(k > |S_B|, |S_B|) = 2^{|S_B|} \leq 3 \times 1.466^{k} 1.365^{\frac{(k+|S_B|)}{2}} \ (k > |S_B|)$$

Since there are at most $3 \times 1.466^{k} 1.365^{\frac{k+|S_B|}{2}}$ branches, each of which can be done in $O(n^c)$ time where $c$ is a constant independent of $k$, algorithm Param-3-Coloring $(G, k, [S_B, S_I])$ terminates in time $O(1.466^{k} 1.365^{\frac{k+|S_B|}{2}} n^{O(1)})$

Now we are ready to apply our algorithm to solve the parameterized 3-coloring problem.

Theorem 1. Param-3-Coloring $(G, k, [\emptyset, \emptyset])$ solves the parameterized 3-coloring problem correctly in time $O(1.713^{k} n^{O(1)})$.

*Proof.* By lemma 3, Param-3-Coloring $(G, k, [\emptyset, \emptyset])$ either finds a complete bipartite-independent partition $[S'_B, V - S'_B]$ which $k$-extends $[\emptyset, \emptyset]$ if such a bipartite-independent partition exists, or reports 'NO' otherwise. In a partition like $[S'_B, V - S'_B]$, we have that $S'_B$ contains at most k vertices and is 2-colorable, and that $V - S'_B$ contains the remaining vertices

and is an independent set. So the algorithm solves the parameterized 3-coloring problem correctly.

By lemma 4, Param-3-Coloring($G, k, [\emptyset, \emptyset]$) terminates in time $O\left(1.466^k 1.365^{\frac{k}{2}} n^{O(1)}\right) = O(1.713^k n^{O(1)})$. This concludes our proof.

# CHAPTER III

# IMPLEMENTATION STUDY

There are two algorithms for the 3-coloring problem we implemented. The first algorithm is a fixed parameterized algorithm proposed by us. The second one is the algorithm proposed by Beigel and Eppstein in [13].

## Measures for comparing Algorithm

Comparing Algorithm: 3-Coloring in Time $O(1.3289^n)$. The algorithm is based on a constraint satisfaction (CSP) formulation of these problems. 3-SAT is equivalent to (2, 3)-CSP while the other problems above are special cases of (3, 2)-CSP; there is also a natural duality transformation from (a, b)-CSP to (b, a)-CSP. We give a fast algorithm for (3, 2)-CSP and use it to improve the time bounds for solving the other problems listed above. The techniques involve a mixture of Davis-Putnam-style backtracking with more sophisticated matching and network flow based ideas.

## Experimental Study of Algorithm

We generate test graphs for both algorithms. The text graphs are 3-colorable graphs, each test graph of n vertices has a complete bipartite-independent partition[ $S_B$ , $S_I$ ] such that k = | $S_B$ | = $\left\lceil \frac{n}{2} \right\rceil$. For twelve cases from n=40 to n=90, 100 graphs are generated at each case; n=95, 60 graphs

1. If the input graph G contains any vertex v with degree less than two, recursively color G \ {v} and assign v a color different from its neighbors .

2. If the input graph contains a cycle or large tree of degree-three vertices, split the problem into smaller instances, recursively attempt to color each smaller instance, and return the first successful coloring found by these recursive calls.

3. Find a maximal bushy forest F in G.

4. Find a maximal set T of K1,3 subgraphs in G \ F .

5. While it is possible to increase the size of T by removing one K1,3 subgraph and using the vertices in G \ (F ∪ T ) to form two more K1,3 subgraphs, do so.

6. Use the network flow algorithm to assign the vertices of G \ (F ∪ N (F ) ∪ T ) to trees in T , forming a forest H of height-two trees.

7. Recursively search through all consistent combinations of colors for the bushy forest roots and internal nodes, and for selected vertices in H. For each coloring of these vertices, form a (3, 2)-CSP instance describing the possible colorings of the uncolored vertices, and use our CSP algorithm to attempt to solve this instance. If one of the CSP instances is solvable, return the resulting coloring. If no CSP instance is solvable, return a flag value indicating that no coloring exists.

Figure 6: Outline of the overall algorithm in 3-Coloring in Time $O(1.3289^n)$

are generated at that case; a total of 1160 generated graphs. These graphs are applied to the exact algorithm by Beigel and Eppstein and our parameterized algorithm, called Eppstein algorithm and parameterized algorithm. We also control the edge densities of test graphs, i.e., the ratio of the number of edges over the number of vertices. In this section, we discuss experimental

performance of these two algorithms. Both algorithms are implemented in C++ on PC with 2GHz of Inter Core2 Duo CPU and 4GB of RAM.

```
Procedure graph-generator(n)
Begin
    Set up n vertices in three sets and make the sum of the first two sets equal to the third one;
    For e:=1 to 2n do
    Choose randomly two vertices x, y from two different sets and check edge(x,y) exist or not
    If not add an edge(x,y);
    End for;
End.
```

Figure 7: The test graphs generator

**Comparison of Eppstein Algorithm and Parameterized Algorithm**

Figure.8 gives results for running time for Eppstein algorithm and parameterized algorithm, when $\frac{number\ of\ vertices}{number\ of\ edges} = \frac{1}{2}$, where "average" shows the variation in average running time for each n as a function of the average number of vertices for each n. From the experimental results, the running time is similar. When $n \leq 65$ the Eppstein algorithm is faster than parameterized algorithm, and when $n \geq 65$ the parameterized algorithm is faster than Eppstein algorithm, but both running time clearly exhibit exponential growth.

**Effects of Edge Densities**

Table 2 summarizes results, where E.A represents Eppstein algorithm; P.A represents parameterized algorithm; N.V represents number of vertices; E.D represents Edge Densities. Table 2 shows that the running time of both algorithms in different densities. In Eppstein algorithm: It shows that as the density becomes bigger, the running time increase dramatically; in parameterized algorithm: It shows that as the density becomes bigger, the running time is quite

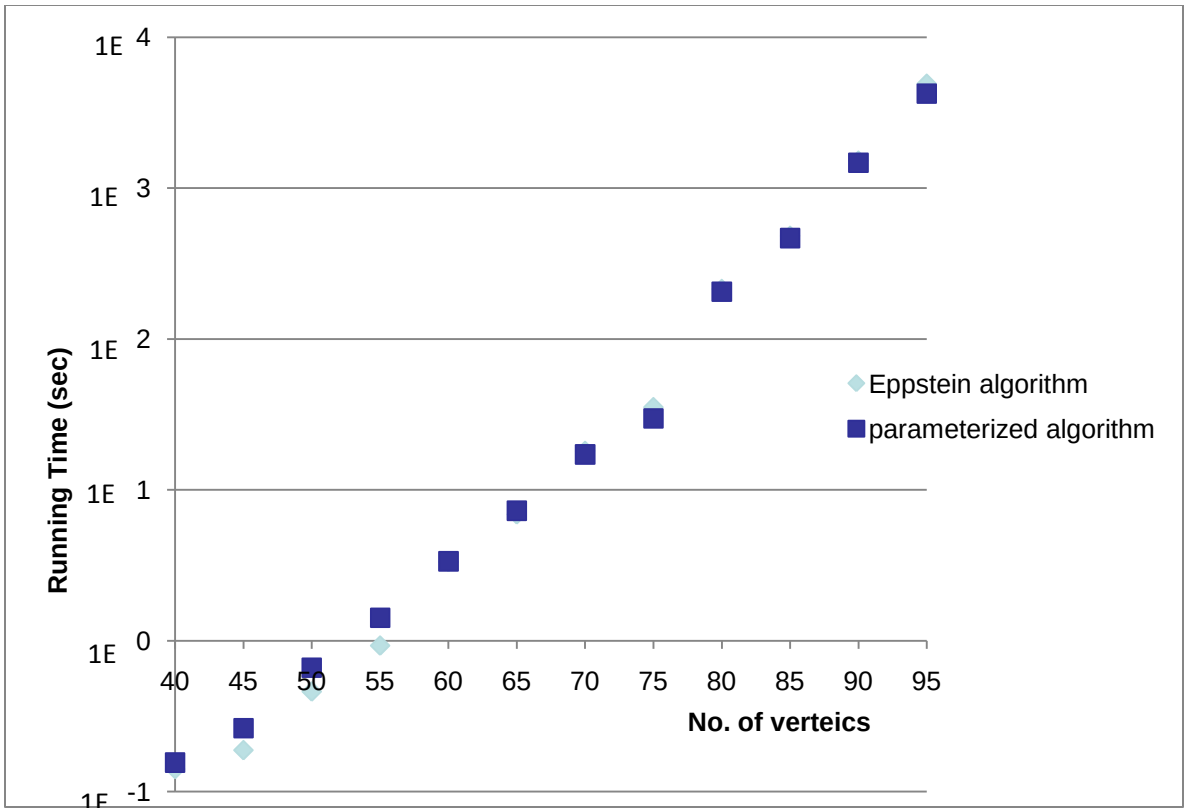


Figure 8: Experimental results on graphs average running time of two algorithms

stable for different densities.

Experiments confirmed that parameterized algorithm is more stably and invariably by increasing edge densities. Table 2 introduces when edge densities increase, the running time of parameterized algorithm increase much slower than Eppstein algorithm for graphs of the same number of vertices. The parameterized algorithm is much better to deal with the large edge density when k=n/2.

Table 3 shows the running time ratios of the exact algorithm for graphs of different densities. It shows as the density becomes bigger, the running time increase dramatically: it increase 13 times as density change from 2 to 7 for graphs of 95 vertices. On the contrary, the modified parameterized algorithm is quite stable for different densities. The running time of the

modified parameterized algorithm increase only around 2 times as density changes from 2 to 7 for graphs of 95 vertices, see Table 4

| Algorithms | E.A | P.A | E.A | P.A | E.A | P.A | E.A | P.A | E.A | P.A | E.A | P.A |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| N.V \ E.D | 1:2 | | 1:3 | | 1:4 | | 1:5 | | 1:6 | | 1:7 | |
| 40 | 0.134 | 0.173 | 0.152 | 0.152 | 0.343 | 0.143 | 0.422 | 0.156 | 0.624 | 0.136 | 0.642 | 0.123 |
| 45 | 0.247 | 0.379 | 0.285 | 0.316 | 0.543 | 0.286 | 0.643 | 0.299 | 1.742 | 0.269 | 2.429 | 0.312 |
| 50 | 0.596 | 0.739 | 0.785 | 0.684 | 1.245 | 0.549 | 2.216 | 0.764 | 3.581 | 0.632 | 5.513 | 0.514 |
| 55 | 0.967 | 1.368 | 1.147 | 1.075 | 1.542 | 0.749 | 5.653 | 1.417 | 9.752 | 1.135 | 12.64 | 1.031 |
| 60 | 4.693 | 4.739 | 5.532 | 4.678 | 7.783 | 4.564 | 14.86 | 4.363 | 25.37 | 3.975 | 37.95 | 3.521 |
| 65 | 7.531 | 7.752 | 11.45 | 7.643 | 12.54 | 6.482 | 39.78 | 5.895 | 41.36 | 6.831 | 67.22 | 6.126 |
| 70 | 23.16 | 21.14 | 41.46 | 23.46 | 67.77 | 20.78 | 109.5 | 28.6 | 128.6 | 25.75 | 153.2 | 23.2 |
| 75 | 49.24 | 42.59 | 63.32 | 41.43 | 143.8 | 42.76 | 235.5 | 41.87 | 255 | 43.74 | 275.4 | 39.28 |
| 80 | 297.2 | 281.7 | 316.3 | 225.7 | 475.7 | 254.3 | 598.4 | 315.6 | 712.5 | 309.5 | 841.5 | 293.5 |
| 85 | 614.4 | 601.4 | 794.2 | 442.4 | 678.5 | 386.7 | 864.7 | 354.7 | 1092 | 339.4 | 1294 | 319.3 |
| 90 | 1624 | 1512 | 2723 | 1564 | 3218 | 1531 | 5228 | 1490 | 6129 | 1238 | 8214 | 1125 |
| 95 | 6215 | 5635 | 7636 | 5387 | 9126 | 4217 | 9865 | 4035 | 10218 | 4029 | 12645 | 3915 |

Table 2: Average running time(sec) of two algorithms with different edge densities

| Number of Vertices | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 40 | 1 | 1.13 | 2.56 | 3.15 | 4.66 | 4.79 |
| 45 | 1 | 1.15 | 2.20 | 2.60 | 7.05 | 9.83 |
| 50 | 1 | 1.32 | 2.09 | 3.72 | 6.01 | 9.25 |
| 55 | 1 | 1.19 | 1.59 | 5.85 | 10.08 | 13.07 |
| 60 | 1 | 1.18 | 1.66 | 3.17 | 5.41 | 8.09 |
| 65 | 1 | 1.52 | 1.67 | 5.28 | 5.49 | 8.93 |
| 70 | 1 | 1.79 | 2.93 | 4.73 | 5.55 | 6.61 |
| 75 | 1 | 1.29 | 2.92 | 4.78 | 5.18 | 5.59 |
| 80 | 1 | 1.06 | 1.60 | 2.01 | 2.40 | 2.83 |
| 85 | 1 | 1.29 | 1.10 | 1.41 | 1.78 | 2.11 |
| 90 | 1 | 1.68 | 1.98 | 3.22 | 3.77 | 5.06 |
| 95 | 1 | 1.23 | 1.47 | 1.59 | 1.64 | 2.03 |

Table 3: Ratio of Exact Algorithm for Different Densities

The experimental results show that our algorithm can help to improve efficiency on resolve a 3-coloring problem when the input size n is large, edge density is large and the parameter k is relative small. It is better than Eppstein's Algorithm for solving 3-coloring when k is less or equal than n/2. Our theoretical analysis proves that, when parameter k is a small value, the implemented with our method has higher speed.

| Number of Vertices | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|
| 40 | 1 | 0.88 | 0.83 | 0.90 | 0.79 | 0.71 |
| 45 | 1 | 0.83 | 0.75 | 0.79 | 0.71 | 0.82 |
| 50 | 1 | 0.93 | 0.74 | 1.03 | 0.86 | 0.70 |
| 55 | 1 | 0.79 | 0.55 | 1.04 | 0.83 | 0.75 |
| 60 | 1 | 0.99 | 0.96 | 0.92 | 0.84 | 0.74 |
| 65 | 1 | 0.99 | 0.84 | 0.76 | 0.88 | 0.79 |
| 70 | 1 | 1.11 | 0.98 | 1.35 | 1.22 | 1.10 |
| 75 | 1 | 0.97 | 1.00 | 0.98 | 1.03 | 0.92 |
| 80 | 1 | 0.80 | 0.90 | 1.12 | 1.10 | 1.04 |
| 85 | 1 | 0.74 | 0.64 | 0.59 | 0.56 | 0.53 |
| 90 | 1 | 1.03 | 1.01 | 0.99 | 0.82 | 0.74 |
| 95 | 1 | 0.96 | 0.75 | 0.72 | 0.71 | 0.69 |

Table 4: Ratio of Parameterized Algorithm for Different Densities

# CHAPTER IV

# FUTURE WORK

Given a graph G, if there exists a 3-coloring of G such that $k \leq 0.527n$, our algorithm is faster than the best algorithm to solve the 3-coloring problem. It is interesting to improve our algorithm to beat the best algorithm for the 3-coloring problem. There are some questions. Can we solve the case dealt by step 6 and 7 is polynomial time or better than $2^k$ / $2^{|S_B|}$? What kind graphs have a large independent set such that the remaining graph is 2-colorable? We have studied the parameterized 3-coloring problem on vertex coloring. How about the same idea on edge coloring?

Another direction is to consider different parameters. One possible candidate is the size of an independent size. Can we find an independent set of size k such that the remaining graph is 2-colorable? If we do not require the k vertices to be an independent set, there is an FPT algorithm of time $O^*(3^k)$ [14]. When we require the k vertices to be an independent set, is there any fpt algorithm for this problem?

REFERENCES


[1] R. Beigel and D. Eppstein, 3-colring in time $O(1.3446^n)$: a no-MIS algorithm, Proc. 36th Symp. Foundations of Computer Science, pp. 444-453, 1995.

[2] R. Beigel and D. Eppstein, 3-coloring in time $O(1.3289^n)$, Journal of Algorithms 54 (2): 168–204, 2005.

[3] A. Bjorklund, T. Husfeldt, and M. Koivisto, Set partitioning via inclusion- exclusion, SIAM J. on Computing, 39(2):546-563, 2009.

[4] A. Blum and D. Karger, An $O(n^{3/14})$- coloring algorithm for 3-colorable graphs, Information Processing Letter, 61(1):49-53, 1997.

[5] J. M. Byskov, Enumerating maximal independent sets with applications to graph colouring, Operations Research Letters, 32:547-556, 2004.

[6] G. J. Chaitin, Register allocation & spilling via graph coloring, Proc. of the 2nd SIGPLAN symposium on Compiler construction, pp.98-105, 1982.

[7] D. Eppstein, Improved algorithms for 3-coloring, 3-edge-coloring, and constraint satisfaction, Proc. 12th Symp. on Discrete Algorithms, pp. 329-337, 2001.

[8] D. Eppstein, Small maximal independent sets and faster exact graph coloring, J. Graph Algorithms and Applications, 7(2):131-140, 2003.

[9] M. Fellows, Blow-ups, win/win's, and crown rules: Some new directions in FPT, Lecture Notes in Computer Science (WG'03), pp.1-12, 2003.

[10] S. Khanna, N. Linal, and S. Safra, On the hardness of approximating the chromatic number, Proc. 2nd Isral Symp. on Theory and Computing Systems, pp. 256-260, 1993.

[11] Lawer, A note on the complexity of the chromatic problem, Information Processing Letter, 5(3):66-67, 1976

[12] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems, Proc. 25th Symp. of Theory of Computing, pp. 286-293, 1993.

[13] C. H. Papadimitriou and M. Yannakakis, On the complexity of database queries, Journal of Computer and System Sciences, 58: 407-427, 1999.

[14]. B. A. Reed, K. Smith, and A. Vetta, Finding odd cycle transversals, Operation Research Letters, 32(4):299-301, 2004.

[15] I. Schiermeyer, Deciding 3-colourability in less than $1.415^n$ steps, Proc. 19th Int. Workshop Graph-Theoretic Concepts in Computer Science, pp. 177-182, 1994.

# BIOGRAPHICAL SKETCH

Qing Wang was born in P.R.China in Oct.1984 as the son of Jianmin Wang and Hongping Du. He received his bachelor Degree in Computer Science of Chongqing University of Posts and Telecommunications, Chongqing, P.R.China in 2008. He graduated from University of Texas-Pan American with Master of Science in Computer Science in May, 2011. His research fields include design and analysis of algorithms.