Heuristic Methods for Colouring Dynamic Random Graphs

Bradley Hardy

A thesis submitted for the degree of PhD.

February 2018

Summary

Many real-world operational research problems can be reformulated into static graph colouring problems. However, such problems might be better represented as dynamic graphs if their size and/or constraints change over time.

In this thesis, we explore heuristics approaches for colouring dynamic random graphs. We consider two different types of dynamic graph: edge dynamic and vertex dynamic. We also consider two different change scenarios for each of these dynamic graph types: without future change information (i. e. random change) and with probabilistic future change information.

By considering a dynamic graph as a series of static graphs, we propose a "modification approach" which modifies a feasible colouring (or solution) for the static representation of a dynamic graph at one time-step into a colouring for the subsequent time-step. In almost all cases, this approach is beneficial with regards to either improving quality or reducing computational effort when compared against using a static graph colouring approach for each time-step independently. In fact, for test instances with small amounts of change between time-steps, this approach can be beneficial with regards to both quality *and* computational effort.

When probabilistic future change information is available, we propose a "two-stage approach" which first attempts to identify a feasible colouring for the current time-step using our "modification approach", and then attempts to increase the robustness of the colouring with regards to potential future changes. For both the edge and vertex dynamic cases, this approach was shown to decrease the "problematic" change introduced between time-steps. A clear trade-off can be observed between the quality of a colouring and its potential robustness, such that a colouring with more colours (i. e. reduced quality) can be made more robust.

Declaration

This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is being submitted concurrently in candidature for any degree or other award.

Signed .....(candidate) Date .....

Statement 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.

Signed .....(candidate) Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated, and the thesis has not been edited by a third party beyond what is permitted by Cardiff University's Policy on the Use of Third Party Editors by Research Degree Students. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed .....(candidate) Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed .....(candidate) Date .....

Acknowledgements

Firstly, I would like to thank Dr. Rhyd Lewis and Dr. Jonathan Thompson for their support and guidance throughout this four-year journey. Thank you for reading and providing feedback on, what has felt like, several hundred of versions of papers, presentations and this thesis. Thank you also for making this experience enjoyable, and for being good friends, as well as great supervisors.

I would like to express my gratitude to EPSRC for their financial support, and to the School of Mathematics for providing the resources necessary to facilitate this research. Thank you also to the members of staff and my fellow PhD students at Cardiff University for making my PhD career such a pleasurable one.

Thank you to my close family. To my mum, her partner: Phil, my dad, and my in-laws: Dave, Val and Jason. You have been (and continue to be) a great source of support and encouragement, and you have pushed me to achieve the best for myself.

I would also like to thank my church family at Tamworth Elim. In particular, I would like to thank Dan, Siân, Amélie, Elijah and the rest of my life-group. Thanks for the cups of tea and coffee, the weekends away, and the mountains of prayer. Thanks especially for the prayer!

Finally, I would like to thank my wife and best friend: Brenda. Thank you for allowing me to live away from you during the first year of our marriage, so that I could finish my studies. Thank you for selflessly giving of yourself during the final months of my writing up, so that I didn't have to concern myself with day-to-day tasks. Thank you for always being there, whether in person or at the end of the phone. And thank you for loving me unconditionally. I would not have accomplished any of this without you. I love you so much.

List of Publications

Hardy, B., Lewis, R., and Thompson, J. Modifying colourings between time-steps to tackle changes in dynamic random graphs. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 186–201. Springer, 2016.

Hardy, B., Lewis, R., and Thompson, J. Tackling the edge dynamic graph colouring problem with and without future adjacency information. *Journal of Heuristics*, pages 1–23, 2017.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

The graph colouring problem (GCP) is a very well-studied combinatorial optimisation problem. Many real-world operational research problems can be reformulated into GCPs, such as exam timetabling and frequency assignment (see Section 1.4). However, in order to use GCP methods to tackle these real-world problems, it is usually assumed that the real-world problem is not subject to change. If the size and/or constraints of a real-world problem are altered in any way then the associated graph will no longer be an accurate representation.

The importance of studying dynamic graphs and their associated problems has been highlighted by Harary and Gupta [1997]. Yet, despite this, there has been very little research, to date, regarding the colouring of dynamic graphs explicitly.

Due to its NP-hard nature, the most common approach to tackling the GCP utilises heuristic methods to identify "good", although not necessarily optimal, solutions. It therefore follows that using heuristic methods to tackle dynamic variants of the GCP will also prove to be an appropriate approach.

1.1 Aims and Structure of Thesis

In this thesis we wish to explore how we might approach the colouring of dynamic graphs. We will consider the dynamic GCP in two flavours: i) edge dynamic, and ii) vertex dynamic. By considering these two flavours we can explore methods that will be useful in the context of real-world problems where either the constraints are variable or the size of the problem itself is variable.

Within each of these flavours we will also consider two different change scenarios: i) without future change information (i. e. random change), and ii) with probabilistic future change information. The first of these change scenarios will give us the opportunity to consider how best to "repair" a solution if it becomes infeasible *after* changes have occurred. The second scenario will allow the future change information to be utilised *before* any changes occur in an attempt to improve the robustness of a solution with regards to those changes.

By considering a dynamic graph as a series of static graphs at different time-steps, we will explore how a feasible solution for the static representation of a dynamic graph at one time-step can be modified into a solution for the static representation in the following time-step. The main aim therefore is to determine whether this "modification approach" is beneficial compared to treating each static representation independently. Also, if any benefits are observed, how much can a graph change between time-steps for this to remain true? In Chapters 3 and 5, we will show that, with appropriate parameter settings, this approach *is* beneficial with regards to either improving solution quality or reducing computational effort, in most cases. In fact, for test instances with small amounts of change between time-steps, this approach can be beneficial with regards to both quality *and* computational effort.

When considering the second change scenario, where probabilistic future change information is provided, we wish to improve the robustness of a solution with regards to potential changes by utilising the previously unavailable information. More specifically, we wish to determine whether this information can be used so that any potential benefits of implementing our "modification approach" can be further improved. In Chapters 4 and 6, we will show that implementing a "two-stage approach" can decrease the amount of "problematic" change introduced between time-steps, which does indeed have knock-on effects for the benefits of our "modification approach". However, we will also show a clear trade-off between solution quality and robustness, such that the further a solution is from optimal, the more robust it can be made.

This thesis has the following structure: the remainder of Chapter 1 will introduce the static GCP and discuss some of its practical applications. Chapter 2 will review methods used for tackling the static GCP with a particular focus on local search heuristics. In Chapter 3 we will introduce the edge dynamic GCP without future change information and will investigate whether our proposed "modification approach" is beneficial in any way. Chapter 4 will then extend upon the work in Chapter 3 by incorporating future change information in an attempt to improve the robustness of colourings between time-steps. Chapters 5 and 6 will follow a similar pattern to Chapters 3 and 4 but with regards to the vertex dynamic GCP, both with and without future change information. Finally, in Chapter 7 we summarise all of the key findings of the work presented and highlight areas of potential future work.

1.2 Graphs and Colouring

A graph $G = (V, E)$ is defined by a vertex set $V = \{v_1, \ldots, v_n\}$ and an edge set $E \subseteq \{V \times V\}$. Vertices $u, v \in V$ are said to be adjacent to one another if there exists an edge $\{u, v\} \in E$. The degree of a vertex $v \in V$, denoted $\deg(v)$, is equal

Table 1.1: Adjacency matrix of the graph shown in Figure 1.1.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_1	0	1	1	1	0	1	1	0	0	0
v_2	1	0	0	0	1	0	0	0	0	0
v_3	1	0	0	1	0	1	1	0	0	0
v_4	1	0	1	0	1	1	1	1	0	0
v_5	0	1	0	1	0	0	1	1	0	0
v_6	1	0	1	1	0	0	1	0	1	0
v_7	1	0	1	1	1	1	0	0	1	0
v_8	0	0	0	1	1	0	0	0	0	1
v_9	0	0	0	0	0	1	1	0	0	1
v_{10}	0	0	0	0	0	0	0	1	1	0

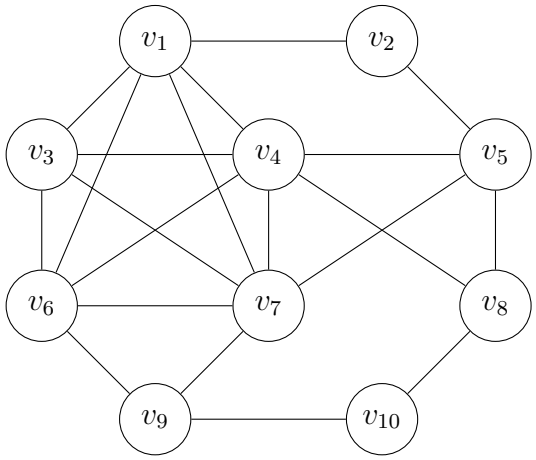

Figure 1.1: An undirected, static graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_9, v_{10}\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_3\}, \ldots, \{v_8, v_{10}\}, \{v_9, v_{10}\}\}$.

to the number of unique vertices adjacent to v.

Throughout this thesis, all graphs $G = (V, E)$ are undirected (i. e. an edge $\{u, v\} \in E$ can also be labelled as $\{v, u\}$), and without loops (i. e. there are no edges $\{u, v\} \in E$ where $u = v$). Unless stated explicitly otherwise, all graphs are also considered to be static (i. e. V and E do not change).

A graph $G = (V, E)$ is commonly represented by a $|V| \times |V|$ adjacency matrix. The (u, v)th entry of the adjacency matrix for G is equal to one if vertices $u, v \in V$ are adjacent to one another, and zero otherwise. The graph in Figure 1.1 is represented by the adjacency matrix in Table 1.1. Note that the diagonal values $(v_1, v_1), (v_2, v_2), \ldots, (v_{10}, v_{10})$ are all equal to zero because the graph in Figure 1.1 does not contain any loops.

The density of a graph $G = (V, E)$ is given by $\frac{2m}{n(n-1)}$ where $n = |V|$ and $m = |E|$. For example, the graph in Figure 1.1 has 10 vertices and 20 edges, therefore its density is $\frac{4}{9} = 0.\dot{4}$.

An independent set of G is a set of mutually non-adjacent vertices $V' \subseteq V$ (i. e. if $u, v \in V'$ then $\{u, v\} \notin E$) and is said to be maximal if no other independent set

V'' exists such that $|V'| < |V''|$. The independence number of G, denoted by $\alpha(G)$, is equal to the number of vertices in the largest maximal independent set of G. In Figure 1.1, the set of vertices $\{v_2, v_3, v_8, v_9\}$ is the largest maximal independent set, therefore the independence number of this graph is 4.

A clique of G is a set of mutually adjacent vertices $V' \subseteq V$ (i. e. for each pair of vertices $u, v \in V'$ there exists an edge $\{u, v\} \in E$). The clique number of G, denoted by $\omega(G)$, is equal to the number of vertices in the largest clique of G. In Figure 1.1, the largest clique is the set of vertices $\{v_1, v_3, v_4, v_6, v_7\}$, therefore the clique number of this graph is 5.

The Graph Colouring Problem

Given a graph $G = (V, E)$, the objective of the graph colouring problem (GCP) is to colour each vertex $v \in V$ such that adjacent vertices are coloured differently and the number of colours used is minimised. The minimum number of colours required to colour a graph G is known as the chromatic number of G, denoted by $\chi(G)$.

The GCP can be considered as a partitioning problem where we wish to partition V into k non-overlapping subsets such that k is minimised and each partition of V is an independent set. This is equivalent to defining a colouring function $c : V \rightarrow \{1, \ldots, k\}$ such that k is minimised and $c(u) \neq c(v)$ if $\{u, v\} \in E$. If adjacent vertices are assigned the same colour, this is called a *clash* and the vertices involved are said to be *clashing*.

In this thesis, a colouring (or solution) $\mathcal{S}$ of $G = (V, E)$ is any partition of V into non-overlapping subsets. If V is partitioned into k subsets (i. e. $\mathcal{S} = \{S_1, \ldots, S_k\}$ where $S_i \subseteq V$ for $i = 1, \ldots, k$) then we call $\mathcal{S}$ a k-colouring of G and S_i the ith colour class of $\mathcal{S}$.

Definition 1.1. A colouring $\mathcal{S}$ of $G = (V, E)$ is *complete* if every vertex $v \in V$ is assigned to a colour class of $\mathcal{S}$, otherwise it is *partial*.

Definition 1.2. A colouring $\mathcal{S}$ of $G = (V, E)$ is *proper* if each colour class of $\mathcal{S}$ is an independent set, otherwise it is *improper*. In other words, a colouring is proper if it contains no clashes.

Definition 1.3. A colouring $\mathcal{S}$ of $G = (V, E)$ is *feasible* if it is both complete and proper, otherwise it is *infeasible*.

The colouring shown in Figure 1.2 can be represented as $\mathcal{S} = \{S_1, S_2, S_3, S_4, S_5\}$ where $S_1 = \{v_1, v_5, v_9\}$, $S_2 = \{v_2, v_4, v_{10}\}$, $S_3 = \{v_3, v_8\}$, $S_4 = \{v_6\}$, and $S_5 = \{v_7\}$

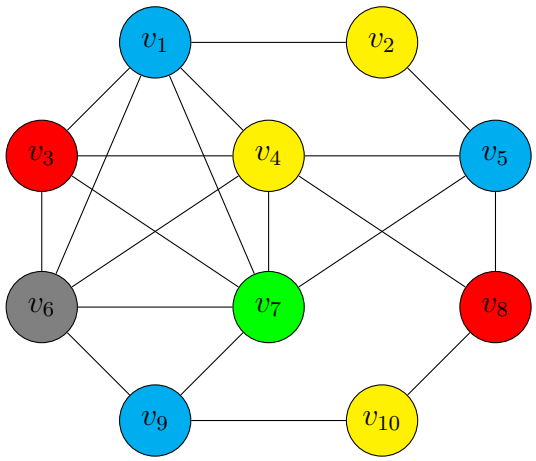

Figure 1.2: A feasible 5-colouring of the graph shown in Figure 1.1.

and the complimentary colouring function would be defined as

$$c(v_i) = \begin{cases} 1 & \text{if } i = 1, 5, 9 \\ 2 & \text{if } i = 2, 4, 10 \\ 3 & \text{if } i = 3, 8 \\ 4 & \text{if } i = 6 \\ 5 & \text{if } i = 7 \end{cases} \tag{1.1}$$

It should be clear that this colouring is feasible as it is both complete (i. e. every vertex $v \in V$ has been assigned to a colour class) and proper (i. e. each colour class is an independent set). The chromatic number for this particular graph is 5, therefore this feasible colouring is also optimal.

Another way of representing feasible colourings involves graph homomorphisms. In this representation, non-adjacent vertices are contracted to produce a complete graph K_k such that each of the k hyper-vertices in K_k represents a colour class [Hell and Nešetril, 2004]. In other words, for every feasible k-colouring of a graph G there exists a homomorphism $G \to K_k$. We have shown that the graph in Figure 1.1 is optimally coloured using 5 colours, therefore there must also exist a homomorphism from this graph to the complete graph K_5.

1.3 Complexity of the GCP

The number of ways of partitioning n objects into k non-empty, non-overlapping subsets is given by the Stirling numbers of the second kind, denoted $\left\{ {n \atop k} \right\}$ (or $S(n, k)$) where

$$\left\{ {n \atop k} \right\} = \begin{cases} 1 & \text{if } k \in \{1, n\} \\ \frac{1}{k!} \sum_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n & \text{if } k \in \{2, 3, \ldots, n-1\} \\ 0 & \text{otherwise} \end{cases} \tag{1.2}$$

For the graph in Figures 1.1 and 1.2 there are $\left\{ {10 \atop 5} \right\} = 42,525$ ways of partitioning the 10 vertices into 5 colour classes. The total number of partitions of n objects is given by the Bell numbers, denoted B_n where

$$B_n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\} \tag{1.3}$$

For the graph in Figures 1.1 and 1.2 there are a total of $B_{10} = 115,975$ possible partitions of the 10 vertices into non-empty, non-overlapping subsets.

Of course, not every possible partition of V into non-empty, non-overlapping subsets need be considered explicitly in order to solve the GCP. One approach is to search all partitions of V into k subsets, starting with $k = 1$ and increasing k by 1 (i.e. $k \leftarrow k + 1$) until a feasible k-colouring is achieved. By implementing this approach, only as many as

$$\sum_{k=1}^{\chi(G)} \left\{ {n \atop k} \right\} \tag{1.4}$$

of the partitions need be considered. However, the Stirling numbers of the second kind still exhibit exponential growth for most values of k, which means that Equation (1.4) will also grow exponentially. This exponential growth demonstrates that a naive "exhaustive search" style algorithm is not suitable for the GCP.

In fact, the decision problem "Can the graph $G = (V, E)$ be feasible coloured using only k colours?" is known to be a NP-complete problem [Garey and Johnson, 1979], and identifying $\chi(G)$ is an NP-hard problem (i.e. it is at least as hard as the corresponding NP-complete decision problem). Due to the NP-hardness of the GCP, excessive amounts of time and computational effort can be required to exactly solve graphs, especially as they grow in size (i.e. as $|V|$ increases). Therefore, in practice, it is usually preferable to use heuristic methods to quickly generate "good" approximations for $\chi(G)$ and identify feasible k-colourings, where $k \geq \chi(G)$, for a given graph G. This will be explored in much more detail in Chapter 2.

Known Bounds for $\chi(G)$

If a given graph $G = (V, E)$ has a particular structure then $\chi(G)$ is known in some cases. For example, $\chi(G) = 1$ if and only if G is a null graph (i.e. $E = \emptyset$), $\chi(G) = 2$ if and only if G is a bipartite graph, and $\chi(G) = |V|$ if and only if G is a complete graph (i.e. $G = K_{|V|}$).

Although calculating $\chi(G)$ for a given graph G is NP-hard, some bounds for the chromatic number are known regardless of the structure of G.

- $\chi(G) \geqslant \omega(G)$
- $\chi(G) \geqslant \lceil \frac{n}{\alpha(G)} \rceil$ where $n = |V|$

- $\chi(G) \geqslant \frac{n^2}{n^2-2m}$ where $m = |E|$ [Geller and Schmeichel, 1971]
- $\chi(G) \leqslant \max_{v \in V}\{\deg(v)\} + 1$ [Brooks, 1941]

If G is connected but not complete, and $\max_{v \in V}\{\deg(v)\} \geqslant 3$ then

- $\chi(G) \leqslant \max_{v \in V}\{\deg(v)\}$ [Brooks, 1941]

If there is a feasible k-colouring for a graph H, and there is a homomorphism from G to H (i.e. $G \to H \to K_k$), then

- $\chi(G) \leq \chi(H) \leq k$ [Hell and Nešetril, 2004]

The first two of these bounds follow from the fact that each vertex within a clique must be assigned to a different colour class in order to avoid clashes, and each colour class must also be an independent set. They are both interesting in theory but are often difficult to use in reality, because calculating the clique number $\omega(G)$ or the independence number $\alpha(G)$ for a given graph G are also NP-hard problems [Garey and Johnson, 1979]. The same can also be concluded with regards to the final of these bounds; since calculating the chromatic number of a different graph H is also an NP-hard problem.

1.4 Applications of the GCP

We can reformulate many real-world problems as GCPs by considering the different aspects of a given problem instance and how they might relate to the components of a graph (vertices, edges and colours). Therefore the methods developed and the insights gained from studying the GCP can be utilised in a much broader sense.

To demonstrate this, we now review two explicit examples of real-world problems that can be reformulated as GCPs: exam timetabling, and frequency assignment.

Exam Timetabling

Exam timetabling [Erben, 2001; Qu et al., 2009] is one of the most commonly cited examples of where GCP techniques can be applied. Consider n exams to be timetabled and, for each pair of exams x and y, it is known how many students must sit both. The objective here is to assign exams to timeslots such that a student is not required to sit more than one exam per timeslot (i.e. the timetable has no clashes).

To reformulate this problem as a GCP, let each exam x be represented by a vertex v_x and, if at least one student must sit both exams x and y, define an edge $\{v_x, v_y\}$. A vertex set V, with $|V| = n$, and an edge set E have now been defined such that the graph $G = (V, E)$ has also been defined. By letting colours represent

Table 1.2: Example list of potential clashes within an exam timetabling problem.

	Bio	Chem	Phys	Maths	Psych	Stats	ComSci
Bio	-	47	32	40	23	0	0
Chem	47	-	36	39	26	0	0
Phys	32	36	-	52	0	0	16
Maths	40	39	52	-	0	31	24
Psych	23	26	0	0	-	28	0
Stats	0	0	0	31	28	-	0
ComSci	0	0	16	24	0	0	-

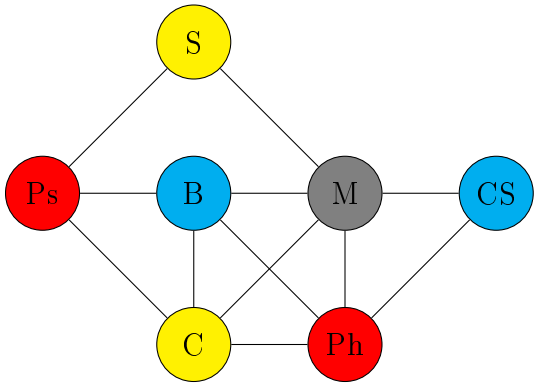

Figure 1.3: Graphical representation and feasible 4-colouring of the exam timetabling problem presented in Table 1.2.

timeslots, GCP methods can now be applied to the newly defined graph G in order to produce a clash-free exam timetable with approximately minimal timeslots.

Table 1.2 is an example of the information required to transform an exam timetabling problem for 7 exams (biology, chemistry, physics, mathematics, psychology, statistics, and computer science) into a GCP. Notice that only the pairs of vertices representing exams in which some students are required to sit both exams are connected by edges in Figure 1.3. In fact, Table 1.2 can easily be transformed into an adjacency matrix by substituting any entry which is larger than zero with a one. By applying GCP methods to the graphical representation of the problem in Figure 1.3, a feasible 4-colouring has been found. This colouring translates to a timetable in which all 7 exams are scheduled over 4 timeslots, such that exams (i. e. vertices) in the same colour class are held simultaneously (e. g. exams for biology and computer science are scheduled in the same timeslot).

Of course, this approach to solving the exam timetabling problem does not take into account any secondary objectives (e. g. room sizes, exam lengths and resting period constraints) but it does guarantee to find a feasible exam timetable under the assumption that there is no upper bound on the number of timeslots (i. e. colours).

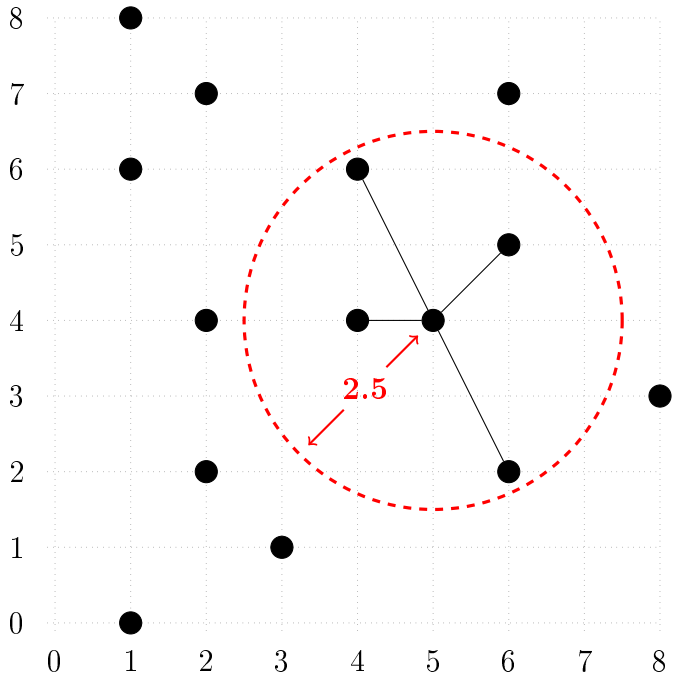

Figure 1.4: Example object locations for a frequency assignment problem. Area within which interference may occur with the object at (5,4) is represented by the area within the red circle.

Frequency Assignment

The frequency assignment problem [Aardal et al., 2007] is another common example of where GCP methods can be used. Here, the objective is to assign communication frequencies (e. g. radio wavelengths) to a set of n physical locations such that there is no interference. If two physical locations are within a certain proximity of one another then they cannot be assigned the same frequency.

We can again apply GCP methods to tackle this problem by considering each physical location x as a vertex, defining an edge $\{v_x, v_y\}$ for each pair of locations x and y that are within a given proximity of one another, and considering colours as communication frequencies. For example, consider the two-dimensional grid of object locations illustrated in Figure 1.4. If interference can occur between objects that are within 2.5 units of one another, then the corresponding graph would include edges from the vertex representing the object at coordinates (5,4) to the vertices representing the objects at coordinates $(4, 4), (4, 6), (6, 2)$ and (6,5). By including these edges, we can ensure that the corresponding vertices are assigned to different colour classes which will then translate to a frequency assignment with no interference between the respective objects.

The full graphical representation of the problem presented in Figure 1.4 can be seen in Figure 1.5, as well as a feasible 4-colouring for it. This feasible 4-colouring can easily be translated into a feasible frequency assignment in which only 4 unique communication frequencies are required. The objects (i. e. vertices) in each colour

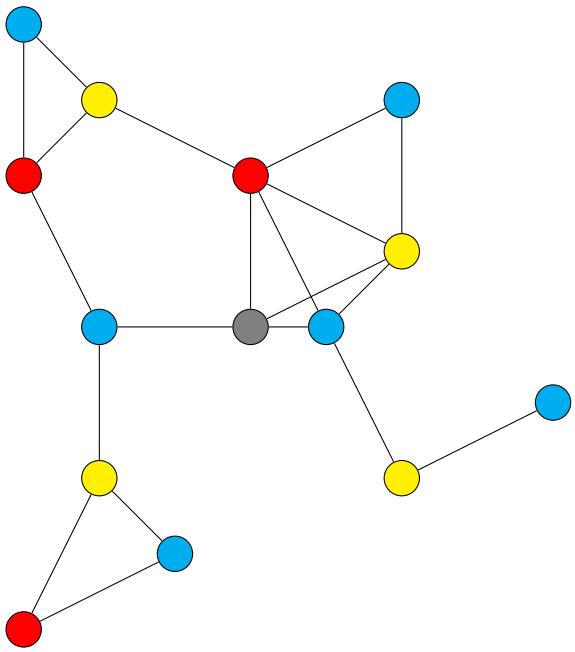

Figure 1.5: Graphical representation and feasible 4-colouring for the frequency assignment problem presented in Figure 1.4.

class can be assigned the same communication frequency without causing interference within the network (e. g. the objects at coordinates $(1, 0), (1, 6)$ and (4,6) can be assigned the same communication frequency).

In a similar fashion to exam timetabling, this approach does not take into account any of the additional constraints of frequency assignment but it does guarantee a feasible solution under the assumption that there is no upper bound on the number of available communication frequencies (which is often untrue in practice).

The dynamic frequency assignment problem [Dupont et al., 2009] extends upon the standard problem to consider the situation where the physical locations change over time (either by increasing/decreasing in number or by moving). This causes some difficulties when attempting to apply GCP methods as we can no longer define a static graph to represent this problem. This example is part of the motivation for studying the colouring of dynamic graphs, which will be discussed in Chapters 3 and 5.

Further Examples of Applicable Real-World Problems

There are many more examples of real-world problems that can be reformulated as GCPs, including: register allocation [Chaitin, 1982], tournament scheduling [Costa, 1995], designing seating plans [Lewis and Carroll, 2016], and grouping individuals within social networks [Tantipathananandh et al., 2007]. However, it is beyond the scope of this section to cover all of these examples in great detail.

1.5 Chapter Summary

In this chapter we have outlined the aims of this thesis and introduced the reader to the static graph colour problem (GCP). The complexity of the GCP was then discussed in order to justify the use of heuristic methods for solving GCPs as opposed to exact methods. Finally, examples were provided of real-world problems that can be transformed into GCPs so as to highlight the importance of studying the GCP in terms of its applications.

In the next chapter we will discuss several heuristic approaches, taken from the literature, for tackling the static GCP.

Chapter 2

Heuristics for the Static Graph Colouring Problem

As mentioned in Section 1.3, due to the NP-hard nature of the GCP, it is generally preferable to use heuristic methods to generate feasible k-colourings for a given graph G such that $k \geq \chi(G)$ is a "good" approximation of $\chi(G)$. In this chapter we will discuss and compare several different heuristic approaches for tackling the static GCP. The solution spaces of these heuristics will also be introduced along with examples of appropriate objective functions and neighbourhood move operators to be used within each space.

Although all of the methods discussed in this chapter are heuristic in nature, exact methods for solving the GCP do exist. Exact methods usually come in the form of backtracking algorithms [Korman, 1979; Kubale and Jackowski, 1985] or integer programming models [Hansen et al., 2009; Malaguti et al., 2011; Mehrotra and Trick, 1996]. The advantage of exact methods is that they will always optimally calculate $\chi(G)$ for a given graph G and in most cases they also identify a feasible $\chi(G)$-colouring for G. However, these advantages are mitigated by the excessive computational effort, and therefore time, requirements of these methods to run in comparison to heuristic methods. That being said, exact methods are sometimes included within a heuristic approach in order to optimise sub-problems of a GCP, such as XRLF presented by Johnson et al. [1991] and MMT presented by Malaguti et al. [2008].

What about approximation algorithms? Unlike heuristic approaches, approximation algorithms have provable bounds on the quality of the solutions they produce compared to optimal. If A is a graph colouring algorithm, let $A(G)$ be the number of colours used when A is applied to a given graph G. Garey and Johnson [1976] have shown that if a polynomial-time algorithm A exists for the GCP that can guarantee $A(G) \leq r \cdot \chi(G) + d$ for some constants $r < 2$ and d, then there also exists a polynomial-time algorithm A' that guarantees $A'(G) = \chi(G)$. However, the GCP is known to be NP-complete. Therefore, unless P = NP, there are no approximation

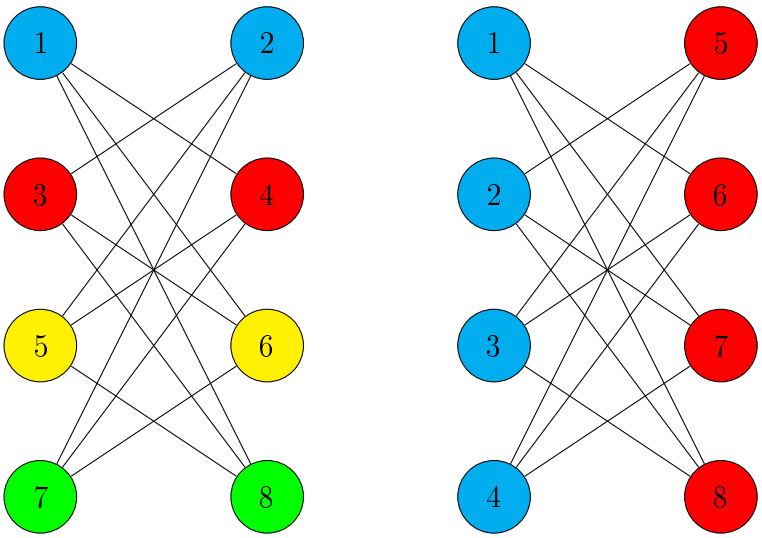

Figure 2.1: Example colourings produced by the GREEDY constructive algorithm with numbers representing the order in which vertices were assigned to colour classes.

algorithms for the GCP with an approximation ratio less than two (i.e. $\nexists A$ such that $A(G) < 2 \cdot \chi(G)$).

2.1 Constructive Algorithms

A constructive (or sequential) algorithm for the GCP builds a colouring for a given graph $G = (V, E)$ by assigning each vertex $v \in V$, in turn, to a colour class. The advantage of constructive algorithms is that they always produce a feasible k-colouring $\mathcal{S}$ for G, and therefore an upper bound for $\chi(G)$. Unfortunately, there is no generalised way to ensure that the upper bound $k = |\mathcal{S}|$ is "close" to the actual value of $\chi(G)$. It should be noted that constructive algorithms do not act in the manner that will be outlined in Section 2.3, however they are often incorporated within that approach (i.e. to find an initial feasible colouring).

The GREEDY (or FIRST-FIT) algorithm [Welsh and Powell, 1967] takes some ordering of the vertices (e.g. random, largest degree first, etc.) and assigns each vertex in turn to the first colour class such that no clashes are incurred. For a graph $G = (V, E)$, there are at least

$$\chi(G)! \prod_{i=1}^{\chi(G)} |S_i|! \tag{2.1}$$

different orderings of the $|V|$ vertices that when passed to the GREEDY algorithm will result in an optimal colouring $\mathcal{S} = \{S_1, \ldots, S_{\chi(G)}\}$ for G [Lewis, 2009]. Of course, not every ordering of the vertices in V will result in an optimal colouring, as illustrated in Figure 2.1. This figure clearly demonstrate how the ordering of the vertices can cause the GREEDY algorithm to produce optimal (RHS) or sub-optimal (LHS) colourings for a given graph.

Colourings with fewer colour classes (see Table 2.1) can often be obtained by dynamically altering the order in which vertices are coloured during the GREEDY algorithm such that "difficult" vertices are coloured earlier. The DSATUR algorithm [Brélaz, 1979] does this by altering the ordering of vertices such that the vertices with the highest saturation degrees are coloured first.

Definition 2.1. Given a partial k-colouring $\mathcal{S} = \{S_1, \ldots, S_k\}$, the *saturation degree* $\deg_{sat}(v, \mathcal{S})$ of an "uncoloured" vertex $v \in V \backslash \{\bigcup_{i=1}^{k} S_i\}$ with respect to $\mathcal{S}$, is equal to the number of colour classes in $\mathcal{S}$ for which there exists at least one vertex $u \in S_i$ such that u is adjacent to v (i.e. $\{u, v\} \in E$). Therefore, $0 \leq \deg_{sat}(v, \mathcal{S}) \leq k$ for all uncoloured vertices $v \in V \backslash \{\bigcup_{i=1}^{k} S_i\}$ with respect to the partial k-colouring $\mathcal{S}$.

A different constructive approach is to build up each colour class (i.e. independent set) one by one such that a new colour class is not considered until no more vertices can be feasibly assigned to the current one. The RLF (recursive largest first) algorithm [Leighton, 1979] is one such method, which successively builds colour classes such that, in some sense, the remaining uncoloured vertices are "[colourable] in as few [colour classes] as possible."

To give an indication of how these constructive algorithms perform against one another, the GREEDY algorithm with random vertex ordering, the DSATUR algorithm and the RLF algorithm have been reproduced and implemented on a number of test instances for comparison[1]. For specific details of how these three algorithms operate, see Appendix A.1. The test instances consist of random graphs with $n \in \{250, 500, 1000\}$ vertices and density $d \in \{0.1, 0.5, 0.9\}$ such that an edge exists between each pair of vertices with probability d. For each pair of graph parameters, n and d, 20 random graphs were produced.

The results of these implementations are presented in Table 2.1. The first two columns of this table indicates the test instance parameters, and the remaining columns show the mean number of colour classes in the colourings produced by each algorithm and the mean time (in seconds) required to produce them.

It can be seen from Table 2.1 that RLF significantly[2] outperforms DSATUR, with regards to the number of colour classes in the colourings achieved, in most cases. DSATUR outperforms the GREEDY algorithm in the same respect. On the other hand, the relationship between the algorithms is reversed with regards to the time required to produce a colouring. These results are similar to those presented in the literature e.g. [Lewis, 2015], and suggest a clear inverse relationship between the number of colour classes in a colouring and the time required to produce it.

[1] All algorithms in this chapter were programmed in C++ and executed on a 2.9 GHZ Windows 7 PC with an Intel Core i5-4570T processor and 8 GB RAM

[2] Unless stated otherwise, all pair-wise comparisons are based on the Wilcoxon signed rank test with significance level $\alpha = 0.01$. This is due to the non-normality of the results data.

Table 2.1: Mean number of colour classes (k) in the colourings produced by constructive algorithms and the mean time (in seconds) required to produce them.

		Greedy		DSatur		RLF	
n	d	k	time	k	time	k	time
250	0.1	12.50	$0^{\dagger}$	9.80	0.001	9.50	0.001
	0.5	42.90	$0^{\dagger}$	37.05	0.002	32.25*	0.003
	0.9	97.70	$0^{\dagger}$	90.25	0.004	85.90*	0.003
500	0.1	19.55	$0^{\dagger}$	16.00	0.002	14.95*	0.002
	0.5	73.00	$0^{\dagger}$	64.95	0.009	60.90*	0.019
	0.9	177.40	$0^{\dagger}$	163.20	0.019	156.40*	0.024
1000	0.1	31.55	$0^{\dagger}$	26.25	0.009	24.45*	0.015
	0.5	126.90	$0.001^{\dagger}$	115.60	0.046	108.75*	0.146
	0.9	320.75	$0.001^{\dagger}$	302.75	0.111	286.15*	0.197

0 represents a time less than 10^{-3} seconds.
* indicates colourings with significantly fewer colour classes than those achieved by all the other constructive algorithms for the same values of n and d.
† indicates a run-time that is significantly less compared to all the other constructive algorithms for the same values of n and d.

2.2 Solution Spaces

Before discussing more complex heuristics for tackling the static GCP we first introduce some of the solution (or search) spaces employed within these methods. When considering heuristic methods for the GCP, Hertz et al. [2008] and Lewis [2015] suggest three main solution spaces:

1. feasible only;
2. complete, improper; and
3. partial, proper.

The number of colour classes k is generally fixed when operating in the latter two solution spaces, though this value is often reduced once a feasible (i. e. zero cost) k-colouring has been obtained (see Section 2.3 for more details).

We will now look at each of these solution spaces in more detail and highlight some of the common objective functions and neighbourhood move operators[3] used within them.

2.2.1 Feasible Only Solution Space

In the feasible only solution space, only feasible colourings are considered (see Definitions 1.1 to 1.3). Recall, a colouring $\mathcal{S} = \{S_1, \ldots, S_k\}$ is considered feasible for a graph $G = (V, E)$ if every vertex is assigned to a colour class (i. e. $V = \bigcup_{i=0}^{k} S_i$)

[3]More specifically, we will only introduce the neighbourhood move operators that will be utilised within our algorithms in Chapters 3 to 6 of this thesis.

and every pair of vertices in a given colour class are non-adjacent (i. e. if $u \in S_i$ and $v \in S_i$ then $\{u, v\} \notin E$). Unlike the complete, improper and partial proper solution spaces, k is often not fixed when operating within this solution space.

When operating in the feasible only solution space, it can be difficult to determine which of two feasible k-colourings is "closer" to becoming a feasible colouring with $k-1$ colour classes. One objective function that has been considered by Morgenstern and Shapiro [1990] is

$$f(\mathcal{S}) = -\sum_{i=0}^{k} |S_i|^2 \tag{2.2}$$

which, if minimised, encourages reduction of the number of vertices in colour classes until they are eventually emptied. Once a colour class has been emptied, it is removed from $\mathcal{S}$ (i. e. if $S_i \in \mathcal{S}$ and $S_i = \emptyset$ then $\mathcal{S} \leftarrow \mathcal{S} \backslash S_i$). An alternative objective function for this solution space, suggested by Erben [2001], is

$$f(\mathcal{S}) = \frac{\sum_{i=1}^{|\mathcal{S}|} (\sum_{v \in S_i} \deg(v))^2}{|\mathcal{S}|} \tag{2.3}$$

which, if maximised, encourages vertices with high degrees to be assigned to the same colour classes, therefore increasing the likelihood of other colour classes becoming empty.

As we will see in Chapters 4 and 6, this solution space is perhaps most useful when considering a secondary objective. This is because the likes of Equations (2.2) and (2.3) can be replaced with a secondary objective function which can then be optimised without violating the constraints of the GCP.

Obviously, all neighbourhood move operators for this solution space must maintain the feasibility of a colouring at all times. We now consider two such neighbourhood move operators, the Kempe-chain interchange and the pair-swap operator.

Kempe-chain Interchange

Given a feasible k-colouring $\mathcal{S}$ for $G = (V, E)$, a vertex $v \in V$ and a colour class S_j such that $j \neq c(v)$, the Kempe-chain from $S_{c(v)}$ to S_j that contains v is the maximal connected subset of V which contains v and whose vertices are either in $S_{c(v)}$ or S_j. This is denoted by $\textsc{Kempe}(v, c(v), j)$. The Kempe-chain interchange takes the vertices in $\textsc{Kempe}(v, c(v), j)$ and transfers those in colour class $S_{c(v)}$ to colour class S_j and vice versa. It should be noted that if $|\textsc{Kempe}(v, c(v), j)| = |S_{c(v)}| + |S_j|$ then the Kempe-chain interchange is simply a re-labelling of the colour classes.

It is shown in [Lewis, 2015] that if $\mathcal{S}$ is feasible for G, then applying the Kempe-chain interchange to $\mathcal{S}$ cannot result in an infeasible colouring for G. Moreover, applying the Kempe-chain interchange to $\mathcal{S}$ will not increase the number of colour classes, and may even reduce the number. Therefore, the Kempe-chain interchange is an ideal neighbourhood move operator for the feasible only solution space.

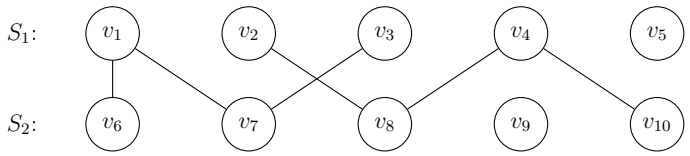

Figure 2.2: A feasible 2-colouring with the vertices in colour class S_1 displayed in the row above those in colour class S_2.

To illustrate, the bipartite graph induced by colour classes S_1 and S_2 in Figure 2.2 contains the following unique Kempe-chains: $\{v_1, v_3, v_6, v_7\}, \{v_2, v_4, v_8, v_{10}\}, \{v_5\}$ and $\{v_9\}$. Applying the Kempe-chain interchange to $\textsc{Kempe}(v_1, 1, 2) = \{v_1, v_3, v_6, v_7\}$ transfers vertices v_1 and v_3 from S_1 to S_2 and vertices v_6 and v_7 from S_2 to S_1. Note that although there are $(k-1) \times n = 10$ distinct Kempe-chain labels for this example, these labels only map onto four unique sets of vertices. For example, $\textsc{Kempe}(v_1, 1, 2)$ also maps to $\textsc{Kempe}(v_3, 1, 2)$, $\textsc{Kempe}(v_6, 2, 1)$ or $\textsc{Kempe}(v_7, 2, 1)$.

Kempe-chains can be identified via breadth first search or depth first search, using $v \in V$ as a starting point and the graph induced by the two colour classes $S_{c(v)}$ and S_j. We have arbitrarily chosen to use breadth first search throughout this thesis. The Kempe-chains for a given graph $G = (V, E)$ and a feasible k-colouring $\mathcal{S}$ for G can be stored in a $|V| \times k$ matrix where the (v, j)th entry is the Kempe-chain $\textsc{Kempe}(v, c(v), j)$. As different Kempe-chain labels can relate to the same sets of vertices, computational saving can be achieved when identifying the entries of this matrix. For example, consider $\textsc{Kempe}(v_1, 1, 2) = \{v_1, v_3, v_6, v_7\}$ in Figure 2.2. Once this has been identified as the $(v_1, 2)$th entry of our matrix, it can then also be assigned to the $(v_3, 2)$th, $(v_6, 1)$th and $(v_7, 1)$th entries. Of course, the $(v, c(v))$th entry of this matrix will be the empty set $\emptyset$ for all $v \in V$.

Pair-swap Operator

An additional move operator for the feasible only solution space is the pair-swap operator. This transfers a single vertex $v \in V$ to a colour class S_j such that $j \neq c(v)$ and no clashes are incurred whilst simultaneously moving a vertex $u \in S_j$ to $S_{c(v)}$ such that, again, no clashes are incurred. There is only one available pair-swap for the graph in Figure 2.2, which transfers vertex v_5 from S_1 to S_2 and v_9 from S_2 to S_1.

It should be noted that a pair-swap is the simultaneous application of two Kempe-chain interchanges applied to $\textsc{Kempe}(v, c(v), c(u))$ and $\textsc{Kempe}(u, c(u), c(v))$ such that $|\textsc{Kempe}(v, c(v), c(u))| = |\textsc{Kempe}(u, c(u), c(v))| = 1$. By observing this relationship to Kempe-chain interchanges, it follows that the pair-swap does not alter the feasibility of an already feasible colouring either. This also means that pair-swaps can be easily identified from the $|V| \times k$ matrix of Kempe-chains for a given graph $G = (V, E)$ and a feasible k-colouring $\mathcal{S}$ for G. If the (v, j)th entry of the

matrix satisfies $|\textsc{Kempe}(v, c(v), j)| = 1$, then we can simply search the $(u, c(v))$th entries of the matrix, where $u \in S_j$, for corresponding Kempe-chains that satisfy $|\textsc{Kempe}(u, j, c(v))| = 1$.

2.2.2 Complete, Improper Solution Space

We now consider our second solution space, that of complete, improper colourings. This solution space is probably the most popular solution space for GCP heuristics having been utilised by Avanthay et al. [2003]; Costa et al. [1995]; Chams et al. [1987]; Dorne and Hao [1998]; Fleurent and Ferland [1996]; Galinier and Hao [1999]; Galinier et al. [2004]; Glass and Prügel-Bennett [2003]; Hertz and de Werra [1987]; and Johnson et al. [1991]. This space consists of every complete colouring (see Definition 1.1) regardless of whether clashes are incurred. Therefore, given a k-colouring $\mathcal{S}$, an appropriate objective function for this solution space might simply be

$$f(\mathcal{S}) = \sum_{i=1}^{k} |E \cap \{S_i \times S_i\}| \tag{2.4}$$

which is equivalent to the number of clashes in the colouring. Indeed, this particular objective function is used in most of the works cited above. If $f(\mathcal{S}) = 0$, then $\mathcal{S}$ is a feasible k-colouring.

Neighbourhood move operators for this solution space must ensure that all vertices remain coloured (i.e. a resultant colouring $\mathcal{S}' = \{S'_1, \ldots, S'_k\}$ must satisfy $V = \bigcup_{i=0}^{k} S'_i$).

1-move Operator

A popular neighbourhood move operator for transforming a colouring $\mathcal{S}$ into a colouring $\mathcal{S}'$ is the 1-move operator. The 1-move transfers a vertex v from its current colour class $S_{c(v)}$ to a different colour class S_j such that $j \neq c(v)$. Examples of work that utilise this move operator can be found in [Chams et al., 1987; Hertz and de Werra, 1987; Johnson et al., 1991].

An example of the 1-move operator is illustrated in Figure 2.3. On the LHS we have a 5-colouring $\mathcal{S}$ for G with two clashes between vertex pairs $\{v_7, v_9\}$ and $\{v_8, v_{10}\}$. By moving v_8 from S_3 to S_5 such that $S'_3 = S_3 \backslash \{v_8\}$ and $S'_5 = S_5 \cup \{v_8\}$ we obtain the 5-colouring $\mathcal{S}'$ on the RHS. In this case, the 1-move has removed one clash from the colouring without introducing any new ones, therefore bringing us "closer" to a feasible 5-colouring for G. Note that, in this example, $S_i = S'_i$ for $i \in \{1, 2, 4\}$.

Many previous works, including [Galinier and Hao, 1999], consider the *critical* 1-move operator, which modifies the 1-move operator slightly such that the vertex v to be transferred can only be selected if it is currently clashing with an adjacent vertex (i.e. $v \in S_i$ can be transferred if and only if $\exists u \in S_i$ such that $u \neq v$

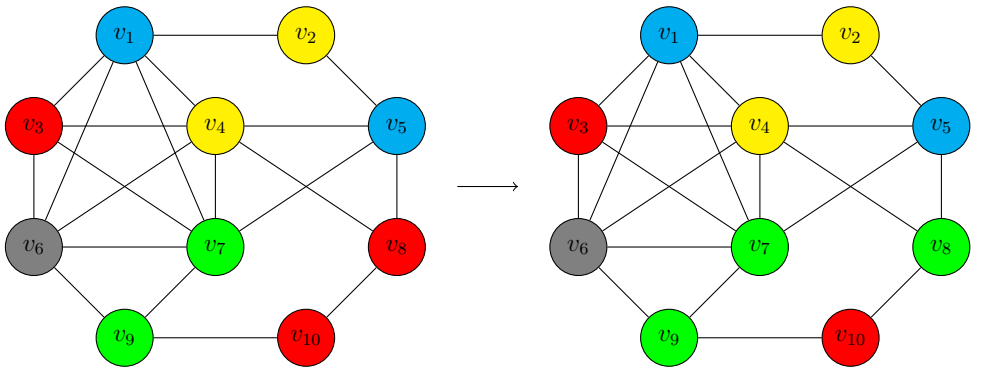

Figure 2.3: 1-move example in the complete, improper solution space for the graph shown in Figure 1.1.

and $\{u, v\} \in E$). In the example illustrated in Figure 2.3, only 1-moves involving v_7, v_8, v_9 or v_{10} could be considered for the graph on the LHS, and only the 1-moves involving v_7 or v_9 could be considered for the graph on the RHS.

2.2.3 Partial, Proper Solution Space

The partial, proper solution space consists of every proper colouring of a graph (see Definition 1.2) although not every vertex is necessarily assigned to a colour class. When considering a colouring $\mathcal{S} = \{S_1, \ldots, S_k\}$ in this solution space, we define the set of "uncoloured" vertices as $U = V \setminus \bigcup_{i=0}^{k} S_i$. In this solution space, one suitable objective function, utilised by Blöchliger and Zufferey [2008], is

$$f(\mathcal{S}) = |U| = |V| - \sum_{i=1}^{k} |S_i| \tag{2.5}$$

which is simply the number of "uncoloured" vertices. An alternative objective function, considered by Morgenstern [1996], is

$$f(\mathcal{S}) = \sum_{v \in U} \deg(v) \tag{2.6}$$

where $\deg(v)$ is the degree of vertex v. If an "uncoloured" vertex is adjacent to fewer vertices then, in practice, it will often be easier to transfer it into a colour class without introducing clashes. As with Equation (2.4), $f(\mathcal{S}) = 0$ indicates that $\mathcal{S}$ is a feasible k-colouring.

Neighbourhood move operators for this solution space must ensure that the resultant colouring also contains no clashes (i. e. the colouring function $c' : V \to \{1, \ldots, k\}$ of the resultant k-colouring $\mathcal{S}'$ must satisfy $c'(u) \neq c'(v)$ if $\{u, v\} \in E$).

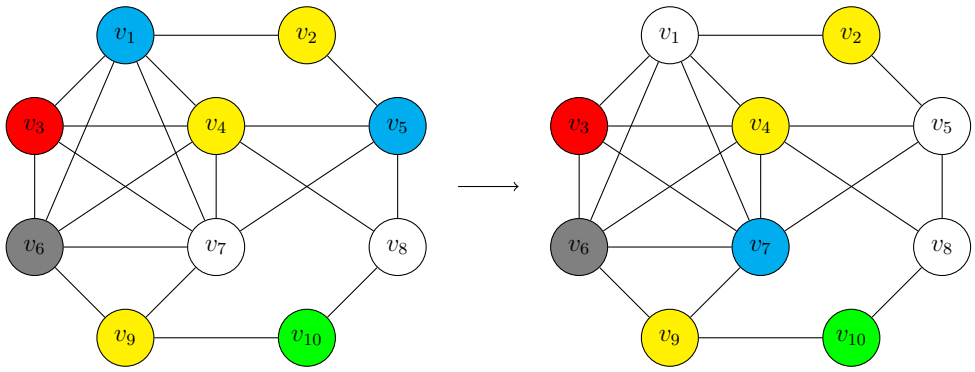

Figure 2.4: i-swap example in the partial, proper solution space for the graph shown in Figure 1.1.

i-swap Operator

A popular neighbourhood move operator for the partial, proper solution space is the i-swap operator. To transform a colouring $\mathcal{S}$ into a colouring $\mathcal{S}'$, the i-swap transfers an "uncoloured" vertex $v \in U$ to a colour class $S_i \in \mathcal{S}$. In order to ensure that $\mathcal{S}'$ remains proper, all vertices in S_i that are adjacent to v are transferred to the set of "uncoloured" vertices U (i. e. if $u \in S_i$ and $\{u, v\} \in E$ then $U \leftarrow U \cup \{u\}$). Examples of work that utilise this move operator can be found in [Blöchliger and Zufferey, 2008; Morgenstern, 1996]. Note that, by only considering the "uncoloured" vertices, the i-swap operator concentrates on the "problematic" parts of a colouring, in a similar fashion to the *critical* 1-move operator for the complete, improper solution space.

An example of the i-swap operator is illustrated in Figure 2.4. On the LHS we have a 5-colouring $\mathcal{S}$ for G with two "uncoloured" vertices $U = \{v_7, v_8\}$. By moving v_7 from U to S_1, two clashes are incurred between the vertex pairs $\{v_1, v_7\}$ and $\{v_5, v_7\}$. By defining $S'_1 = (S_1 \cup \{v_7\}) \backslash \{v_1, v_5\}$ and $U' = (U \backslash \{v_7\}) \cup \{v_1, v_5\}$ we ensure that the resultant 5-colouring $\mathcal{S}'$ on the RHS remains proper. This particular i-swap has increased the number of "uncoloured" vertices, therefore moving us "further" from a feasible colouring for G. Note that, in this example, $S_i = S'_i$ for $i = 2, \ldots, 5$.

2.3 General Static GCP Approach

In this section we review a common approach for tackling the static GCP via heuristic methods. The approach in question aims to solve a series of k-GCPs for a graph G such that k is decreasing. The goal of a k-GCP is to determine whether G can be feasibly coloured using k colours. This particular approach was used by Lewis et al. [2012] in their comprehensive comparison of six high-performing GCP algorithms.

In practice, an initial value for k might be determined using a constructive operator which is guaranteed to produce a feasible colouring for G, such as the algorithms described in Section 2.1. Once an initial, feasible value of k has been identified, an attempt can be made to find a feasible colouring with $k-1$ colour classes. If this is successful then an attempt to find a feasible colouring with one fewer colour class again can be made, and so on, until some stopping criteria (e. g. a time or computational limit) is reached. This approach is outlined in Algorithm 2.1.

Algorithm 2.1 Generic Static GCP Algorithm

Input: a graph G
Output: a feasible colouring $\mathcal{S}$ for G

```
1: S produced by a constructive operator
2: S_best ← S
3: k ← |S| − 1
4: while not stopping criterion do
5:    attempt to make S a feasible k-colouring for G_{t+1}
6:    if S is a feasible k-colouring for G_{t+1} then
7:       S_best ← S
8:       k ← k − 1
9: S ← S_best
10: return S
```

As mentioned in Section 2.2, algorithms designed to work in the complete, improper or partial, proper solution spaces generally operate with a fixed number of colour classes k. Because of this, these algorithms are only "good" at tackling the GCP for a graph G if the value of k is also a "good" estimate of $\chi(G)$. This approach allows such algorithms to be utilised for tackling k-GCPs whilst systematically reducing k in order to determine "better and better" estimates for $\chi(G)$. In the best case scenario, $\chi(G)$ might even be identified (i. e. if the k-GCP for G can be solved but the $(k-1)$-GCP cannot[4], then $\chi(G) = k$).

2.4 Local Search Heuristics

In the following sections we will discuss several different heuristic approaches for tackling the static GCP that do not follow the basic constructive approach of the methods discussed in Section 2.1. Firstly, we will discuss local search, a common approach for solving combinatorial optimisation problems, such as the GCP.

Local search heuristics apply small perturbations to a single solution within a neighbourhood of solutions in an attempt to improve it in some way. With regards to the GCP, this means working within one of the solution spaces described in Section 2.2 and applying an appropriate neighbourhood move operator to a single colouring in an attempt to identify a feasible colouring.

[4]Remember, solving the k-GCP for a given graph G is NP-hard (see Section 1.3). Therefore, proving that G is *not* k-colourable is also NP-hard.

Simulated annealing, which mimics the process of annealing in metallurgy, is a popular local search method first proposed for combinatorial optimisation problems by Kirkpatrick et al. [1983]. Simulated annealing is an extension of random descent, which randomly generates a neighbourhood move and accepts the move if it improves the objective function. The difference between random descent and simulated annealing is that simulated annealing also probabilistically accepts moves that do *not* improve the objective function. Johnson et al. [1991] propose three different simulated annealing algorithms for the GCP, one of which operates in the feasible only solution space and two that operate in the complete, improper solution space. Of these three algorithms, none was shown to clearly dominate the others. Two further simulated annealing algorithms for the GCP were proposed by Chams et al. [1987], and Morgenstern and Shapiro [1990].

Tabu search is another popular local search method first proposed by Glover [1989, 1990]. Tabu search chooses the "best" move at each iteration from all available non-tabu moves regardless of its affect on the objective function. A move that is currently considered "tabu" cannot be chosen in the current iteration, unless an *aspiration criterion* is met. The tabu list, which stores all currently "tabu" moves, is usually updated during execution so as to decrease the chances of the algorithm getting "stuck" at a local optimum. For the GCP, there are two popular tabu search algorithms: TabuCol [Hertz and de Werra, 1987] and PartialCol [Blöchliger and Zufferey, 2008] (see Sections 2.4.1 and 2.4.2 respectively for detailed descriptions). Both of these algorithms have proved to be very competitive against other local search methods, and even against more sophisticated methods in some cases[5].

Pseudocode and general details regarding both simulated annealing and tabu search can be found in Appendix A.2.

Lewis [2009] proposes a hill-climbing algorithm for the GCP which operates in the feasible only solution space. This algorithm uses a combination of the Greedy constructive algorithm, a form of independent set extraction (see Section 2.5), and local search that utilises Kempe-chain interchanges and pair-swaps. This algorithm's objective function is simply $|\mathcal{S}|$ (i. e. the number of colour classes in the feasible colouring $\mathcal{S}$). The number of colour classes cannot increase during execution, hence the objective function does not increase during execution either, thus providing its "hill-climbing" characteristics.

All of the local search methods described so far operate in a single solution space and often only employ a single neighbourhood move operator (with the exception of the hill-climbing algorithm which uses two). On the other hand, variable neighbour search (VNS), first introduced by Mladenović and Hansen [1997], is a general purpose metaheuristic that utilises several neighbourhoods in order to explore the solution space more freely. Avanthay et al. [2003] have proposed a VNS approach for

[5]In general, algorithms for the static GCP tend to be compared against one another using a set of benchmark instances which can be found at http://mat.gsia.cmu.edu/COLOR/instances.html.

tackling the GCP which combines TabuCol (see Section 2.4.1) with twelve different neighbourhood move operators in the complete, improper solution space. This approach has been shown to be more efficient than using TabuCol alone, at least within the context of the experimental conditions of their paper.

A variable space search (VSS) approach for the GCP [Hertz et al., 2008] alternates between three different tabu search procedures operating in three different solution spaces, including TabuCol and PartialCol. The main idea behind this approach is the possibility that the local optimum in one solution space may not be a local optimum in another. Therefore, if a search becomes "stuck" in one solution space, jumping to another may allow more freedom and lead to further improvements. As with the VNS approach, this VSS approach has been shown to achieve "better" results than either TabuCol or PartialCol on their own, at least within the context of the paper's experimental conditions.

Other local search methods for the GCP exist, many of which are discussed in a survey by Galinier and Hertz [2006]. This survey covers some of the solution spaces, objective functions and neighbourhood move operators not discussed here. This survey also provides references for more sophisticated approaches (e. g. hybrid evolutionary algorithms) that incorporate local search methods, and also highlights why TabuCol, a relatively old algorithm for solving the GCP, is still popular today.

2.4.1 TabuCol

As mentioned above, one of the most popular local search heuristics for the GCP is TabuCol, a tabu search algorithm that operates in the complete, improper solution space. TabuCol was first introduced by Hertz and de Werra [1987] and then later improved by Galinier and Hao [1999]. One such improvement is that all *critical* 1-moves are considered in [Galinier and Hao, 1999], as opposed to just a random sample of *critical* 1-moves in [Hertz and de Werra, 1987].

As TabuCol is designed to operate in the complete, improper solution space, the objective function f is given by Equation (2.4) (i. e. the number of clashes). During each iteration, the "best" non-tabu *critical* 1-move is applied to the current k-colouring $\mathcal{S}$ to obtain the k-colouring $\mathcal{S}'$, regardless of whether $f(\mathcal{S}') < f(\mathcal{S})$ or not, and ties are broken randomly. The inverse 1-move (i. e. the 1-move that would transform $\mathcal{S}'$ back into $\mathcal{S}$) is then made "tabu" for a number of subsequent iterations. In fact, the tabu tenure varies dynamically, based on the objective function[6]. For the example illustrated in Figure 2.3, the inverse 1-move would be the one that transfers v_8 back into colour class S_3. The tabu list is stored as an $|V| \times k$ matrix where the (v, j)th entry contains the first iteration that the 1-move that transfers vertex v to colour class S_j is no longer tabu.

[6]The specific tabu tenure used by TabuCol and PartialCol is $0.6 \cdot f(\mathcal{S}') + r$ iterations, where f is Equation (2.4) or Equation (2.5), respectively, and r is a random integer from 0 to 9.

It is worth noting that a "tabu" move can be selected if it satisfies an *aspiration criterion*. The aspiration criterion for TabuCol is to accept a "tabu" move if the resultant colouring $\mathcal{S}'$ satisfies $f(\mathcal{S}') < f(\mathcal{S}_{best})$, where $\mathcal{S}_{best}$ is the "best" colouring observed up until that point. An even simpler aspiration criterion might be to accept a "tabu" move if the resultant colouring $\mathcal{S}'$ satisfies $f(\mathcal{S}') = 0$ (i. e. $\mathcal{S}'$ is a feasible colouring).

TabuCol has been incorporated within many hybrid evolutionary algorithms, including those proposed in [Dorne and Hao, 1998; Dowsland and Thompson, 2008; Galinier and Hao, 1999; Galinier et al., 2004; Lü and Hao, 2010]. Blöchliger and Zufferey [2008] have also presented results which suggest TabuCol can be very successful even when implemented on its own.

2.4.2 PartialCol

PartialCol proposed by Blöchliger and Zufferey [2008], is a tabu search algorithm for the static GCP that operates in the partial, proper solution space and acts in a similar fashion to TabuCol. As it is designed for the partial, proper solution space, PartialCol uses Equation (2.5) as its objective function f.

At each iteration, the "best" non-tabu i-swap is implemented to transform the current k-colouring $\mathcal{S}$ into the k-colouring $\mathcal{S}'$, with ties broken randomly. Again, as with TabuCol, this "best" move is implemented regardless of whether $f(\mathcal{S}') < f(\mathcal{S})$. The inverse i-swaps are then made "tabu" for the same tabu number of iterations as used in TabuCol (see Footnote 6). For the example illustrated in Figure 2.4, the inverse i-swaps would be those that transfer vertices v_1 or v_5 back into colour class S_1.

Blöchliger and Zufferey [2008] also propose Foo-PartialCol which dynamically changes the tabu tenure of the inverse i-swaps based on the fluctuation of the objective function. If the objective function is seen to fluctuate less and less between iterations then the tabu tenure is increased in an attempt to leave the region of the solution space that the algorithm has become "stuck" in. Similarly, if the objective function is seen to fluctuate more and more between iterations then the tabu tenure is decreased in order to reduce these fluctuations and therefore refocus the optimisation process. This method requires more parameter settings compared to the standard PartialCol.

To give a flavour of the relative performance of TabuCol and PartialCol, they have been reproduced and compared on a number of random test instances, the results of which can be seen in Table 2.2. The algorithms follow the process outlined in Algorithm 2.1 with initial colourings produced by DSatur and a time limit of 5 minutes (300 seconds) being applied. Once a feasible k-colouring $\mathcal{S} = \{S_1, \ldots, S_k\}$ is identified, it is transformed into a colouring with $k-1$ colour classes. When operating in the complete, improper space (i. e. for TabuCol) a feasible

Table 2.2: Number of colour classes in the "best" colourings achieved by TabuCol and PartialCol.

		TabuCol			PartialCol		
n	d	min	mean	max	min	mean	max
250	0.1	8	8.00	8	8	8.00	8
	0.5	28	28.05*	29	28	28.55	29
	0.9	72	72.95	74	72	73.20	75
500	0.1	13	13.00	13	12	12.70	13
	0.5	49	49.60*	50	50	50.00	50
	0.9	125	126.65*	127	126	127.80	129
1000	0.1	21	21.05	22	21	21.00	21
	0.5	90	91.10	92	90	90.35*	91
	0.9	227	228.00*	230	227	229.35	231

* indicates colourings with significantly fewer colour classes than the those achieved by the other tabu search algorithm for the same values of n and d.

k-colouring is transformed as follows: a random colour class $S_i \in \mathcal{S}$ is removed such that $\mathcal{S} \leftarrow \mathcal{S} \backslash S_i$, then each of the vertices $v \in S_i$ are randomly assigned to one of the remaining $k-1$ colour classes. When operating in the partial, proper solution space (i. e. for PartialCol) a feasible k-colouring is transformed in a similar manner except now the removed colour class $S_i \in \mathcal{S}$ is considered as the set of "uncoloured" vertices such that $U \leftarrow S_i$. The test instances used here are the same as those used in Section 2.1 for comparing the constructive algorithms.

As can be seen in Table 2.2, there does not appear to be any overall dominance of one algorithm over the other as they have both been shown to outperform one another, with regards to the number of colour classes in the "best" colourings achieved, in some cases. However, it should be noted that TabuCol does outperform PartialCol in more cases than the reverse.

The literature would suggest that there is very little difference in the results obtained by TabuCol and PartialCol. We do not consider our results to necessarily contradict the literature because our test instances and conditions (e. g. time limit) are different from those generally used in the literature.

2.5 Independent Set Extraction

In this section we discuss a heuristic approach which is sometimes used to reduce the problem size of GCPs. By reducing the size of a given GCP, it is assumed that the resultant ("smaller") GCP will be easier to tackle.

Independent set extraction can be considered as a pre-processing method of reducing a graph G to a smaller residual graph $\tilde{G}$, which can then be passed to a different colouring procedure (e. g. a local search method). In Figure 2.5, the maximal independent set $I = \{v_2, v_3, v_8, v_9\}$ has been extracted from G to leave the residual graph $\tilde{G}$ on the RHS. This residual graph $\tilde{G}$ can now be passed to another

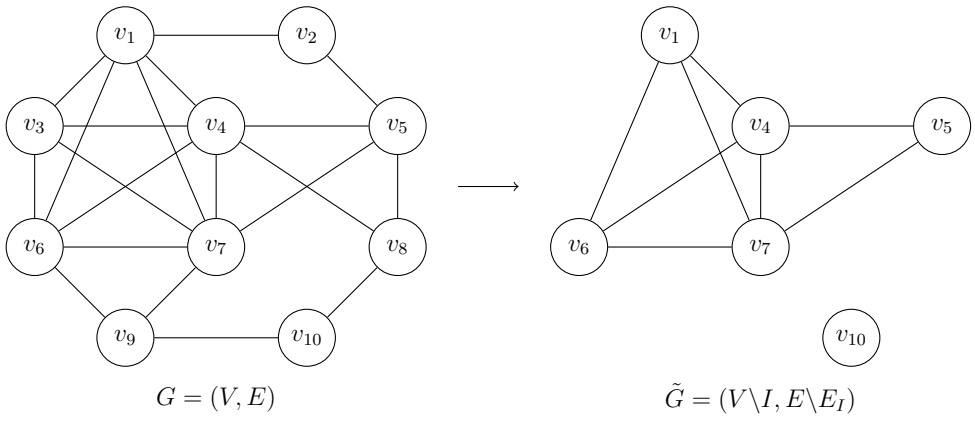

Figure 2.5: Independent set extraction example with maximal $I = \{v_2, v_3, v_8, v_9\}$

colouring procedure (e. g. a constructive algorithm or local search method). If a feasible k-colouring $\tilde{\mathcal{S}}$ for $\tilde{G}$ can be found, then combining $\tilde{\mathcal{S}}$ with I will produce a feasible colouring $\mathcal{S}$ for G with $k + 1$ colour classes. In general, if x independent sets are extracted from a graph G and a feasible k-colouring can be identified for the residual graph $\tilde{G}$ then a feasible colouring for G exists with $k + x$ colour classes.

An early tabu search algorithm for identifying large independent sets was proposed by Hertz and de Werra [1987]. This works in a similar fashion to Algorithm 2.1 for the GCP, i. e. first a legal independent set with l vertices is identified, then the algorithm attempts to identify an independent set with $l + 1$ vertices, and so on until some stopping criteria is met. Due to the way RLF [Leighton, 1979] constructs colourings (i. e. one colour class at a time), it can also be used to extract independent sets.

The method of Hertz and de Werra [1987] has been reproduced to extract a user-defined proportion $0 \leq r \leq 1$ of vertices, such that if $I_1, \ldots, I_x$ are the extracted independent sets then $\sum_{i=1}^{x} |I_i| \geq r|V|$ (for more details, see Appendix A.3). The stopping criterion, used here, is an iteration limit of 1000 iterations per independent set. Once the extraction phase is complete, the residual graph is passed to Algorithm 2.1 which produces an initial colouring via DSatur and utilises TabuCol to tackle the subsequent k-GCPs. The overall time limit is set to 5 minutes (300 seconds), with the independent set extraction phase taking place within this limit.

All experimental results presented in this section are done so with the aim of demonstrating that under the same time constraints used for comparing TabuCol and PartialCol in Section 2.4, the inclusion of independent set extraction is rarely beneficial with regards to the quality of the colourings achieved. We use the same test instances as in the previous comparisons of this chapter.

The results of extracting different proportions r of vertices can be seen in Table 2.3. The columns of the table show the mean number of colour classes in the "best" colourings obtained. Here; that is equal to the number of independent sets extracted plus the number of colour classes in the "best" colouring obtained for the

Table 2.3: Mean number of colour classes in the "best" colouring achieved with a proportion r of the vertices extracted via independent set extraction. The final column is the Spearman rank correlation coefficient ρ between r and the number of colour classes in the "best" colouring achieved, which are all significant at the $\alpha = 0.01$ level.

		r						
n	d	0	0.1	0.2	0.3	0.4	0.5	ρ
250	0.1	8.00	8.00	8.15	8.15	8.95	9.00	0.796
	0.5	28.05	28.70	29.10	29.75	30.20	30.90	0.896
	0.9	72.95	74.75	76.40	78.25	80.05	82.05	0.936
500	0.1	13.00	13.00	13.00	13.00	13.00	13.35	0.388
	0.5	49.60	50.00	50.50	50.95	51.70	52.50	0.887
	0.9	126.65	129.50	131.95	135.05	138.35	141.65	0.976
1000	0.1	21.05	21.25	21.30	21.45	21.45	21.80	0.355
	0.5	91.10	91.30	91.55	91.95	92.25	92.85	0.695
	0.9	228.00	232.90	237.60	242.20	246.60	251.65	0.981

residual graph.

It can be seen in Table 2.3 that the proportion of vertices extracted has a significant positive relationship with the number of colour classes in the "best" colourings achieved (i. e. as r increases so too does the number of colour classes). The poor quality of the colourings obtained after applying independent set extraction is probably due to the lack of freedom in the solution space because the extracted independent sets are equivalent to colour classes that can no longer be altered.

In later work [Hao and Wu, 2012; Wu and Hao, 2013], it has been suggested that this observation could be tackled with the addition of an expansion phase. During the expansion phase, the previously extracted independent sets are reintroduced to the residual graph for further perturbation.

We have produced two different expansion schemes for comparison: sequential and simultaneous. For both of these expansion schemes, half of the time limit (so 150 seconds in our case) is dedicated to first extracting the independent sets and then applying TabuCol to the initial residual graph. The sequential expansion scheme reintroduces the extracted independent sets one-by-one in the reverse order to which they were extracted. After each reintroduction, TabuCol is reapplied for the remainder of the time limit divided by the number of initially extracted independent sets x, so in our case that would be $\frac{150}{x}$ seconds. The simultaneous expansion scheme reintroduces all of the extracted independent sets at once and TabuCol is reapplied for the remainder of the time limit.

As can be seen in Table 2.4, the strategy of not using any expansion scheme is generally outperformed by using the sequential expansion scheme, with regards to the number of colour classes in the "best" colouring achieved, which in turn is generally outperformed by using the simultaneous expansion scheme. These findings, along with those presented in Table 2.3, are clearly illustrated in Figure 2.6 for test

Table 2.4: Mean number of colour classes in the "best" colouring achieved using different expansion schemes with a proportion of $r = 0.5$ of the vertices extracted via independent set extraction.

		Expansion Scheme		
n	d	No Expansion	Sequential	Simultaneous
250	0.1	9.00	8.00	8.00
	0.5	30.90	28.50	28.05*
	0.9	82.05	73.30	72.95
500	0.1	13.35	13.00	13.00
	0.5	52.50	50.65	49.80*
	0.9	141.65	128.75	126.90*
1000	0.1	21.80	21.05	21.00
	0.5	92.85	92.05	91.45*
	0.9	251.65	238.95	229.15*

* indicates colourings with significantly fewer colour classes than those achieved when implementing all others expansion schemes for the same values of n and d.

instances with $n = 500$ and $d = 0.5$. Observe that the proportion of extracted vertices r appears to have little to no effect on the quality of the colourings achieved when using the simultaneous expansion scheme.

Under these test conditions, including independent set extraction (both with or without an expansion phase) leads to results that cannot outperform and are regularly bested by simply applying TabuCol to the original graph. However, a slightly different extraction phase described in [Wu and Hao, 2012], which is combined with the memetic algorithm proposed by Lü and Hao [2010] instead of TabuCol, has been shown to outperform many methods for tackling the GCP on very large graphs (i. e. $n = |V| \geq 1000$).

2.6 Evolutionary Methods

Overall, perhaps the best results reported in the literature for the GCP have been obtained using evolutionary (or population-based) methods. These methods use the characteristics of a population of colourings, rather than a single colouring, to inform the generation of new colourings.

Evolutionary algorithms recombine two or more "parent" solutions to produce one or more "offspring" solutions. Heuristics are generally utilised such that "better parent" solutions are selected for recombination with higher probabilities (e. g. using roulette wheel selection). Random mutations to the "offspring" are then allowed in an attempt to improve diversity within the population. Hybrid evolutionary algorithms (HEAs) work in a similar fashion but apply an local search method in place of random mutation. As mentioned in Section 2.4.1, such algorithms designed for tackling the GCP often replace random mutation with TabuCol [Hertz and de Werra, 1987].

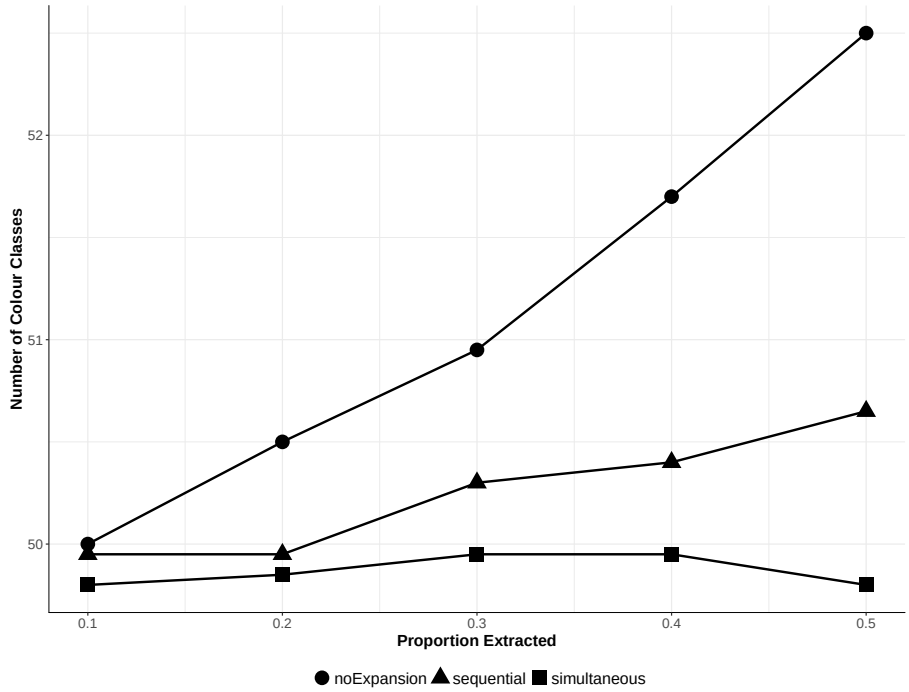

Figure 2.6: Mean number of colour classes in the "best" colouring achieved with different extraction proportions and expansion schemes on test instances with $n = 500$ and $d = 0.5$.

For evolutionary and hybrid evolutionary algorithms, an appropriate crossover operator must be defined. An effective crossover for the GCP is the greedy partition crossover (GPX) operator introduced by Galinier and Hao [1999], which produces an "offspring" colouring by inheriting the largest remaining colour class from two "parent" colourings in an alternating fashion.

An example of the GPX operator recombining "parent" colourings $\mathcal{S}_{p_1}$ and $\mathcal{S}_{p_2}$ to produce an "offspring" colouring $\mathcal{S}_o$ can be seen in Table 2.5. Step 0 shows the original "parent" colourings $\mathcal{S}_{p_1}$ and $\mathcal{S}_{p_2}$. During step 1, $\mathcal{S}_o$ "inherits" the largest colour class S_2 from $\mathcal{S}_{p_1}$ and the vertices in this colour class are then removed from both $\mathcal{S}_{p_1}$ and $\mathcal{S}_{p_2}$. $\mathcal{S}_o$ then "inherits" colour class S_3 from $\mathcal{S}_{p_2}$ and S_1 from $\mathcal{S}_{p_1}$ during steps 2 and 3 respectively. Notice that v_9 has not been inherited by $\mathcal{S}_o$ so that $\mathcal{S}_o$ is a partial colouring and v_9 is "uncoloured". In practice, after the GPX operator has been executed, any "uncoloured" vertices need to be assigned to a colour class of the "offspring" colouring (e. g. at random).

The adaptive multi-parent crossover (AMPaX) operator proposed by Lü and Hao [2010] works in a similar fashion to the GPX operator but recombines between 2 and $maxP$ (a user-defined upper bound) "parent" colourings to produce an "offspring" colouring. Unlike GPX, AMPaX does not cycle through all "parent" colouring in turn; instead, it selects the colour class with the most vertices across all "parent" colourings at each step, and incorporates a simple heuristic (similar to tabu search) to ensure that the "parent" colouring that was just selected cannot be selected again for a number of subsequent steps. It should be noted, that if $maxP = 2$

Table 2.5: GPX operator example with "parent" colourings $\mathcal{S}_{p_1}$ and $\mathcal{S}_{p_2}$ recombined to produce an "offspring" colouring $\mathcal{S}_o$.

		Colour Class		
step	colouring	S_1	S_2	S_3
0	$\mathcal{S}_{p_1}$:	$\{v_1, v_2, v_3\}$	$\{v_4, v_5, v_6, v_7\}$	$\{v_8, v_9, v_{10}\}$
	$\mathcal{S}_{p_2}$:	$\{v_3, v_4, v_5, v_7\}$	$\{v_1, v_6, v_9\}$	$\{v_2, v_8, v_{10}\}$
	$\mathcal{S}_o$:	$\{\}$	$\{\}$	$\{\}$
1	$\mathcal{S}_{p_1}$:	$\{v_1, v_2, v_3\}$	$\{\}$	$\{v_8, v_9, v_{10}\}$
	$\mathcal{S}_{p_2}$:	$\{v_3\}$	$\{v_1, v_9\}$	$\{v_2, v_8, v_{10}\}$
	$\mathcal{S}_o$:	$\{v_4, v_5, v_6, v_7\}$	$\{\}$	$\{\}$
2	$\mathcal{S}_{p_1}$:	$\{v_1, v_3\}$	$\{\}$	$\{v_9\}$
	$\mathcal{S}_{p_2}$:	$\{v_3\}$	$\{v_1, v_9\}$	$\{\}$
	$\mathcal{S}_o$:	$\{v_4, v_5, v_6, v_7\}$	$\{v_2, v_8, v_{10}\}$	$\{\}$
3	$\mathcal{S}_{p_1}$:	$\{\}$	$\{\}$	$\{v_9\}$
	$\mathcal{S}_{p_2}$:	$\{\}$	$\{v_9\}$	$\{\}$
	$\mathcal{S}_o$:	$\{v_4, v_5, v_6, v_7\}$	$\{v_2, v_8, v_{10}\}$	$\{v_1, v_3\}$

Table 2.6: Initial number of colour classes k for our hybrid evolutionary algorithm.

	d		
n	0.1	0.5	0.9
250	8	29	74
500	13	50	129
1000	22	92	232

then AMPaX acts in the exact same manner as GPX.

In order to determine whether a HEA can outperform local search with regards to the quality of the colourings achieved, as might be expected by looking at the results reported in the literature, we have reproduced a HEA that is similar to the ones presented in [Galinier and Hao, 1999; Lü and Hao, 2010]. This algorithm uses the AMPaX operator and, like these publications, applies TabuCol in place of random mutation. Unlike the process outlined in Algorithm 2.1, this method starts by attempting to find a colouring with a user-defined number of colour classes k. If a feasible k-colouring can be identified then k is reduced in the same manner as before. The initial values of k used in our trials are given in Table 2.6, which were influenced by the maximum colourings documented in Table 2.2. It should be noted that the number of colour classes in the "best" colourings achieved by our HEA will be greatly influenced by the "quality" of these initial values of k.

The initial population of colourings is generated using a modified version of DSatur which can only create colourings with at most k colour classes. Vertices that can not be feasibly assigned to one of the k colour classes are randomly assigned to a colour classes at the end, therefore creating a complete, improper colouring. The AMPaX operator randomly chooses between 2 and $maxP$ "parent" colourings from the population to create an "offspring" colouring. The population of colourings is updated by replacing the "parent" colouring with the most clashes, with the "off-

Table 2.7: Mean number of colour classes in the "best" colourings achieved by our hybrid evolutionary algorithm with $maxP = 2$ and 5000 TabuCol iterations. The final column is the Spearman rank correlation coefficient ρ between population size and the number of colour classes in the "best" colourings achieved, if significant at the $\alpha = 0.01$ level.

		Population Size			
n	d	5	10	20	ρ
250	0.1	8.00	8.00	8.00	-
	0.5	28.50	28.20	28.15	-
	0.9	73.05	73.15	73.10	-
500	0.1	13.00	12.80	12.60*	-0.408
	0.5	49.40	49.00*	48.35*	-0.734
	0.9	127.75†	127.35†	127.00	-0.415
1000	0.1	21.00	20.90	20.25*	-0.679
	0.5	88.35*	86.70*	88.00*	-
	0.9	230.10†	228.25	229.60†	-

* indicates colourings with significantly fewer colour classes than those achieved when using TabuCol alone for the same values of n and d.
† indicates colourings with significantly more colour classes than those achieved when using TabuCol alone for the same values of n and d.

spring" colouring. All new colourings (i. e. the initial population colourings and each "offspring" colouring) are passed to TabuCol for a fixed number of iterations. This HEA has an overall time limit of 5 minutes (300 seconds) in order to be comparable with the other algorithms discussed in this chapter.

Using the same test instances used throughout the chapter so far, comparisons of the number of colour classes in the "best" colourings achieved when this HEA is implemented with different population sizes, values of $maxP$ for AMPaX, and number of TabuCol iterations can be seen in Tables 2.7 to 2.9 respectively. More specific parameter setting for each of these comparisons can be found in the appropriate table descriptions.

It can be seen in Table 2.7 that for some test instances, especially those with $n = 500$, there is a significant negative relationship between the population size and the number of colour classes in the "best" colourings achieved (i. e. as the population size increases, the number of colour classes decreases). In general, a larger population could also imply higher diversity within the population, which in turn would allow this method to explore more of the solution space for feasible colourings.

The results presented in Table 2.8 indicate that increasing $maxP$ does not appear to have any positive impact on the quality of the "best" colourings achieved in most cases. On the contrary, in some cases there appears to be a detrimental effect on the number of colour classes in the "best" colourings achieved (see rows with $\rho > 0$).

The results in Table 2.9 show that in some cases, increasing the number of TabuCol iterations, decreases the number of colour classes in the "best" colourings achieved. This is likely due to the quality of the colourings that TabuCol is

Table 2.8: Mean number of colour classes in the "best" colourings achieved by our hybrid evolutionary algorithm with a population size of 20 and 5000 TabuCol iterations. The final column is the Spearman rank correlation coefficient ρ between population size and the number of colour classes in the "best" colourings achieved, if significant at the $\alpha = 0.01$ level.

		maxP for AMPaX				
n	d	2	5	10	20	ρ
250	0.1	8.00	8.00	8.00	8.00	-
	0.5	28.15	28.20	28.30	28.40	-
	0.9	73.10	73.05	73.15	73.15	-
500	0.1	12.60*	12.80*	12.95	12.95	0.353
	0.5	48.35*	48.90*	49.05*	49.15*	0.615
	0.9	127.00	127.25†	127.35†	127.55†	-
1000	0.1	20.25*	20.85	21.00	21.00	0.643
	0.5	88.00*	87.00*	87.40*	87.90*	-
	0.9	229.60†	228.95†	228.60†	228.85†	-0.430

* indicates colourings with significantly fewer colour classes than those achieved when using TabuCol alone for the same values of n and d.
† indicates colourings with significantly more colour classes than those achieved when using TabuCol alone for the same values of n and d.

known to achieve as a standalone method for solving the GCP (see Section 2.4.1 and [Blöchliger and Zufferey, 2008]). However, for test instances with $n = 1000$ and $d \in \{0.5, 0.9\}$ a positive relationship was observed (i. e. $\rho > 0$). This seems contradictory to what might be expected, but the reasons for the decrease in colouring quality may be due to other parts of the HEA (e. g. crossover) being implemented less often.

As mentioned at the start of this section, evolutionary methods have been shown to produce some of the best results for the GCP. Under the parameters explored here, the HEA described outperforms TabuCol most notably for test instances with $n \in \{500, 1000\}$ and $d = 0.5$. Based on the literature, the HEA is expected to outperform the relatively simple local search method on all random graphs. However, as with the comparison between TabuCol and PartialCol, we do not consider these results to be contradictory to the literature because our time limit is much shorter than those usually used in the literature (i. e. our time limit is a few minutes as opposed to days). It is also worth highlighting that our HEA is regularly outperformed, with regards to the number of colour classes in the "best" colourings achieved, by TabuCol on test instances with $n \in \{500, 1000\}$ and $d = 0.9$.

In the literature, experimentation with different local search methods in place of TabuCol within the HEA framework have also been explored (e. g. simple vertex descent [Glass and Prügel-Bennett, 2003]). This particular research concluded that it is not the implementation of TabuCol specifically that leads to such "good" results in [Galinier and Hao, 1999], but rather the combination of a crossover operator with a local search method.

Table 2.9: Mean number of colour classes in the "best" colourings achieved by our hybrid evolutionary algorithm with a population size of 20 and $maxP = 2$. The final column is the Spearman rank correlation coefficient ρ between population size and the number of colour classes in the "best" colourings achieved, if significant at the $\alpha = 0.01$ level.

		TabuCol Iterations			
n	d	1000	2000	5000	ρ
250	0.1	8.00	8.00	8.00	-
	0.5	28.45†	28.40	28.15	-
	0.9	73.30	73.15	73.10	-
500	0.1	13.00	12.85	12.60*	-0.422
	0.5	49.05*	48.85*	48.35*	-0.610
	0.9	127.85†	127.15†	127.00	-0.456
1000	0.1	21.00	20.60*	20.25*	-0.630
	0.5	87.45*	87.00*	88.00*	0.473
	0.9	228.55	228.65†	229.60†	0.522

* indicates colourings with significantly fewer colour classes than those achieved when using TabuCol alone for the same values of n and d.
† indicates colourings with significantly more colour classes than those achieved when using TabuCol alone for the same values of n and d.

Another example of an evolutionary method for the GCP is the adaptive memory algorithm proposed by Galinier et al. [2004], which utilises a population of independent sets (i. e. colour classes) instead of full colourings. An ant colony optimisation algorithm has also been designed for tackling the GCP by Costa and Hertz [1997], and then later improved upon by Dowsland and Thompson [2008]. Ant colony optimisation [Dorigo et al., 2006], which mimics the cooperative behaviour of real ants, uses the relative success of previous colourings to generate new colourings that embody the "good" aspects of the population.

2.7 Other Heuristic Methods

Many more heuristic methods for tackling the GCP have also been proposed, including: iterated local search [Chiarandini and Stützle, 2002], neural networks [Jagota, 1996], greedy randomised adaptive search procedure (GRASP) for low density graphs [Laguna and Martí, 2001], quantum annealing [Titiloye and Crispin, 2011], and hyper heuristics [Qu et al., 2009]. To discuss and compare all of these methods is beyond the scope of this chapter. However, an overview of the most recent methods for solving the GCP can be found in [Galinier et al., 2013], and a comprehensive comparison of some of the most successful methods (including TabuCol, PartialCol and hybrid evolutionary algorithms) can be found in [Lewis et al., 2012].

2.8 Chapter Summary

In this chapter we have reviewed several different heuristic methods for tackling the static GCP. We have also reproduced and compared several of these methods under relatively tight time constraints, at least in comparison to the time constraints used in the literature. Shorter time limits were considered here, because when we look at dynamic versions of the GCP in Chapters 3 to 6, quick run-times will be of increased importance.

The best results observed in this chapter were achieved by TabuCol, PartialCol and our HEA. However, of these three algorithms, none was shown to clearly dominate the other two over all test instances. One particular drawback of the HEA, compared to the other two algorithms, is the large number of parameter settings that must be "tuned" in order to achieve the "best" results. As we saw in Section 2.6, the population size, the value of $maxP$ used within the AMPaX operator, and the number of TabuCol iterations all effect the quality of the colourings achieved by the HEA.

In subsequent chapters, we will be reducing the time limit even further (from 5 minutes to 10 seconds). It has been shown by Lewis [2015] that local search methods are generally quicker at locating local optima, compared to more complex methods. Therefore, we will cease to use the HEA in the following chapters, but continue to use TabuCol and PartialCol in their own rights.

Chapter 3

Modifying Colourings for Edge Dynamic Random Graphs without Future Change Information

The importance of studying dynamic graphs and their associated problems has been highlighted by Harary and Gupta [1997]. In their paper, they describe many practical application areas, especially within computer science, and postulate that techniques applied to static graphs should be extended for their dynamic counterparts. Despite this, there has been very little research regarding methods designed explicitly for colouring dynamic graphs.

As discussed in Section 1.4, many real-world operational research problems can be reformulated as static GCPs. This method of tackling real-world problems assumes that the size and constraints of said problems are fixed (i. e. the vertex set V and edge set E of the associated graph $G = (V, E)$ do not change). However, this is not always appropriate.

Consider the dynamic frequency assignment problem [Dupont et al., 2009], an extension of the frequency assignment problem discussed in Section 1.4. Initially, a set of physical locations within a communication network will be known, such that an initial static graph $G_0 = (V_0, E_0)$ can be defined to represent the initial (static) frequency assignment problem. As time moves forward, additional locations may be added to the communication network and / or locations may move within the network, therefore altering the size of the problem and its "interference" constraints. If these changes were to occur, then the initial static graph G_0 will no longer accurately represent the problem. Could it be beneficial in some way to "repair" a solution (i. e. a colouring) for the initial problem, rather than attempting to find a new solution, from scratch, for the new problem?

In this chapter we will introduce the concept of dynamic graphs and their associated colouring problems. We will then introduce our modification approach for solving dynamic GCPs and present results for the edge dynamic GCP without future

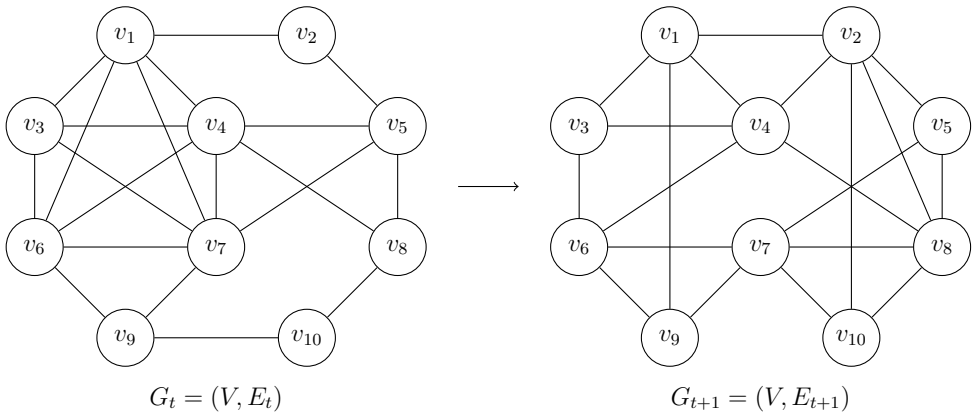

Figure 3.1: Edge dynamic graph example.

change information.

3.1 Dynamic Graph Colouring Problems

For the purpose of this thesis, we define a dynamic graph $\mathcal{G} = (G_0, G_1, \ldots, G_T)$ as a series of $T + 1$ static graphs where $G_t = (V_t, E_t) \in \mathcal{G}$ is the static representation of $\mathcal{G}$ at time-step $t \in \{0, 1, \ldots, T\}$. At every time-step the objective of the dynamic GCP is analogous to the static GCP (i. e. we wish to minimise the number of colours used). In terms of methodology, this means that we are interested in finding a feasible k_t-colouring for each time-step t, where k_t is a good approximation of $\chi(G_t)$ (see Section 1.2 for a formal definition of a "feasible" colouring). Objectively, this is an attempt to minimise $\sum_{t=0}^{T} k_t$.

The concept of dynamic graphs will be considered as two separate cases: edge dynamic graphs, which will be the focus of this chapter, and vertex dynamic graphs, which will be explored in Chapter 5.

Edge Dynamic Graphs

For an edge dynamic graph, changes can only occur on the edge set such that E_t may be different to E_{t+1}; therefore $V_t = V_{t+1}$ for all time-steps $t \in \{0, 1, \ldots, T-1\}$, and we will simply refer to the vertex set of an edge dynamic graph as V. Given an edge dynamic graph $\mathcal{G}$, consider the graph $G_t = (V, E_t)$ at time-step t. To get to time-step $t + 1$ we define a set of deleted edges $E^-_{t+1} \subseteq E_t$ and a set of new edges $E^+_{t+1} \subseteq (\mathcal{E} \backslash E_t)$ where $\mathcal{E}$ is the set of all $\binom{|V|}{2}$ possible edges between vertices in V. The edge set for time-step $t + 1$ is then defined as $E_{t+1} = (E_t \backslash E^-_{t+1}) \cup E^+_{t+1}$.

An example of how an edge dynamic graph can change between time-steps is illustrated in Figure 3.1. Here, we define the set of deleted edges as $E^-_{t+1} = \{\{v_1, v_6\}, \{v_1, v_7\}, \{v_3, v_7\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_9, v_{10}\}\}$ and the set of new edges as $E^+_{t+1} = \{\{v_1, v_9\}, \{v_2, v_4\}, \{v_2, v_8\}, \{v_2, v_{10}\}, \{v_7, v_8\}, \{v_7, v_{10}\}\}$. As mentioned, the

vertex set V remains fixed between time-steps.

An edge dynamic graph could be used to represent a real-world problem where the constraints change over time. For example, consider an exam timetabling problem for a university over a number of consecutive years. A university may have decided to only offer the same modules each year, which can be represented by a fixed vertex set V. However, the modules may be combined in a number of different ways by students each year, therefore the constraints that ensure no conflicting exams are scheduled in the same timeslots will also change on a yearly basis. These changing constraints could then be represented by a changing edge set E_t where t represents a specific academic year.

3.2 Modification Approach

Our general approach for tackling a dynamic graph $\mathcal{G} = (G_0, G_1, \ldots, G_T)$ follows the process outlined in Algorithm 3.1. Note that an approach for solving static graphs is used for G_0 because there is no colouring from a previous time-step to be modified.

Algorithm 3.1 Generic Dynamic GCP Algorithm

Input: a dynamic graph $\mathcal{G} = (G_0, G_1, \ldots, G_T)$
Output: a set $\mathbf{S} = \{\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_T\}$ where $\mathcal{S}_t$ is a feasible colouring for $G_t \in \mathcal{G}$
1: $\mathcal{S}_0 \leftarrow$ Static GCP Algorithm $[G_0]$ (i. e. variation of Alg. 2.1)
2: **for** $t = 1$ **to** T **do**
3: $\quad \mathcal{S}_t \leftarrow$ Dynamic GCP Time-step Algorithm $[G_t, \mathcal{S}_{t-1}]$ (i. e. variation of Alg. 3.2)
4: **return** $\mathbf{S} = \{\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_T\}$

For time-steps $t = 1, \ldots, T$ we can use the approach outlined in Algorithm 3.2 which is a modified version of the general approach for solving a static GCP (see Algorithm 2.1). Here, we have replaced the constructive operator at the beginning of the algorithm with a modification operator which modifies the "best" feasible colouring for the previous time-step into an initial, though not necessarily feasible, colouring for the current time-step.

Consider time-step $t + 1$. The "best" feasible colouring $\mathcal{S}_t$ for G_t is modified into an initial colouring $\mathcal{S}_{t+1}$ for G_{t+1}. This initial colouring is then passed to an optimisation operator (i. e. a k-GCP algorithm) which attempts to find a feasible k-colouring for G_{t+1} where the initial value of k is determined by the modification operator and satisfies $k \geq |\mathcal{S}_t|$.

One potential issue with this methodology can arise if the initial value of k is chosen such that $k < \chi(G_{t+1})$. This eventuality is dealt with by periodically allowing the value of k to increase if no feasible colouring for G_{t+1} has been found. This is shown on Lines 9 and 10 of Algorithm 3.2.

Although there are very few algorithms for tackling dynamic graph colouring problems in the current literature, we will now summarise and compare these algo-

Algorithm 3.2 Generic Dynamic GCP Time-step Algorithm

Input: a graph G_{t+1} and a feasible colouring $\mathcal{S}_t$ for G_t
Output: a feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1}

```
1: S_best ← ∅
2: S_{t+1} ← S_t modified in some way (see Sections 3.3 and 5.2)
3: k ← |S_{t+1}|
4: while not stopping criterion do
5:    attempt to make S_{t+1} a feasible k-colouring for G_{t+1}
6:    if S_{t+1} is a feasible k-colouring for G_{t+1} then
7:       S_best ← S_{t+1}
8:       k ← k − 1
9:    if S_best = ∅ and a computation limit is reached then
10:      k ← k + 1
11: S_{t+1} ← S_best
12: return S_{t+1}
```

rithms against our proposed modification approach.

Other Algorithms for Tackling Dynamic GCPs, a Literature Review

As mentioned previously, Harary and Gupta [1997] have highlighted the importance of studying dynamic graphs, their associated problems, and adapting algorithms from their static counterparts. Yet, very few static graph colouring algorithms have been adapted for dynamic GCPs.

Theoretical studies by Barba et al. [2017] and Bhattacharya et al. [2018] have proposed a few algorithms for tackling dynamic GCPs. Their works focuses on upper-bounds for the number of vertices that must be recoloured or the "update" time required, respectively, at each time-step in order to maintain a feasible colouring with a fixed number of colour classes. As might be expected, a clear negative relationship between these two objectives is presented, such that more "maintenance" is required as the number of colour classes decreases (i. e. becomes closer to optimal). Due to the theoretical nature of these works, it would be inappropriate to compare their results with ours, which are experimental in nature.

Algorithms for the on-line graph colouring problem, a special case of the vertex dynamic GCP, are well documented in the literature. Most research in this area is theoretical and concerns worst-case behaviour of proposed algorithms. This problem, and relevant literature, will be discussed in more detail in Section 5.1, once vertex dynamic GCPs have been formally introduced.

The few heuristic approaches for dynamic GCPs that we were able to find in the literature are slightly different from our proposed modification approach. Indeed, most of these approaches are only interested in acquiring an initial feasible colouring and do not allow local optimisation to take place between time-steps. For example, Preuveneers and Berbers [2004] propose an agent-based approach (ACODYGRA) for the edge dynamic GCP which "repairs" colourings between time-steps in a vertex-

by-vertex manner. ACODYGRA first attempts to "recolour" a vertex (i. e. transfer to a different colour class) and, if this is not possible, a new colour class is created which the vertex is then transferred to. This approach is similar in some ways to Method 4 (*solveClashes*) which will be described towards the end of Section 3.3.

A decentralised algorithm for dynamic GCPs has been proposed by Dutot et al. [2007]. The experimental results presented for this algorithm are very brief and are only compared against the GREEDY constructive algorithm [Welsh and Powell, 1967]. This decentralised algorithm does outperform the GREEDY algorithm with regards to the number of colour classes used, however no information is presented regarding the computational effort required to do so. As with ACODYGRA, no optimisation is permitted between time-steps.

Monical and Stonedahl [2014] proposed a genetic algorithm approach for tackling the vertex dynamic GCP and experimented with both static and dynamic populations of vertex orderings (which are passed to a GREEDY-style constructive algorithm). This approach differs from the one outlined in Algorithm 3.1 because, instead of passing a single colourings from time-step t to time-step $t+1$, their algorithm passes a whole population of vertex orderings. Again, as with ACODYGRA and the decentralised algorithm discussed above, this algorithm is only concerned with achieving an initial colouring and is therefore only compared against DSATUR [Brélaz, 1979].

3.3 Modification Operators

The focus of this particular chapter is to compare the performance of the different modification operators used on Line 2 of Algorithm 3.2. In doing so, we wish to determine whether there are any benefits of using a feasible colouring for a graph G_t to find a feasible colouring for a similar graph G_{t+1} and, if so, under what conditions these benefits occur.

It is worth noting that a feasible colouring at time-step t is likely to become a complete, improper colouring at time-step $t+1$ because the new edges introduced are likely to lead to clashes. Let us reconsider the edge dynamic graph in Figure 3.1. As is illustrated in Figure 3.2, the feasible colouring of G_t on the LHS has become an complete, improper colouring for G_{t+1} on the RHS. Every vertex of G_{t+1} remains coloured; however the introduction of edges $\{v_1, v_9\}$, $\{v_2, v_4\}$ and $\{v_2, v_{10}\}$ has led to clashes. Therefore, perhaps the most "natural" solution space to consider when using a colouring from time-step t as a starting point for a colouring at time-step $t+1$ is the complete, improper one.

We now introduce five modification operators (or methods) that "modify" a feasible colouring for G_t into a colouring for G_{t+1}. The colourings produced by these modification operators are not necessarily feasible for G_{t+1}, in which case they are

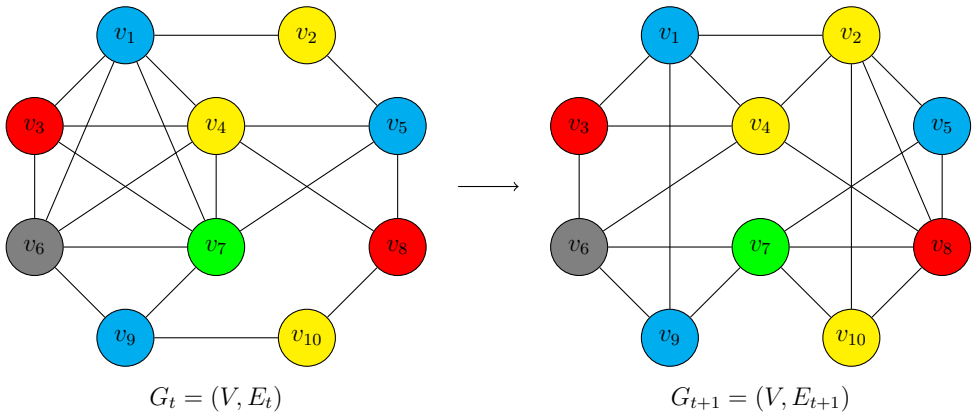

Figure 3.2: Edge dynamic graph colouring example

used as a starting point from which to find a feasible colouring for G_{t+1}. The first of these modification operators (Method 0) will be used as a control throughout this thesis.

Method 0 (*reset*)**.** Ignoring $\mathcal{S}_t$, use a constructive operator to produce $\mathcal{S}_{t+1}$.

By ignoring the colourings for the previous time-step and simply using a constructive algorithm (see Section 2.1 for examples), this approach can be used as a base-line comparison against the remaining methods for time-steps $t = 1, \ldots, T$. Essentially, implementing Method 0 within Algorithm 3.2 is identical to the approach used for static graphs outlined in Algorithm 2.1. This method is guaranteed to produce an initial, feasible colouring for all time-steps due to the nature of constructive algorithms.

Method 1 (*calculateClashes*)**.** Set $\mathcal{S}_{t+1} = \mathcal{S}_t$ and calculate the number of clashes via Equation (2.4).

As mentioned above, a feasible colouring $\mathcal{S}_t$ for G_t is likely to be a complete, improper colouring for G_{t+1}. By implementing Method 1, the resultant colouring $\mathcal{S}_{t+1} = \mathcal{S}_t$ can be passed directly to an optimisation operator that operates in the complete, improper solution space. This optimisation operator will attempt to remove all clashes from $\mathcal{S}_{t+1}$ to achieve a feasible k-colouring for G_{t+1} where $k = |\mathcal{S}_t|$.

For the example illustrated in Figure 3.2, the optimisation operator would attempt to find a feasible 5-colouring for G_{t+1} starting from the feasible colouring for G_t on the LHS which has 3 clashes for G_{t+1} on the RHS.

Method 2 (*uncolourClashes*)**.** Identify a pair of clashing vertices in $\mathcal{S}_t$ and transfer one vertex from this pair to a set of "uncoloured" vertices U. Repeat until the resultant colouring $\mathcal{S}_{t+1}$ has no clashes.

Under this method, the resultant colouring $\mathcal{S}_{t+1}$ is a partial, proper colouring for G_{t+1} which, along with U, can be passed to an optimisation operator that operates

in the partial, proper solution space. This optimisation operator will attempt to feasibly colour all "uncoloured" vertices in U to achieve a feasible k-colouring for G_{t+1} where $k = |\mathcal{S}_t|$. By moving from the "natural" solution space to the partial, proper one, we can explore an entirely different neighbourhood of colourings.

Due to the random nature of how vertices are "uncoloured" here, there are several different colourings that Method 2 could produce. For the example illustrated in Figure 3.2, there are 8 different partial, proper colourings that could be produced, resulting in U being any of the following: $\{v_1, v_2\}, \{v_1, v_2, v_4\}, \{v_1, v_2, v_{10}\}, \{v_1, v_4, v_{10}\}$, $\{v_2, v_9\}, \{v_2, v_4, v_9\}, \{v_2, v_9, v_{10}\}$, or $\{v_4, v_9, v_{10}\}$.

This modification operator is very similar to the "translator" proposed by Hertz et al. [2008], which converts a complete, improper k-colouring for a graph G into a partial, proper k-colouring for G.

Method 3 (*uncolourMostClashing*)**.** Identify *all* pairs of clashing vertices in $\mathcal{S}_t$ and transfer the "most clashing" vertex (i.e. the vertex that is included in the highest number of clashing pairs) to the set of "uncoloured" vertices U. Repeat until the resultant colouring $\mathcal{S}_{t+1}$ has no clashes.

Much like Method 2, we wish to "uncolour" clashing vertices in $\mathcal{S}_t$ to achieve a partial, proper colouring for G_{t+1}. By using this method, the number of "uncoloured" vertices, $|U|$, should be reduced in comparison to using the random approach of Method 2.

For the example illustrated in Figure 3.2, Method 3 could produce 2 different partial, proper colourings for G_{t+1} with either $U = \{v_1, v_2\}$ or $\{v_2, v_9\}$. Therefore, in this particular case, this method is guaranteed to produce a partial, proper colouring $\mathcal{S}_{t+1}$ for G_{t+1} with $|U| = 2$. On the other hand, implementing Method 2 on the same graph can lead to partial, proper colourings for G_{t+1} with either $|U| = 2$ or $|U| = 3$ at a ratio of $1 : 2$.

Method 4 (*solveClashes*)**.** Using Method 2 on $\mathcal{S}_t$, obtain a partial, proper colouring $\mathcal{S}_{temp}$ for G_{t+1} and a set of "uncoloured" vertices U. Next, randomly order the vertices in U and attempt to transfer the first vertex in U to a colour class of $\mathcal{S}_{temp}$ in a "greedy" fashion such that no clashes are introduced. Repeat for each vertex in U sequentially. Pass the residual graph $\tilde{G}$ induced by the remaining vertices in U to a constructive operator to produce a colouring $\tilde{\mathcal{S}}$ for $\tilde{G}$. Combine $\mathcal{S}_{temp}$ and $\tilde{\mathcal{S}}$ to produce the colouring $\mathcal{S}_{t+1}$.

This method bears some similarity to independent set extraction (see Section 2.5) where a graph is considered in separate parts which are then combined to produce a colouring for the whole graph. Here, the clashing vertices are considered separately to the vertices which remain feasibly coloured between time-steps.

This method is guaranteed to produce a feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1}. Firstly, by implementing Method 2, all clashes in $\mathcal{S}_t$ for G_{t+1} are removed. Also, no new

clashes are introduced when a vertex in U is transferred to a colour class in $\mathcal{S}_{temp}$, otherwise it would not be transferred. A constructive algorithm is also guaranteed to produce a feasible colouring for any given graph, so the colouring $\tilde{\mathcal{S}}$ is a feasible colouring of the remaining "uncoloured" vertices. Finally, when $\mathcal{S}_{temp}$ and $\tilde{\mathcal{S}}$ are combined, the resultant colouring $\mathcal{S}_{t+1}$ has no clashes and all vertices in V are assigned to a colour class. Therefore, $\mathcal{S}_{t+1}$ is a feasible colouring for G_{t+1} with $k = |\mathcal{S}_t| + |\tilde{\mathcal{S}}|$ colour classes.

For the example illustrated in Figure 3.2, regardless of which vertices are transferred to the set of "uncoloured" vertices U, all vertices in U can be feasibly transferred to a colour class in $\mathcal{S}_{temp}$ to produce a feasible 5-colouring for G_{t+1}. Consider $U = \{v_2, v_9\}$, in this example v_2 can be transferred to either the grey or the green colour class without introducing clashes, and v_9 can be transferred to the red colour class similarly. Thus, for this example there is no residual graph $\tilde{G}$ to pass to a constructive operator.

As mentioned in Section 3.2, this method is similar to ACODYGRA [Preuveneers and Berbers, 2004]. Both approaches attempts to transfer each "problematic" vertex to a colour class in the current colouring first, and then, if necessary, they create new colour classes for any remaining "problematic" vertices. In ACODYGRA the "problematic" vertices are those which are clashing, and here the "problematic" vertices are the newly "uncoloured" ones.

The specific details regarding the constructive operators and optimisation operators used in Methods 0 to 4 will be discussed in Section 3.4.2.

3.4 Trial Information

3.4.1 Test Instances

The graphs that make up our test instances have four parameters: n the number of vertices, d the desired density, p the change probability, and T the number of time-steps. The following parameter values were used for our test instances: $n = \{250, 500, 1000\}$, $d \in \{0.1, 0.5, 0.9\}$, $p \in \{0.005, 0.01, \ldots, 0.05\}$ and $T = 10$. These values of n and d are broadly in line with the parameters of the static random graphs presented in the benchmark instances referenced in Chapter 2. Preliminary trials indicated that larger values of p greatly diminished the benefits of using a modification operator between time-steps. For each combination of these parameters, 20 dynamic graphs were produced.

In our case, G_0 is constructed such that $|V| = n$, and every edge $\{u, v\}$ of the $\binom{n}{2}$ possible edges in $\mathcal{E}$ is included in E_0 with probability d. At time-step t, each edge $\{u, v\} \in E_t$ is included in the set of deleted edges E^-_{t+1} with probability p and each edge $\{u, v\} \in \mathcal{E} \backslash E_t$ is included in the set of new edges E^+_{t+1} with probability $\frac{pd}{1-d}$. The probability $\frac{pd}{1-d}$ is used to ensure that the density of the various graphs

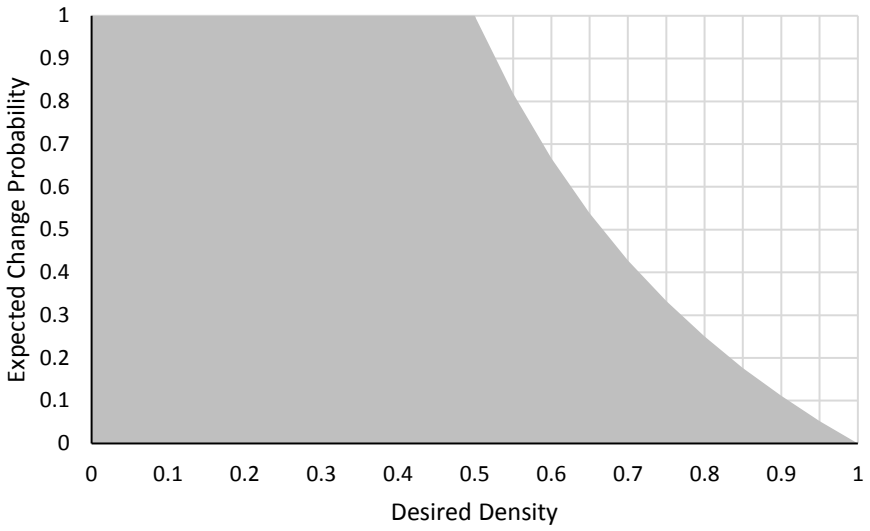

Figure 3.3: Legal combinations of change probability p and desired density d that maintain approximately equal density of an edge dynamic graph over all time-steps.

remains approximately equal over all time-steps.

Legal Parameter Combinations for Edge Dynamic Graphs

Note that at time-step t there are $|E_t|$ active edges out of the $|\mathcal{E}|$ possible edges between all the vertices in V, where $|\mathcal{E}| = \binom{|V|}{2}$. If at each time-step a proportion p of the active edges are deleted and a proportion q of the non-active edges are added then

$$\begin{aligned} |E_{t+1}| &= |E_t| - |E^-_{t+1}| + |E^+_{t+1}| \\ &= |E_t| - p|E_t| + q(|\mathcal{E}| - |E_t|). \end{aligned}$$

If we want the density of $\mathcal{G}$ to remain approximately equal to d over all time-steps then it is necessary for $|E_t| \approx d|\mathcal{E}|$ for all t. The above equation therefore becomes

$$|E_{t+1}| = d|\mathcal{E}| - pd|\mathcal{E}| + q(|\mathcal{E}| - d|\mathcal{E}|).$$

Setting $|E_{t+1}| = d|\mathcal{E}|$ and rearranging gives $q = \frac{pd}{1-d}$. Hence, if each edge in E_t is added to E^-_{t+1} with probability p, then each non-active edge in $\mathcal{E} \backslash E_t$ should be added to E^+_{t+1} with probability $\frac{pd}{1-d}$.

As q is a probability, it must hold that $0 \leq q \leq 1$. It therefore follows that p and d must satisfy $\frac{pd}{1-d} \leq 1$ and $p \leq \frac{1-d}{d}$. Because p is also a probability (satisfying $0 \leq p \leq 1$) these inequalities can only be reasonably violated when $d > 0.5$. Figure 3.3 clearly illustrates the legal combinations of p and d (represented by the shaded area). All combinations of the parameter values of our test instances satisfy these inequalities.

3.4.2 Algorithm Parameters

For our experiments we used RLF [Leighton, 1979] as our constructive operator. This constructive operator is used for producing initial colourings for G_0 of each test instance (i. e. RLF replaces Line 2 in Algorithm 3.2 for G_0) and also within Method 4. Of the three constructive algorithms discussed in Section 2.1, RLF was shown to produce colourings with the fewest number of colour classes though, in many cases, it also requires the most time to run.

TabuCol [Hertz and de Werra, 1987] and PartialCol [Blöchliger and Zufferey, 2008] are implemented to tackle the k-GCPs within our approach (i. e. Line 5 of Algorithm 3.2) when operating in the complete, improper and partial, proper solution spaces respectively. These algorithms operate as described in Sections 2.4.1 and 2.4.2 and include an aspiration criterion that allows a "tabu" move to be selected and executed if the resultant colouring has fewer clashes or "uncoloured" vertices than the "best" colouring observed up until that iteration. With regards to TabuCol, only the *critical* 1-moves are considered, in a similar way that only the i-moves for currently "uncoloured" vertices are considered by PartialCol.

During execution, k is adjusted in the following way: if a feasible k-colouring, where the initial value of k is defined by the modification operator, cannot be obtained within half of the allotted time limit then k is increased by 1. If a feasible k-colouring cannot then be obtained within half of the remaining time limit then k is again increased by 1, and so on (i. e. Lines 9 and 10 of Algorithm 3.2). Less time is allocated to solving the k-GCP for higher value of k because, in general, it should be easier to identify a feasible k-colouring as k increases. This adjustment is particularly useful if a feasible k-colouring for G_t is passed to Algorithm 3.2 such that $k < \chi(G_{t+1})$ (see Section 3.2).

We used a time limit of 10 seconds[1] per time-step (i. e. Line 4 in Algorithm 3.2). If this time limit had been set much longer, say hours, then the advantage of modifying colourings between time-steps obviously diminishes.

It should be noted that Method 1 exclusively produces complete, improper colourings, and Methods 2 and 3 exclusively produce partial, proper colourings. On the other hand, Methods 0 and 4 produce feasible colourings which can then be passed to a tabu search operator that operates within either solution space, as required. Therefore, only comparisons between modification operators that produce colourings within the same solution space or feasible colourings will be compared (e. g. Methods 1 and 2 will not be compared against one another).

[1] All algorithms were programmed in C++ and executed on a 3.3GHZ Windows 7 PC with an Intel Core i3-2120 processor and 8GB RAM.

3.5 Results

In this section we present an analysis of our experimental results. The majority of our data was found to be non-normally distributed, therefore non-parametric statistical techniques are employed. Unless stated otherwise, all pairwise statistical comparisons are based on the Wilcoxon signed rank test with significance level $\alpha = 0.01$. Results refer to the graphs $G_1, \ldots, G_T$ of each dynamic graph $\mathcal{G} = \{G_0, G_1, \ldots, G_T\}$, with G_0 ignored because there are no colouring from a previous time-step to be modified. As such, the results will refer to single time-steps rather than the entire dynamic graph (e. g. $|S_t|$ for G_t will be reported rather than $\sum_{i=1}^{T} |S_i|$).

3.5.1 Initial Colourings

Let us first consider the initial feasible colourings produced for the edge dynamic GCP. For all densities d and change probabilities p, Methods 1 to 3 were found to produce initial, feasible colourings with significantly fewer colour classes than both Methods 0 and 4. This is clearly illustrated in Figure 3.4. However we also see that there is a significant increase in the time required by Methods 1 to 3 to achieve their initial, feasible colourings compared to Methods 0 and 4 for all values of d and p, as seen in Table 3.1. A main contributing factor to this is found in the nature of the different methods: Methods 0 and 4 both start from feasible colourings, whereas Methods 1 to 3 do not and therefore require more time to move to a feasible region of the solution space. For similar reasons, as p increases, so too does the time required by Methods 1 to 3 to achieve an initial, feasible colouring.

For $d = 0.1$ with $p = 0.005$, $d = 0.5$ with $p \leq 0.02$, and $d = 0.9$ with $p \leq 0.01$ Method 4 was found to produce initial, feasible colourings with significantly fewer colour classes than Method 0. However, for higher settings of p, specifically for $d = 0.1$ with $p \geq 0.01$, $d = 0.5$ with $p \geq 0.03$, and $d = 0.9$ with $p \geq 0.015$, the opposite holds. This is again clearly illustrated in Figure 3.4. Hence we can conclude that for these high levels of p, modifying feasible colourings for G_t via Method 4 is of no benefit when attempting to achieve initial, feasible colourings for G_{t+1}.

Considering computational effort, we found that the time required by Method 4 to achieve initial, feasible colourings is significantly less compared to Method 0 for $d \in \{0.5, 0.9\}$ with all values of p. Both Methods 0 and 4 employ RLF; however, Method 0 applies it to the whole graph $G_t = (V, E_t)$ at each time-step t as opposed to Method 4 which only applies it to a residual graph $\tilde{G} = (\tilde{V}, \tilde{E})$ of G_t where $\tilde{V} \subseteq V$ (which implies $|\tilde{V}| \leq |V|$). We therefore see that applying Method 4 with low levels of p is advantageous with regards to both the number of colour classes in initial, feasible colourings and the time required to obtain them.

With the exception of Table 3.2, all of the results, figures and tables presented so far are for test instances with $n = 500$. In general, the relationships observed

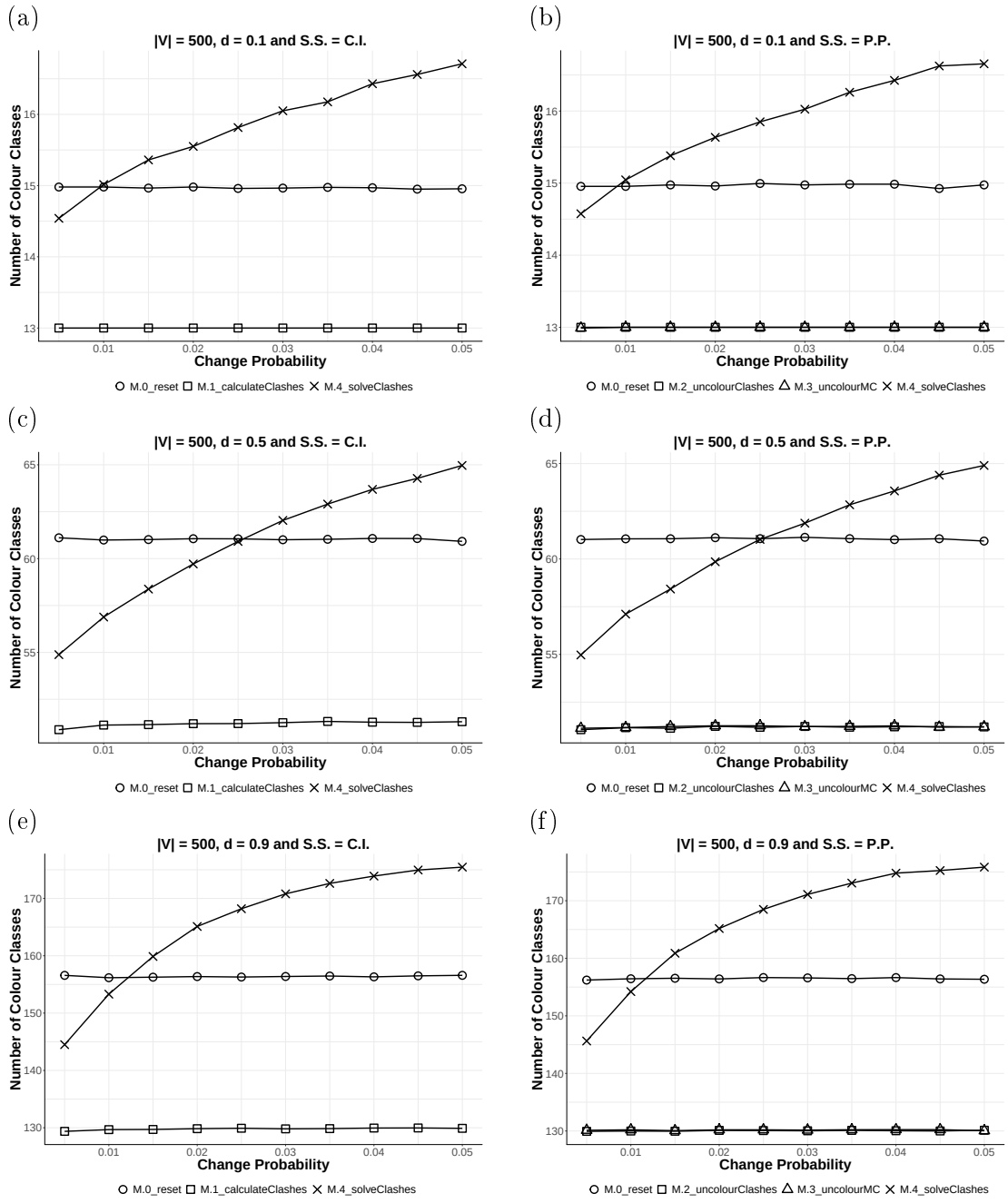

Figure 3.4: Mean number of colour classes in initial, feasible colourings for the edge dynamic GCP on test instances with $|V| = 500$. Graphs (a), (c) and (e) represent methods that operate in the complete, improper solution space, graphs (b), (d) and (f) represent methods that operate in the partial, proper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively (e. g. graphs (a) and (b) represent test instances with $d = 0.1$).

Table 3.1: Median time (in seconds) required to obtain an initial, feasible colouring for the edge dynamic GCP on test instances with $|V| = 500$. The second column indicates the method (M.) that was implemented.

		p									
d	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	0	0	0	0	0	0	0	0	0	0	0
	1	0	0.015	0.016	0.031	0.031	0.031	0.031	0.047	0.047	0.047
	2	0	0	0.015	0.015	0.015	0.015	0.015	0.016	0.016	0.016
	3	0	0	0.015	0.016	0.016	0.016	0.016	0.031	0.031	0.031
	4	0	0	0	0	0	0	0	0	0	0
0.5	0	0.016	0.016	0.031	0.031	0.031	0.016	0.016	0.016	0.016	0.016
	1	1.692	2.246	2.777	2.948	3.182	3.268	3.363	3.791	4.181	3.713
	2	1.545	1.872	2.083	2.996	2.325	2.590	2.824	2.519	2.730	2.972
	3	1.794	2.012	2.083	2.730	2.504	2.840	2.605	2.855	3.066	2.886
	4	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$
0.9	0	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031
	1	5.008	5.125	5.335	5.140	5.288	5.421	5.366	5.171	5.327	5.304
	2	4.376	4.235	5.016	5.070	5.047	5.031	5.008	4.789	5.038	5.023
	3	4.290	4.938	5.047	5.007	5.039	5.054	5.016	5.039	5.008	4.961
	4	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$	$0^\star$

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than all others for the same values of d and p.

Table 3.2: Significant differences between the number of colour classes in the initial colourings achieved by Methods 0 and 4. The third column indicates the solution space (S.S.) that Methods 0 and 4 are operating in.

			p									
n	d	S.S.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	C.I.	X	-	O	O	O	O	O	O	O	O
		P.P.	X	O	O	O	O	O	O	O	O	O
	0.5	C.I.	X	X	X	X	-	O	O	O	O	O
		P.P.	X	X	X	X	-	O	O	O	O	O
	0.9	C.I.	X	-	O	O	O	O	O	O	O	O
		P.P.	X	-	O	O	O	O	O	O	O	O
500	0.1	C.I.	X	-	O	O	O	O	O	O	O	O
		P.P.	X	-	O	O	O	O	O	O	O	O
	0.5	C.I.	X	X	X	X	-	O	O	O	O	O
		P.P.	X	X	X	X	-	O	O	O	O	O
	0.9	C.I.	X	X	O	O	O	O	O	O	O	O
		P.P.	X	X	O	O	O	O	O	O	O	O
1000	0.1	C.I.	X	O	O	O	O	O	O	O	O	O
		P.P.	X	X	-	O	O	O	O	O	O	O
	0.5	C.I.	X	X	X	X	O	O	O	O	O	O
		P.P.	X	X	X	X	O	O	O	O	O	O
	0.9	C.I.	X	X	O	O	O	O	O	O	O	O
		P.P.	X	X	O	O	O	O	O	O	O	O

X - indicates that Method 4 achieves an initial colouring with significantly fewer colour classes than Method 0.
O - indicates that Method 4 achieves an initial colouring with significantly more colour classes than Method 0.

between the different modification operators were seen to be very similar regardless of the size of the test instances, as can be seen by comparing Figure 3.4 and Table 3.1 with Figures B.1 and B.2 and Table B.1 in Appendix B, respectively. The only difference of note is that Method 4 does not clearly require the least amount of time to achieve initial feasible colourings for test instances with $n = 250$. This is likely due to the relatively small time values recorded for these test instances, which may not be accurate enough for reliable conclusions to be drawn.

Method 2 (*uncolourClashes*) vs. Method 3 (*uncolourMostClashing*)

We have observed that Methods 2 and 3 behave in a very similar fashion when compared against Methods 0 and 4. Now we will compare Methods 2 and 3 against one another to explore whether the manner in which vertices are "uncoloured" between time-steps has any affect on the initial colourings achieved. It is hypothesised that the more sophisticated manner employed by Method 3, which reduces the initial number of "uncoloured" vertices, should therefore reduce the amount of time required by the tabu search operator to subsequently achieve a feasible colouring.

At the $\alpha = 0.01$ significance level, we found that there was no significant difference between the number of colour classes in the initial, feasible colourings achieved when using either modification operator on most test instances. However, for a few test instance Method 2 achieved initial, feasible colourings with significantly fewer colour classes than Method 3. More specifically, this occurred on test instance with the following parameter combinations: $n = 500$, $d = 0.9$, $p \in \{0.01, 0.04, 0.045\}$, and $n = 1000$, $d = 0.5$, $p = 0.03$.

With the exception of these few test instances, it stands to reason that the initial number of "uncoloured" vertices should not affect the number of colour classes in the initial, feasible colourings achieved. Indeed, for both Methods 2 and 3, the initial target number of colour classes is determined by the "best" colouring achieved in the previous time-step (see descriptions in Section 3.3) and not the modification operator itself.

With regards to computational effort, we observed no significant difference in the time required to achieve an initial feasible colouring between the two methods in most cases. Nor does there appear to be any significant difference in the number of iterations of PartialCol required to achieve an initial feasible colouring in most cases. However, there are a few exceptions which illuminate the behavioural difference between Methods 2 and 3.

For test instances with $n \in \{250, 500\}$ and $d = 0.1$, significantly more time is required to achieve an initial, feasible colouring when using Method 3. However, there is no significant difference in the the number of iterations of PartialCol required to identify a feasible colouring from the initial partial, proper colourings produced by Methods 2 and 3. Both of these observations are clearly illustrated in

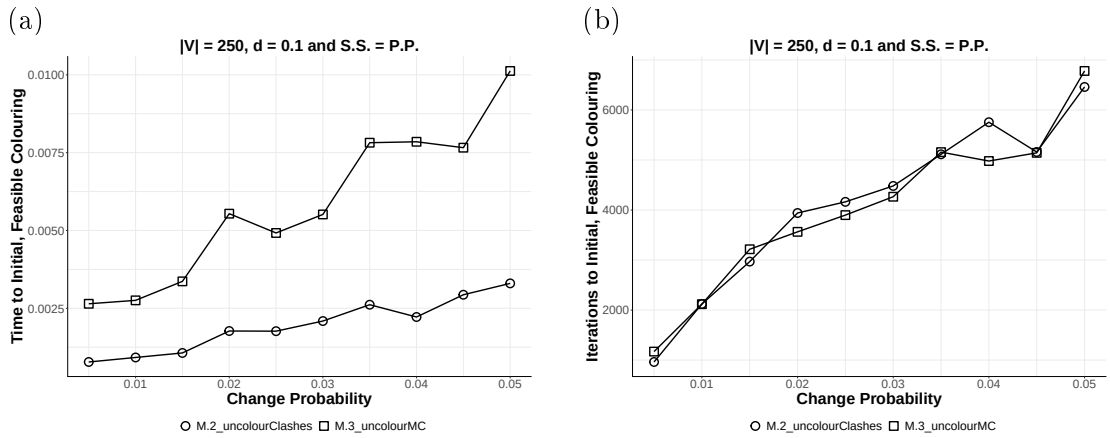

Figure 3.5: Mean time (a) and median number of PartialCol iterations (b) required to achieve an initial, feasible colouring for the edge dynamic GCP on test instances with $|V| = 250$ and $d = 0.1$.

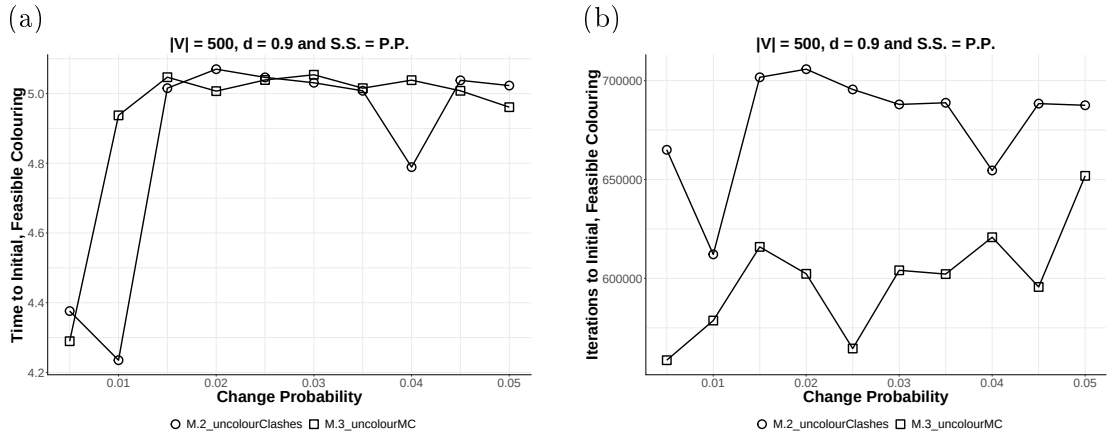

Figure 3.6: Median time (a) and number of PartialCol iterations (b) required to achieve an initial, feasible colouring for the edge dynamic GCP on test instances with $|V| = 500$ and $d = 0.9$.

Figure 3.5. We can therefore conclude that the additional time required when using Method 3 is likely due to the modification operator itself, which is more complex in its transformation of $\mathcal{S}_t$ to $\mathcal{S}_{t+1}$.

For test instances with $n = 500$ and $d = 0.9$ (see Figure 3.6) there is no significant difference in the time required by Methods 2 and 3 to achieve an initial, feasible colouring, but PartialCol requires significantly fewer iterations to do so when Method 3 is implemented. This would again suggest that Method 3 requires more execution time than Method 2.

3.5.2 Final Colourings

Let us now consider final colourings for the edge dynamic GCP. The Friedman test with $\alpha = 0.01$ shows that for test instances with $n = 250$ and all values of

d, and $n = 500, d = 0.1$ there is no significant difference between the number of colour classes in the final, feasible colourings achieved when applying any of the modification operators.

In a small number of cases, using Methods 1 to 3 led to final, feasible colourings with significantly fewer colour classes than Method 0. In all of these cases, the test instance parameters satisfy $n \in \{500, 1000\}, d \in \{0.5, 0.9\}$ and $p \leq 0.03$. However, in most cases there was either no significant difference or a significant increase when compared to Method 0. For example, Methods 1 to 3 all achieve final, feasible colourings with significantly more colour classes than Method 0 for test instances with $n = 1000$ and $d = 0.9$, with the exception of Methods 2 and 3 when $p = 0.005$. This continues to hold for Methods 1 and 3 on most test instances with $n = 500$ and $d = 0.9$. Methods 1 to 3 also achieve final, feasible colourings with significantly more colour classes than those achieved by Method 4 for some test instances with $n \in \{500, 1000\}$ and $d = 0.9$. Method 1 is also outperformed by Method 4 on some test instances with $n \in \{500, 1000\}$ and $d = 0.5$.

These observations are likely due to the relatively large amount of time required by Methods 1 to 3 to identify an initial, feasible colouring compared to Methods 0 and 4 as discussed in Section 3.5.1. For most test instances with $n = 500$ and $d = 0.9$, the median time required to achieve an initial feasible colourings is greater than 5 seconds (see Table 3.1), which indicates that the initial value of k generally needed to be increased before a feasible colouring could be identified in these cases. This "wasted" time then translates to time which cannot be allocated to identifying feasible colourings with fewer colour classes.

For some test instances with $n \in \{500, 1000\}, d \in \{0.5, 0.9\}$ and $p \leq 0.025$, Method 4 was found to achieve final, feasible colourings with significantly fewer colour classes than Method 0. This is unsurprising as Method 4 produces initial, feasible colourings with significantly fewer colour classes than Method 0 and requires significantly less time to do so under these parameter settings, which then allows more time to tackle k-GCPs with decreasing values of k whilst also starting with a smaller lower bound for k.

We now consider the computational effort required to achieve the "best" feasible colouring for each time-step. Here, the following time comparisons correspond only to time-steps where the number of colour classes in the final, feasible colourings achieved by the compared methods were equal to one another. Therefore, less data will be utilised for these comparisons due to the exclusion of time-steps for which the final colourings had differing numbers of colour classes.

Methods 1 to 3 were found to identify final, feasible colourings significantly faster than Method 0 for some test instances with $d \in \{0.1, 0.5\}$, as seen in Table 3.3 for test instances with $n = 500$. These methods were also able to reach final, feasible colourings significantly faster than Method 4 for test instances with $d = 0.1$

Table 3.3: Median time (in seconds) required to obtain final, feasible colourings with the same numbers of colour classes for the edge dynamic GCP on test instances with $|V| = 500$.

			p									
d	S.S.	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	C.I.	0	0.047	0.046	0.047	0.046	0.047	0.046	0.047	0.046	0.047	0.047
		1	$0^{\star}$	$0.015^{\dagger}$	$0.016^{\star}$	$0.031^{\dagger}$	$0.031^{\star}$	$0.031^{\star}$	$0.031^{\star}$	0.047	0.047	0.047
		4	$0.015^{\star}$	$0.016^{\star}$	$0.031^{\star}$	$0.031^{\star}$	0.046	0.031	0.047	0.047	0.047	0.047
	P.P.	0	0.031	0.016	0.031	0.016	0.031	0.031	0.031	0.031	0.031	0.031
		2	$0^{\dagger}$	$0^{\dagger}$	$0.015^{\dagger}$	$0.015^{\dagger}$	$0.015^{\star}$	$0.015^{\dagger}$	$0.015^{\dagger}$	$0.016^{\dagger}$	$0.016^{\dagger}$	$0.016^{\dagger}$
		3	$0^{\dagger}$	$0^{\dagger}$	$0.015^{\star}$	$0.016^{\star}$	$0.016^{\star}$	$0.016^{\star}$	$0.016^{\star}$	0.031	0.031	0.031
		4	$0^{\star}$	$0.015^{\star}$	$0.015^{\star}$	$0.015^{\star}$	$0.016^{\star}$	0.016	$0.016^{\star}$	0.031	0.031	0.031
0.5	C.I.	0	3.478	2.996	3.034	3.128	2.442	2.855	2.528	3.136	2.941	2.754
		1	$1.653^{\star}$	2.371	3.190	3.097	3.424	3.417	3.869	4.259	4.321	4.275
		4	$1.077^{\star}$	$1.794^{\star}$	$2.130^{\star}$	2.683	2.239	2.652	2.754	2.465	3.284	2.762
	P.P.	0	3.058	2.964	2.800	2.949	2.192	2.730	2.636	2.605	2.356	2.387
		2	1.872	1.918	2.160	3.620	2.247	2.714	3.120	2.574	2.512	2.972
		3	1.841	2.075	2.192	3.089	3.182	3.229	2.745	3.495	3.206	3.666
		4	$1.435^{\star}$	$2.278^{\star}$	$2.021^{\star}$	$2.372^{\star}$	2.246	2.870	2.184	2.340	2.067	2.551
0.9	C.I.	0	5.492	5.476	5.008	4.836	5.569	5.912	4.851	5.694	4.430	4.602
		1	6.225	7.122	7.691	7.074	7.964	7.550	7.535	8.455	7.176	7.488
		4	4.181	4.906	4.415	4.353	5.694	5.195	4.649	5.234	5.039	4.882
	P.P.	0	4.852	5.952	5.179	4.095	4.384	4.173	3.681	4.696	4.665	4.633
		2	5.117	5.343	6.505	6.224	5.257	5.203	5.710	5.483	5.577	5.928
		3	4.617	6.178	6.802	5.717	6.458	6.381	6.131	7.184	6.287	5.679
		4	3.885	4.805	5.663	4.064	4.352	4.228	3.775	4.087	5.117	4.914

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than Method 0 for the same values of d and p whilst operating in the same solution space.
† indicates a time that is significantly less than both Methods 0 and 4 for the same values of d and p whilst operating in the same solution space.

and some values of p. These observations are likely because the initial, feasible colourings achieved by Methods 1 to 3 are also the final, feasible colourings achieved for test instances with $d \in \{0.1, 0.5\}$ and low values of p. Results regarding the time requirements on test instances with $n \in \{250, 1000\}$ can be seen in Table B.2 in Appendix B.

On the other hand, Methods 1 to 3 were found to require significantly more time than Method 0 to achieve final, feasible colourings for test instances with $d = 0.5$ and high values of p, and $d = 0.9$ with most values of p. In a similar fashion, these three methods require significantly more time to achieve final, feasible colourings than Method 4 for test instances with $d \in \{0.5, 09\}$ and most values of p. This is probably due to the same arguments presented with regards to the number of colour classes in the final, feasible colourings achieved by these methods for $d = 0.9$. A clear illustration of how d and p affect the time required to achieve a final feasible colouring can be seen in Table 3.4.

Unlike Methods 1 to 3, Method 4 was found only to require significantly more time than Method 0. In contrast, for test instances with $d \in \{0.1, 0.5\}$ and $p \leq 0.035$

Table 3.4: Significant differences in the time required by Methods 0 and 3 to achieve final, feasible colouring with the same number of colour classes on test instances with $|V| = 500$.

	p									
d	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	O	O	O	O	O	O	O	-	-	-
0.5	-	-	-	-	-	-	-	-	X	X
0.9	-	-	X	X	X	X	X	X	X	X

X - indicates that Method 3 requires significantly less time than Method 0.
O - indicates that Method 3 requires significantly more time than Method 0.

or $p \leq 0.02$, respectively, Method 4 requires significantly less time to identify a final, feasible colourings. It should be highlighted that, as with the number of colour classes, these are similar parameter settings for which Method 4 is able to produce initial, feasible colourings with significantly fewer colour classes than Method 0.

Method 2 (*uncolourClashes*) vs. Method 3 (*uncolourMostClashing*)

As with the initial colourings, there is no significant difference in the number of colour classes in final feasible colourings achieved when implementing Methods 2 and 3. In a small number of test instances with $n = 500$ and $d \in \{0.5, 0.9\}$, Method 2 was able to achieve final feasible colourings with significantly fewer colour classes than Method 3. However, there appears to be no link between the number of colour classes in the initial feasible colouring achieved because none of the instances mentioned here are the same as those mentioned in Section 3.5.

Again, in most cases, no significant difference was observed between the time and number of PartialCol iterations required to reach the final feasible colourings. In the few cases where a significant difference was observed, it is for the same test instances for which the same significant difference was observed with regards to achieving an initial feasible colouring. For example, Method 2 requires significantly less time than Method 3 to achieve final feasible colourings for test instances with $n = 250, d = 0.1$, and some test instances with $n = 500, d = 0.1$. These are the same test instances where Method 2 requires significantly less time than Method 3 to achieve *initial* feasible colourings (see Figure 3.5).

In general, there appears to be no obvious preference with regards to using either Method 2 or 3 as the modification operator within our proposed approach. If forced to choose, Method 2 is easier to implement and requires significantly less time to achieve both initial and final feasible colourings for a handful of test instances.

3.6 Adding Empty Colour Classes

We saw in Section 3.5.1 that Methods 1 to 3 will achieve initial, feasible colourings with significantly fewer colour classes than Method 0, but require significantly more time to do so.

The optimisation that takes place during each time-step often results in Methods 1 to 3 passing an infeasible colouring $\mathcal{S}_{t+1}$ to a tabu search operator such that $|\mathcal{S}_{t+1}| = |\mathcal{S}_t|$ is a "good" approximation to $\chi(G_{t+1})$. The tabu search operator then attempts to solve the k-GCP for G_{t+1} where $k = |\mathcal{S}_{t+1}| \approx \chi(G_{t+1})$. As k becomes "closer" to $\chi(G_{t+1})$, there are fewer feasible k-colourings for G_{t+1} and the k-GCP becomes more difficult to solve (i. e. more time is required to identify a feasible k-colouring for G_{t+1}). Similarly, as k becomes greater than $\chi(G_{t+1})$, the associated k-GCP becomes easier to solve.

In our approach, if $k = |\mathcal{S}_t|$ is too low for a feasible k-colouring to be found for G_{t+1} then half of the time limit is "wasted" before k is increased. As can be seen in Table 3.1, this is the situation encountered for most test instances with $d = 0.9$.

In this regard, we now propose variations of Methods 1 and 2 such that the initial colouring $\mathcal{S}_{t+1}$ passed to the tabu search operator has more colour classes than the previous "best" colouring $\mathcal{S}_t$ for G_t (i. e. $|\mathcal{S}_{t+1}| > |\mathcal{S}_t|$) in an attempt to reduce the amount of time required to achieve an initial feasible colouring for G_{t+1}. Due to the similarity of the results between Methods 2 and 3, we omit Method 3 from these comparisons in an attempt to reduce the volume of results presented in this section.

3.6.1 Empty Colour Classes Approach

In this method, $\mathcal{S}_{t+1}$ and U are initially defined in the same manner as when using the respective modification operators given in Section 3.3. Then $\mathcal{S}_{t+1}$ is combined with $x > 0$ empty colour classes where x is a user-defined integer. When $\mathcal{S}_{t+1}$ is passed to the tabu search operator, along with U where appropriate, an attempt will be made to find a k-colouring for G_{t+1} where $k = |\mathcal{S}_t| + x$.

In the figures presented in this section, we will use the same names introduced previously to refer to the methods followed by "_x" to indicate the number of empty colour classes used (e. g. Method 1 with x empty colour classes will be referred to as *M.1_ calculateClashes_* x).

Here, we use the same test instances as described in Section 3.4.1 with $n = 500$, and all algorithm parameters also remain the same as in Section 3.4.2.

The number of empty colour classes x added to the colourings produced by Methods 1 and 2 are as follows:

- $x \in \{0, 1, 2\}$ for test instances with $d = 0.1$,
- $x \in \{0, 2, 4, 6\}$ for test instances with $d = 0.5$, and

- $x \in \{0, 3, 6, 9, 12\}$ for test instances with $d = 0.9$.

When applying the modification with empty colour classes approach, the number of empty colour classes x may be chosen such that a feasible colouring is identified in which some of the empty colour classes remain empty. If say, $0 < r \leq x$ of the empty colour classes remain empty, then the tabu search operator will report finding a feasible colouring with $k - r = |\mathcal{S}_t| + x - r$ colour classes as opposed to a k-colouring.

3.6.2 Results

As with Section 3.5, most of our data was found to be non-normally distributed. Therefore we continue to use non-parametric statistical techniques and, unless stated otherwise, all pair-wise statistical comparisons are based on the Wilcoxon signed rank test with $\alpha = 0.01$.

Initial Colourings

As can be seen in Figure 3.7, Methods 1 and 2 with empty colour classes still achieve initial, feasible colouring with significantly fewer colour classes than Methods 0 and 4 in some cases. The number of empty colour classes x for which this holds true depends on the density d of the test instance.

More specifically, for test instances with $d = 0.1$, implementing Methods 1 and 2 with $x = 1$ empty colour class achieves initial, feasible colouring with significantly fewer colour classes than Methods 0 and 4 for all values of p. However, with $x = 2$, Methods 1 and 2 only achieve initial colouring with significantly fewer colour classes than Method 0 for $p = 0.005$, but either no significant difference or significantly more colour classes for $p \geq 0.01$. This is illustrated in Figure 3.7(a,b) where the lines for Method 0 (i. e. the line of circles) is met by the lines for Methods 1 and 2 with $x = 2$ (i. e. the line of plus symbols). In comparison to Method 4, they achieve significantly more colour classes for $p = 0.005$, there is no significant difference for $p = 0.01$, and significantly fewer colour classes for $p \geq 0.015$, which is similar to the relationship observed between Methods 0 and 4 (see Section 3.5.1). This is again illustrated in Figure 3.7(a,b) where the lines for Method 4 (i. e. the line of crosses) is met and then crossed at $p = 0.01$ by the lines for Methods 1 and 2 with $x = 2$.

For test instances with $d \in \{0.5, 0.9\}$, Methods 1 and 2 achieved initial, feasible colouring with significantly fewer colour classes than Method 0 for all combinations of x and p tested, and for $d = 0.9$ this also holds against Method 4. For $d = 0.5$ with $(x = 4, p = 0.005)$ and $(x = 6, p = 0.01)$ there is no significant difference in the number of colour classes compared to Method 4 and significantly more for $x = 6$ and $p = 0.005$. This can be seen in Figure 3.7(c,d) where the lines for Method 4

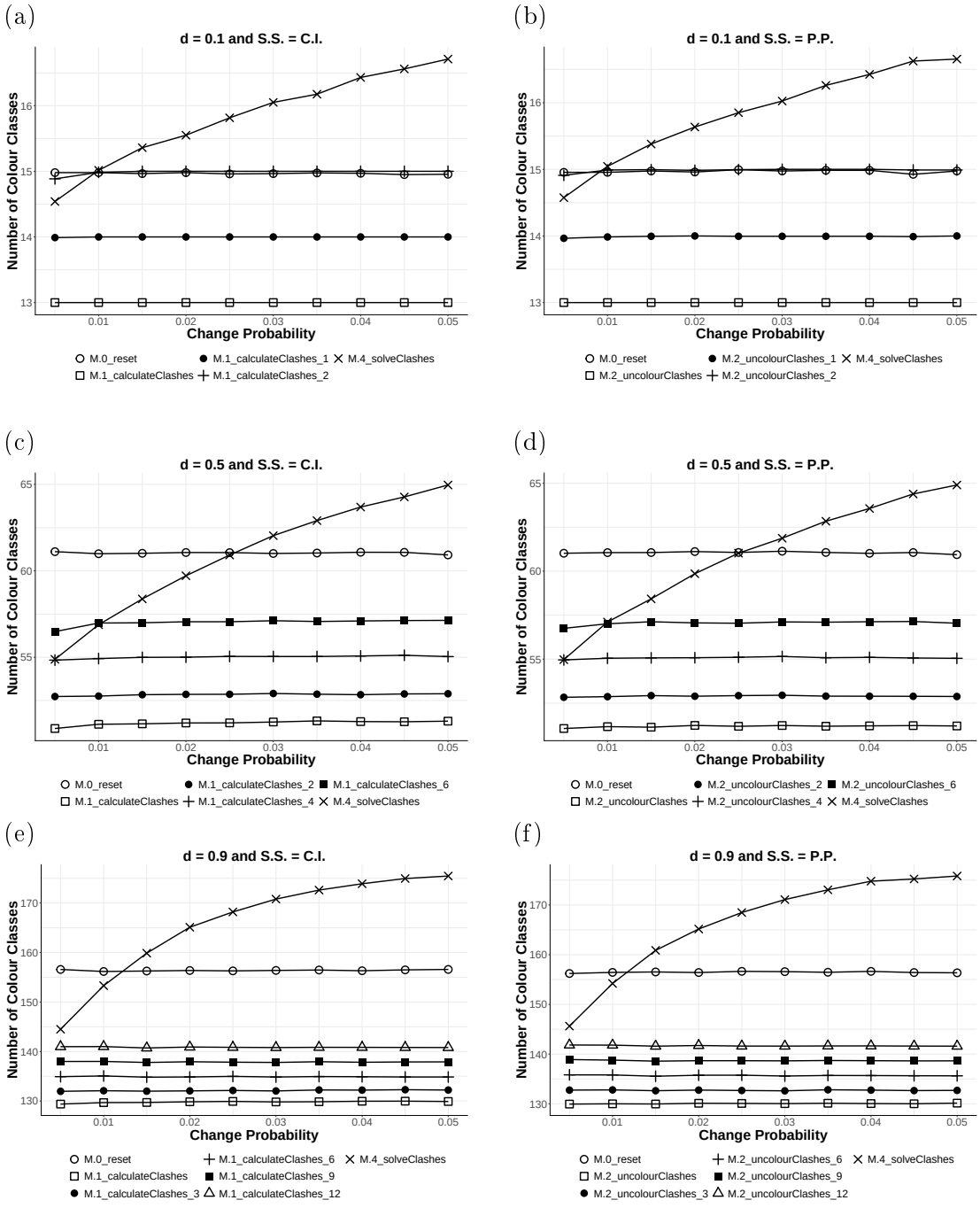

Figure 3.7: Mean number of colour classes in initial, feasible colourings for the edge dynamic GCP with Methods 1 and 2 including empty colour classes on test instances with $|V| = 500$. Graphs (a), (c) and (e) represent methods that operate in the complete, improper solution space, graphs (b), (d) and (f) represent methods that operate in the partial, proper solution space, and rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

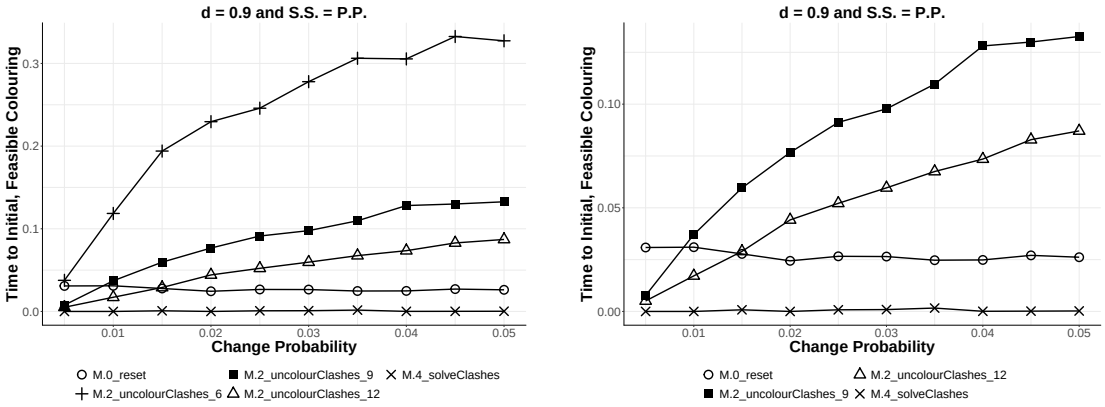

Figure 3.8: Mean time (in seconds) for Method 2 with $x \in \{6, 9, 12\}$ empty colour classes to achieve an initial, feasible colouring on test instances with $d = 0.9$.

(i. e. the line of crosses) is met and then crossed by the lines for Methods 1 and 2 with $x = 4$ and 6 (i. e. the lines of plus symbols and squares, respectively).

As was shown in Section 3.5.1, Methods 1 and 2 without empty colour classes (i. e. with $x = 0$) require significantly more time to achieve an initial, feasible colouring compared to Method 0 for all change probabilities p tested. However, this does not remain true once empty colour classes are included and the initial value of k is increased.

The amount of time required by Methods 1 and 2 with empty colour classes to achieve an initial, feasible colouring is now also dependent on the number of empty colour classes x, as well as the change probability p. The relationship between x and the amount of time required is negatively correlated such that as x increases, the amount of time required decreases. An example of this is illustrated in Figure 3.8 for Method 2 on test instances with $d = 0.9$.

For test instances with $d \in \{0.5, 0.9\}$[2] and some values of x, Methods 1 and 2 are able to achieve an initial, feasible colouring in significantly less time than Method 0. In fact, as x increases so too does the value of p for which this holds true (see Table 3.5). However, Methods 1 and 2 remain unable to identify an initial, feasible colouring in significantly less time than Method 4 for any values of x or p tested.

These results appear to confirm our proposition that the tabu search operator will require less time to identify a k-colouring for G_{t+1} where $k > |\mathcal{S}_t|$ and $\mathcal{S}_t$ is the colouring with the fewest colour classes identified for G_t.

Final Colourings

Somewhat surprisingly, the empty colour classes approach has a detrimental effect on the final feasible colourings achieved, with regards to both quality and computational

[2]For test instances with $d = 0.1$, the times required to achieve initial, feasible colourings were generally less than 10^{-3} seconds and therefore output as 0 seconds. Due to the inaccuracy of these outputs, any conclusions drawn from statistical comparisons would also be inaccurate.

Table 3.5: Significant differences in the time required by Method 1 with x empty colour classes to achieve an initial, feasible colouring compared against Method 0 on test instances with $d = 0.5$.

	p									
x	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0	O	O	O	O	O	O	O	O	O	O
2	X	-	O	O	O	O	O	O	O	O
4	X	X	X	X	X	X	-	O	O	O
6	X	X	X	X	X	X	X	X	X	X

X - indicates that Method 1 requires significantly less time than Method 0.
O - indicates that Method 1 requires significantly more time than Method 0.

effort, as x increases for most test instances, especially those with $d \in \{0.5, 0.9\}$.

For example, Methods 1 and 2 achieve final feasible colourings with significantly fewer colour classes than those achieved by Method 0, or no significant difference is observed (i.e. the colourings never have significantly more colour classes) for test instances with $d = 0.5$. However, Methods 1 and 2 with $x \geq 4$ empty colour classes achieve final feasible colourings with significantly more colour classes than those achieved by Method 0 for most test instances with $d = 0.5$. This increase in colour classes is similarly observed for $x \geq 3$ on test instances with $d = 0.9$.

With regards to the time required to achieve final feasible colouring with the same number of colour classes, there again appears to be a detrimental relationship with x such that as x increases so too does the time required. For example, Methods 1 and 2 with $x \in \{0, 1\}$ require significantly less time than Method 0 to achieve final feasible colourings for most test instances with $d = 0.1$, but the number of test instances for which this continues to hold drops-off when $x = 2$.

These observations seem contradictory to what might be expected because for higher values of x, Methods 1 and 2 achieve initial feasible colourings with significantly fewer colour classes and require significantly less time to do so compared to Method 0. Of course, it will naturally require more time to achieve a colouring with the fewest number of colour classes when starting from an initial colouring with more colour classes (i.e. $|\mathcal{S}_t| < |\mathcal{S}_t| + x$ for $x > 0$). Perhaps time is also being "wasted" identifying empty colour classes that remain within feasible colourings (i.e. if $\exists S_i \in \mathcal{S}_t$ such that $S_i = \emptyset$).

In general, it would appear that the empty colour classes approach yields improved results with regards to the initial feasible colourings achieved but these do not lead to improved results with regards to the final feasible colourings achieved.

3.7 Conclusions

In this chapter we have discussed edge dynamic graphs and introduced an associated graph colouring problem. We have proposed a modification approach for tackling

the edge dynamic GCP along with four modification operators (*calculateClashes, uncolourClashes, uncolourMostClashing* and *solveClashes*) to be used within this approach. These modification operators modify a feasible colouring for one time-step into an initial, though not necessarily feasible, colouring for the following time-step. These four modification operators were then compared against one another, where appropriate, and against an approach that treats each time-step of a dynamic graph as a static graph, i. e. Method 0 (*reset*).

Our experiments have shown that initial colourings with significantly fewer colour classes can be achieved by using Methods 1 to 3 (*calculateClashes*, *uncolourClashes* and *uncolourMostClashing*, respectively), which all modify a feasible k-colouring for G_t into an infeasible k-colouring for G_{t+1} and then pass this colouring directly to an optimisation operator (tabu search in our experiments). However, there is a significant trade-off with respect to the time required to achieve an initial, feasible colouring when these modification operators are applied. These operators were also found to achieve final, feasible colourings with both significantly fewer or significantly more colour classes depending on the density d of the test instance and its change probability p. In general, more beneficial results are observed for lower values of d and p, whereas the number of vertices n in a test instance has little affect, at least with regards to the relationships observed between modification operators.

It has also been shown that Method 4 (*solveClashes*), which modifies a feasible k-colouring for G_t into a feasible k'-colouring for G_{t+1} such that $k' \geq k$, can also achieve initial, feasible colourings with significantly fewer colour classes for small values of p. This modification operator was also shown to require significantly less time to produce initial, feasible colourings for all values of d and p. Finally, using this modification operator led to final, feasible colourings with the same or significantly fewer colour classes than Method 0 (*reset*) and requires significantly less time to do so for low values of d and p. Of the four modification operators introduced, Method 4 (*solveClashes*) was the only one that improved upon both the quality (i. e. number of colour classes) and effort (i. e. computation time) required compared to treating each time-step as an individual static graph in several cases, i. e. on test instances with low values of p.

In order to combat the trade-off between quality and computational effort observed for the first three modification operators, an empty colour classes approach was also proposed. This approach uses Methods 1 and 2 to modify a feasible k-colouring for G_t and then includes $x > 0$ empty colour classes such that the optimisation operator attempts to find a feasible colouring for G_{t+1} with $k + x$ colour classes. With appropriate values of x, this approach was able to achieve initial feasible colourings with significantly fewer colour classes than those produced by Method 0 and required less time to do so. In fact, as x increases, this approach is able to achieve initial feasible colourings significantly faster than Method 0 for

test instances with increasing change probabilities p. Unfortunately, as x increases, this approach appears to have a detrimental effect on the final feasible colourings achieved with regards to both quality and computational effort.

In comparison to Method 4, the empty colour classes approach achieved initial feasible colouring with significantly fewer colour classes than Method 0 dependent on x rather than p. However, in order to identify an initial feasible colouring in significantly less time than Method 0, the empty colour classes approach depends on p whereas Method 4 does not, at least not for $p \leq 0.05$.

Key Findings and Recommendations

- Methods 1 to 3 achieve initial feasible colourings with significantly fewer colour classes than Method 0 but require significantly longer to do so.
- For low values of p, Method 4 achieves initial feasible colourings with significantly fewer colour classes than Method 0 *and* significantly less time is required to do so.
- Adding an "appropriate" number of empty colour classes to the colouring produced by Methods 1 and 2 leads to initial feasible colourings with significantly fewer colour classes than Method 0 which are *also* achieved in significantly less time.
- For low values of p, we suggest using Method 4 over Methods 1 and 2 with empty colour classes, as the latter option requires extra knowledge concerning the "appropriate" number of empty colour classes to use.
- With regards to final feasible colourings, very few benefits can be gained from using our modification approach.

In the next chapter, we will look at how future information regarding the changes to the edge set between time-steps might be utilised to improve the robustness of a colouring to change.

Chapter 4

Colouring Edge Dynamic Random Graphs with Future Adjacency Information

Up to this point we have approached the edge dynamic GCP under the assumption that we have no prior information regarding future changes to the edge set. Now let us consider the case where we have some probabilistic information concerning how likely each non-active edge is to be included in the next time-step.

The focus of this chapter is to determine whether this information can be utilised in some beneficial way, e. g. to decrease the likelihood of clashes being introduced between time-steps. At each time-step t we wish to identify a feasible colouring for G_t whilst also trying to increase the robustness of this colouring with regards to G_{t+1}.

Consider the example illustrated in Figure 4.1. The dotted edges represent the non-active edges at time-step t that are "highly likely" to become active in time-step $t+1$. Although $\mathcal{S}_t$ and $\mathcal{S}'_t$ are both feasible 5-colourings for G_t, only $\mathcal{S}'_t$ will remain feasible at time-step $t+1$ if all of the dotted edges are included in E^+_{t+1}. The objective of this chapter is to determine a way of transforming $\mathcal{S}_t$ on the LHS into $\mathcal{S}'_t$ on the RHS without compromising feasibility for the current time-step at any stage of the transformation.

4.1 Future Adjacency

As with Chapter 3, let $\mathcal{G} = \{G_0, G_1, \ldots, G_T\}$ be a dynamic graph where G_t is the static representation of $\mathcal{G}$ at time-step t. For a graph $G_t = (V, E_t) \in \mathcal{G}$, let $p_{t+1}(u, v) = p_{t+1}(v, u)$ be the probability that $\{u, v\} \in E_{t+1}$. At time-step t, we say that vertices u and v are future adjacent with probability $p_{t+1}(u, v)$. If $p_{t+1}(u, v)$ is known for every possible edge $\{u, v\} \in \mathcal{E}$ then we can define the $|V| \times |V|$ future adjacency matrix P_{t+1} such that the (u, v)th entry of P_{t+1} is equal to $p_{t+1}(u, v)$.

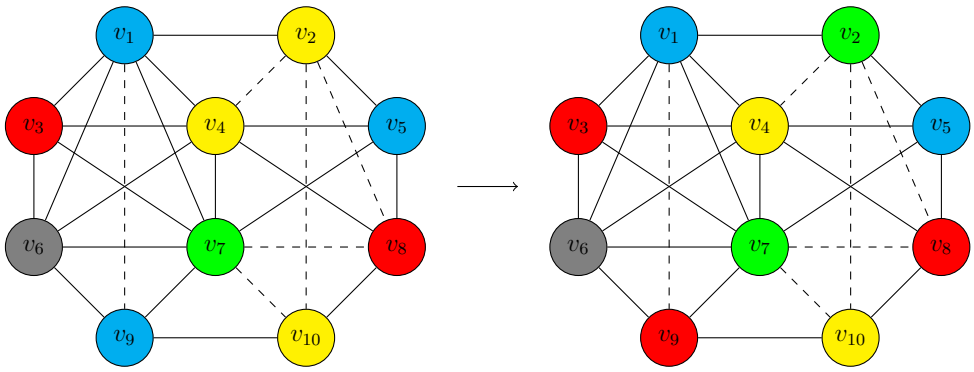

Figure 4.1: Edge dynamic graph with future adjacency information example. Solid lines represent edges that are included in E_t and dotted lines represent edges that are "highly likely" to be included in E^+_{t+1}.

Regardless of whether P_{t+1} is known, given the distribution of future adjacency probabilities for $\{u, v\} \in \mathcal{E} \backslash E_t$, we can estimate the number of clashes in a feasible k-colouring $\mathcal{S}_t$ (for G_t) at time-step $t + 1$ as

$$\mathcal{F}(\mathcal{S}_t) = \mathbb{E}(p_{t+1}) \cdot \sum_{i=1}^{k} \binom{|S_i|}{2} \tag{4.1}$$

where p_{t+1} is simply the distribution of future adjacency probabilities for edges in $\mathcal{E} \backslash E_t$ and $\mathbb{E}(p_{t+1})$ is the expected value of this distribution. If the number of vertices in each colour class is approximately equal (i. e. $|S_i| \approx \frac{n}{k}$ for $i = 1, \ldots, k$) then Equation (4.1) can be simplified to

$$\mathcal{F}(\mathcal{S}_t) = \mathbb{E}(p_{t+1}) \cdot \frac{n(n-k)}{2k} \tag{4.2}$$

It should be apparent that if P_{t+1} is unknown then, other than increasing k, little can be done with regards to minimising $\mathcal{F}(\mathcal{S}_t)$ because every edge needs to be treated as having the same future adjacency probability. On the other hand, if P_{t+1} is known then $\mathcal{F}(\mathcal{S}_t)$ is given by

$$\mathcal{F}(\mathcal{S}_t) = \sum_{i=1}^{k} \sum_{\substack{u,v \in S_i \\ u \neq v}} p_{t+1}(u, v) \tag{4.3}$$

which is more useful. Vertices in $\mathcal{S}_t$ can now be "recoloured" in an attempt to reduce Equation (4.3), without increasing k or violating the constraints of the GCP. In doing so, a more robust, feasible k-colouring for G_t can be produced that is likely to have fewer clashes in time-step $t + 1$ and therefore be "closer" to a feasible colouring for G_{t+1} also.

4.2 Two Stage Approach

As noted in Section 4.1, if P_{t+1} is known then we can also attempt to reduce the estimated number of clashes in $\mathcal{S}_t$ for time-step $t+1$ by reducing $\mathcal{F}(\mathcal{S}_t)$, given by Equation (4.3). The approach outlined in Algorithm 4.1 is a two stage approach that first attempts to find a feasible colouring for the current time-step with a user-defined target number of colour classes (Lines 1-10). It then attempts to reduce the estimated number of clashes for the following time-step (Lines 11-16).

Algorithm 4.1 Two Stage Approach for "Robust" Colourings

Input: a graph G_{t+1}, a target number of colour classes $k^\star$, a feasible colouring $\mathcal{S}_t$ for G_t such that $|\mathcal{S}_t| \geq k^\star$ and the future change information P_{t+2}
Output: a feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1} such that $|\mathcal{S}_{t+1}| \geq k^\star$

1: $\mathcal{S}_{\text{best}} \leftarrow \emptyset$
2: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t$ modified in some way (see Sections 3.3 and 5.2)
3: $k \leftarrow |\mathcal{S}_{t+1}|$
4: **while not** stopping criterion **and** $|\mathcal{S}_{\text{best}}| \neq k^\star$ **do**
5: Attempt to make $\mathcal{S}_{t+1}$ a feasible k-colouring for G_{t+1}
6: **if** $\mathcal{S}_{t+1}$ is a feasible k-colouring for G_{t+1} **then**
7: $\mathcal{S}_{\text{best}} \leftarrow \mathcal{S}_{t+1}$
8: $k \leftarrow k-1$
9: **if** $\mathcal{S}_{\text{best}} = \emptyset$ **and** a computational limit is reached **then**
10: $k \leftarrow k+1$
11: $\mathcal{F}_{\text{best}} \leftarrow \mathcal{F}(\mathcal{S}_{\text{best}})$ (see Equations (4.3) and (6.2))
12: **while not** stopping criterion **do**
13: Attempt to reduce $\mathcal{F}(\mathcal{S}_{t+1})$
14: **if** $\mathcal{F}(\mathcal{S}_{t+1}) < \mathcal{F}_{\text{best}}$ **then**
15: $\mathcal{S}_{\text{best}} \leftarrow \mathcal{S}_{t+1}$
16: $\mathcal{F}_{\text{best}} \leftarrow \mathcal{F}(\mathcal{S}_{t+1})$
17: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{\text{best}}$
18: **return** $\mathcal{S}_{t+1}$

Note that Stage 1 of this approach (Lines 1-10) is almost identical to Algorithm 3.2 with the exception that a user-defined, target number of colour classes $k^\star$ must also be provided. For Stage 2 (Lines 11-16) a tabu search operator is implemented that, at each iteration, attempts to reduce the expected number of future clashes. Specific details regarding the design and performance of this tabu search operator with be discussed in Section 4.3.

A two stage approach has previously been proposed by Thompson and Dowsland [1998] for solving exam timetabling problems which, as shown in Section 1.4, can be reformulated as GCPs. The first stage attempts to find a feasible timetable (i. e. one in which no student must attend more than one exam simultaneously), and the second stage attempts to optimise secondary objectives (e. g. reducing the number of consecutive exams for students). This is essentially the same as the approach outlined above, but here we only have one secondary objective: increase robustness by reducing $\mathcal{F}(\mathcal{S}_{t+1})$.

4.3 Method for Reducing Future Adjacency

The backbone of our method for increasing the robustness of a colouring at time-step t is the tabu search operator that attempts to reduce $\mathcal{F}(\mathcal{S}_t)$, the expected number of future clashes in time-step $t+1$.

The main challenge when attempting to reduce $\mathcal{F}(\mathcal{S}_t)$ is the requirement for $\mathcal{S}_t$ to remain feasible for G_t. As mentioned in Section 2.2.1, neighbourhood move operators designed to work in the feasible only solution space are useful here because we wish to tackle a secondary objective rather than reduce the number of colour classes. Therefore, we consider a tabu search operator that uses the Kempe-chain interchange and pair-swap move operators.

4.3.1 Design Process

Although it is clear what objective function and neighbourhood move operators should be used, in implementing our algorithm, experiments were conducted in order to determine a few other parameters (e. g. tabu tenure). In this section we discuss which options were considered and which options were ultimately chosen.

In all of the following discussions we compare the final expected number of future clashes for the "best" colourings achieved and the time required by these methods to complete 5000 iterations. To do this, we use a small set of test instances each comprising of a single graph G_0 and its future adjacency matrix P_1. These test instances have $n = 500$ vertices, a desired density of $d \in \{0.1, 0.5, 0.9\}$, and change probability $p \in \{0.01, 0.025, 0.05\}$. For each test instance, G_0 is constructed with $|V| = |V_0| = n$ and each edge $\{u, v\} \in \mathcal{E}$ is included in E_0 with probability d. The future adjacency matrix P_1 is populated with values sampled from the uniform distribution $U[0, \frac{2pd}{1-d}]$. Five test instances were produced for each combination of n, d and p, totalling 45 test instances.

Initial colourings for these test instances were produced using the DSatur algorithm [Brélaz, 1979] described in Section 2.1, therefore the colourings are guaranteed to be feasible. The number of colour classes remains fixed throughout the tabu search procedure and equals the number of colour classes in the initial colouring produced by DSatur.

Unless stated otherwise, all pair-wise comparisons were conducted using the Wilcoxon signed rank test due to the non-normality of most of the data. The level of significance is set at $\alpha = 0.01$.

"Tabu" Inverse Moves vs. "Tabu" Costs

Firstly, we wish to compare tabu search algorithms that consider either inverse moves or costs (i. e. values of $\mathcal{F}(\mathcal{S})$) for the tabu list. At each iteration, either the inverse moves are made "tabu" for a fixed number of subsequent iterations, or the

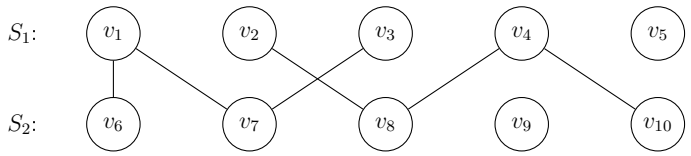

Figure 4.2: A feasible 2-colouring with the vertices in colour class S_1 displayed in the row above those in colour class S_2. This figure is identical to Figure 2.2.

resultant colouring's cost is made "tabu" such that no move operator that results in a colouring with a cost equal to any cost on the tabu list can be chosen[1] for a fixed number of subsequent iterations. Making inverse moves tabu will be referred to as the *TM* (i.e. tabu moves) method, and making costs tabu will be referred to as the *TC* (i.e. tabu costs) method.

When we refer to the inverse moves of the Kempe-chain interchange or pair-swap that transforms $\mathcal{S}$ into $\mathcal{S}'$, these are the labels of the associated Kempe-chain interchanges[2] that will transform $\mathcal{S}'$ back into $\mathcal{S}$. For example, consider the Kempe-chain interchange applied to $\textsc{Kempe}(v_1, 1, 2) = \{v_1, v_3, v_6, v_7\}$ in Figure 4.2. The inverse moves of this Kempe-chain interchange would be the Kempe-chain interchange applied to $\textsc{Kempe}(v_1, 1, 2)$, $\textsc{Kempe}(v_3, 1, 2)$, $\textsc{Kempe}(v_6, 2, 1)$, or $\textsc{Kempe}(v_7, 2, 1)$. The advantage of using the labels in this way is that the tabu list can be stored in a $|V| \times k$ matrix where $k = |\mathcal{S}| = |\mathcal{S}'|$ and the (i, j)th entry refers to the Kempe-chain interchange applied to $\textsc{Kempe}(v_i, c(v_i), j)$.

As this is the first comparison to be made, trials were conducted using two different methods of choosing a neighbourhood move (which was shown to have no influence here) and four different tabu tenures (50, 100, 250, and 500). Note that the tabu search procedure was unable to complete 5000 iterations for test instances with $d = 0.1$ when using a tabu tenure of 500. In this case, it is likely that a colouring is found such that every possible neighbourhood move is an inverse move of at least one of the previous 500 (or fewer) moves executed. Therefore, all neighbourhood moves are simultaneously "tabu" and the algorithm is unable to continue. Note, this issue is dealt with when implementing the final tabu search procedure in Section 4.5 by allowing random moves to be executed if such a scenario arises.

For almost all test instances and tabu tenures, *TM* was shown to identify final colourings with a significantly smaller number of expected future clashes compared to *TC* (see Table 4.1). The one exception to this is for test instances with $d = 0.9$ when using a tabu tenure of 50 iterations, in which case no significant difference was observed between the methods. *TM* makes the labels of Kempe-chains tabu rather than the Kempe-chains themselves, therefore as the colouring continues to be

[1] More specifically, no move operator that results in a colouring that is within $\pm 10^{-6}$ of any cost on the tabu list can be chosen.

[2] Remember that a pair-swap is the simultaneous execution of two Kempe-chain interchanges applied to $\textsc{Kempe}(v, c(v), c(u))$ and $\textsc{Kempe}(u, c(u), c(v))$ such that $|\textsc{Kempe}(v, c(v), c(u))| = |\textsc{Kempe}(u, c(u), c(v))| = 1$.

Table 4.1: Median expected number of future clashes when using *TM* and *TC* within the tabu search procedure. The first column indicates the fixed tabu tenure (T.T.) used.

			Best Move (*BM*)			Prioritise Swaps (*PS*)		
			p			p		
T.T.	d	M.	0.010	0.025	0.050	0.010	0.025	0.050
50	0.1	*TM*	7.367	18.455	36.986	7.809	19.458	38.957
		TC	7.697	19.357	38.539	8.673	21.673	43.237
	0.5	*TM*	13.961	34.534	70.181	14.071	35.117	69.225
		TC	14.790	37.125	74.250	14.684	37.065	72.552
	0.9	*TM*	32.735	81.056	165.663	32.010	82.869	171.194
		TC	34.003	85.006	170.013	31.654	82.581	165.586
100	0.1	*TM*	7.338	18.362	36.708	7.967	19.885	39.856
		TC	7.662	19.192	38.436	8.649	21.554	43.214
	0.5	*TM*	13.184	33.297	65.846	13.014	32.694	66.334
		TC	14.619	36.734	73.467	14.505	36.311	72.552
	0.9	*TM*	28.466	70.468	139.313	28.636	69.862	139.836
		TC	31.441	77.566	156.796	31.018	76.263	157.689
250	0.1	*TM*	7.332	18.331	36.801	7.685	19.287	38.446
		TC	7.612	19.070	38.284	8.590	21.513	43.056
	0.5	*TM*	12.701	31.641	63.765	12.685	31.392	63.833
		TC	14.409	36.137	71.719	14.357	35.377	71.129
	0.9	*TM*	25.153	63.107	126.631	24.316	62.833	126.804
		TC	28.798	72.079	142.704	28.951	70.508	134.021
500	0.1	*TM*	-	-	-	-	-	-
		TC	-	-	-	-	-	-
	0.5	*TM*	12.665	31.536	63.570	12.567	31.587	61.876
		TC	14.250	35.089	70.798	14.358	35.362	69.685
	0.9	*TM*	24.463	61.425	122.329	24.631	59.598	121.617
		TC	27.931	69.471	138.996	27.636	67.869	131.076

altered, the labels may no longer refer to the original Kempe-chains (i. e. the same sets of vertices that the labels referred to when they were made tabu). On the other hand, for *TC*, once a Kempe-chain changes such that the resultant cost is no longer on the tabu list it can be executed again. Therefore labelled moves remain tabu for longer under *TM* than *TC*, which means that more of the solution space may be explored under *TM*.

It was also shown that for graphs with $d \in \{0.1, 0.5\}$, the *TM* method was able to complete 5000 iterations significantly faster than the *TC* method. The same can also be shown for test instances with $d = 0.9$ when using a tabu tenure of 250 or 500 iterations. However, when using a tabu tenure of 50 or 100 iterations on the same instances, it can be seen that the *TC* method requires significantly less time to complete 5000 iterations compared to the *TM* method. These results can be observed in Table 4.2. The general increase in time required by *TC* is likely caused by the way in which the tabu list is stored compared to *TM*. In *TM*, a $|V| \times k$ matrix stores the tabu list such that determining whether a move applied to KEMPE$(v_i, c(v_i), j)$ is tabu is as simple as looking up the (i, j)th entry of this matrix. On the other hand, *TC* stores the tabu costs as an ordered list equal in size to the tabu tenure. In order to determine whether a move is tabu under *TC* requires comparing the cost of the move's resultant colouring against the costs on

Table 4.2: Median time (in seconds) required for the tabu search procedure to complete 5000 iterations when using *TM* and *TC*.

			Best Move (*BM*)			Prioritise Swaps (*PS*)		
			p			*p*		
T.T.	*d*	M.	0.010	0.025	0.050	0.010	0.025	0.050
50	0.1	*TM*	5.175	5.124	5.135	4.942	4.961	4.938
		TC	10.670	10.655	10.538	6.309	6.306	6.325
	0.5	*TM*	4.706	4.705	4.697	4.683	4.717	4.705
		TC	6.040	5.874	5.853	5.968	5.853	5.921
	0.9	*TM*	6.625	6.718	6.725	6.620	6.767	6.711
		TC	6.026	5.990	6.011	6.084	6.031	6.123
100	0.1	*TM*	5.106	5.108	5.103	4.979	4.983	4.944
		TC	16.324	16.190	16.202	7.906	7.965	8.043
	0.5	*TM*	4.725	4.740	4.717	4.702	4.734	4.737
		TC	7.457	7.442	7.318	7.583	7.475	7.214
	0.9	*TM*	6.613	6.697	6.677	6.608	6.729	6.755
		TC	6.338	6.338	6.332	6.483	6.512	6.450
250	0.1	*TM*	5.092	5.114	5.064	5.101	5.093	5.090
		TC	31.725	30.601	31.198	13.172	13.012	12.665
	0.5	*TM*	4.766	4.764	4.780	4.759	4.786	4.792
		TC	11.846	11.841	11.501	10.625	10.702	10.399
	0.9	*TM*	6.599	6.719	6.715	6.672	6.753	6.732
		TC	7.333	7.343	7.326	7.439	7.372	7.258
500	0.1	*TM*	-	-	-	-	-	-
		TC	-	-	-	-	-	-
	0.5	*TM*	4.867	4.852	4.830	4.849	4.853	4.850
		TC	18.110	17.629	17.496	15.527	14.475	14.133
	0.9	*TM*	6.649	6.727	6.719	6.660	6.794	6.741
		TC	8.240	8.364	8.281	8.320	8.230	7.968

the ordered list, which clearly requires far more checks compared to *TM*.

It was decided that the *TM* method (i. e. making inverse moves "tabu") would be used within our tabu search procedure because it was shown to outperform the *TC* method with regards to the quality of colourings and time requirements in almost all cases tested. The few cases in which the *TC* method was able to outperform the *TM* method with regards to run-time are associated with tabu tenures that are unlikely to be used in practice (see later discussion).

Best Move vs. Prioritise Pair-Swap

Now, we compare two different methods for choosing which neighbourhood move is to be executed during each iteration of the tabu search procedure. The first method chooses the best of all available non-tabu neighbourhood moves (i. e. the Kempe-chain interchange or pair-swap that will result in the best cost change). The second method prioritises pair-swaps, such that if a pair-swap is available (i. e. not all pair-swaps are currently "tabu") then this will be chosen over all possible Kempe-chain interchanges even if an available Kempe-chain interchanges would result in a better cost change. This second method was considered because preliminary trials indicated that pair-swaps were utilised far less often than Kempe-chain interchanges. The first method will be referred to as the best move (*BM*) method and the second

will be referred to as the prioritise swaps (PS) method.

As with the previous comparison, trials were conducted using four different tabu tenures (50, 100, 250, and 500). Results presented are for trials using the TM method from the previous discussion.

For test instances with $d = 0.1$, the BM method was able to achieve colourings with a significantly smaller number of expected future clashes compared to the PS method. On the other hand, no significant differences were observed for test instances with $d \in \{0.5, 0.9\}$. Again, this can be seen in Table 4.1. Note that here, the tabu tenure used within the tabu search procedure has no observable influence on the results.

With regards to the time required to complete 5000 iterations, the PS method requires significantly less time than the BM method on some test instances with $d = 0.1$ when using a tabu tenure of 50 or 100 iterations. In all other cases, there is no significant difference in the time required by either method (see Table 4.2).

It was decided that the BM method (i.e. choose the best Kempe-chain interchange or pair-swap) would be used within our tabu search procedure because it was shown to have no significant difference or to outperform the PS method with regards to the quality of the colourings it achieves. For the cases in which the PS method was able to outperform the BM method with regards to run-time, these cases are again associated with tabu tenures that are unlikely to be used in practice (see discussion below).

Tabu Tenure Length

Finally, we compare different tabu tenures for our tabu search procedure. During each iteration, the neighbourhood move (or moves) that would reverse the most recently executed neighbourhood move are made "tabu" for a fixed number of subsequent iterations defined by the given tabu tenure.

Results presented are for trials using the TM and BM methods from the previous discussions.

Using the Friedman test with $\alpha = 0.01$, no significant difference was found for the number of expected future clashes when using different tabu tenures on test instances with $d = 0.1$. On the other hand, significant and negative correlations are observed between tabu tenure and the number of expected future clashes on test instances with $d \in \{0.5, 0.9\}$ (i.e. as tabu tenure is increased, the expected number of future clashes decreases). The corresponding correlation coefficients range between -0.469 and -0.557 for test instances with $d = 0.5$ and between -0.714 and -0.766 for test instances with $d = 0.9$.

In general, there is a negative relationship between the tabu tenure used and the expected number of future clashes of the colourings achieved by the tabu search procedure. This is to be expected as longer tabu tenures reduce the probability of

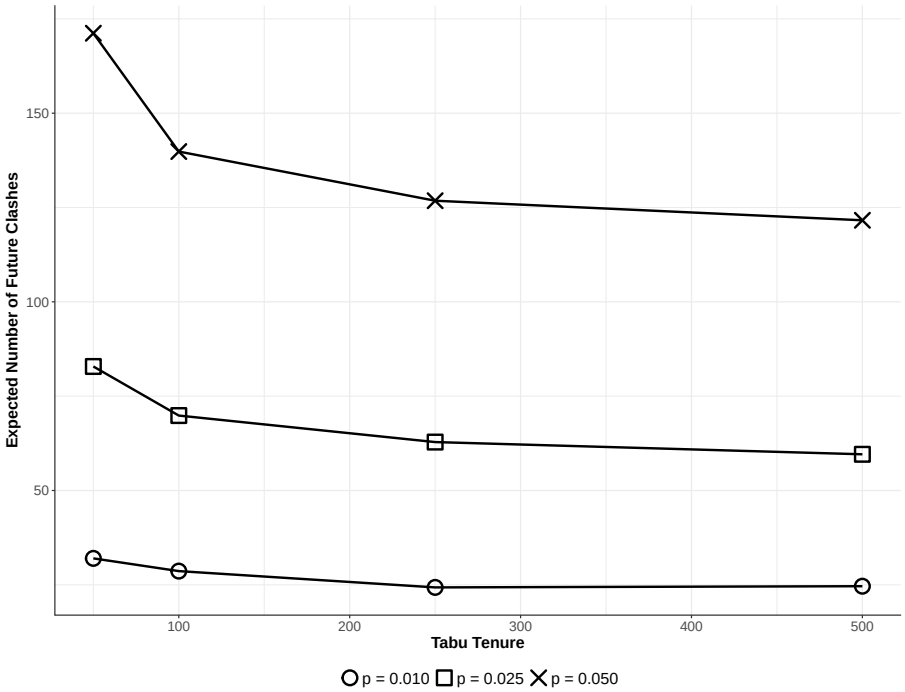

Figure 4.3: Median expected number of future clashes of the "best" colouring achieved via our tabu search procedure against tabu tenure for test instances with $d = 0.9$.

"cycling" and increase the exploration of the solution space. However, as can be seen in Figure 4.3, there seems to be a diminishing return with regard to this relationship.

For most test instances with $d = 0.1$, no significant difference was observed in the time required to complete 5000 iterations when using different tabu tenures. In the cases where a significant difference was observed, for test instances $p = 0.01$, a negative correlation with coefficient -0.535 was also observed. Conversely, for test instances with $d = 0.5$, significant positive correlation was observed (i.e. as the tabu tenure increases so too does the time required to complete 5000 iterations), as shown in Figure 4.4. For these test instances, the correlation coefficients are in the range of 0.514 to 0.553. Tabu tenure appears to have no significant effect on the run-times for test instances with $d = 0.9$.

It is unclear why there is a trade-off between quality and the time required to complete 5000 iterations on test instances with $d = 0.5$. The number of checks (i.e. checking whether a move is "tabu" or not) should remain the same regardless of the tabu tenure as all moves must be considered either way. In fact, one would hypothesise that there would be a negative relationship between run-time and tabu tenure as only the costs of "non-tabu" moves are compared for selection.

For the following experiments, we decided to use a higher tabu tenure because the main objective of this tabu search procedure is to decrease the number of expected future clashes. Because we can see that there is a diminishing return for increasing the tabu tenure, we have opted for 250 iterations as it is suitable for all test instances

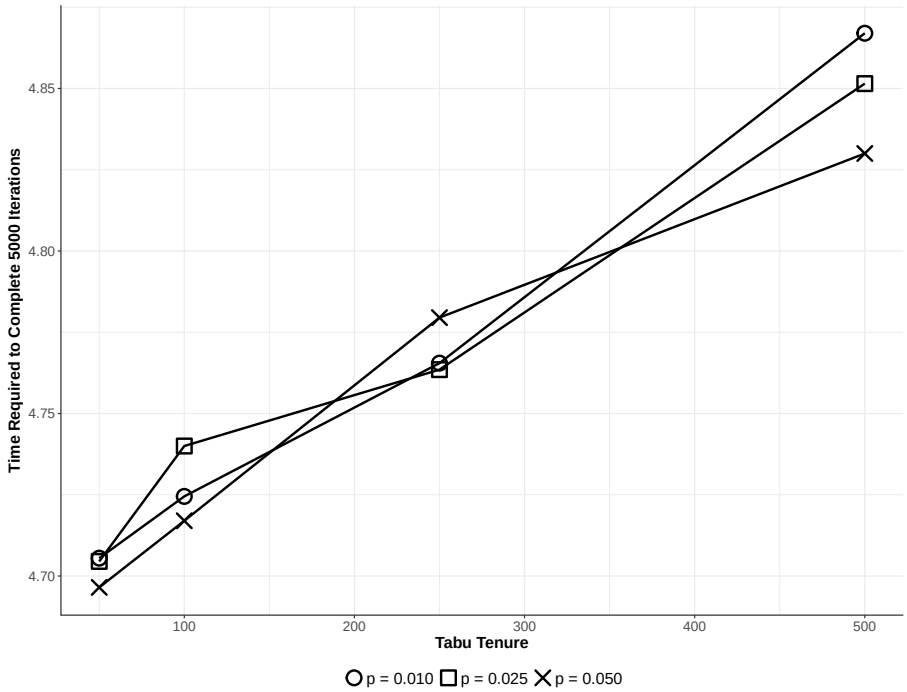

Figure 4.4: Median time (in seconds) required to complete 5000 iterations of our tabu search procedure against tabu tenure for test instances with $d = 0.5$.

regardless of their desired density d.

4.3.2 Performance and Behaviour Analysis

In this section we now explore how our tabu search procedure behaves with regards to reducing the expected number of clashes in a subsequent time-step.

To do this we use test instances consisting of a single graph G_0 and its future adjacency matrix P_1. These test instances have $n = 500$ vertices, desired densities $d \in \{0.1, 0.5, 0.9\}$, and change probability $p = 0.05$. For each test instance, G_0 is constructed with $|V| = |V_0| = n$ and each edge $\{u, v\} \in \mathcal{E}$ is included in E_0 with probability d. The future adjacency matrix P_1 is populated with values sampled from the uniform distribution $U[0, \frac{2pd}{1-d}]$. Twenty test instances were produced for each combination of n, d and p, totalling 60 test instances.

Unlike the previous comparisons in Section 4.3.1, the number of colour classes in a colouring will be user-defined as opposed to being set by the colourings produced by DSatur. By doing this, it will be easier to determine the relationship between the number of colour classes in a colouring and its expected number of future clashes. Specifically, we use $k^\star = 13, 14, 15, \ldots, 23$ for test instances with $d = 0.1$, $k^\star = 49, 51, 53, \ldots, 79$ for test instances with $d = 0.5$, and $k^\star = 125, 128, 131, \ldots, 185$ for test instances with $d = 0.9$.

An initial colouring is produced using a modified version of the Greedy algorithm discussed in Section 2.1. The Greedy algorithm is used so that the vertices

will occupy as many of the $k^\star$ colour classes as possible before our tabu search procedure is applied (i. e. we wish to reduce the number of empty colour classes in the initial colourings). This modified version, outlined in Algorithm 4.2, attempts to produce a feasible colouring for a given graph in a "greedy" fashion so long as the number of colour classes does not exceed $k^\star$. If it is not possible to feasibly colour a vertex without introducing a $(k^\star + 1)$th colour class then it is placed in a set of "uncoloured" vertices U. If $|U| > 0$ after each vertex has been considered, then each vertex in U is randomly assigned to one of the first $k^\star$ colour classes, which will introduce a total of $|U|$ or more clashes. If a colouring is produced such that the number of colour classes is less than $k^\star$, then an appropriate number of empty classes are added to this colouring.

Algorithm 4.2 Modified GREEDY Algorithm

Input: a graph $G = (V, E)$, an ordering of the $|V|$ vertices $\{v_1, \ldots, v_{|V|}\}$, and a target number of colour classes $k^\star$

Output: a complete, improper $k^\star$-colouring $\mathcal{S} = \{S_1, \ldots, S_{k^\star}\}$ for G

```
1:  S_1 ← ∅
2:  S ← {S_1}
3:  U ← ∅
4:  for i ← 1 to |V| do
5:      success ← false
6:      for j ← 1 to |S| do
7:          if S_j ∪ {v_i} is an independent set then
8:              S_j ← S_j ∪ {v_i}
9:              success ← true
10:             break
11:     if not success and |S| < k* then
12:         S_{|S|+1} ← {v_i}
13:         S ← S ∪ S_{|S|+1}
14:     else if not success and |S| = k* then
15:         U ← U ∪ {v_i}
16: if |U| > 0 then
17:     for all v ∈ U do
18:         x ← random integer between 0 and k*
19:         S_x ← S_x ∪ {v}
20: while |S| < k* do
21:     S_{|S|+1} ← ∅
22:     S ← S ∪ S_{|S|+1}
23: return S = {S_1, ..., S_{k*}}
```

If the colouring produced by Algorithm 4.2 is infeasible (i. e. it contains clashes) then the resultant colouring is passed to TABUCOL [Hertz and de Werra, 1987] which attempts to find a feasible $k^\star$-colouring for G. Here, we allow 900 seconds for TABUCOL to successfully identify a feasible $k^\star$-colouring, if this is not achieved then we do not implement our tabu search procedure for reducing the expected number of future clashes.

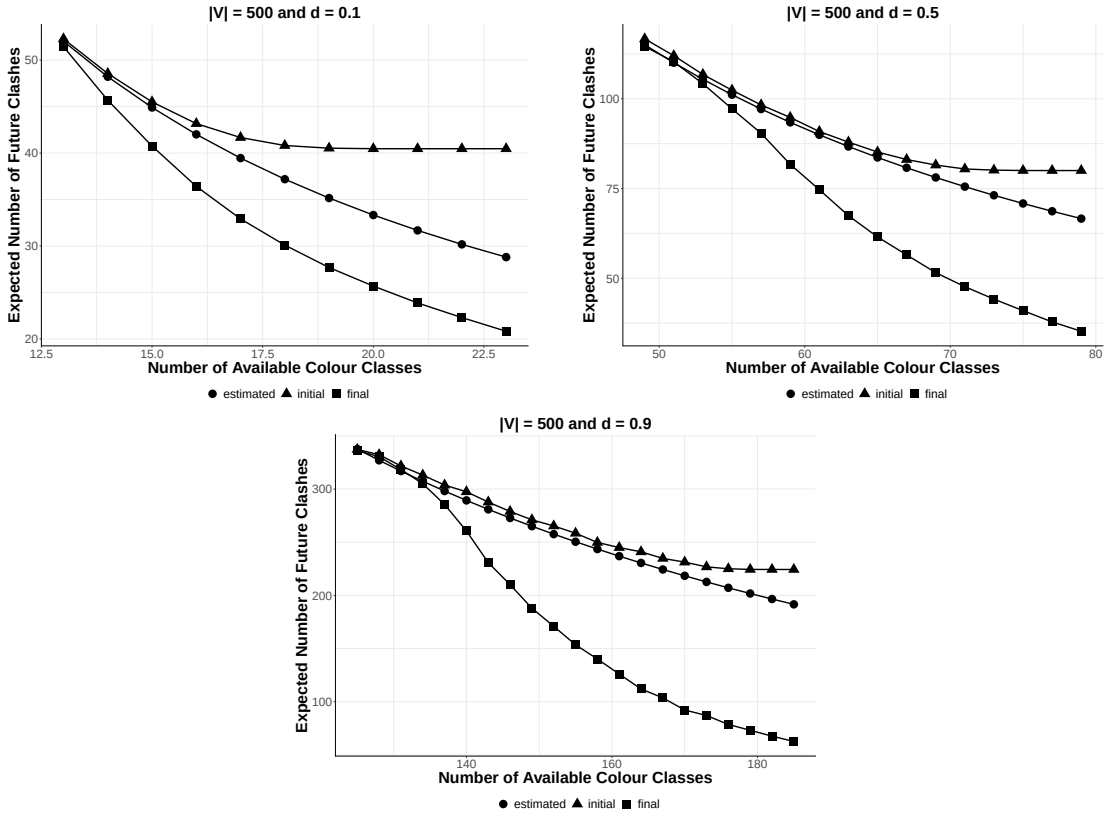

Figure 4.5: Mean expected number of future clashes against target number of colour classes (in clockwise order, starting from the top-left, graphs represent trials conducted on test instances with $d = 0.1, 0.5$ and 0.9, respectively).

Once a feasible $k^\star$-colouring has been obtained, either by the modified GREEDY algorithm or TABUCOL, it is passed to our tabu search procedure for 10,000 iterations. The "best" value for the expected number of future clashes observed and the run time was recorded every 250 iterations.

In all but one case, our tabu search procedure was able to reduce the expected number of clashes in the next time-step, i. e. Equation (4.3), from that of the initial colourings. The expected number of future clashes of the initial and final colourings, along with an "estimated" value[3], have been plotted against $k^\star$ in Figure 4.5. As can be observed from this figure, the amount of reduction between the initial colouring and the final colouring (i. e. the "best" colouring achieved after 10,000 iterations) increases as the value of $k^\star$ increases. In fact, the correlation coefficients between $k^\star$ and the amount of reduction (i. e. the distance between the lines of the graphs in Figure 4.5) lie between 0.985 and 0.993 for the different values of d.

It should be noted that the similarity between the "estimated" and initial number of expected future clashes wanes as $k^\star$ increases. This is because the modified

[3]The "estimated" expected number of future clashes in Figure 4.5 is calculated using Equation (4.2). For our test instances, the expected adjacency values are sampled from a uniform distribution with a lower bound of 0 and an upper bound of $\frac{2pd}{1-d}$, i. e. $p_1 = U[0, \frac{2pd}{1-d}]$, therefore $\mathbb{E}(p_1) = \frac{pd}{1-d}$.

Table 4.3: Median time (in seconds) required for our tabu search procedure to complete 10,000 iterations.

$d = 0.1$	$k^\star$	13	14	15	16	17	18	19	20
	runTime	9.029	8.816	9.061	9.352	9.564	9.746	9.957	10.187
	$k^\star$	21	22	23					
	runTime	10.445	10.776	11.127					
$d = 0.5$	$k^\star$	49	51	53	55	57	59	61	63
	runTime	8.152	8.166	8.297	8.452	8.518	8.625	8.717	8.843
	$k^\star$	65	67	69	71	73	75	77	79
	runTime	9.015	9.150	9.291	9.446	9.594	9.810	10.027	10.202
$d = 0.9$	$k^\star$	125	128	131	134	137	140	143	146
	runTime	11.255	11.354	11.197	11.148	11.240	11.401	11.509	11.669
	$k^\star$	149	152	155	158	161	164	167	170
	runTime	11.833	12.010	12.209	12.300	12.444	12.596	12.767	12.939
	$k^\star$	173	176	179	182	185			
	runTime	13.146	13.327	13.587	13.767	14.062			

GREEDY algorithm adds empty colour classes in these cases in order to achieve the target number of colour classes (see Lines 15-17 of Algorithm 4.2), whereas the "estimated" value assumes all colour classes contain approximately equal numbers of vertices.

A significant, positive relationship can also be observed between $k^\star$ and the time required to complete 10,000 iterations, i. e. as $k^\star$ increases, so too does the amount of time required to complete 10,000 iterations (see Table 4.3). The Spearman rho correlation coefficients range between 0.934 and 0.990 for the three different values of d. This observation is to be expected, because the number of potential Kempe-chain interchanges for a $k^\star$-colouring is of the order $|V| \times k^\star$, which means, along with the potential pair-swaps, the total number of neighbourhood moves to be considered during each iteration also increases as $k^\star$ increases.

Finally, it is worth highlighting that the majority of the reduction to the expected number of future clashes takes place in the early stages of the tabu search procedure. The reduction between iterations decreases as the tabu search procedure continues, a pattern that is often observed with local search methods. This is clearly displayed in Figure 4.6.

4.4 Trial Information

Now that we have shown that the expected number of future clashes $\mathcal{F}(\mathcal{S})$ of a colouring $\mathcal{S}$ can be successfully reduced by the tabu search procedure described above, we wish to investigate how this affects colourings when it is included within our proposed two stage approach, i. e. Algorithm 4.1.

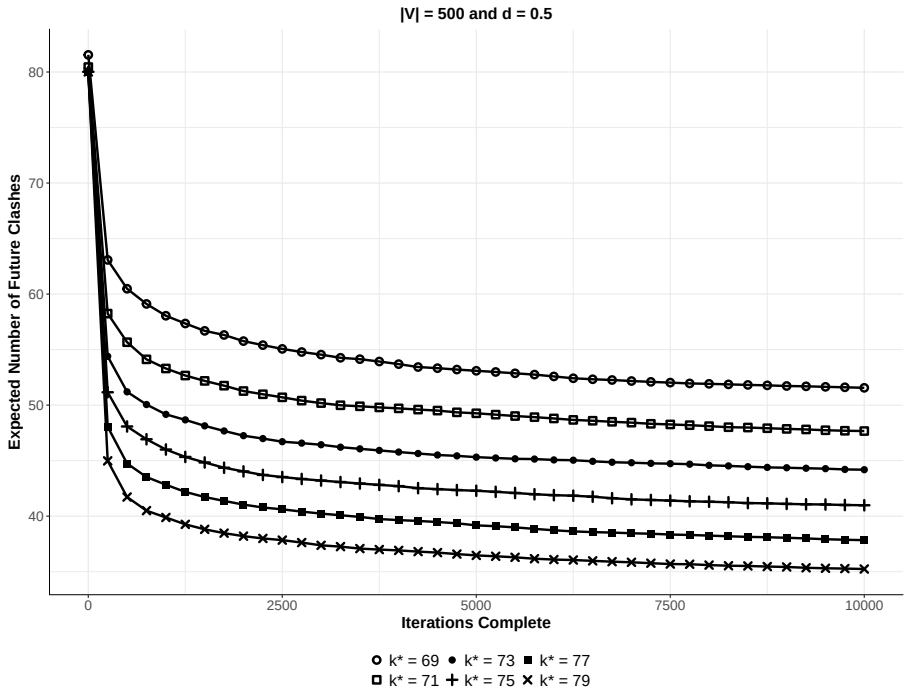

Figure 4.6: Mean expected number of future clashes against the number of iterations of our tabu search procedure completed for test instances with $d = 0.5$.

4.4.1 Test Instances

In these experiments our test instances are edge dynamic random graphs. For each test instance we specify the number of vertices $n = |V|$, a desired density d, an "expected" change probability p and a number of time-steps T. As in Chapter 3, we use $n \in \{250, 500, 1000\}$[4], $d \in \{0.1, 0.5, 0.9\}$, $p \in \{0.005, 0.01, \ldots, 0.05\}$ and $T = 10$. For each combination of these parameters, 20 dynamic graphs were produced.

In each case, G_0 is constructed such that every edge $\{u, v\} \in \mathcal{E}$ exists with probability d. Then for $t = 0, 1, \ldots, T-1$, every edge in E_t is copied to the set of deleted edges E^-_{t+1} with probability p and the edges in $\mathcal{E} \backslash E_t$ are copied to the set of new edges E^+_{t+1} with probabilities sampled from the uniform distribution $U[0, \frac{2pd}{1-d}]$. By using this distribution, each edge in $\mathcal{E} \backslash E_t$ is copied to E^+_{t+1} with an expected probability of $\frac{pd}{1-d}$ which ensures that the density remains approximately equal over all time-steps (see Section 3.4.1).

4.4.2 Algorithm Parameters

Here, we use DSatur [Brélaz, 1979], as opposed to RLF [Leighton, 1979], as our constructive operator to produce an initial colouring for G_0 of each test instance and within Method 4. With the difference observed between the "estimated" and

[4]Unless stated otherwise, the results presented in Section 4.5 concern test instances with $n = 500$. The relationships observed between the different modification operators were similar regardless of the value of n. Therefore, to reduce the volume of results presented, only particularly interesting differences will be addressed for test instances with $n \in \{250, 1000\}$.

Table 4.4: Target number of colour classes $k^\star$ used based on n and d. The final column indicates the size of the increments (inc.) between the values of $k^\star$ used.

		$k^\star$		
n	d	min.	max.	inc.
250	0.1	8	13	1
	0.5	28	46	2
	0.9	76	106	3
500	0.1	12	18	1
	0.5	48	70	2
	0.9	125	170	3
1000	0.1	21	32	1
	0.5	92	134	3
	0.9	240	325	5

initial expected number of future clashes for high values of $k^\star$ in Figure 4.5, it was believed that this would also be observed for even lower values of $k^\star$ if RLF was used, which was undesirable. On the other hand, using the GREEDY algorithm did not seem appropriate as this is known to produce initial colouring with significantly more colour classes than both DSATUR and RLF in many cases (see Section 2.1). Therefore, DSATUR seemed like the best "middle-ground" choice.

The rest of the algorithm parameters for the first stage of Algorithm 4.1 are the same as those used in Chapter 3 for Algorithm 3.2: TABUCOL [Hertz and de Werra, 1987] and PARTIALCOL [Blöchliger and Zufferey, 2008] are implemented to tackle the k-GCPs (i. e. Line 5 of Algorithm 4.1) and k is adjusted such that, if a feasible k-colouring cannot be obtained within half of the allotted time limit, then k is increased by 1, and so on (i. e. Lines 9 and 10 of Algorithm 4.1).

Unlike in Chapter 3, k is now capped such that it cannot be reduced below $k^\star$. On the other hand, if an initial feasible colouring achieved has fewer than $k^\star$ colour classes then an appropriate number of empty colour classes will be added. The values of $k^\star$ used for a test instance with n vertices and desired density d are shown in Table 4.4.

As described in Section 4.3, the tabu search operator for reducing the future number of expected clashes (i. e. the second stage of Algorithm 4.1) makes inverse moves "tabu" for a fixed tabu tenure, chooses the "best" available Kempe-chain interchange or pair-swap at each iteration, and uses a tabu tenure of $\frac{n}{2}$ iterations. If during an iteration there are no available moves (i. e. all Kempe-chain interchanges and pair-swaps are currently "tabu") then a random Kempe-chain interchange is performed. As with TABUCOL and PARTIALCOL, this tabu search operator has an aspiration criteria such that if the resultant colouring $\mathcal{S}'$ of a "tabu" move satisfies $\mathcal{F}(\mathcal{S}') < \mathcal{F}(\mathcal{S}_{\text{best}})$, then that move will be executed regardless of its "tabu" status.

The time limit remains fixed as 10 seconds per time-step. Here, the time limit is cumulative over both stages of Algorithm 4.1 such that if more time is dedicated to identifying a feasible $k^\star$-colouring $\mathcal{S}_t$ for G_t then less time is available for reducing the $\mathcal{F}(\mathcal{S}_t)$. Of course, if a feasible $k^\star$-colouring cannot be found then the second

Table 4.5: Median time (in seconds) required to obtain an initial, feasible colouring. Results for test instances with $d = 0.1$ are omitted due to their relatively small values, which were deemed to be insufficiently accurate to present.

		p									
d	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.5	0	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015	0.015
	1	1.732	2.683	2.504	2.887	3.136	3.362	3.572	3.409	4.126	3.534
	2	1.920	2.278	2.511	2.582	3.058	3.097	2.324	2.723	3.058	3.058
	3	1.950	2.192	2.644	2.434	3.027	3.144	3.401	3.237	3.175	2.395
	4	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
0.9	0	0.016	0.016	0.016	0.016	0.016	0.031	0.016	0.016	0.016	0.016
	1	5.055	5.351	5.312	5.226	5.733	5.406	5.320	5.296	5.421	5.554
	2	4.431	5.008	5.039	5.008	5.008	5.039	5.008	5.008	5.086	5.008
	3	4.368	4.922	5.039	5.055	5.047	5.023	5.024	5.008	5.024	5.047
	4	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than all others for the same values of d and p.

stage of Algorithm 4.1 cannot be executed.

4.5 Results

4.5.1 Without Secondary Optimisation

Here we aim to illustrate that the relationships observed between the modification operators in Chapter 3 remain the same for these test instances. We also wish to show that replacing RLF with DSatur causes the relationships to become slightly shifted with regards to p. In Section 2.1 it was shown that the colourings produced by DSatur generally have significantly more colour classes than those produced by RLF but, on the other hand, significantly less time is required for DSatur to be executed.

By setting $k^\star$ equal to the minimum values in Table 4.4, our two stage approach behaves in a similar manner to Algorithm 3.2. This happens because the minimum values of $k^\star$ in Table 4.4 were chosen such that a feasible $k^\star$-colouring is extremely unlikely to be identified within the 10 second time limit, therefore the entire time limit will be dedicated to minimising the number colour classes.

Initial, Feasible Colourings

For all test instances, when Methods 1 to 3 are executed, the initial feasible colourings achieved continue to have significantly fewer colour classes compared to using Methods 0 and 4, and require significantly more time to do so. These observations can be seen in Figure 4.7 and Table 4.5, respectively.

The initial feasible colourings produce by Method 4 have both significantly fewer

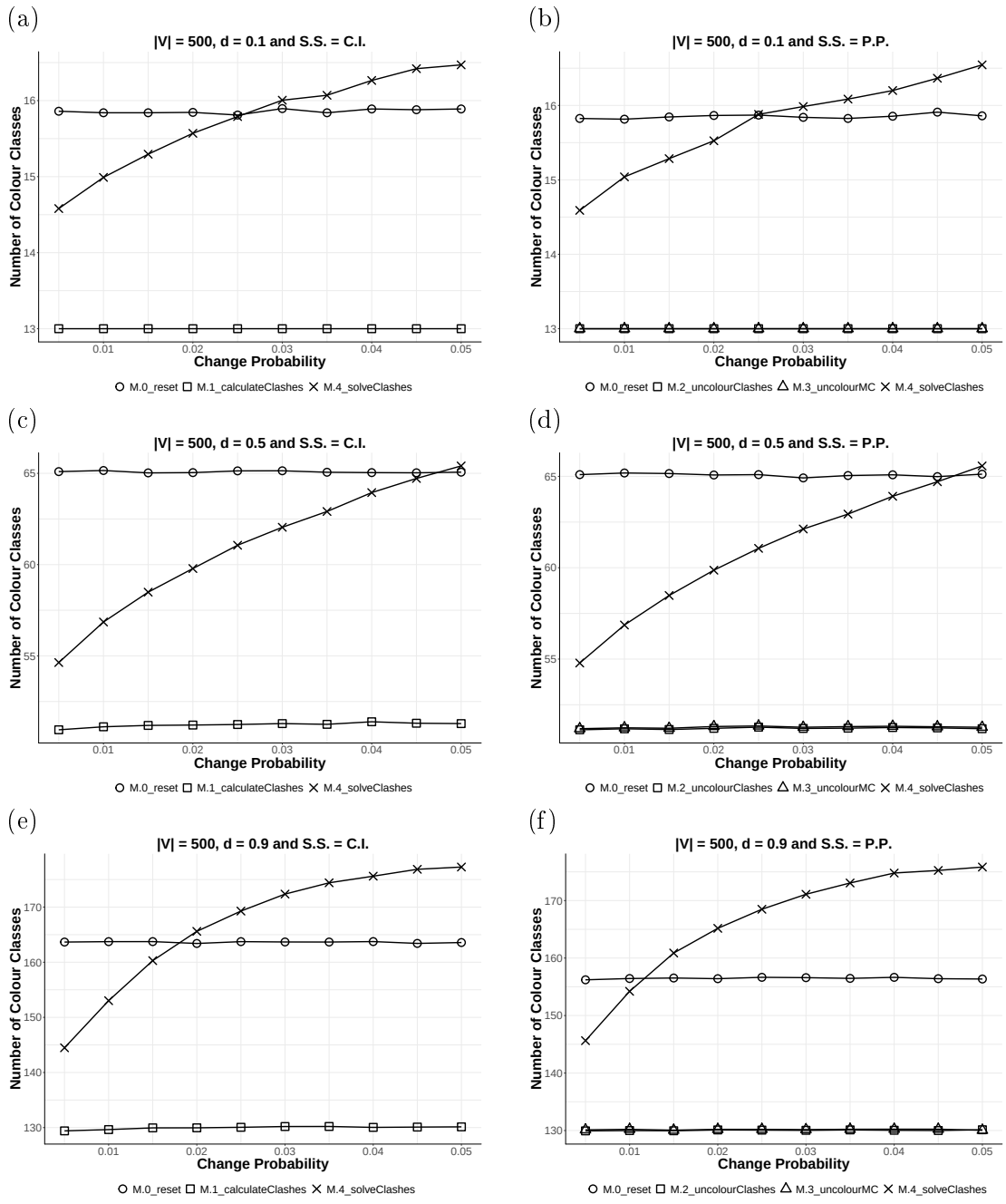

Figure 4.7: Mean number of colour classes in initial, feasible colourings on test instances with $|V| = 500$. Graphs (a), (c) and (e) represent methods that operate in the complete, improper solution space, graphs (b), (d) and (f) represent methods that operate in the partial, proper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

Table 4.6: Significant differences between the number of colour classes in the initial feasible colourings achieved by Methods 0 and 4.

			p									
n	d	S.S.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	C.I.	X	X	X	-	O	O	O	O	O	O
		P.P.	X	X	X	-	-	O	O	O	O	O
	0.5	C.I.	X	X	X	X	X	X	X	X	X	-
		P.P.	X	X	X	X	X	X	X	X	-	-
	0.9	C.I.	X	X	O	O	O	O	O	O	O	O
		P.P.	X	X	O	O	O	O	O	O	O	O
500	0.1	C.I.	X	X	X	X	-	O	O	O	O	O
		P.P.	X	X	X	X	-	O	O	O	O	O
	0.5	C.I.	X	X	X	X	X	X	X	X	X	O
		P.P.	X	X	X	X	X	X	X	X	X	O
	0.9	C.I.	X	X	X	O	O	O	O	O	O	O
		P.P.	X	X	X	O	O	O	O	O	O	O
1000	0.1	C.I.	X	X	X	X	X	X	O	O	O	O
		P.P.	X	X	X	X	X	X	X	X	X	O
	0.5	C.I.	X	X	X	X	X	X	X	O	O	O
		P.P.	X	X	X	X	X	X	X	X	O	O
	0.9	C.I.	X	X	X	O	O	O	O	O	O	O
		P.P.	X	X	X	O	O	O	O	O	O	O

X - indicates that Method 4 achieves an initial colouring with significantly fewer colour classes than Method 0.
O - indicates that Method 4 achieves an initial colouring with significantly more colour classes than Method 0.

or significantly more colour classes compared to using Method 0 dependent on p. In this case, the values of p for which the number of colour classes is significantly reduced is increased in comparison to the results presented in Chapter 3. This can clearly be observed by comparing Figures 3.4 and 4.7, and Tables 3.2 and 4.6. This observation is most likely due to the significant difference in the number of colour classes in the colourings produced by DSatur in comparison to those produced by RLF. As in Chapter 3, the amount of time required by Method 4 to produce an initial feasible colourings is significantly less compared to Method 0.

The similarity of the results when using Methods 2 and 3 continue here. Significant differences in the number of colour classes in the initial colourings are only observed for some test instances with $n \in \{500, 1000\}$ and $d = 0.9$, where Method 2 has significantly fewer. There is either no significant difference in the time required to achieve an initial feasible colouring or Method 2 is significantly faster. Also, there is either no significant difference in the number of PartialCol iterations required to achieve an initial feasible colouring or Method 2 requires significantly more.

One piece of information that was not available for comparison in Chapter 3 is the initial number of "uncoloured" vertices (i. e. $|U|$) after Methods 2 and 3 were implemented. As can be seen in Table 4.7, Method 3 produces initial colourings with significantly fewer "uncoloured" vertices than Method 2 dependent on n, d and p. This makes sense as $|E^+_{t+1}| \approx \frac{pdn(n-1)}{2}$ and is therefore dependent on n, d and p

Table 4.7: Significant differences between the number of "uncoloured" vertices in the initial colourings produced by Methods 2 and 3.

		p									
n	d	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	-	-	-	-	-	X	X	X	X	X
	0.5	-	-	X	X	X	X	X	X	X	X
	0.9	X	X	X	X	X	X	X	X	X	X
500	0.1	-	-	-	-	X	X	X	X	X	X
	0.5	-	X	X	X	X	X	X	X	X	X
	0.9	X	X	X	X	X	X	X	X	X	X
1000	0.1	-	X	X	X	X	X	X	X	X	X
	0.5	X	X	X	X	X	X	X	X	X	X
	0.9	X	X	X	X	X	X	X	X	X	X

X - indicates that Method 3 achieves an initial colouring with significantly fewer "uncoloured" vertices than Method 2.

also. Although the information is not available for comparison, it is believed that these results will hold true for the experiments in Chapter 3 because the constructive operator should bare little to no influence on this particular metric.

Final, Feasible Colourings

The Friedman test with significance level $\alpha = 0.01$ shows no significant difference in the number of colour classes of the final, feasible colourings achieved when using any of the modification operators on test instances with $n = 250$ and all values of d and p, $n = 500, d = 0.1$ and all values of p, and $n = 500, d = 0.5$ and most values of p.

In comparison to Method 0, Method 1 was found to produce final, feasible colourings with significantly fewer colour classes, for test instances with $d = 0.5$ and small values of p. On the other hand, Methods 1 to 3 achieve final, feasible colourings with significantly more colour classes than Method 0 for $d = 0.9$ with some values of p. Using Method 4 was also shown to achieve final, feasible colourings with significantly fewer colour classes than those achieved when using Methods 0 to 3 for test instances with $d \in \{0.5, 0.9\}$ and some values of p. These observations are again almost identical to the those presented in Section 3.5.2.

When using Methods 1 to 3, the time required[5] to achieve a final, feasible colouring was found to be significantly less for test instances with $d = 0.1$ and most values p compared to Methods 0 and 4. This is also true on the same test instances for Method 3 compared to Method 0. Unlike Methods 1 to 3, Method 4 did not require significantly more time than Method 0 for any test instances in order to achieve final, feasible colourings. Moreover, for test instances with $d = 0.1$ and all values of p, and $d = 0.5$ and some low values of p, it required significantly less time. These

[5]Remember that time comparisons for final colourings only correspond to trials where the final, feasible colourings achieved have an equal number of colour classes.

Table 4.8: Median time (in seconds) required to obtain final, feasible colourings with an equal numbers of colour classes across all modification operators for the given solution space. As with Table 4.5, results for test instances with $d = 0.1$ are omitted as they were deemed to be insufficiently accurate to present.

			p									
d	S.S.	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.5	C.I.	0	3.947	3.682	3.245	3.276	3.799	3.417	3.316	3.931	2.839	2.901
		1	1.607	2.824	3.058	3.385	3.167	3.783	3.635	3.931	4.227	3.572
		4	0.874*	1.622	2.348	2.325	3.526	2.801	2.987	3.151	3.166	3.105
	P.P.	0	3.151	3.284	2.824	2.551	2.925	2.770	2.652	2.87	3.237	2.910
		2	2.074	2.075	2.589	3.019	3.791	3.370	2.418	3.089	3.440	2.722
		3	2.262	2.668	2.871	2.73	2.909	3.433	3.759	3.245	3.136	2.793
		4	1.638	2.442	2.418	2.301*	2.777	2.754	3.198	2.387	2.847	2.761
0.9	C.I.	0	5.662	5.351	4.977	6.052	4.586	4.882	5.710	4.695	5.195	4.984
		1	6.661	7.191	7.129	7.442	7.894	7.378	7.559	7.550	7.488	7.933
		4	6.224	3.993*	4.711	5.507	4.150	5.616	6.139	5.070	5.132	4.851
	P.P.	0	4.142	4.820	6.131	5.179	4.773	4.087	4.664	4.376	5.398	4.953
		2	4.431	5.647	5.234	5.889	5.881	6.661	6.318	6.053	6.396	6.170
		3	5.624	5.460	6.076	5.226	5.819	6.645	6.910	7.075	6.763	5.850
		4	4.290	4.399	4.564	5.491	4.040	3.759	4.649	4.228	4.836	5.359

* indicates a time that is significantly less than all others for the same solution space, and values of d and p.

results are displayed in Table 4.8.

4.5.2 With Secondary Optimisation

We now consider the situation where the future adjacency matrix $P_{(t+1)}$ is known for every time-step $t \in \{0, \ldots, T-1\}$. By using this additional information we have implemented the approach outlined in Algorithm 4.1 in an attempt to produce more robust colourings. Here we explore how our approach affects the initial number of clashes at the start of each time-step, the number of colour classes in the initial, feasible colourings achieved, and the time required to achieve these colourings. Final, feasible colourings will not be discussed, because altering the value of $k^\star$ will naturally cap the number of colour classes in these colourings.

Initial Clashes

The secondary objective of the approach outlined in Algorithm 4.1 is to reduce the value of $\mathcal{F}(\mathcal{S}_t)$ such that the returned colouring $\mathcal{S}_t$ has fewer expected future clashes for G_{t+1}. To measure the effectiveness of our approach, we compare the number of clashes at the start of the following time-step when using our approach against an algorithm that does not include a secondary optimisation stage (i. e. Algorithm 4.1 with Lines 11-16 omitted).

Our results show that the number of clashes at the start of the following time-step is significantly reduced dependent on both $k^\star$ and p. As $k^\star$ becomes larger than $\chi(G_{t+1})$, we begin to observe fewer clashes at the start of the time-step $t+1$ for

Table 4.9: Significant differences between the number of clashes at the start of each time-step when using modification operator 4 (*solveClashes*) within our approach (Algorithm 4.1) compared against an algorithm without any secondary optimisation on test instances with $d = 0.1$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
12	-	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-	X
14	-	-	-	X	X	X	X	X	X	X
15	-	X	X	X	X	X	X	X	X	X
16	-	X	X	X	X	X	X	X	X	X
17	X	X	X	X	X	X	X	X	X	X
18	X	X	X	X	X	X	X	X	X	X

X - indicates significantly fewer clashes at the start of each time-step when our two stage approach is employed compared to when it is not.

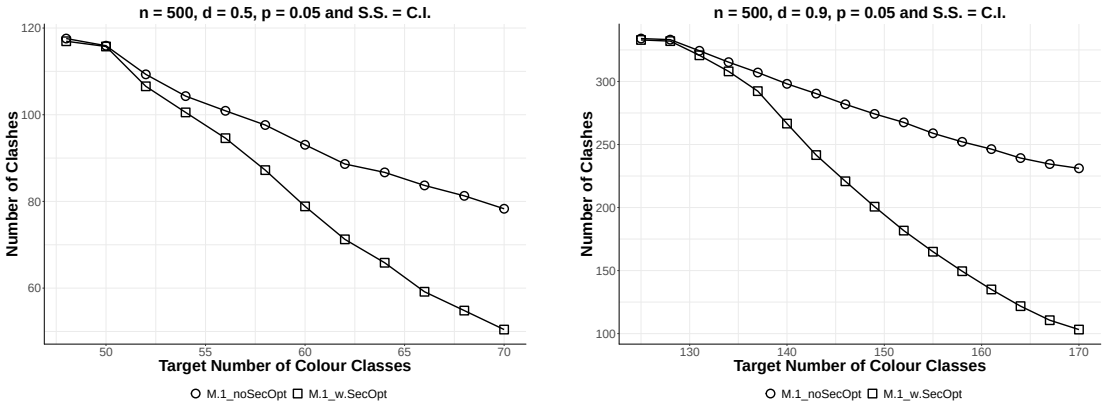

Figure 4.8: Mean number of clashes in $\mathcal{S}_t$ for time-step $t + 1$ against the target number of colour classes $k^\star$ for test instances with $d \in \{0.5, 0.9\}$ and $p = 0.05$ whilst implementing modification operator 1 (*calculateClashes*).

higher values of p. This is likely due to the fact that as $k^\star$ increases, so too does the number of feasible $k^\star$-colourings, which grants more opportunities for our method to reduce $\mathcal{F}(\mathcal{S}_t)$. Moreover, as $k^\star$ increases, significant reductions are observed on test instances with decreasing values of p.

For example, consider test instances with $n = 500$ and $d = 0.1$ where it is likely that $\chi(G_t) \approx 12$ or 13 for all $t \in \{1, \ldots, T\}$ (illustrated in Table 4.9). When using Method 4 within our approach and operating in the complete, improper solution space, significantly fewer clashes were observed for $k^\star = 13$ with $p = 0.05$, $k^\star = 14$ with $p \geq 0.02$, $k^\star \in \{15, 16\}$ with $p \geq 0.01$, and $k^\star \in \{17, 18\}$ with all values of p.

It can also be shown that the magnitude of the reduction is significant and positively correlated with $k^\star$ for all modification operators. This can be seen in Figure 4.8 where the two lines in each graph (corresponding to the omission and inclusion of secondary optimisation) diverge as the value of $k^\star$ increases.

Reducing the number of clashes at the start of a time-step has a knock-on affect

Table 4.10: Number of "uncoloured" vertices in the initial colourings produced by Method 3 (*uncolourMostClashing*) on test instances with $d = 0.1$. The second column indicates whether secondary optimisation (S.O.) was employed or not (Y and N correspond to yes and no respectively).

		p									
$k^\star$	S.O.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
12	N	5.460	10.290	15.010	19.905	23.360	28.385	32.405	36.555	39.855	44.270
	Y	5.395	9.895	14.990	19.570	23.730	27.845	32.155	36.695	40.395	44.080
13	N	5.360	10.325	14.830	19.550	24.215	28.285	32.430	36.855	40.460	44.760
	Y	5.290	9.745	14.350	19.260	23.175	27.265	31.725	35.970	39.545	43.235*
14	N	5.060	9.295	13.500	18.345	22.455	26.615	30.170	34.400	37.170	40.860
	Y	4.780	9.025	12.885	17.345*	20.970*	25.350	27.920*	31.855*	36.095	39.455*
15	N	4.620	8.655	12.755	17.135	20.760	24.925	28.375	32.515	35.585	38.625
	Y	4.100*	8.370	11.440*	15.480*	19.200*	22.965*	25.525*	29.490*	33.030*	35.820*
16	N	4.440	8.150	11.975	16.245	19.565	23.210	26.680	30.530	33.380	37.655
	Y	4.040	7.470	10.440*	13.970*	17.405*	20.400*	23.160*	26.860*	29.100*	32.255*
17	N	4.330	8.005	11.435	15.810	18.930	22.445	25.815	29.235	31.760	35.540
	Y	3.415*	6.835*	9.370*	12.580*	15.690*	18.585*	21.415*	23.920*	26.575*	30.045*
18	N	4.275	7.910	11.235	15.295	18.405	21.665	24.770	28.340	30.785	34.405
	Y	3.000*	6.140*	8.685*	11.765*	14.235*	17.180*	19.290*	22.425*	25.235*	27.165*

* indicates colourings with significantly fewer "uncoloured" vertices for the same values of $k^\star$ and p.

with regards to the number of "uncoloured" vertices in the colourings produced by Methods 2 and 3. Significant reductions in the number of "uncoloured" vertices is dependent on $k^\star$ and p in the same way as for clash reduction, i.e. as $k^\star$ increases the value of p for which significant differences are observed decreases. This is clearly shown in Table 4.10. Starting with a partial, proper colouring with significantly fewer "uncoloured" vertices should also be beneficial with regards to the time required to identify an initial feasible colouring.

Initial, Feasible Colourings

Here, we assume that if a colouring $\mathcal{S}_t$ has fewer clashes for the following time-step, then the optimisation operators should require less time to identify initial, feasible colourings. It might also be possible that these initial, feasible colourings will have fewer colour classes. We now investigate whether our results support these hypotheses.

In our experiments we found that secondary optimisation of $\mathcal{F}(\mathcal{S}_t)$ significantly reduced the time required to achieve initial, feasible colourings when using Methods 1 to 3 for most test instances with the following parameters $d = 0.9$, $k^\star \geq 143$ and $p \geq 0.01, 0.02$ and 0.015, respectively. For these test instances, we also observed a high level of reduction with regards to the number of clashes at the start of a time-step, which supports our hypothesis that reducing clashes leads to reduced time requirements. Conversely, there appears to be no effect on the number of colour classes in the initial, feasible colouring achieved when using Methods 1 to 3. This is

Table 4.11: Significant differences between the time required to achieve an initial, feasible colouring when using modification operator 1 (*calculateClashes*) within our approach (Algorithm 4.1) compared against Method 0 on test instances with $d = 0.9$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
125	O	O	O	O	O	O	O	O	O	O
128	O	O	O	O	O	O	O	O	O	O
131	O	O	O	O	O	O	O	O	O	O
134	O	O	O	O	O	O	O	O	O	O
137	O	O	O	O	O	O	O	O	O	O
140	-	O	O	O	O	O	O	O	O	O
143	X	X	O	O	O	O	O	O	O	O
146	X	X	-	O	O	O	O	O	O	O
149	X	X	X	X	-	O	O	O	O	O
152	X	X	X	X	X	-	-	O	O	O
155	X	X	X	X	X	X	X	X	-	O
158	X	X	X	X	X	X	X	X	X	-
161	X	X	X	X	X	X	X	X	X	X
164	X	X	X	X	X	X	X	X	X	X
167	X	X	X	X	X	X	X	X	X	X
170	X	X	X	X	X	X	X	X	X	X

X - indicates that our two stage approach with Method 1 requires significantly less time than Method 0.
O - indicates that our two stage approach with Method 1 requires significantly more time than Method 0.

unsurprising because the number of colour classes in the initial feasible colouring is usually equal to the number of colour classes in the "best" colouring for the previous time-step which is now capped by $k^\star$.

However, these modification operators are still able to produce initial, feasible colourings with significantly fewer colour classes than Method 0, provided that $k^\star$ is small enough (i. e. $k^\star \leq 15$, 66 and 164 for test instances with $d = 0.1$, 0.5 and 0.9 respectively). In addition, the time required to achieve these colourings is dependent on both $k^\star$ and p, as illustrated in Table 4.11. Therefore, our secondary optimisation when used in conjunction with Methods 1 to 3 can produce initial, feasible colourings with fewer colour classes and requires less time to do so in comparison to Method 0 for low values of $k^\star$ and p.

For Method 4 there is no significant difference in the time required to reach an initial, feasible colouring when including or omitting the secondary optimisation phase in most cases. In the few instances where differences do occur (for test instances with $d = 0.9$, operating in the complete, improper solution space), there are no observable patterns with regards to the values of $k^\star$ and p. In comparison to Method 0, for all values of $k^\star$, d and p, Method 4 requires significantly less time to achieve initial, feasible colourings for the same reasons outlined in Sections 3.5.1 and 4.5.1.

With regards to the number of colour class in the initial, feasible colourings achieved by Method 4, any significant differences when compared against omitting

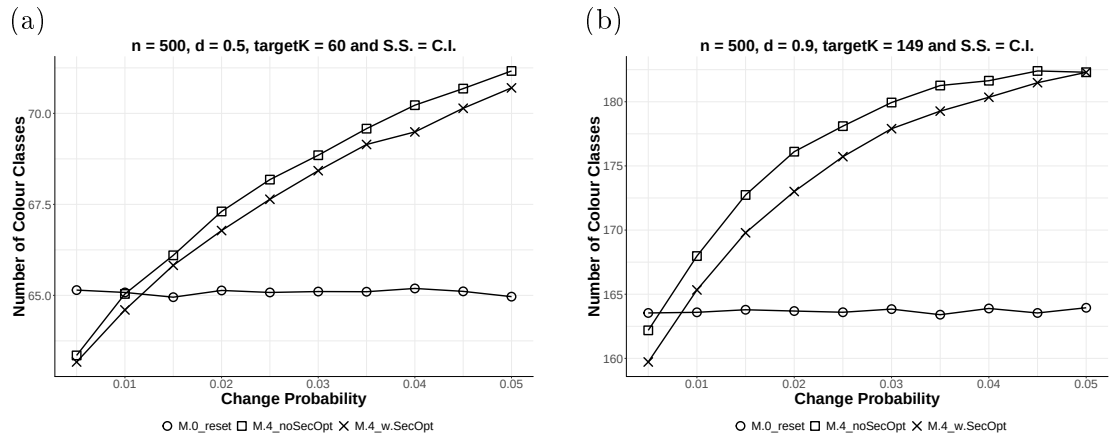

Figure 4.9: Mean number of colour classes in initial, feasible colourings for test instances with $d = 0.5, k^\star = 60$ (a) and $d = 0.9, k^\star = 149$ (b) whilst implementing modification operator 4 (*solveClashes*) in the complete, improper solution space.

the secondary optimisation appear to be dependent on $k^\star$. For low values of $k^\star$ there are no significant differences. For mid-range to high values of $k^\star$ there is a significant reduction (examples of which are illustrated in Figure 4.9). On the other hand, significant increases were observed for test instances with $d = 0.1$, the highest values of $k^\star$ and low values of p. When $k^\star$ is much larger than $\chi(G)$ for a given graph G, our algorithm is more likely to find an initial, feasible k-colouring for G such that $k < k^\star$ and, therefore, empty colour classes will be added to this colouring (see Section 4.4.2). During secondary optimisation, it is likely that vertices are then moved into these empty colour classes and, subsequently, these colour classes are less likely to feasibly accommodate "newly clashing" vertices at the start of the following time-step. Therefore, depending on the test instance, our experiments appear to both support and contradict our hypothesis with regards to secondary optimisation leading to initial, feasible colourings with fewer colour classes.

In comparison to Method 0, the number of colour classes in the initial, feasible colouring produced by this modification operator are dependent on $k^\star$ and p, similar to the case without future adjacency information. For low values of $k^\star$ and p, there are significantly fewer colour classes and vice versa for high values of $k^\star$ and p. As $k^\star$ increases, the value of p decreases, for which these significant differences can be observed, as illustrated in Table 4.12.

Method 2 (*uncolourClashes*) vs. Method 3 (*uncolourMostClashing*)

The value of $k^\star$ has a few notable effects on the behaviour of Methods 2 and 3 in comparison with one another. As we saw in Section 4.5.1, dependent on d and p, Method 3 produces initial colouring with significantly fewer "uncoloured" vertices compared to those produced by Method 2. Yet again, $k^\star$ also effects this relationship, such that as $k^\star$ increases the values of p for which this significant difference is

Table 4.12: Significant differences between the number of colour classes in the initial, feasible colourings achieved when using modification operator 4 (*solveClashes*) within our approach (Algorithm 4.1) compared against Method 0 on test instances with $d = 0.5$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
48	X	X	X	X	X	X	X	X	-	O
50	X	X	X	X	X	X	X	X	X	O
52	X	X	X	X	X	X	X	X	-	O
54	X	X	X	X	X	X	-	O	O	O
56	X	X	X	X	-	O	O	O	O	O
58	X	X	X	O	O	O	O	O	O	O
60	X	X	O	O	O	O	O	O	O	O
62	-	O	O	O	O	O	O	O	O	O
64	O	O	O	O	O	O	O	O	O	O
66	O	O	O	O	O	O	O	O	O	O
68	O	O	O	O	O	O	O	O	O	O
70	O	O	O	O	O	O	O	O	O	O

X - indicates that our two stage approach with Method 4 achieves colourings with significantly fewer colour classes than Method 0.
O - indicates that our two stage approach with Method 4 achieves colourings with significantly more colour classes than Method 0.

observed increases (see example in Table 4.13).

The number of cases where Method 2 achieves initial feasible colourings with significantly fewer colour classes than Method 3 decreases as the value of $k^\star$ increases. This is likely due to the natural capping of $k^\star$ with regards to colour classes.

As $k^\star$ increases, the number of cases in which Method 2 requires significantly less time to achieve an initial feasible colouring compared to Method 3 increases. This is likely due to the extra time required by Method 3 to transform a feasible colouring for G_t into a partial, proper colouring for G_{t+1} (see Section 3.5.1). As the number of clashing vertices decreases, the time required by Method 2 to produce a partial, proper colouring will also decrease which will allow PartialCol to begin solving a k-GCP earlier than if Method 3 is implemented.

Finally, as $k^\star$ increases the number of cases in which Method 3 requires significantly fewer iterations of PartialCol to identify a feasible colouring also increases, especially for test instances with $d = 0.9, p \geq 0.04$ and $k^\star \geq 158$. This is likely due to the fact that the number of "uncoloured" vertices decreases as $k^\star$ increases, which would imply that the initial colouring produced by Method 3 is "closer" to a feasible colouring. For the test instances mentioned, the colour classes are likely to contain fewer vertices which would imply that it is also more likely for an "uncoloured" vertex to be transferred to an existing colour class without introducing clashes, and therefore different vertices being transferred to the set of "uncoloured" vertices. Despite this reduction in the number of PartialCol iterations, Method 2 generally does not require any additional time to achieve an initial feasible colouring in the

Table 4.13: Significant differences between the number of "uncoloured" vertices in the colourings produced by Methods 2 and 3 after secondary optimisation has been employed on test instances with $d = 0.5$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
48	-	-	X	X	X	X	X	X	X	X
50	-	-	X	X	X	X	X	X	X	X
52	-	X	X	X	X	X	X	X	X	X
54	-	-	X	X	X	X	X	X	X	X
56	-	X	X	X	X	X	X	X	X	X
58	-	-	X	X	X	X	X	X	X	X
60	-	-	-	X	X	X	X	X	X	X
62	-	-	X	X	X	X	X	X	X	X
64	-	-	-	-	X	X	X	X	X	X
66	-	-	-	X	-	X	X	X	X	X
68	-	-	-	X	-	X	X	X	X	X
70	-	-	-	-	-	-	X	X	X	X

X - indicates that Method 3 achieves initial colourings with significantly fewer "uncoloured" vertices than Method 2.

same cases.

As concluded in Chapter 3, it appears that it is more beneficial to employ Method 2 rather than Method 3. Not only are its results better more often than vice versa, but it is easier to implement and therefore requires less time to be executed.

4.6 Conclusions

In this chapter we have introduced the concept of future adjacency and proposed a two-stage approach for the edge dynamic GCP which first attempts to identify a feasible colouring and then, using future adjacency information, attempts to make the colouring more robust with regards to potential future change. In order to achieve this secondary objective, we have introduced a tabu search procedure which maintains the feasibility of a colouring for the current time-step whilst attempting to reduce the estimated number of clashes in the subsequent time-step.

Our experiments have shown that changing the constructive operator does not affect the relationships observed in Chapter 3 between the modification operators other than shifting the value of p for which the observations hold, in some cases.

By using future adjacency information to reduce the estimated number of clashes in the following time-step, the number of clashes observed at the start of each time-step is significantly reduced for higher values of $k^\star$ and p. In some cases, reducing the initial number of clashes has also reduced the amount of time required to achieve initial, feasible colourings and the number of colour classes in these colourings.

All of the relationships observed between Methods 1 to 4 against Method 0 in Chapter 3 continue to hold when our two-stage approach is used here. However, the relationships also become dependent on the value of $k^\star$, with the strength of the

statements diminishing as the value of $k^\star$ increases.

There also appears to be a clear relationship between the value of $k^\star$ and the "robustness" of the colourings achieved. By design, if more time is dedicated to reducing the number of colour classes in a colouring, such as for cases with low values of $k^\star$, then less time will be available for the secondary optimisation, which tackles "robustness". However, we have also observed that as $k^\star$ approaches $\chi(G_t)$ for a given graph G_t then there are fewer feasible $k^\star$-colourings for G_t and therefore the neighbourhood of moves for the secondary optimisation is also reduced.

We believe that our two-stage approach is not hampered by its design, because even if more time was available for the secondary optimisation stage, it is unlikely to perform much better for low values of $k^\star$. In other words, as $k^\star$ approaches $\chi(G_t)$ for a given graph G_t, the amount of improvement attainable with regards to the secondary objective decreases. Also, it was shown that most of the improvements with regard to the secondary objective will be realised within earlier iterations, such that running the secondary optimisation stage for longer periods of time has diminishing returns. Therefore, by design, when there is the most to be gained with regards to the secondary objective (i. e. when $k^\star$ is large), more time is available for secondary optimisation, and vice versa.

Key Findings and Recommendations

- Using future adjacency information within our two-stage approach leads to a significant decrease in the number of clashes observed in the following time-step (i. e. increased robustness).
- By increasing the robustness of a colouring to future changes, benefits are observed with regards to the quality and time required to achieve an initial feasible colouring via our modification approach in the following time-step.
- A clear inverse relationship between quality and robustness has been demonstrated, we therefore suggest that the user prioritises between the two objectives based on their preferences.

Chapter 5

Modifying Colourings for Vertex Dynamic Random Graphs without Future Change Information

In Section 3.1, two separate cases of dynamic graphs were proposed: edge dynamic and vertex dynamic. In this chapter we shift our focus from the edge dynamic GCP to the vertex dynamic GCP.

Vertex dynamic graphs may be of particular interest when considering real-world problems in which the "objects" within that problem change over time. For example, the dynamic frequency assignment problem [Dupont et al., 2009] deals with a communication network where additional locations (the "objects" of a frequency assignment problem that are represented by vertices in the associated GCP) can be added to an existing network, therefore altering the vertex set, and subsequently the edge set, of the associated GCP.

In this chapter we will introduce vertex dynamic graphs, as well as new modification operators (or methods) to be used within the modification approach outlined in Chapter 3. Results will then be presented for the modification approach, as it is applied to the vertex dynamic GCP without future change information.

5.1 Vertex Dynamic Graphs

With vertex dynamic graphs the changes are applied to the vertex set V_t. This in turn affects the edge set E_t, as edges incident to deleted vertices will themselves need to be deleted. Similarly, new edges will be needed to connect any new vertices to the existing ones, unless the new vertices are intended to be isolated.

For a vertex dynamic graph $\mathcal{G}$, consider the graph $G_t = (V_t, E_t)$ at time-step t. To get to time-step $t+1$ we must define a set of deleted vertices $V_{t+1}^- \subseteq V_t$ and a set of new vertices V_{t+1}^+. The vertex set for time-step $t+1$ is then defined, in a similar fashion to E_{t+1} for edge dynamic graphs (see Section 3.1), as $V_{t+1} = (V_t \backslash V_{t+1}^-) \cup V_{t+1}^+$.

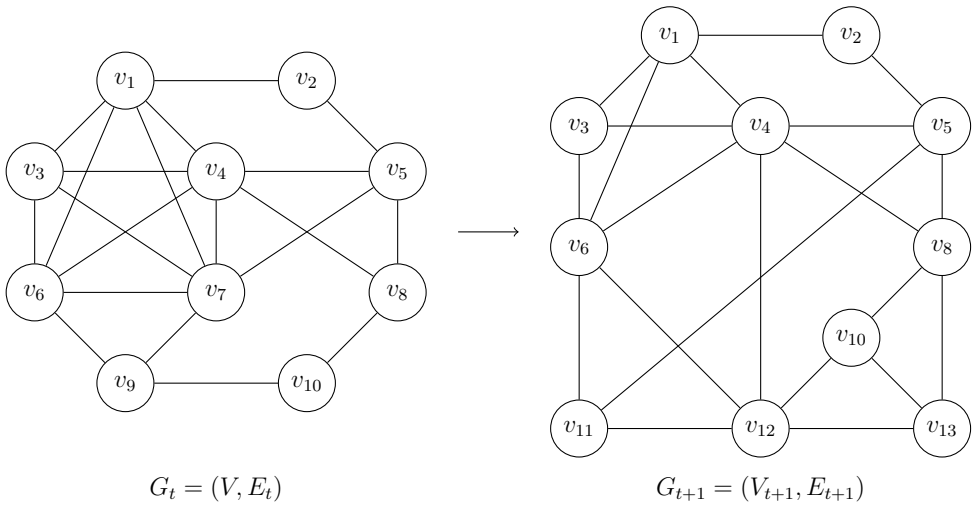

Figure 5.1: Vertex dynamic graph example

Once V_{t+1}^- and V_{t+1}^+ have been defined, sets of deleted edges E_{t+1}^- and new edges E_{t+1}^+ also need to be defined. The set of deleted edges is defined as $E_{t+1}^- = \{\{u, v\} \in E_t : u \in V_{t+1}^- \text{ or } v \in V_{t+1}^-\}$ (i.e. E_{t+1}^- is the set of edges in E_t that are incident to the deleted vertices) and the set of new edges is defined as $E_{t+1}^+ \subseteq \{\{u, v\} \in \mathcal{E}_{t+1} : u \in V_{t+1}^+ \text{ or } v \in V_{t+1}^+\}$ where $\mathcal{E}_{t+1}$ is the set of all $\binom{|V_{t+1}|}{2}$ possible edges between vertices in V_{t+1} (i.e. E_{t+1}^+ is a set of connecting edges to the set of new vertices). The edge set for time-step $t+1$ is then defined as for edge dynamic graphs, $E_{t+1} = (E_t \backslash E_{t+1}^-) \cup E_{t+1}^+$.

An example of how a vertex dynamic graph can change between time-steps is illustrated in Figure 5.1. Here, we define the set of deleted vertices as $V_{t+1}^- = \{v_7, v_9\}$. Subsequently, the set of deleted edges is defined as $E_{t+1}^- = \{\{v_1, v_7\}, \{v_3, v_7\}, \{v_4, v_7\}, \{v_5, v_7\}, \{v_6, v_7\}, \{v_6, v_9\}, \{v_7, v_9\}, \{v_9, v_{10}\}\}$. Finally, the sets of new vertices and edges are defined as $V_{t+1}^+ = \{v_{11}, v_{12}, v_{13}\}$ and $E_{t+1}^+ = \{\{v_4, v_{12}\}, \{v_5, v_{11}\}, \{v_6, v_{11}\}, \{v_6, v_{12}\}, \{v_8, v_{13}\}, \{v_{10}, v_{12}\}, \{v_{10}, v_{13}\}, \{v_{11}, v_{12}\}, \{v_{12}, v_{13}\}\}$ respectively.

A vertex dynamic graph could be used to model a real-world problem where the size of the problem changes over time. For example, consider a communication network where the number of objects in the network (e.g. telephone masts) changes over time. The objects within the network can be represented by a changing vertex set V_t where t represents a specific snapshot in time. For this particular problem, an edge will be present between two vertices if their associated objects are located within a certain proximity of each other. Assuming the objects do not move once they have been established, the constraints (represented by an edge set E_t) will only alter with respect to the objects being added to and/or removed from the network (i.e. the edges incident to the vertices in V_{t+1}^- and V_{t+1}^+).

A well-known variant of the GCP is the on-line graph colouring problem. This problem is actually a special case of the vertex dynamic graph colouring problem. At each time-step, a single vertex is introduced along with its connecting edges and

must be coloured based on the information available up to and including the current time-step. To use our notation for this problem we would define $V^-_{t+1} = \emptyset$ and V^+_{t+1} such that $|V^+_{t+1}| = 1$. Most research on the on-line graph colouring problem has concerned worst-case behaviour of algorithms. For example, Lovász et al. [1989] have shown that if $G = (V, E)$ is a bipartite graph with $|V| = n$, then there exists an algorithm A such that

$$\chi_A(G) \leq 2 \log_2 n \tag{5.1}$$

where $\chi_A(G)$ is the maximum number of colour classes in a colouring achieved by A for G. Bounds regarding the behaviour of the GREEDY constructive algorithm have been presented by Gyárfás and Lehel [1988], and empirical results in the same area have been presented by Ouerfelli and Bouziri [2011].

Vertex Dynamic GCP vs. Edge Dynamic GCP

Before we explore the vertex dynamic GCP in its own right, it is worth considering it in comparison to the edge dynamic GCP, as discussed in Chapter 3. In particular, it is worth highlighting the theoretical difference between using a change probability of p for the vertex dynamic GCP compared to that of an edge dynamic GCP.

Suppose we have an edge dynamic graph with parameters n, d and p (as described in Section 3.4.1) and a feasible k-colouring $\mathcal{S}_t = \{S_1, \ldots, S_k\}$ for G_t. In the worst case scenario, every new edge in E^+_{t+1} will relate to a pair of vertices that are in the same colour class of $\mathcal{S}_t$. Therefore, the maximum number of clashes that can be introduced between time-steps t and $t + 1$ is given by

$$\min\{\sum_{i=1}^{k} \binom{|S_i|}{2}, |E^+_{t+1}|\} \approx \min\{\frac{n(n-k)}{2k}, \frac{n(n-1)pd}{2}\} \tag{5.2}$$

The approximation on the RHS of Equation (5.2) assumes that each colour class of $\mathcal{S}_t$ contains approximately the same numbers of vertices, i. e. $|S_i| \approx \frac{n}{k}$ for $i = 1, \ldots, k$.

Now consider a vertex dynamic graph with parameters n, d and p, and let $\mathcal{S}_t$ be a feasible k-colouring for G_t. Here, the worst case scenario would be realised if every new vertex in V^+_{t+1} were assigned to the colour class $S_i \in \mathcal{S}_t$ that contains the most vertices (i. e. $|S_i| \geq |S_j|$ for all $j \in \{1, \ldots, k\} \backslash \{i\}$) *and* that each vertex in S_i is also adjacent to every vertex in V^+_{t+1} (i. e. $\{S_i \times V^+_{t+1}\} \cap E^+_{t+1} = \{S_i \times V^+_{t+1}\}$). Therefore, the maximum number of clashes that can be introduced between time-steps t and $t + 1$ is given by

$$|V^+_{t+1}| \cdot \max_{i=1,\ldots,k} \{|S_i|\} \approx \frac{n^2 p}{k} \tag{5.3}$$

As with Equation (5.2), the approximation on the RHS of Equation (5.3) assumes that each colour class of $\mathcal{S}_t$ contains approximately the same numbers of vertices. It also assumes that $|V^+_{t+1}| \approx np$, as will be the case for our test instances (see Section 5.3.1).

As the parameters n and d increase for $\mathcal{G} = \{G_0, G_1, \ldots, G_T\}$, so too does the feasible number of colour classes required in each time-step of $\mathcal{G}$, i. e. $\chi(G_t)$. For test instances with $d \in \{0.5, 0.9\}$, we can show that $np < \frac{n-k}{2}$ and $\frac{n}{k} < \frac{(n-1)d}{2}$ for all values of n, p and k used in our experiments, such that $k \geq \chi(G_t)$. This implies that the worst case scenario for the vertex dynamic GCP, with regards to clashes being introduced, is less extreme than that of the edge dynamic GCP on test instances with the same parameter values.

Let us now consider "uncoloured" vertices instead of clashes. For an edge dynamic graph with parameters n, d and p, the maximum number of vertices in the set of "uncoloured" vertices U when transforming a feasible k-colouring $\mathcal{S}_t$ for G_t into a partial, proper k-colouring for G_{t+1} is given by

$$\min\{n-k, |E^+_{t+1}|\} \approx \min\{n-k, \frac{n(n-1)pd}{2}\} \tag{5.4}$$

In this situation, every new edge in E^+_{t+1} will again relate to a pair of vertices which are in the same colour class of $\mathcal{S}_t$. Within each of these pairs, at least one vertex involved will be transferred to U, which if done in a certain manner will result in $|U| = |E^+_{t+1}|$.

On the other hand, the worst case scenario for the vertex dynamic problem is that each new vertex in $|V^+_{t+1}|$ is adjacent to at least one vertex in each colour class of S_t. Therefore, the maximum number of vertices in the set of "uncoloured" vertices U when transforming $\mathcal{S}_t$ into a partial, proper k-colouring for G_{t+1} is given by

$$|V^+_{t+1}| \approx np \tag{5.5}$$

It is clear that the worst case scenario for the vertex dynamic GCP, with regards to "uncoloured" vertices, is again less extreme in comparison to the edge dynamic GCP on test instances with the same parameter values. It is therefore unsurprising that the results presented towards the end of this chapter are similar to those presented for the edge dynamic GCP in Section 3.5 while allowing for higher values of p.

5.2 Modification Operators for the Vertex Dynamic GCP

As with the edge dynamic GCP, it is worth noting that a feasible colouring at time-step t will not remain feasible at time-step $t+1$ under the assumption that V^-_{t+1} and V^+_{t+1} are non-empty. In this case, a feasible colouring for G_t will not include any of the new vertices in V^+_{t+1} and will still include the deleted vertices in V^-_{t+1}. However, no clashes will be introduced because every new edge in E^+_{t+1} will have at least one

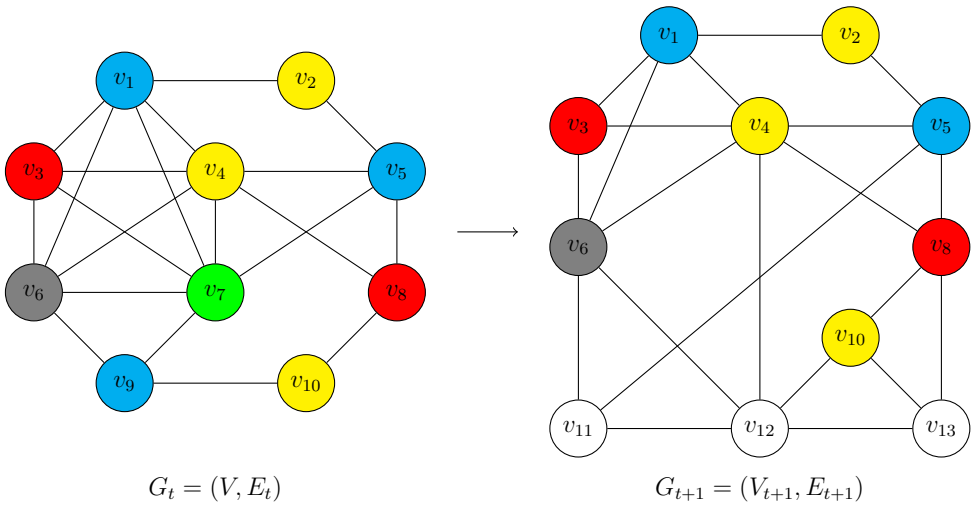

Figure 5.2: Vertex dynamic graph colouring example

end-point in V_{t+1}^+. If $\mathcal{S}_t = \{S_1, \ldots, S_k\}$ is a feasible colouring for G_t then $\mathcal{S}_t$ with the set of deleted vertices V_{t+1}^- removed, denoted by $\mathcal{S}_t \backslash V_{t+1}^- = \{S_1 \backslash V_{t+1}^-, \ldots, S_k \backslash V_{t+1}^-\}$, is a partial, proper colouring for G_{t+1}.

Let us reconsider the vertex dynamic graph in Figure 5.1. As illustrated in Figure 5.2, removing the deleted vertices from the feasible colouring of G_t (on the LHS) has led to a partial, proper colouring for G_{t+1} (on the RHS). The new edges do not introduce any clashes because the new vertices are considered "uncoloured", as they were not included within the colouring for G_t.

Unlike the edge dynamic GCP, here the "natural" solution space to consider when using a colouring from time-step t as a starting point for a colouring at time-step $t+1$ is the partial, proper one.

Method 5 (*uncolourNew*)**.** Set $\mathcal{S}_{t+1} = \mathcal{S}_t \backslash V_{t+1}^-$ and let $U = V_{t+1}^+$ be the set of "uncoloured" vertices.

As mentioned above, if $\mathcal{S}_t$ is a feasible colouring for G_t then $\mathcal{S}_t \backslash V_{t+1}^-$ is a partial, proper colouring for G_{t+1}. By implementing Method 5, the resultant colouring $\mathcal{S}_{t+1} = \mathcal{S}_t \backslash V_{t+1}^-$ and set of "uncoloured" vertices $U = V_{t+1}^+$ can be passed directly to an optimisation operator that operates in the partial, proper solution space. This optimisation operator will then attempt to feasibly colour all "uncoloured" vertices in U to achieve a feasible k-colouring for G_{t+1} where $k = |\mathcal{S}_t|$.

For the example illustrated in Figure 5.2, the tabu search operator would attempt to find a feasible 5-colouring for G_{t+1} starting from $\mathcal{S}_t \backslash \{v_7, v_9\}$ with $U = \{v_{11}, v_{12}, v_{13}\}$, shown on the RHS, where $\mathcal{S}_t$ is the feasible colouring for G_t on the LHS.

Method 6 (*randomAssign*)**.** Randomly order the vertices in V_{t+1}^+. Transfer (or assign) the first vertex in V_{t+1}^+ to a random colour class of $\mathcal{S}_t \backslash V_{t+1}^-$. Repeat for each

vertex in V_{t+1}^+ sequentially, then calculate the number of clashes in the resultant colouring $\mathcal{S}_{t+1}$.

The resultant colouring $\mathcal{S}_{t+1}$ is a complete, improper colouring for G_{t+1} and can be passed directly to an optimisation operator that operates in the complete, improper solution space. This optimisation operator will attempt to remove all clashes from $\mathcal{S}_{t+1}$ to achieve a feasible k-colouring for G_{t+1} where $k = |\mathcal{S}_t|$. As with Method 2 for the edge dynamic GCP, by moving out of the "natural" solution space, this modification operator allows us to explore an entirely different neighbourhood of colourings.

Due to the random assignment of the new vertices in V_{t+1}^+ to colour classes in $\mathcal{S}_t \backslash V_{t+1}^-$, there are $|\mathcal{S}_t|^{|V_{t+1}^+|}$ different colourings that could be produced by Method 6. In the example illustrated in Figure 5.2 there are $5^3 = 125$ potential resultant colourings. A best case scenario could transfer v_{11} to the yellow colour class, v_{12} to the blue colour class, and v_{13} to the grey colour class, thus producing a feasible 4-colouring for G_{t+1}. On the other hand, if all 3 new vertices in V_{t+1}^+ were transferred to the yellow colour class then the resultant colouring $\mathcal{S}_{t+1}$ would be a complete, improper colouring for G_{t+1} with 5 clashes: $\{v_4, v_{12}\}, \{v_{10}, v_{12}\}, \{v_{10}, v_{13}\}, \{v_{11}, v_{12}\}$, and $\{v_{12}, v_{13}\}$.

Method 7 (*leastClashesAssign*)**.** Randomly order the vertices in V_{t+1}^+. Transfer the first vertex in V_{t+1}^+ to the colour class of $\mathcal{S}_t \backslash V_{t+1}^-$ with the "least clashes" (i. e. the colour class with the least adjacent vertices). Repeat for each vertex in V_{t+1}^+ sequentially, then calculate the number of clashes in the resultant colouring $\mathcal{S}_{t+1}$.

In the same way that Method 3 is a "more sophisticated" version of Method 2 for the edge dynamic GCP, so too is Method 7 with regards to Method 6 for the vertex dynamic problem. As with Method 6, here we wish to assign the new vertices in V_{t+1}^+ to colour classes in $\mathcal{S}_t \backslash V_{t+1}^-$ to achieve a complete, improper colouring for G_{t+1}. By using this method, the number of clashes introduced should be reduced in comparison to using the random approach of Method 6.

For the example illustrated in Figure 5.2, Method 7 will always produce a feasible colouring for G_{t+1} with either 4 or 5 colour classes, dependent on the order in which the new vertices and the colour classes of $\mathcal{S}_t \backslash \{v_7, v_9\}$ are considered. Method 6, on the other hand, could produce colourings with as many as 5 clashes for the same example.

Method 8 (*solveNew*)**.** Randomly order the vertices in V_{t+1}^+. Attempt to transfer the first vertex in V_{t+1}^+ to a colour class in $\mathcal{S}_t \backslash V_{t+1}^-$ in a "greedy" fashion such that no clashes are introduced. Repeat for each vertex in V_{t+1}^+ sequentially. Pass the residual graph $\tilde{G}$ induced by the remaining vertices in V_{t+1}^+ to a constructive operator to produce a colouring $\tilde{\mathcal{S}}$ for $\tilde{G}$. Combine $\mathcal{S}_t \backslash V_{t+1}^-$ and $\tilde{\mathcal{S}}$ to produce the colouring $\mathcal{S}_{t+1}$.

The approach of Methods 4 and 8 is to identify and focus on the "problematic" parts of $\mathcal{S}_t$ and $\mathcal{S}_t \backslash V^-_{t+1}$ with regards to G_{t+1}, respectively. Method 8 focuses on the new, and therefore "uncoloured", vertices in V^+_{t+1}, whereas Method 4 focuses on the new clashes that may have been introduced between time-steps t and $t+1$.

As with Method 4 for the edge dynamic GCP, Method 8 is also guaranteed to produce a feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1} here, though it may increase the number of colour classes (i. e. $|\mathcal{S}_{t+1}| \geq |\mathcal{S}_t|$). Firstly, all deleted vertices are removed from $\mathcal{S}_t$ by definition of $\mathcal{S}_t \backslash V^-_{t+1}$. All new edges have at least one end-point which is a new vertex, therefore $\mathcal{S}_t \backslash V^-_{t+1}$ has no clashes for G_{t+1} because $\mathcal{S}_t$ was feasible (i. e. it had no clashes) for G_t and it does not contain any of the new vertices. No new clashes are introduced when a new vertex in V^+_{t+1} is transferred to a colour class in $\mathcal{S}_t \backslash V^-_{t+1}$, otherwise it would not be transferred. A constructive algorithm is also guaranteed to produce a feasible colouring for any given graph, so the colouring $\tilde{\mathcal{S}}$ is a feasible colouring of the remaining new vertices. Finally, when $\mathcal{S}_t \backslash V^-_{t+1}$ and $\tilde{\mathcal{S}}$ are combined, the resultant colouring $\mathcal{S}_{t+1}$ has no clashes and all vertices in $V_{t+1} = V_t \backslash (V^-_{t+1}) \cup V^+_{t+1}$ are assigned to a colour class. Therefore, $\mathcal{S}_{t+1}$ is a feasible colouring for G_{t+1} with $k = |\mathcal{S}_t| + |\tilde{\mathcal{S}}|$ colour classes.

As we saw with Method 7, for the example illustrated in Figure 5.2, each new vertex in $V^+_{t+1} = \{v_{11}, v_{12}, v_{13}\}$ can be feasibly transferred to a colour class of $\mathcal{S}_t \backslash \{v_7, v_9\}$ to produce either a feasible 4- or 5-colouring for G_{t+1}. Method 8 also attempts to feasibly transfer each new vertex to a colour class before considering the residual graph induced by any remaining new vertices. Therefore, Method 8 will also produce a feasible 4- or 5-colouring for the example illustrated in Figure 5.2, again depending on the order in which the new vertices and the colour classes of $\mathcal{S}_t \backslash \{v_7, v_9\}$ are considered.

Details regarding the specific parameters of the constructive and optimisation operators used within these modification operators will be outlined in Section 5.3.2.

As we have seen, parallels can be drawn between these modification operators and those defined for the edge dynamic GCP in Section 3.3. They can be grouped by the types of colouring they produce: Methods 1, 6 and 7 produce complete, improper colourings for G_{t+1}, Methods 2, 3 and 5 produce partial, proper colourings for G_{t+1}, and Methods 4 and 8 produce feasible colourings for G_{t+1}. They can also be grouped by their approach: Methods 1 and 5 pass $\mathcal{S}_t$ and $\mathcal{S}_t \backslash V^-_{t+1}$ directly to the optimisation operators that operate in their "natural" solution spaces, Methods 2, 3, 6 and 7 convert $\mathcal{S}_t$ and $\mathcal{S}_t \backslash V^-_{t+1}$ to colourings that are optimised in an alternative solution space, and Methods 4 and 8 treat the "problematic" vertices as residual graphs to be recombined with $\mathcal{S}_t$ and $\mathcal{S}_t \backslash V^-_{t+1}$ respectively. These parallels and comparisons are summarised in Table 5.1.

It is worth noting that we consider more modification operators that convert G_t into a colouring that does *not* belong to the "natural" solution space of the respective

Table 5.1: Summary of modification operators.

		Edge Dynamic		Vertex Dynamic	
Modification description	Size of $k = \lvert\mathcal{S}_{t+1}\rvert$	M.	S.S.	M.	S.S.
Constructive operator	k independent of $\mathcal{S}_t$	0	Feasible	0	Feasible
Direct to optimisation operator	$k = \lvert\mathcal{S}_t\rvert$	1	C.I.	5	P.P.
Simple conversion (i. e. random)	$k = \lvert\mathcal{S}_t\rvert$	2	P.P.	6	C.I.
"More sophisticated" conversion	$k = \lvert\mathcal{S}_t\rvert$	3	P.P.	7	C.I.
Separate out "problematic" parts	$k \geq \lvert\mathcal{S}_t\rvert$	4	Feasible	8	Feasible

problem. This is because a decision must be made regarding how the conversion is to be completed. Methods 2 and 6 employ a random approach and perform the conversion moves on a first identified basis, whereas Methods 3 and 7 employ a basic search heuristic which chooses the "best" available conversion move at each step. As mentioned previously, we can therefore consider Methods 3 and 7 to be "more sophisticated" versions of Methods 2 and 6, respectively.

Adding Empty Colour Classes

Due to the success of including empty colour classes with Methods 1 and 2 that produce initial, infeasible colourings for the edge dynamic GCP, in our approach we also consider the inclusion of empty colour classes with Methods 5 and 6 that are likely to produce infeasible colourings for the vertex dynamic GCP.

Initially, $\mathcal{S}_{t+1}$ and U are defined in the same manner as when using the respective modification operators that were introduced earlier in this section. Then $\mathcal{S}_{t+1}$ is combined with $x > 0$ empty colour classes where x is a user-defined integer. When $\mathcal{S}_{t+1}$ is passed to the optimisation operator, along with U where appropriate, an attempt will then be made to find a k-colouring for G_t where $k = |\mathcal{S}_t| + x$. The specific values of x used will be discussed in Section 5.3.2.

In the following figures and tables, we will use the same names introduced previously to refer to the methods followed by "_x" to indicate the number of empty colour classes used (e. g. Method 6 with x empty colour classes will be referred to as *M.6_ randomAssign_ x*).

5.3 Trial Information

5.3.1 Test Instances

As with our edge dynamic test instances, our test instances for the vertex dynamic GCP also have four parameters: n the desired number of vertices[1], d the desired density, p the change probability, and T the number of time-steps. For consistency, the same parameter values used in Section 3.4.1 are used for these test instances,

[1]Unlike the edge dynamic GCP in Chapters 3 and 4, the number of vertices is no longer fixed over all time-steps. Therefore, the value of n is now simply the desired number of vertices at each time-step such that $|V_t| \approx n$ for $t = 0, 1, \ldots, T$.

that is: $n \in \{250, 500, 1000\}$, $d \in \{0.1, 0.5, 0.9\}$, $p \in \{0.005, 0.01, \ldots, 0.05\}$ and $T = 10$. For each test instance, G_0 is constructed such that $|V_0| = n$ and every edge $\{u, v\}$ of the $\binom{n}{2}$ possible edges in $\mathcal{E}_0$ is included in E_0 with probability d.

For vertex dynamic graphs, at time-step t, each vertex $v \in V_t$ is included in the set of deleted vertices V^-_{t+1} with probability p and the set of new vertices V^+_{t+1} is constructed such that $|V^+_{t+1}|$ is an integer between $np(1-p)$ and $np(1+p)$. By constructing V^+_{t+1} such that $|V^+_{t+1}| \approx np$ we can ensure that the number of vertices remains approximately equal over all time-steps. Each edge $\{u, v\} \in \mathcal{E}_{t+1}$ with at least one end-point in V^+_{t+1} is then included in the set of new edges E^+_{t+1} with probability d.

5.3.2 Algorithm Parameters

The algorithm parameters used here are the same as those used for the edge dynamic GCP without future change information, see Section 3.4.2. To briefly recap:

- RLF [Leighton, 1979] is used as our constructive operator for Method 8 (as it also is for Method 0).
- TabuCol [Hertz and de Werra, 1987] and PartialCol [Blöchliger and Zufferey, 2008] are implemented to tackle the k-GCPs within our approach (i. e. Line 5 of Algorithm 3.2) when operating in the complete, improper and partial, proper solution spaces respectively.
- During execution, k is adjusted such that if a feasible k-colouring cannot be obtained within half of the allotted time limit then k is increased by 1, and so on (i. e. Lines 9 and 10 of Algorithm 3.2).
- The time limit is set to 10 seconds per time-step[2] (i. e. Line 4 in Algorithm 3.2).
- When considering Methods 5 and 6 with empty colour classes, the number of empty colour classes x added to the produced colourings are as follows:
  - $x \in \{0, 1, 2\}$ for test instances with $d = 0.1$,
  - $x \in \{0, 2, 4, 6\}$ for test instances with $d = 0.5$, and
  - $x \in \{0, 3, 6, 9, 12\}$ for test instances with $d = 0.9$.

It should be noted that Method 5 exclusively produces partial, proper colourings, and Methods 6 and 7 exclusively produce complete, improper colourings. On the other hand, Method 8, as with Method 0, produces feasible colourings which can then be passed to a tabu search operator that operates within either solution space, as required. Therefore, only comparisons between modification operators that produce

[2]As with Chapters 3 and 4, all algorithms were programmed in C++ and executed on a 3.3GHZ Windows 7 PC with an Intel Core i3-2120 processor and 8GB RAM.

colourings within the same solution space or feasible colourings will be compared e. g. Methods 5 and 6 will not be compared against one another.

5.4 Results

Unless stated otherwise, all pairwise statistical comparisons are based on Wilcoxon signed rank tests with significance level $\alpha = 0.01$ due to the non-normality of our data.

5.4.1 Initial Colourings

Let us first consider initial colourings for the vertex dynamic GCP.

Similarly to Methods 1 to 3 for the edge dynamic problem, the initial, feasible colourings achieved by Methods 5 to 7 have significantly fewer colour classes than Methods 0 and 8 but require significantly more time to obtain them. The only cases for which this does not hold is for test instances with $n = 250, d = 0.1$ and $p \geq 0.015$ where no significant difference is observed[3]. The time required by Methods 5 to 7 also has a positive relationship with the change probability p. These observations can be seen in Figure 5.3 and Table 5.2 for test instances with $n = 500$.

The reasons for this behaviour are the same as those given for Methods 1 to 3 in Section 3.5.1, i. e. unlike Methods 0 and 8, Methods 5 to 7 are not guaranteed to produce an initial, feasible colouring and therefore require more time to move to a feasible region of the solution space via tabu search.

Again, as with Method 4 for the edge dynamic problem, Method 8 produces initial, feasible colourings with both significantly fewer and significantly more colour classes than Method 0 depending on the change probability p. However, Method 8 only produces initial, feasible colourings with significantly more colour classes for test instances with $d = 0.1$ and high values of p (see Table 5.3). In fact, for test instances with $d = 0.1$ and and low values of p, and $d \in \{0.5, 0.9\}$ with all values of p, Method 8 achieves initial, feasible colourings with significantly fewer colour classes. This is clearly illustrated for test instances with $n = 500$ in Figure 5.3.

As with Method 4, Method 8 requires significantly less time than Method 0 except for test instances with parameters $n = 250, d = 0.1$, $n = 250, d = 0.5, p \leq 0.03$, and $n = 500, d = 0.1, p \leq 0.03$ (as seen in Table 5.2) where no significant difference is observed[4]. This is most likely to occur because Method 8 applies RLF

[3] In these cases the time required to obtain an initial, feasible colouring is generally less than 10^{-3} seconds which is represented in the output as 0 seconds. The conclusions drawn from the Wilcoxon Signed Rank test may be unreliable because the data used in these cases may not be accurate enough.

[4] As with Methods 5 to 7, the time required to obtain an initial, feasible colouring in these cases is generally less than 10^{-3} seconds so the conclusions drawn from the statistical tests may be unreliable.

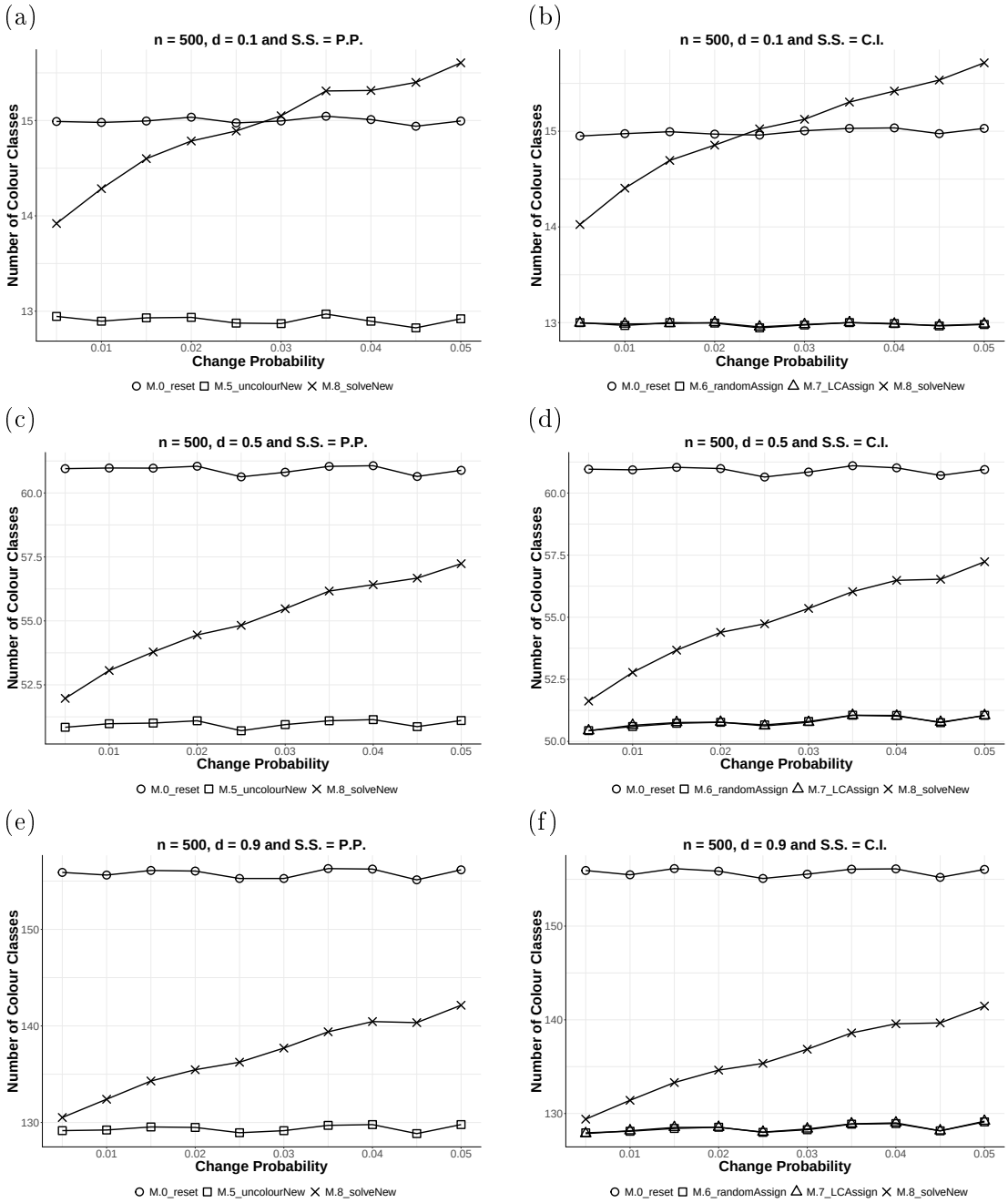

Figure 5.3: Mean number of colour classes in initial, feasible colourings for the vertex dynamic GCP on test instances with $n = 500$. Graphs (a), (c) and (e) represent methods that operate in the partial, proper solution space, graphs (b), (d) and (f) represent methods that operate in the complete, improper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively. For graphs regarding test instances with $n \in \{250, 1000\}$, see Figures B.3 and B.4 in Appendix B.

Table 5.2: Median time (in seconds) required to obtain an initial, feasible colouring for the vertex dynamic GCP on test instances with $n = 500$. For results regarding test instances with $n \in \{250, 1000\}$, see Table B.3 in Appendix B.

		p									
d	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	0	0	0	0	0	0	0	0	0	0.015	0.015
	5	0	0	0.015	0.015	0.015	0.016	0.015	0.016	0.016	0.016
	6	0	0	0.015	0.015	0.016	0.031	0.031	0.031	0.031	0.046
	7	0	0	0.015	0.015	0.016	0.031	0.031	0.031	0.031	0.032
	8	0	0	0	0	0	0	0*	0*	0*	0*
0.5	0	0.016	0.031	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016
	5	0.663	0.983	1.537	1.529	1.630	1.537	1.973	1.794	1.841	2.340
	6	0.320	1.131	1.240	1.724	1.482	1.724	1.935	2.411	2.114	2.785
	7	0.359	0.936	1.311	1.607	1.911	1.802	2.434	2.840	2.223	2.449
	8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
0.9	0	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031
	5	1.163	1.731	2.644	3.222	2.387	3.416	3.339	3.424	2.816	3.346
	6	1.069	1.997	2.941	3.830	3.565	4.189	4.820	4.938	4.852	5.007
	7	1.350	2.294	3.923	3.830	3.331	4.922	5.007	5.008	4.665	5.007
	8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than all others for the same values of d and p.

Table 5.3: Significant differences between the number of colour classes in the initial colourings achieved by Methods 0 and 8 on test instances with $d = 0.1$.

		p									
n	S.S.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	C.I.	X	X	X	-	-	O	O	O	O	O
	P.P.	X	X	X	-	-	O	O	O	O	O
500	C.I.	X	X	X	X	-	O	O	O	O	O
	P.P.	X	X	X	X	-	-	O	O	O	O
1000	C.I.	X	X	X	X	X	-	O	O	O	O
	P.P.	X	X	X	X	X	X	X	-	-	O

X - indicates that Method 8 achieves an initial colouring with significantly fewer colour classes than Method 0.
O - indicates that Method 8 achieves an initial colouring with significantly more colour classes than Method 0.

to a smaller residual graph $\tilde{G} = (\tilde{V}, \tilde{E})$ of G_t with $|\tilde{V}| \leq |V_t^+| \approx np$ as opposed to applying it to G_t with $|V_t| \approx n$.

Method 6 (*randomAssign*) vs. Method 7 (*leastClashesAssign*)

As with Methods 2 and 3 for the edge dynamic GCP, here we wish to explore whether the manner in which a feasible colouring for time-step t is modified into a colouring that belongs to the "unnatural" solution space of the given problem has any affect on the initial, feasible colourings achieved. Here specifically, we will explore whether the manner in which new vertices are assigned to the colour classes of $\mathcal{S}_t \backslash V_{t+1}^-$ has any affect on the initial, feasible colourings achieved. It is hypothesised that starting with a complete, improper colouring for G_{t+1} with fewer clashes should be advantageous with regards to quality and / or computational effort, therefore we expect to observe Method 7 outperforming Method 6 in some way.

For all test instance parameter combinations but two ($n = 250, d = 0.9, p = 0.025$; and $n = 1000, d = 0.9, p = 0.035$), we found there to be no significant difference between the number of colour classes in the initial, feasible colourings achieved when implementing either Method 6 or 7, at the $\alpha = 0.01$ level. The initial number of colour classes k for which TabuCol will attempt to find a feasible k-colouring is determined by the colouring $\mathcal{S}_t$ which is passed to Methods 6 and 7 and not by the modification operators themselves. It is therefore unsurprising that there is no significant difference between the number of colour classes in the initial, feasible colourings achieved.

What is surprising is that there appears to be no significant difference in the time required to achieve an initial, feasible colouring. Nor is there any significant difference in the number of iterations of TabuCol required to identify a feasible colouring after the initial colourings are produced by the modification operators, as illustrated in Figure 5.4 for test instances with $n = 500$. Therefore, we cannot conclude that Method 7 requires significantly more time than Method 6 to be executed, which is in contrast to Methods 2 and 3 for the edge dynamic GCP.

The graphs in Figure 5.4, clearly show that the number of TabuCol iterations required to identify a feasible colouring is positively related to p. This can be used as supporting evidence, at least for Methods 6 and 7 on test instances with $n = 500$, for why the relationship between length of time required to achieve an initial, feasible colouring and p is also positively correlated.

With Empty Colour Classes

Due to the similarity of the results observed between Methods 6 and 7 so far, we have chosen to omit Method 7 from these comparisons in an attempt to reduce the volume of results presented. Also, as in Section 3.6 for the edge dynamic GCP, only the test instances from Section 5.3.1 with $n = 500$ are used in this section.

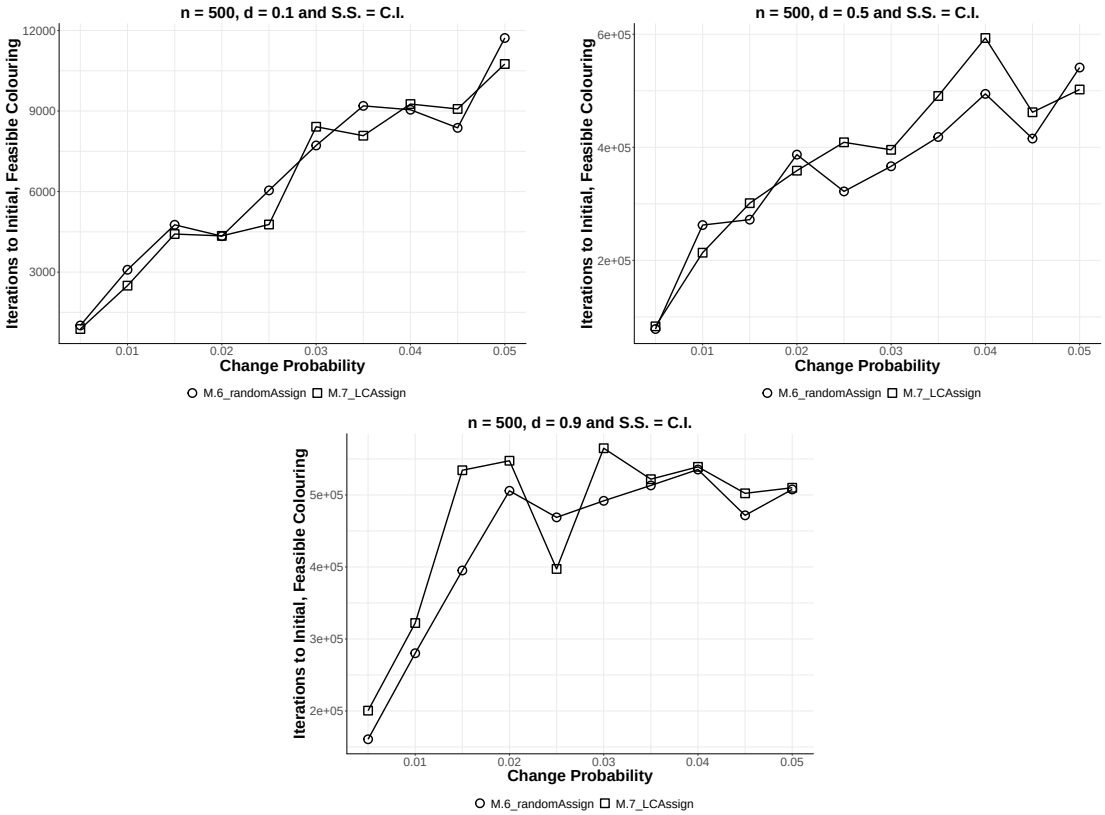

Figure 5.4: Median number of TabuCol iterations required to achieve an initial, feasible colouring for the vertex dynamic GCP (in clockwise order, starting from the top-left, graphs represent trials conducted on test instances with $n = 500$ and $d = 0.1, 0.5$ and 0.9, respectively).

Table 5.4: Significant differences between the number of colour classes in the initial, feasible colourings achieved by Method 5 with x empty colour classes against Method 8.

		p									
d	x	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	0	X	X	X	X	X	X	X	X	X	X
	1	-	X	X	X	X	X	X	X	X	X
	2	O	O	O	O	-	X	X	X	X	X
0.5	0	X	X	X	X	X	X	X	X	X	X
	2	O	X	X	X	X	X	X	X	X	X
	4	O	O	O	O	-	X	X	X	X	X
	6	O	O	O	O	O	O	O	O	-	X
0.9	0	X	X	X	X	X	X	X	X	X	X
	3	O	X	X	X	X	X	X	X	X	X
	6	O	O	O	X	X	X	X	X	X	X
	9	O	O	O	O	O	-	X	X	X	X
	12	O	O	O	O	O	O	O	O	-	X

X - indicates that Method 5 achieves an initial colouring with significantly fewer colour classes than Method 8.
O - indicates that Method 5 achieves an initial colouring with significantly more colour classes than Method 8.

When implementing Methods 5 and 6 with x empty colour classes, our modification approach continues to identify initial, feasible colourings with significantly fewer colour classes than Method 0 for most test instances and values of x. This is clearly illustrated in Figure 5.5 where the lines for Methods 5 and 6 remain below the lines for Method 0 (i. e. the line of circles), in the most part. The only exception to this is for Method 6 with $x = 2$ empty colour classes on some of the test instance with $d = 0.1$ and $p \geq 0.015$, and even in these cases there is no significant increase (i. e. no significant difference was observed).

In comparison to Method 8, the initial feasible colouring achieved when implementing Methods 5 and 6 with x empty colour classes have both significantly fewer or significantly more colour classes dependent on x and p. This is again clearly illustrated in Figure 5.5, where the corresponding lines meet and cross one another, and more specifically for Method 5 in Table 5.4.

Unlike when applying the empty colour class approach for the edge dynamic GCP, here the approach does not (almost) exclusively produce feasible colouring with $k = |\mathcal{S}_t| + x$ colour classes for G_{t+1} where $\mathcal{S}_t$ is a feasible colouring for G_t. This can be observed in the differences between Figures 3.7 and 5.5, where the (almost) parallel line pattern present in Figure 3.7 for the different values of x is no-longer present in Figure 5.5. On the contrary, as x increases for the vertex dynamic GCP, the number of these additional empty colour classes that are utilised decreases with the change probability p. For low values of p and high values of x, Methods 5 and 6 with x empty colour classes produce feasible colourings with $k < |\mathcal{S}_t| + x$ colour classes for G_{t+1}.

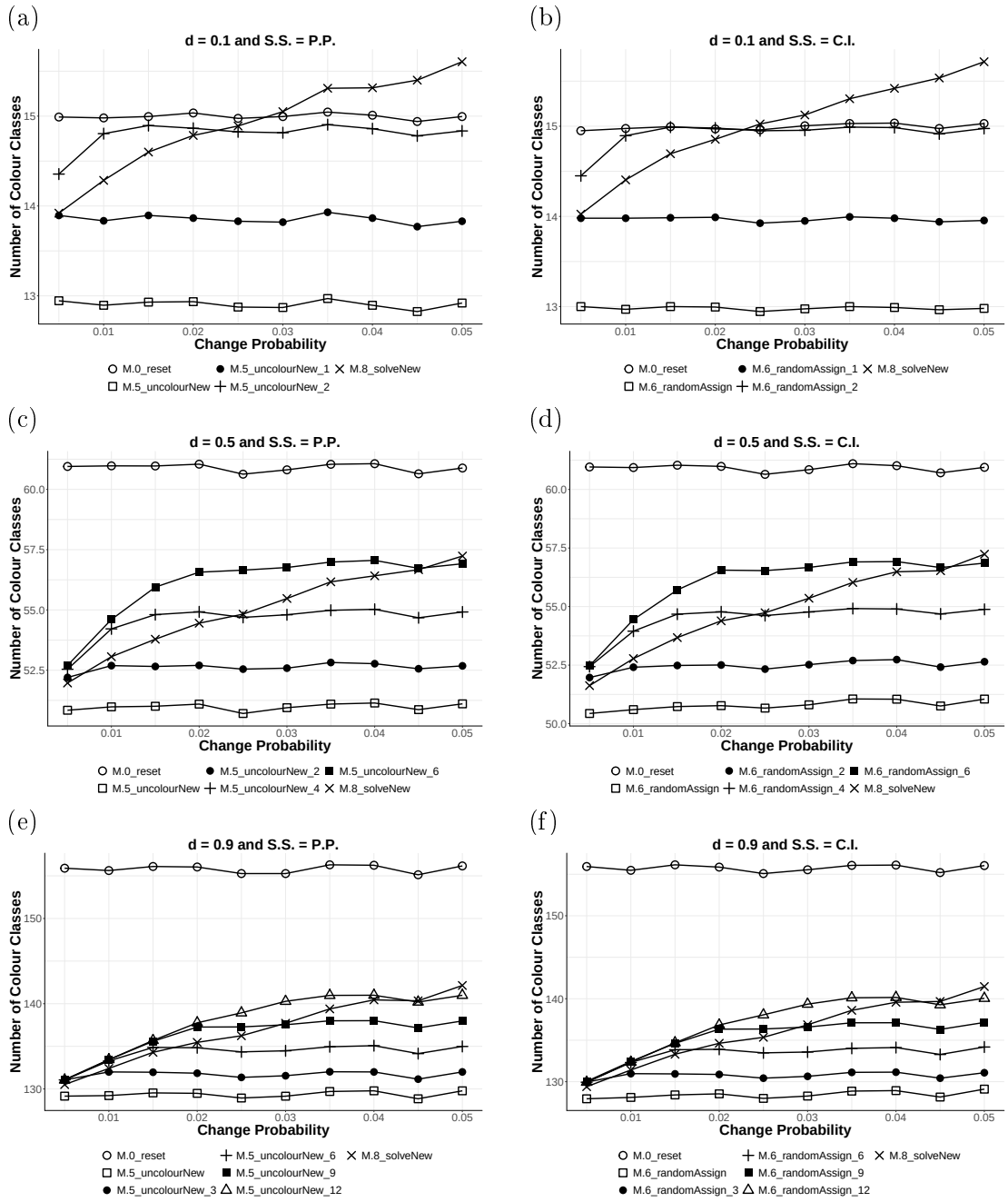

Figure 5.5: Mean number of colour classes in initial, feasible colourings for the vertex dynamic GCP with Methods 5 and 6 including empty colour classes on test instances with $n = 500$. Graphs (a), (c) and (e) represent methods that operate in the partial, proper solution space, graphs (b), (d) and (f) represent methods that operate in the complete, improper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

For the vertex dynamic GCP, the only problematic portions (clashes or "uncoloured" vertices) of an initial colouring $\mathcal{S}_{t+1}$ produced by Methods 5 and 6 will be caused by the new vertices in V_{t+1}^{+}. It is therefore likely that the tabu search operators will transfer approximately the same number or fewer vertices to the x empty colour classes than are introduced in V_{t+1}^{+}. In other words,

$$\sum_{i=|\mathcal{S}_t|+1}^{x} |S_i| \lesssim |V_{t+1}^{+}| \approx np \tag{5.6}$$

where $\mathcal{S}_t$ is a feasible colouring for G_t and S_i is the ith colour class of the initial feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1}. For particularly small values of p and high values of x (e. g. $p \leq 0.02$ and $x = 12$), it is clear that $x > np$ which implies that not all empty colour classes will be utilised.

Of course this assumes that each new vertex in V_{t+1}^{+} is adjacent to at least one vertex in each of the colour classes of $\mathcal{S}_t \backslash V_{t+1}^{-}$, and that all of the new vertices are adjacent to one another. This would imply that the tabu search operators will transfer vertices to the empty colour classes such that each empty colour class is occupied by a single vertex. This is a worst case scenario, which would explain why our empty colour class approach still identifies initial feasible colourings with $k < |\mathcal{S}_t| + x$ colour classes for G_{t+1} in cases where $x \leq np$.

With regards to the amount of time required to identify an initial feasible colouring, accurate comparisons were not achievable for test instances with $d = 0.1$ and $x \geq 1$ free colour classes because in most cases the time required was less than 10^{-3} seconds. For the remaining cases, there is a clear negative relationship between the time required to identify an initial feasible colouring and the number of empty colour classes x, as can be seen in Figure 5.6. There is also a negative relationship between the number of tabu search iterations required and the number of empty colour classes (see Figure 5.7). As k increases, so too does the number of feasible k-colourings, which means less computational effort is required to identify one. Therefore, it makes sense that the time (and number of tabu search iterations) required decreases as x increases because the initial k-GCP attempted by the tabu search operators is for $k = |\mathcal{S}_t| + x$.

Due to this, there are now observable cases where Methods 5 and 6 with x empty colour classes identify initial, feasible colourings in significantly less time than Method 0. The positive relationship between computational effort and the change probability p also comes into play here such that, as x increases, so too does the largest value of p for which Methods 5 and 6 are significantly faster than Method 0. The results presented in Table 5.5 for Method 5 hold true for Method 6 also with the exception of using $x = 3$ empty colour classes for test instances with $d = 0.9$ and $p = 0.025$, where Method 0 is significantly faster.

These reductions in time do not affect the relationship between Methods 5 and 6

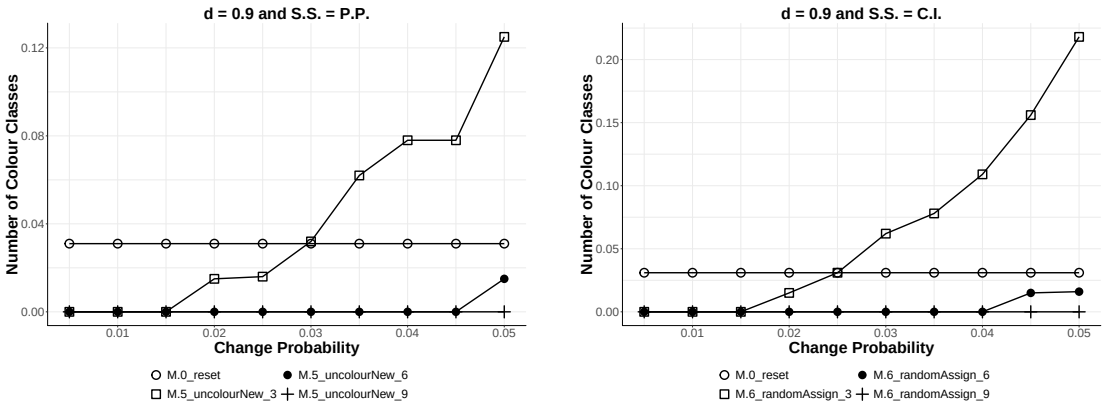

Figure 5.6: Median time (in seconds) for Methods 5 and 6 with $x \in \{3, 6, 9\}$ empty colour classes to achieve an initial, feasible colouring on test instances with $n = 500$ and $d = 0.9$.

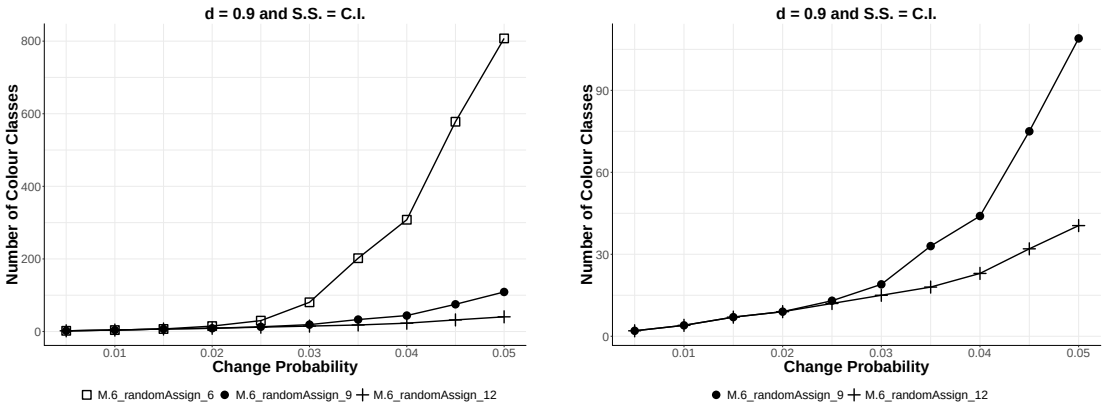

Figure 5.7: Median number of TabuCol iterations for Method 6 with $x \in \{6, 9, 12\}$ empty colour classes to achieve an initial, feasible colouring on test instances with $n = 500$ and $d = 0.9$.

Table 5.5: Significant differences in the time required by Method 5 with x empty colour classes to identify an initial, feasible colouring against Method 0.

		p									
d	x	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.5	0	O	O	O	O	O	O	O	O	O	O
	2	X	X	X	X	X	X	X	-	-	O
	4	X	X	X	X	X	X	X	X	X	X
	6	X	X	X	X	X	X	X	X	X	X
0.9	0	O	O	O	O	O	O	O	O	O	O
	3	X	X	X	X	-	O	O	O	O	O
	6	X	X	X	X	X	X	X	X	X	X
	9	X	X	X	X	X	X	X	X	X	X
	12	X	X	X	X	X	X	X	X	X	X

X - indicates that Method 5 requires significantly less time than Method 0.
O - indicates that Method 5 requires significantly more time than Method 0.

and Method 8 with regards to the time required, i. e. Method 8 remains significantly faster regardless of the number of empty colour classes.

5.4.2 Final Colourings

Finally, let us consider final colourings for the vertex dynamic GCP. As mentioned previously in Section 5.1, a small change to the edge set will usually affect more vertices than a comparable change to its vertex set.

Method 5 was found to achieve final, feasible colourings with significantly fewer colour classes than Method 0 for test instances with $n = 500, d = 0.9$ and some values of p (no obvious pattern); $n = 1000, d = 0.1, p \leq 0.02$; and $n = 1000, d \in \{0.5, 0.9\}$ with most[5] values of p. Similarly, Methods 6 and 7 were also found to achieve final, feasible colourings with significantly fewer colour classes than Method 0 for test instances with $n = 500, d = 0.5, p \leq 0.04$; $n = 500, d = 0.9, p \leq 0.03$; $n = 1000, d = 0.1, p \leq 0.01$; and $n = 1000, d \in \{0.5, 0.9\}$ with most values of p (see Footnote 5). On the other hand, Method 5 achieves final, feasible colourings with significantly more colour classes than Method 8 for test instances with $n = 1000, d = 0.9$ and $p \geq 0.03$. This is also true for Methods 6 and 7 on test instances with $n = 500, d = 0.9, p \geq 0.035$; and $n = 1000, d = 0.9, p \geq 0.015$. Although Methods 5 to 7 require significantly more time to produce initial, feasible colourings (see Table 5.2) it is likely that Methods 0 and 8 still require more time to reach a feasible colouring with equivalent numbers of colour classes for low levels of p. This would imply that the tabu search operators will attempt to solve k-GCPs for smaller values of k at an earlier stage when Methods 5 to 7 are executed compared to Methods 0 and 8. In other words, an initial feasible colouring is likely to have the same number of colour classes as the final feasible colouring when Methods 5 to 7 are executed, whereas several k-GCPs with decreasing values of k will have to be solved when Methods 0 and 8 are executed. Further analysis should be conducted in order to investigate the validity of this proposition.

The relationships between the number of colour classes in the final colourings achieved when Methods 0 and 5 to 8 are implemented is illustrated for test instances with $n = 1000$ and $d \in \{0.5, 0.9\}$ in Figure 5.8.

Unlike Method 4 for the edge dynamic problem, Method 8 was only found to reach final, feasible colourings with the same or significantly fewer colour classes than Method 0, especially for test instances with $n \in \{500, 1000\}$ and $d \in \{0.5, 0.9\}$ (see Figure 5.8). Both Methods 0 and 8 start each time-step from a feasible colouring; however, Method 8 achieves initial colouring with significantly fewer colour classes than Method 0 for most test instances (see Figure 5.3 and Table 5.3). Therefore,

[5]The only test instances with $n = 1000$ and $d \in \{0.5, 0.9\}$ that Methods 5 to 7 did not achieve final colourings with significantly fewer colour classes than Method 0 were for parameters $d = 0.5, p = 0.025$ for Method 5, $d = 0.5, p = 0.04$ for Method 6, and $d = 0.9, p = 0.05$ for Method 7.

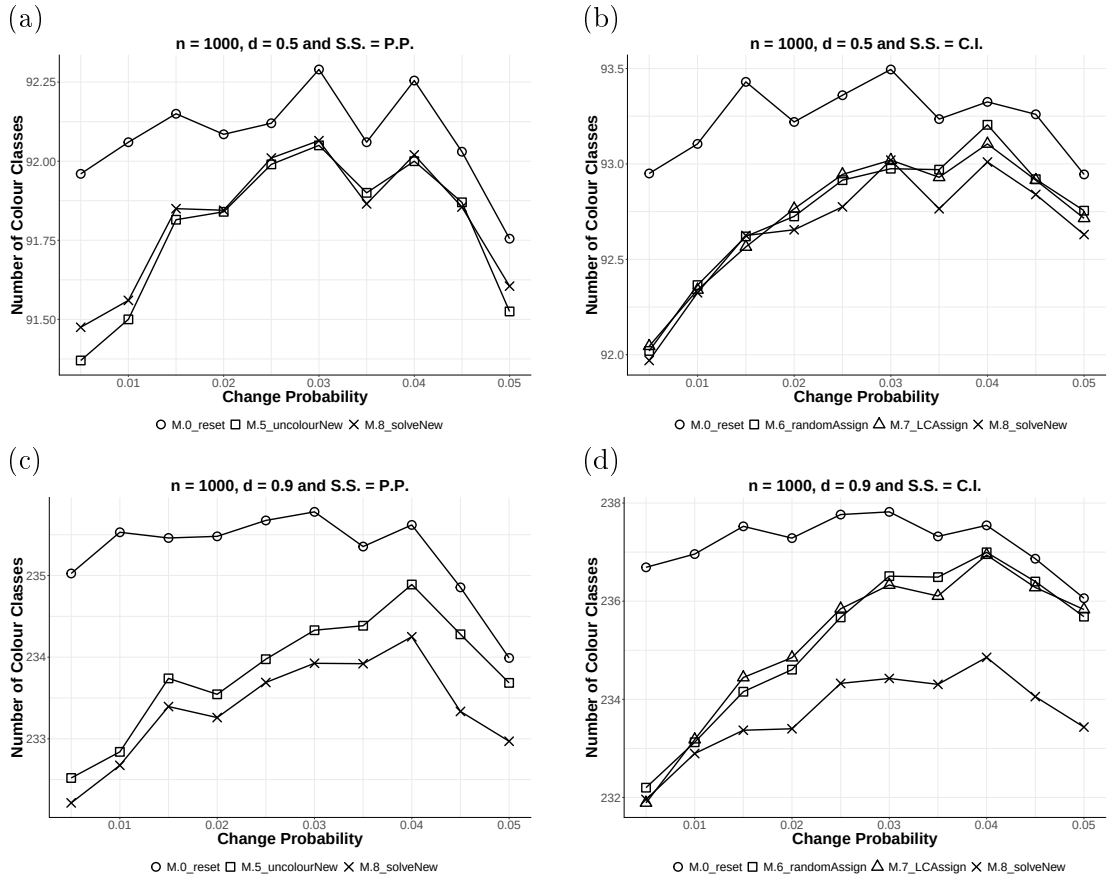

Figure 5.8: Mean number of colour classes in final, feasible colourings for the vertex dynamic GCP on test instances with $n = 1000$ and $d \in \{0.5, 0.9\}$. Graphs (a) and (c) represent methods that operate in the partial, proper solution space, graphs (b) and (d) represent methods that operate in the complete, improper solution space, and the rows from top to bottom represent test instances with $d = 0.5$ and 0.9, respectively.

Table 5.6: Significant differences in the time required by Methods 0 and 7 to achieve a final feasible colouring with an equal number of colour classes.

		p									
n	d	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	X	X	X	X	X	X	X	X	X	-
	0.5	X	X	X	X	-	-	-	-	-	-
	0.9	X	X	-	-	O	O	-	O	O	O
500	0.1	X	X	X	X	X	X	X	X	X	X
	0.5	X	X	X	X	X	-	-	-	-	-
	0.9	X	X	X	-	-	-	-	-	-	-
1000	0.1	X	X	X	X	X	X	-	-	-	-
	0.5	X	X	X	X	X	-	-	-	-	-
	0.9	-	-	X	-	-	-	-	-	-	-

X - indicates that Method 7 achieves final, feasible in significantly less time than Method 0.
O - indicates that Method 7 achieves final, feasible in significantly more time than Method 0.

the tabu search operator will attempt to solve k-GCPs for smaller values of k at an earlier stage when Method 8 is implemented in these cases.

As with the time comparison analysis for final feasible colourings in Chapters 3 and 4, here we only compare the test instance time-steps for which the number of colour classes in the final feasible colourings achieved are equal when both modification operators are implemented.

It was found that Methods 5 to 7 achieve final, feasible colourings in significantly less time than Method 0 on test instances with low values of n, d and p. To see how these different parameters affect the time requirements, consider Tables 5.6 and 5.7. The reason for these observations is likely to be the same as that given with regards to the number of colour classes in the final, feasible colourings achieved with low levels of p: i.e. the tabu search operators attempt to solve k-GCPs for smaller values of k at an earlier stage and therefore arrive at the final feasible colourings at an earlier stage too.

On the contrary, Methods 5 to 7 require significantly more time to achieve final, feasible colourings compared to Method 8 for high values of n, d and p. In comparison to Method 0, Method 8 starts from a feasible colouring with significantly fewer colour classes for most test instances which likely translates to the tabu search operators attempting to solve k-GCPs for smaller values of k before an initial, feasible colouring has been identified after the implementation of Methods 5 to 7.

Method 8 was also found to require significantly less time to achieve final, feasible colourings than Method 0 for most test instances (again, see Table 5.7), with the exception of some test instances with $d = 0.9$ or high values of p. This is probably due to the same argument given throughout this section with regards to the tabu search operators attempting to solve k-GCPs for lower values of k.

These findings suggest a knock-on effect from the reductions achieved with re-

Table 5.7: Median time (in seconds) required to obtain final, feasible colourings with the same numbers of colour classes for the vertex dynamic GCP on test instances with $n = 500$. For results regarding test instances with $n \in \{250, 1000\}$, see Table B.4 in Appendix B.

			p									
d	S.S.	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	P.P.	0	0.016	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031
		5	0*	0*	0.015*	0.015*	0.016*	0.016*	0.016*	0.031*	0.031*	0.031*
		8	0*	0.015*	0.015*	0.015*	0.016*	0.016*	0.016*	0.031*	0.031*	0.031*
	C.I.	0	0.046	0.046	0.047	0.047	0.047	0.047	0.047	0.047	0.047	0.062
		6	0*	0*	0.015*	0.015†	0.016†	0.031*	0.031*	0.031*	0.031*	0.047*
		7	0*	0*	0.015*	0.015†	0.016†	0.031*	0.031†	0.031*	0.031*	0.046*
		8	0*	0*	0.015*	0.016*	0.031*	0.031*	0.031*	0.046*	0.039*	0.047
0.5	P.P.	0	2.652	3.073	3.276	3.151	2.980	2.504	2.964	2.457	2.933	2.855
		5	1.505*	1.264*	2.574	2.621	2.341	3.066	2.566	2.551	3.167	2.980
		8	0.203*	1.068*	1.092*	1.529*	1.544*	1.420*	1.381*	2.192	2.387	2.293
	C.I.	0	5.141	3.525	4.181	4.509	4.227	3.378	3.884	3.900	3.776	3.261
		6	0.305*	1.263*	1.856*	1.731*	2.184*	2.496	3.323	2.902	4.009	5.007
		7	0.484*	1.185*	1.654*	2.246*	3.307*	3.370	2.839	3.682	3.354	3.261
		8	0.328*	1.264*	1.155*	1.560*	2.106*	1.521*	2.106*	2.480*	1.997*	2.246*
0.9	P.P.	0	4.867	5.281	4.680	5.492	5.585	4.267	4.181	4.665	4.602	4.462
		5	2.964*	3.362*	4.665	5.194	4.446	4.196	6.255	5.585	5.054	5.281
		8	1.373*	2.013*	2.566*	1.981*	3.261*	3.330*	3.416*	2.745*	3.884	4.189*
	C.I.	0	5.569	6.458	6.318	6.255	5.756	5.538	6.404	5.055	5.788	6.240
		6	1.256*	4.883*	4.345*	4.275*	5.351	5.007	5.538	6.568	5.842	6.100
		7	1.865*	3.744*	4.165*	5.226	5.149	5.710	5.445	6.380	6.069	5.007
		8	1.498*	2.090*	2.176*	3.478*	4.696*	3.681*	3.651*	3.573*	3.682*	4.009*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than Method 0 for the same values of d and p whilst operating in the same solution space.
† indicates a time that is significantly less than Methods 0 and 8 for the same values of d and p whilst operating in the same solution space.

gards to initial feasible colourings and the subsequent reductions that are then observed with regards to the final feasible colourings. Both reducing the number of colour classes in the initial, feasible colourings and the time required to achieve them seems to positively affect the quality and computational efforts required for the final feasible colourings.

Method 6 (*randomAssign*) vs. Method 7 (*leastClashesAssign*)

As with the initial feasible colourings achieved when implementing Methods 6 and 7, there are very few observable differences with regards to the final feasible colourings achieved when implementing these modification operators. In the rare cases where a significant difference is observed, there is no obvious pattern to explain why they have occurred.

For example, Method 6 achieves final feasible colourings with significantly fewer colour classes than those achieved under Method 7 for test instances with $n = 500, d = 0.1$ and $p = 0.01$, but significantly more for test instances with $n = 500, d = 0.9, p = 0.05$; and $n = 1000, d = 0.9, p \in \{0.005, 0.035\}$. Similarly, for test instances with $n = 500, d = 0.5$ and $p = 0.05$, Method 7 achieves final feasible colourings in significantly less time compared to Method 6, but in no other cases.

We therefore conclude that there is no obvious advantage to be gained from using Method 7 rather than Method 6 (or vice versa), with regards to either the initial or final feasible colourings achieved.

With Empty Colour Classes

In comparison to using the empty colour classes approach for the edge dynamic GCP, it appears to be less detrimental with regards to the final feasible colourings when applied to the vertex dynamic GCP.

For test instances with $d = 0.1$, Methods 5 and 6 with one or two empty colour classes (i. e. $x \in \{1, 2\}$) achieve final feasible colourings with no significant difference in the number of colour classes compared to those achieved under Method 0. For test instances with $d = 0.5$ there is a detrimental effect as x increases, such that Methods 5 and 6 with $x \in \{4, 6\}$ empty colour classes achieve final feasible colourings with significantly more colour classes than those achieved under Method 0 when $p \geq 0.02$ and 0.045 respectively. On the other hand, for test instances with $d = 0.9$, final feasible colourings with significantly fewer colour classes compared to those achieved under Method 0 are achieved for increasing values of p.

Similarly, Methods 5 and 6 continue to achieve final feasible colourings in significantly less time than Method 0 for test instances with $d = 0.1$ regardless of the number of empty colour classes x, at least for $x \leq 2$. They are able to achieve final feasible colourings in significantly less time compared to Method 0 on test instances with $d = 0.9$ for increasing values of p. For test instances with $d = 0.5$ the affect

Table 5.8: Significant differences in the time required to achieve a final feasible colourings with the same number of colour classes achieved under Methods 0 and 5 on test instances with $d \in \{0.5, 0.9\}$.

		p									
d	x	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.5	0	X	X	-	-	-	-	-	-	-	-
	2	X	X	X	X	-	X	X	X	-	-
	4	X	X	X	-	-	-	-	-	-	-
	6	X	X	X	-	-	-	-	-	-	O
0.9	0	X	X	-	-	-	-	-	O	-	-
	3	X	X	X	X	X	-	X	-	-	-
	6	X	X	X	X	X	X	X	X	X	X
	9	X	X	X	X	X	-	X	X	X	X
	12	X	X	X	X	X	X	X	-	X	X

X - indicates that Method 5 achieves final, feasible colourings in significantly less time than Method 0.
O - indicates that Method 5 achieves final, feasible colourings in significantly more time than Method 0.

of x is more complicated. For $x = 2$, they can achieve final feasible colourings in significantly less time for increased values of p compared to when $x = 0$. However, as x continues to increase, this pattern is reversed. These patterns are illustrated for Method 5 in Table 5.8.

In general, for test instances with $d = 0.1$ our empty colour classes approach is neither beneficial nor detrimental with regards to final feasible colourings achieved under Methods 5 and 6 compared to Method 0; for test instances with $d = 0.5$ it is detrimental; and for test instances with $d = 0.9$ it is beneficial.

The benefits observed on test instances with $d = 0.9$ is a clear knock-on effect from the initial feasible colourings achieved, which also have significantly fewer colour classes than those produced by Method 0 and require less time to be identified as x increases. However, for test instances with $d = 0.5$, as with the detrimental affects observed in Section 3.6 for the final feasible colourings of the edge dynamic GCP, there is no obvious reason why this behaviour is observed. In fact, it is contradictory to what is expected considering the benefits of the approach with regards to the initial feasible colourings achieved.

5.5 Conclusions

In this chapter we have discussed an alternative to edge dynamic graphs and their associated GCP i. e. vertex dynamic graphs and the vertex dynamic GCP. We have proposed using the same modification approach outlined in Section 3.2 and introduced four modification operators tailored specifically for the vertex GCP (*randomAssign, leastClashesAssign, uncolourNew* and *solveNew*) to be used within this approach. These modification operators modify a feasible colouring for one time-step

into an initial, though not necessarily feasible, colouring for the following time-step. These four modification operators have been compared against one another, where appropriate, and against an approach which treats each time-step of a dynamic graph as a static graph i.e. Method 0 (*reset*) introduced in Chapter 3.

Our experiments have shown that initial colourings with significantly fewer colour classes can be achieved by using Methods 5 to 7 (*uncolourNew*, *randomAssign* and *leastClashesAssign*, respectively), which all modify a feasible k-colouring for G_t into an infeasible k-colouring for G_{t+1} and then pass this colouring directly to a tabu search operator. However, there is a significant trade-off with respect to the time required to achieve an initial, feasible colouring when these modification operators are applied. This is identical behaviour to that observed for Methods 1 to 3 (*calculateClashes*, *uncolourClashes* and *uncolourMostClashing*, respectively) in the edge dynamic case, which also modify a feasible k-colouring for G_t into an infeasible k-colouring for G_{t+1}.

With regards to final feasible colourings, those achieved under Methods 5 to 7 were found to have both significantly fewer or significantly more colour classes than those achieved under Method 0 dependent on test instance parameters d and p (desired density and change probability, respectively). In particular, significantly fewer colour classes were observed for test instances with higher values of d and lower values of p. In comparison to Methods 1 to 3 for the edge dynamic GCP, reductions in the number of colour classes in the final feasible colourings were observed far more often. In fact, for most test instances with $n = 500, d = 0.9$; and $n = 1000, d \in \{0.5, 0.9\}$, Methods 5 to 7 were able to outperform Method 0 in this regard.

The strategy of adding $x > 0$ empty colour classes was again explored, in order to combat the trade-off between quality and computational effort with regards to initial feasible colourings. This time around, the empty colour classes were added to the colouring produced by Methods 5 and 6. As with the edge dynamic case, increasing the value of x decreased the computational effort required to identify an initial feasible colouring. This led to cases, with appropriate values of x, such that the initial feasible colourings achieved had significantly fewer colour classes than those achieved under Method 0 *and* less time was required to achieve them. Unlike the edge dynamic case, this approach has fewer detrimental effects with regards to the final feasible colourings achieved. In fact, detrimental effects are only observed on test instances with $d = 0.5$ when the number of empty colour classes satisfies $x \geq 4$. Furthermore, there are now observable benefits with regards to both quality and computational effort on test instances with $d = 0.9$.

It has also been shown, that Method 8 (*solveNew*), which modifies a feasible k-colouring for G_t into a feasible k'-colouring for G_{t+1} such that $k' \geq k$, can also achieve initial, feasible colourings with significantly fewer colour classes for low values of p on test instances with $d = 0.1$ and *all* values of p tested otherwise. This observation is

unlikely to hold as p increases beyond 0.05 on test instances with $d \in \{0.5, 0.9\}$. This modification operator was also shown to require significantly less time to produce initial, feasible colourings for test instances with $n \in \{500, 1000\}$.

Finally, using this modification operator led to final, feasible colourings with the same or significantly fewer colour classes than Method 0 on test instances with high values of n and d. It was also shown to require significantly less time to do so in most cases, though this is more frequently observed for low values of d and p. In comparison to Method 4 for the edge dynamic GCP, these improved cases are observed far more often.

Of the four modification operators introduced, Method 8 (*solveNew*) was the only one that improved upon both the quality (i. e. number of colour classes) and effort (i. e. computation time) required for initial feasible colourings, compared to treating each time-step as an individual static graph in most cases, especially on test instances with $d \in \{0.5, 0.9\}$. With regards to final feasible colourings, all four modification operators led to significant improvements to quality and significant reductions to the effort required, for test instances with high values of d and low values of p.

In general, these modification operators produce more beneficial results than their counterparts for the edge dynamic case. This is most likely due to the amount of "problematic" change (e. g. clashes or "uncoloured" vertices) being introduced. For our test instances, the amount of "problematic" change introduced between time-steps is likely to be larger for an edge dynamic GCP with change probability p compared to a vertex dynamic GCP with the same change probability. Therefore, in order to see results for the vertex dynamic GCP that more closely resemble those of the edge dynamic GCP with change probability p, test instances with larger change probabilities (i. e. greater than p) should be used.

Key Findings and Recommendations

- With regards to initial feasible colourings, Methods 5 to 7 behave in a similar fashion to Methods 1 to 3 for the edge dynamic GCP, as does Method 8 in comparison to Method 4.

- Although adding an "appropriate" number of empty colour classes to the colourings produced by Methods 5 and 6 increases their benefits, we suggest using Method 8 instead, as the results observed are comparable and the former requires extra knowledge concerning the "appropriate" number of empty colour classes to use.

- For high values of d and low values of p, all four modification operators for the vertex dynamic GCP achieve final feasible colourings with significantly fewer colour classes *and* require significantly less time to do so.

- In general, the benefits of using the modification approach to tackle the vertex dynamic GCP outweigh those observed when tackling the edge dynamic GCP, especially for increased values of p.

Here, change has occurred at random between time-steps. In the next chapter, we will extend the vertex dynamic GCP to include future information regarding potential changes to the vertex set. We aim to determine whether this information can be utilised, in the same way that future adjacency information was utilised for the edge dynamic GCP in Chapter 4, to improve the robustness of a colouring to change.

Chapter 6

Colouring Vertex Dynamic Random Graphs with Future Vertex Inclusion Information

In this chapter we will extend the work of the previous chapter in the same manner that the work of Chapter 3 was extended in Chapter 4, i. e. by considering future change information. First we propose a more suitable definition of vertex dynamic graphs; we then discuss how future information might be utilised in order to improve the robustness of a colouring with regards to potential change. Finally, the two stage approach introduced Section 4.2 will be implemented for the vertex dynamic GCP and experimental results will be presented.

6.1 Vertex Pool and Future Inclusion

As we saw in Chapters 3 and 4, the number of vertices $|V|$ is fixed over all time-steps for the edge dynamic GCP, which means that all potential edges $\mathcal{E} = \{\{u, v\} : u, v \in V, u \neq v\}$ are known. The probability of an non-active edge at time-step t (i. e. an edge $\{u, v\} \notin E_t$) being included in time-step $t + 1$ (i. e. the probability of $\{u, v\} \in E^+_{t+1}$) is an obvious piece of information that could be utilised if known, as we explored in Chapter 4.

Due to the nature of the way that our vertex dynamic graphs have been constructed up until this point (see Section 5.3.1), there is no such obvious piece of information that could be utilised for the vertex dynamic GCP. We therefore consider a slightly different type of vertex dynamic graph in this chapter.

We will continue to define a dynamic graph $\mathcal{G} = (G_0, G_1, \ldots, G_T)$ as a series of $T + 1$ static graphs where $G_t = (V_t, E_t)$ is the static representation of $\mathcal{G}$ at time-step t. We will now also consider a *pool* of vertices $\mathcal{V}$ such that, at each time-step, V_t is a subset of $\mathcal{V}$ (i. e. $V_t \subseteq \mathcal{V}$ for $t = 0, 1, \ldots, T$). At time-step t, we refer to the vertices in V_t as the *active* vertices.

To get from G_t to G_{t+1}, we define the set of deleted vertices $V_{t+1}^- \subseteq V_t$ as before (see Section 5.1), but now the set of new vertices is defined as a subset of the currently non-active vertices $V_{t+1}^+ \subseteq (\mathcal{V} \backslash V_t)$. The vertex set for G_{t+1} remains defined in the same way as before, i. e. $V_{t+1} = (V_t \cup V_{t+1}^+) \backslash V_{t+1}^-$.

The introduction of a vertex pool means that information regarding the likelihood of a given vertex being "active" in the following time-step can now be provided, which shall be referred to as future vertex inclusion information. However, without knowing which vertices a currently non-active vertex will be adjacent to, this is not particularly useful future change information. Therefore, we suggest predetermining a set of edges between all vertices in $\mathcal{V}$ so that future adjacency information can also be gained from future vertex inclusion information.

Consider a set of globally active edges $E_{\mathcal{V}} \subseteq \{\{u, v\} : u, v \in \mathcal{V}, u \neq v\}$ as a fixed subset of all $\binom{|\mathcal{V}|}{2}$ potential edges between the vertices in $\mathcal{V}$. Then, at each time-step, the edge set E_t is the subset of $E_{\mathcal{V}}$ such that if $\{u, v\} \in E_t$ then both u and v are active vertices at time-step t (i. e. $u, v \in V_t$). In other words, $E_t = E_{\mathcal{V}} \cap \mathcal{E}_t$ where $\mathcal{E}_t = \{\{u, v\} : u, v \in V_t, u \neq v\}$ is the set of all $\binom{|V_t|}{2}$ possible edges between vertices in V_t.

Given a graph $G_t = (V_t, E_t) \in \mathcal{G}$, let $p_{t+1}(v)$ be the probability that vertex $v \in \mathcal{V}$ is active at time-step $t + 1$ (i. e. the probability of $v \in V_{t+1}$). If $p_{t+1}(v)$ is known for every vertex $v \in \mathcal{V}$ then we can can define the $1 \times |\mathcal{V}|$ future inclusion vector P_{t+1} such that the vth entry of P_{t+1} is equal to $p_{t+1}(v)$. Given P_{t+1} and $E_{\mathcal{V}}$, if $\{u, v\} \in E_{\mathcal{V}}$ then vertices $u, v \in \mathcal{V}$ are future adjacent with probability[1] $p_{t+1}(u) \cdot p_{t+1}(v)$, otherwise $u, v \in \mathcal{V}$ are future adjacent with probability 0.

6.2 Method for Reducing Future Saturation

When a non-active vertex at time-step t is made active in the following time-step $t + 1$, it needs to be assigned to a colour class. Therefore, it would be advantageous for there to be at least one colour class that the newly active vertex could be assigned to such that no clashes are introduced. In fact, the more such colour classes that exist, the better.

Let us reconsider the saturation degree of an "uncoloured" vertex (i. e. Definition 2.1). Given a partial k-colouring $\mathcal{S} = \{S_1, \ldots, S_k\}$ of a graph $G = (V, E)$, the saturation degree $\deg_{sat}(v, \mathcal{S})$ of an "uncoloured" vertex $v \in V \backslash \{\bigcup_{i=1}^{k} Si\}$ with respect to $\mathcal{S}$, is equal to the number of colour classes in $\mathcal{S}$ for which there exists at least one vertex $u \in S_i$ such that u is adjacent to v (i. e. $\{u, v\} \in E$). The lower the saturation degree of a given vertex v, the more colour classes exist that v can be assigned to without introducing clashes. For a partial k-colouring $\mathcal{S}$, there exists

[1] The inclusion probabilities are independent here, because our test instances are constructed such that the probabilities are randomly sampled from a uniform distribution. Note, for test instances constructed in a different manner, this may not hold.

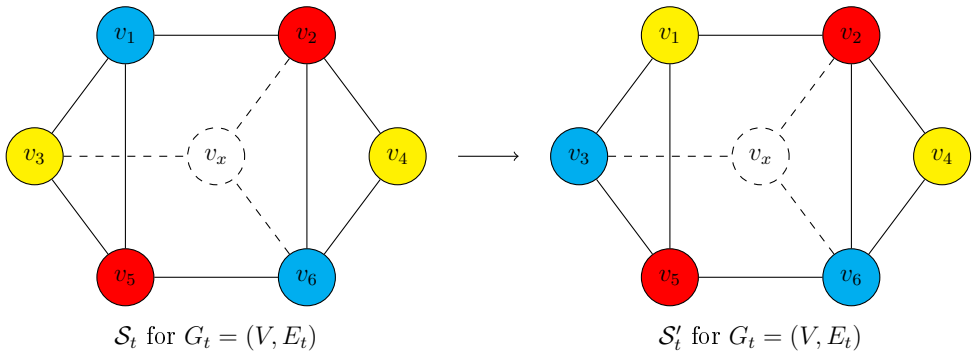

Figure 6.1: Example of a vertex dynamic graph with future inclusion information. Vertices with solid outlines represent the active vertices (i. e. those in V_t), the vertex with a dotted outline represents a vertex that is "highly likely" to be included in V^+_{t+1}, and the dotted lines represent edges in $E_{\mathcal{V}} \backslash E_t$.

exactly $k - \deg_{sat}(v, \mathcal{S})$ such colour classes.

Given a vertex dynamic graph $\mathcal{G}$ with a pool of vertices $\mathcal{V}$ and a set of globally active edges $E_{\mathcal{V}}$, at time-step t, a non-active vertex $v \in \mathcal{V} \backslash V_t$ can be considered as an "uncoloured" vertex with regards to a feasible colouring for G_t and, because $E_{\mathcal{V}}$ is known, we can utilise the same definition of saturation degree. More importantly, if we can reduce the saturation degree of a given non-active vertex, we can increase the number of colour classes that that vertex can be assigned to without introducing clashes.

Consider the example illustrated in Figure 6.1. The dotted vertex v_x is a non-active vertex at time-step t that is "highly likely" to become active in time-step $t+1$. Because $E_{\mathcal{V}}$ is known, we know which vertices will be adjacent to v_x if they are also included in V_{t+1} (i. e. the dotted edges). Both $\mathcal{S}_t$ and $\mathcal{S}'_t$ are feasible 3-colourings for G_t with $\deg_{sat}(v_x, \mathcal{S}_t) = 3$ and $\deg_{sat}(v_x, \mathcal{S}'_t) = 2$ respectively. It should be clear that if v_x is added to any of the colour classes of $\mathcal{S}_t$ it will introduce a clash, whereas it can be introduced to the yellow colour class of $\mathcal{S}'_t$ without doing so. As with the edge dynamic GCP with future adjacency information in Chapter 4, we wish to find a way to transform $\mathcal{S}_t$ into $\mathcal{S}'_t$ without compromising feasibility for the current time-step.

The total saturation of all non-active vertices at time-step t with regards to a feasible colouring $\mathcal{S}_t$ for G_t is given by

$$\mathcal{F}(\mathcal{S}_t) = \sum_{v \in \mathcal{V} \backslash V_t} \deg_{sat}(v, \mathcal{S}_t) \tag{6.1}$$

where $\deg_{sat}(v, \mathcal{S}_t)$ is the saturation degree of $v \in \mathcal{V} \backslash V_t$ with regards to $\mathcal{S}_t$. An attempt to reduce Equation (6.1) is an attempt to reduce the total saturation under the assumption that all non-active vertices have an equal probability of being included in V_{t+1}. If the future inclusion information P_{t+1} is known, then we can define

the expected future saturation degree of a non-active vertex $v \in \mathcal{V} \backslash V_t$ with regards to $\mathcal{S}_t$ as $p_{t+1}(v) \cdot \deg_{sat}(v, \mathcal{S}_t)$. The total expected future saturation is therefore given by

$$\mathcal{F}(\mathcal{S}_t) = \sum_{v \in \mathcal{V} \backslash V_t} p_{t+1}(v) \cdot \deg_{sat}(v, \mathcal{S}_t) \tag{6.2}$$

and, by attempting to reduce Equation (6.2), as opposed to Equation (6.1), priority is given to reducing the saturation degree of non-active vertices that are more likely to be included in V_{t+1}.

It is worth noting that one slight drawback of using Equation (6.2), is that potential clashes between two non-active vertices (i. e. vertices both in the set $\mathcal{V} \backslash V_t$) cannot be considered. However, as all non-active vertices are also "uncoloured", a pair of *adjacent* non-active vertices at time-step t will only clash with one another if they both become active in time-step $t+1$ and are then assigned to the same colour class by Method 6 or 7. It is therefore unlikely that this will cause much of an issue in practice.

6.2.1 Alternative Secondary Objective Functions

The following "secondary objective functions" were also considered, as alternatives to Equation (6.2) which could be used within our two stage approach.

If $\mathcal{S}_t = \{S_1, \ldots, S_k\}$ is a feasible k-colouring of $G_t \in \mathcal{G}$, we could attempt to reduce either

$$\sum_{v \in (\mathcal{V} \backslash V_t)} p_{t+1}(v) \max_{i \in \{1,\ldots,k\}} \{ \sum_{u \in S_i} \text{adj}(u, v) \} \tag{6.3}$$

or

$$\sum_{v \in (\mathcal{V} \backslash V_t)} p_{t+1}(v) \min_{i \in \{1,\ldots,k\}} \{ \sum_{u \in S_i} \text{adj}(u, v) \} \tag{6.4}$$

where $\text{adj}(u, v)$ is an identifier function equal to 1 if the edge $\{u, v\} \in E_{\mathcal{V}}$ and 0 otherwise. By using Equation (6.3), we would be attempting to minimise the number of clashes in the "worst" colour class for each non-active vertex (i. e. minimise the worst case scenario), and by using Equation (6.4) we would be attempting to minimise the number of clashes in the "best" colour class for each non-active vertex (i. e. maximise the best case scenario).

In practice, both of these secondary objective functions require much more time to be evaluated for each available Kempe-chain interchange or pair-swap. Here, the resultant "cost" change of each Kempe-chain interchange will be stored in a $|V| \times k$ matrix. The additional time required to evaluate Equations (6.3) and (6.4), compared to Equation (6.2), subsequently amplifies the time required to update this "cost" matrix after each iteration (i. e. many more checks are required to ensure the minimums or maximums are correctly identified after a neighbourhood move has been executed).

6.2.2 Performance and Behaviour Analysis

Here we will briefly look at how the tabu search procedure described in Section 4.3 behaves when the objective function Equation (4.3) is exchanged for Equation (6.2). As shown before, this tabu search procedure attempts to reduce a secondary objective, the total expected future saturation in this case, without affecting the feasibility of a colouring.

For these experiments we used a set of instances each consisting of a single graph G_0 and its future inclusion vector P_1. These test instances have a vertex pool size of $N \in \{550, 625, 750, 1000\}$, number of active vertices $n = 500$, a desired density $d \in \{0.1, 0.5, 0.9\}$, and change probability $p = 0.05$. For each test instance, the vertex pool is constructed with $|\mathcal{V}| = N$ and the set of globally active edges $E_{\mathcal{V}}$ contains each edge $\{u, v\}$ of the $\binom{|\mathcal{V}|}{2}$ potential edges between the vertices in $\mathcal{V}$ with probability d. G_0 is then constructed such that each vertex $v \in \mathcal{V}$ is included in V_0 with probability $\frac{N}{n}$ and the edge set is defined as $E_0 = \{\{u, v\} \in E_{\mathcal{V}} : u, v \in V_0\}$. The future inclusion vector P_1 is populated with values sampled from the uniform distribution $U[0, \frac{2np}{N-n}]$[2]. Twenty test instances were produced for each combination of N, n, d and p, totalling 240 test instances.

Other than the objective function and test instances, the rest of the parameters used in Section 4.3 remain the same here:

- Initial colourings are produced using the modified GREEDY algorithm described in Algorithm 4.2.
- The following target number of colour classes $k^\star$ are used:
  - $k^\star = 13, 14, 15, \ldots, 23$ for test instances with $d = 0.1$,
  - $k^\star = 49, 51, 53, \ldots, 79$ for test instances with $d = 0.5$, and
  - $k^\star = 125, 128, 131, \ldots, 185$ for test instances with $d = 0.9$.
- If the colouring produced by Algorithm 4.2 is infeasible then it is passed to TABUCOL [Hertz and de Werra, 1987] which attempts to find a feasible $k^\star$-colouring within a 900 second time limit.
- If a feasible $k^\star$-colouring has been identified then our tabu search procedure for reducing total expected future saturation is implemented for 10,000 iterations.
- The tabu tenure for inverse Kempe-chain interchanges and pair-swaps within our tabu search procedure is set to $\lfloor \frac{N}{2} \rfloor$ iterations.

The results of our experiments show that in all cases, our tabu search procedure was able to identify a colouring with a reduced total expected future saturation (i. e.

[2]See Section 6.3.1 for the arguments regarding why this upper bound is used within this distribution.

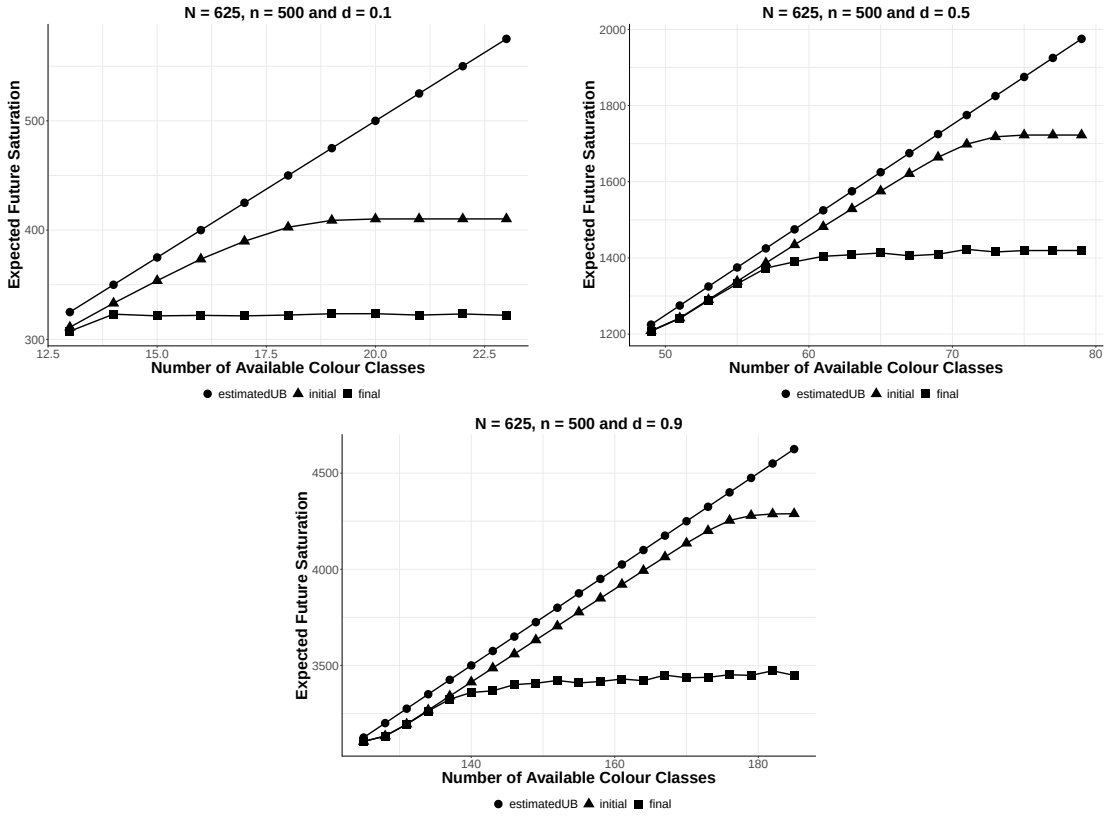

Figure 6.2: Mean expected total saturation against target number of colour classes (in clockwise order, starting from the top-left, graphs represent trials conducted on test instances with $N = 625$ and $d = 0.1, 0.5$ and 0.9, respectively).

Equation (6.2)) compared to the initial colouring achieved via the modified GREEDY algorithm or TABUCOL. In Figure 6.2, the mean total expected future saturation of the initial and final colourings, along with an "estimated" upper bound, are plotted against $k^\star$ for test instances with $N = 625$. The "estimated" upper bound for total expected future saturation at time-step t, where each non-active vertex is adjacent to at least one vertex in every colour class of a colouring $\mathcal{S}_t$, is given by

$$\mathcal{F}(\mathcal{S}_t)_{UB} = |\mathcal{V} \backslash V_t| \cdot k^\star \cdot \mathbb{E}(p_{t+1}) \tag{6.5}$$

where p_{t+1} is the distribution of future inclusion probabilities for non-active vertices in $\mathcal{V} \backslash V_t$ and $\mathbb{E}(p_{t+1})$ is the expected value of this distribution[3].

In a similar fashion to Figure 4.5, Figure 6.2 demonstrates a positive relationship between the value of $k^\star$ and the amount of reduction observed between the initial and final colourings. The correlation coefficients for this particular relationship range between 0.828 and 0.970 for the different values of N and d, with the coefficients tending towards the upper bound of this range for higher values of d and vice versa.

The initial total expected future saturations becomes less similar to the "esti-

[3]For our test instances $|\mathcal{V}| = N, |V_0| \approx n$, and $\mathbb{E}(p_1) = \frac{np}{N-n}$ where $p_1 = U[0, \frac{2np}{N-n}]$. Therefore, the "estimated" upper bound for the the total expected future saturation is equal to $np \cdot k^\star$ here.

Table 6.1: Median time (in seconds) required for our tabu search procedure to complete 10,000 iterations.

$d = 0.1$	$k^\star$	13	14	15	16	17	18	19	20
	runTime	13.363	13.732	14.360	16.380	17.871	19.318	20.591	22.071
	$k^\star$	21	22	23					
	runTime	23.287	24.327	25.629					
$d = 0.5$	$k^\star$	49	51	53	55	57	59	61	63
	runTime	29.748	30.019	30.384	30.763	31.142	32.311	33.087	34.063
	$k^\star$	65	67	69	71	73	75	77	79
	runTime	34.833	35.470	36.563	37.210	37.897	38.557	39.099	39.777
$d = 0.9$	$k^\star$	125	128	131	134	137	140	143	146
	runTime	27.157	27.332	27.125	26.815	27.006	27.559	27.866	28.624
	$k^\star$	149	152	155	158	161	164	167	170
	runTime	29.256	29.988	30.457	30.721	31.366	31.939	32.423	32.937
	$k^\star$	173	176	179	182	185			
	runTime	33.295	33.894	34.076	34.785	35.322			

mated" upper bound for the largest values of $k^\star$ used. In fact, these are the same values of $k^\star$ for which the initial number of expected future clashes became less similar to its "estimated" value in Section 4.3.2 (compare Figures 4.5 and 6.2). This observation is again due to the fact that for large values of $k^\star$, empty colour classes will be added to the initial colouring (i. e. Lines 15-17 of Algorithm 4.2 are utilised), which are not accounted for within the "estimates".

It is worth noting that the total expected future saturation naturally decreases with $k^\star$, as can be seen in Figure 6.2. However, this does not indicate that the robustness of a colouring can be increased by simply reducing the number of available colour classes. On the contrary, as $k^\star$ decreases, it becomes more likely that the saturation degree of each non-active vertex with regards to a feasible $k^\star$-colouring is equal to $k^\star$ itself. As discussed in Section 6.2, it is more desirable for the saturation degree of each non-active vertex to be *less* than $k^\star$.

In all other observations, the tabu search procedure continues to act in the same manner demonstrated in Section 4.3.2 despite changing the objective function, i. e. there is still a positive relationship between $k^\star$ and the time required to complete 10,000 iterations (see Table 6.1) and the majority of reduction takes place in the early stages of the procedure (see Figure 6.3). All previous explanations for these relationships carry over from Section 4.3.2 also.

We therefore propose to continue using the tabu search procedure introduced in Section 4.3.2 for the second stage of Algorithm 4.1 but simply swap Equation (4.3) for Equation (6.2).

6.3 Trial Information

We now look at whether reducing the total expected future saturation within our two stage approach leads to more robust colouring with regards to potential future changes. Moreover, we hope to see improvements with regards to the quality of the

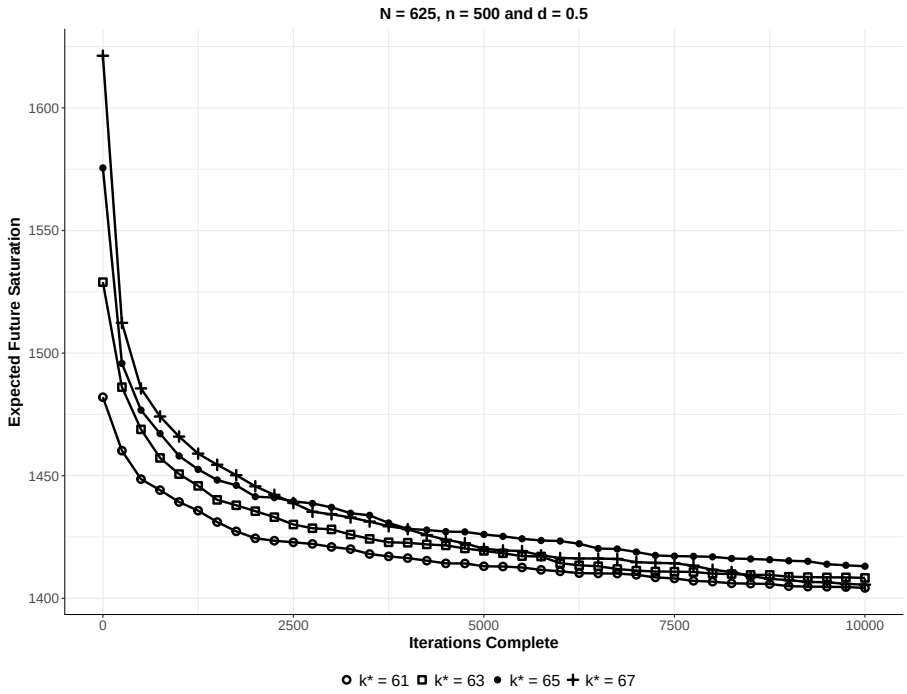

Figure 6.3: Mean expected total saturation against the number of iterations of tabu search procedure completed for test instances with $N = 625$ and $d = 0.5$.

initial feasible colourings achieves (i. e. fewer colour classes) and the computational effort required to achieve them (i. e. less time). As with the results presented in Chapter 4, we expect any benefits of our two stage approach to be dependent on the value of $k^\star$.

6.3.1 Test Instances

The graphs that make up our test instances have five parameters: N the size of the vertex pool, n the desired number of active vertices, d the desired density, p the change probability, and T the number of time-steps. The following parameter values were used for our test instances: $N \in \{550, 625, 750, 1000\}$, $n = 500$, $d \in \{0.1, 0.5, 0.9\}$, $p = 0.005, 0.01, \ldots, 0.05$ and $T = 10$. The set of globally active edges $E_{\mathcal{V}}$ is constructed such that every edge $\{u, v\}$ of the $\binom{N}{2}$ possible edges between the vertices in $\mathcal{V}$ is included with probability d. G_0 is then constructed such that each vertex $v \in \mathcal{V}$ is included in V_0 with probability $\frac{n}{N}$ and E_0 contains the edges $\{u, v\} \in E_{\mathcal{V}}$ such that $u, v \in V_0$.

At each time-step $t = 0, 1, \ldots, T-1$, every active vertex $v \in V_t$ is copied to the set of deleted vertices V_{t+1}^- with probability p. Simultaneously, every non-active vertex in $\mathcal{V} \backslash V_t$ is copied to the set of new vertices V_{t+1}^+ with a probability sampled from the uniform distribution $U[0, \frac{2np}{N-n}]$. By using this distribution, each vertex in $\mathcal{V} \backslash V_t$ is copied to V_{t+1}^+ with an expected probability of $\frac{np}{N-n}$ which ensures that the number of active vertices remains approximately equal over all time-steps (see

below). The sets of deleted edges and new edges are then defined as $E^-_{t+1} = \{\{u, v\} \in E_t : u \text{ or } v \in V^-_{t+1}\}$ and $E^+_{t+1} = \{\{u, v\} \in E_{\mathcal{V}} \backslash E_t : u \text{ or } v \in V^+_{t+1}\}$ respectively.

Legal Parameter Combinations for Vertex Dynamic Graphs with a Pool of Vertices

Unlike the test instances for the edge dynamic GCP in Chapters 3 and 4, there were no obvious relationships that the test instance parameters must adhere to for the vertex dynamic GCP in Chapter 5. The set of new vertices V^+_{t+1} was simply constructed such that $|V^+_{t+1}| \approx np$, in order to maintain an approximately equal number of active vertices over all time-steps. However, now, V^+_{t+1} is a subset of a fixed vertex pool $\mathcal{V}$, so we get the following.

As noted previously, at time-step t, there are $|V_t|$ active vertices out of the $|\mathcal{V}| = N$ available vertices in the vertex pool. The number of vertices at time-step $t + 1$ is therefore calculated as

$$|V_{t+1}| \;=\; |V_t| - |V^-_{t+1}| + |V^+_{t+1}|$$

If each active vertex at time-step t is included in V^-_{t+1} with probability p and each non-active vertex is included in V^+_{t+1} with probability q then the above can be approximated to

$$\begin{aligned} |V_{t+1}| &= |V_t| - p|V_t| + q(|\mathcal{V}| - |V_t|) \\ &= (1 - p)|V_t| + q(N - |V_t|) \end{aligned}$$

If we desire the number of active vertices to remain approximately equal over all time-steps, then we can substitute $|V_t|$ and $|V_{t+1}|$ for n, where n is the desired number of active vertices. Rearranging then gives $q = \frac{np}{N-n}$. Therefore, if each active vertex in V_t is added to V^-_{t+1} with probability p then each non-active vertex in $\mathcal{V} \backslash V_t$ should be added to V^+_{t+1} with probability $q = \frac{np}{N-n}$.

Given that q is a probability, it must satisfy $0 \leq q \leq 1$. It therefore follows that N, n and p must also satisfy $0 \leq \frac{np}{N-n} \leq 1$. The number of active vertices is non-negative and cannot exceed the size of the vertex pool (i. e. $0 \leq n \leq N$), p is a probability that satisfies $0 \leq p \leq 1$ so we can conclude that $\frac{np}{N-n} \geq 0$ always holds. It follows that the only concern is satisfying $p \leq \frac{N-n}{n}$.

The graph in Figure 6.4 illustrates the legal combinations of the change probability p and the vertex pool size N (represented by the shaded area) for a fixed number of desired active vertices n. All combinations of the parameter values of our test instances satisfy these inequalities. More specifically, they fall within the feasible region of Figure 6.4 as we have fixed $n = 500$ for all instances.

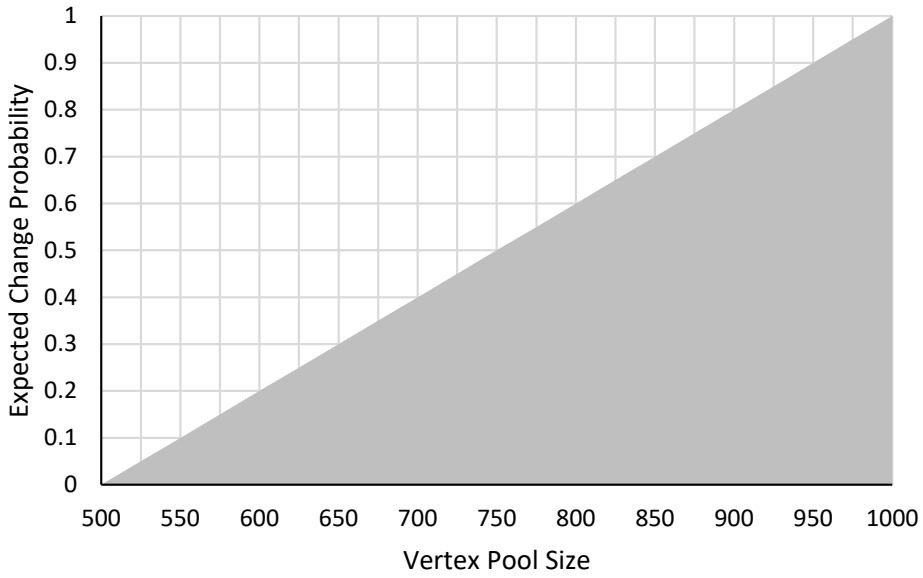

Figure 6.4: Legal combinations of change probability p and vertex pool size N that maintain an approximately equal number of active vertices $n = 500$ over all time-steps for a vertex dynamic graph with a vertex pool.

6.3.2 Algorithm Parameters

The algorithm parameters used here are the same as those used for the two-stage approach applied to the edge dynamic GCP with future adjacency information in Section 4.4.2. To briefly summarise:

- DSatur [Brélaz, 1979] is used instead of RLF [Leighton, 1979] as our constructive operator for Methods 0 and 8.
- TabuCol [Hertz and de Werra, 1987] and PartialCol [Blöchliger and Zufferey, 2008] are implemented to tackle the k-GCPs within our two-stage approach (i. e. Line 5 of Algorithm 4.1) when operating in the complete, improper and partial, proper solution spaces respectively.
- During execution, if no feasible colouring has been identified within half of the allotted time limit then k will be increased by 1, and so on (i. e. Lines 9 and 10 of Algorithm 4.1).
- The value of k will never be reduced below the target number of colour classes $k^\star$. If an initial feasible k-colouring is obtained such that $k < k^\star$ then $k^\star - k$ empty colour classes will be added to the colouring.
- As $n = 500$ for all test instances in this chapter, the target number of colour classes $k^\star$ used are:
  - $k^\star = 12, 13, \ldots, 18$ for test instances with $d = 0.1$,

- $k^\star = 48, 50, \ldots, 70$ for test instances with $d = 0.5$, and
- $k^\star = 125, 128, \ldots, 170$ for test instances with $d = 0.9$.

- The tabu search operator used for reducing total future saturation (i. e. Line 13 of Algorithm 4.1) has the following properties:
  - the objective is to reduce Equation (6.2), the total expected future saturation,
  - the best Kempe-chain interchange or pair-swap is executed during each iteration,
  - the associated inverse moves are then made "tabu" for a fixed tabu tenure of $\lfloor \frac{N}{2} \rfloor$ iterations[4],
  - if all moves are currently "tabu" then a random Kempe-chain interchange is performed, and
  - the aspiration criterion allows currently "tabu" moves to be performed if the resultant colouring $\mathcal{S}'$ satisfies $\mathcal{F}(\mathcal{S}') < \mathcal{F}(\mathcal{S}_{best})$.
- The time limit is set to 10 seconds and is cumulative over both stages of Algorithm 4.1, i. e. the time remaining after a feasible $k^\star$-colouring has been identified is then used to reduce total future saturation.

6.4 Results

6.4.1 Without Secondary Optimisation

In a similar fashion to Section 4.5.1, here we simply aim to illustrate that the relationships observed between the modification operators in Chapter 5 remain the same for these test instances. We also wish to show that there is again a slight shift in the relationships with regards to p, due to the use of DSatur as our constructive operator as opposed to RLF. Finally, we hypothesise that the size of the vertex pool N has little-to-no effect on the previously presented results, which we will therefore aim to demonstrate.

As discussed in Section 4.5.1, our two-stage approach behaves in a similar manner to Algorithm 3.2 if $k^\star$ is set to the minimum values in Table 4.4 i. e. $k^\star = 12, 48$ and 125 for test instances with $d = 0.1, 0.5$ and 0.9 respectively.

Initial Colourings

As in Chapter 5, the initial feasible colourings achieved when implementing Methods 5 to 7 have significantly fewer colour classes than Methods 0 and 8 but require

[4]Here we arbitrarily chose to link the length of the tabu tenure with the size of the vertex pool N as opposed to the number of desired vertices n.

Table 6.2: Median time (in seconds) required to obtain an initial, feasible colouring for the vertex dynamic GCP on test instances with $N = 625$.

		p									
d	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	0	0	0	0	0	0	0	0	0	0	0
	5	0	0.016	0.016	0.031	0.016	0.024	0.031	0.031	0.031	0.031
	6	0	0.016	0.016	0.031	0.047	0.031	0.031	0.047	0.047	0.047
	7	0	0.016	0.031	0.031	0.031	0.047	0.047	0.047	0.047	0.062
	8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
0.5	0	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016	0.016
	5	0.468	0.936	1.326	1.903	1.942	2.254	2.036	2.402	2.652	3.323
	6	0.382	1.194	1.310	1.880	2.005	1.958	1.973	2.472	2.496	2.684
	7	0.850	1.319	1.942	1.506	2.270	1.911	1.872	2.855	2.161	2.114
	8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
0.9	0	0.031	0.031	0.031	0.032	0.031	0.031	0.031	0.031	0.032	0.031
	5	1.007	2.746	2.785	2.637	3.121	3.105	3.299	3.448	3.978	3.534
	6	1.927	3.042	3.869	3.768	3.877	3.292	4.774	5.007	5.008	5.007
	7	1.685	2.816	3.877	3.550	3.604	3.869	4.485	5.008	5.007	5.007
	8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than all others for the same values of d and p.

significantly more time to be identified. This can be seen in Figure 6.5 and Table 6.2 for test instances with $N = 625$.

Method 8 now produces initial feasible colourings with significantly fewer colour classes than Method 0 for all test instance, whereas in Chapter 5 this was not the case for test instances with $d = 0.1$ and higher values of p. As can be seen from Figure 6.5, the number of colour classes in the initial colourings are still positively correlated to p so we would expect Method 8 to produce colourings with significantly more colour classes than Method 0 for some value of $p > 0.05$. This shift is similar to the one observed for Method 4 in Section 4.5.1 and is again likely due to the fact that the colourings produced by DSatur generally have significantly more colour classes than those produced by RLF (see Section 2.1).

For most test instances, there is no significant difference in the number of colour classes in the initial feasible colourings achieved when implementing Methods 6 and 7, nor is there any significant difference in the time required to do so. On the rare occasions when significant differences are observed, Method 6 achieves initial feasible colourings with significantly fewer colour classes than Method 7, but Method 7 requires significantly less time to identify an initial feasible colouring. There does not appear to be any obvious pattern with regards to when these differences are observed, however.

Information regarding the number of clashes in the complete, improper colouring produced by Methods 6 and 7 was unavailable in Chapter 5. Here we were able to show that Method 7 produces colourings with significantly fewer clashes than Method 6 for all test instances. Examples of this clear relationship can be seen in

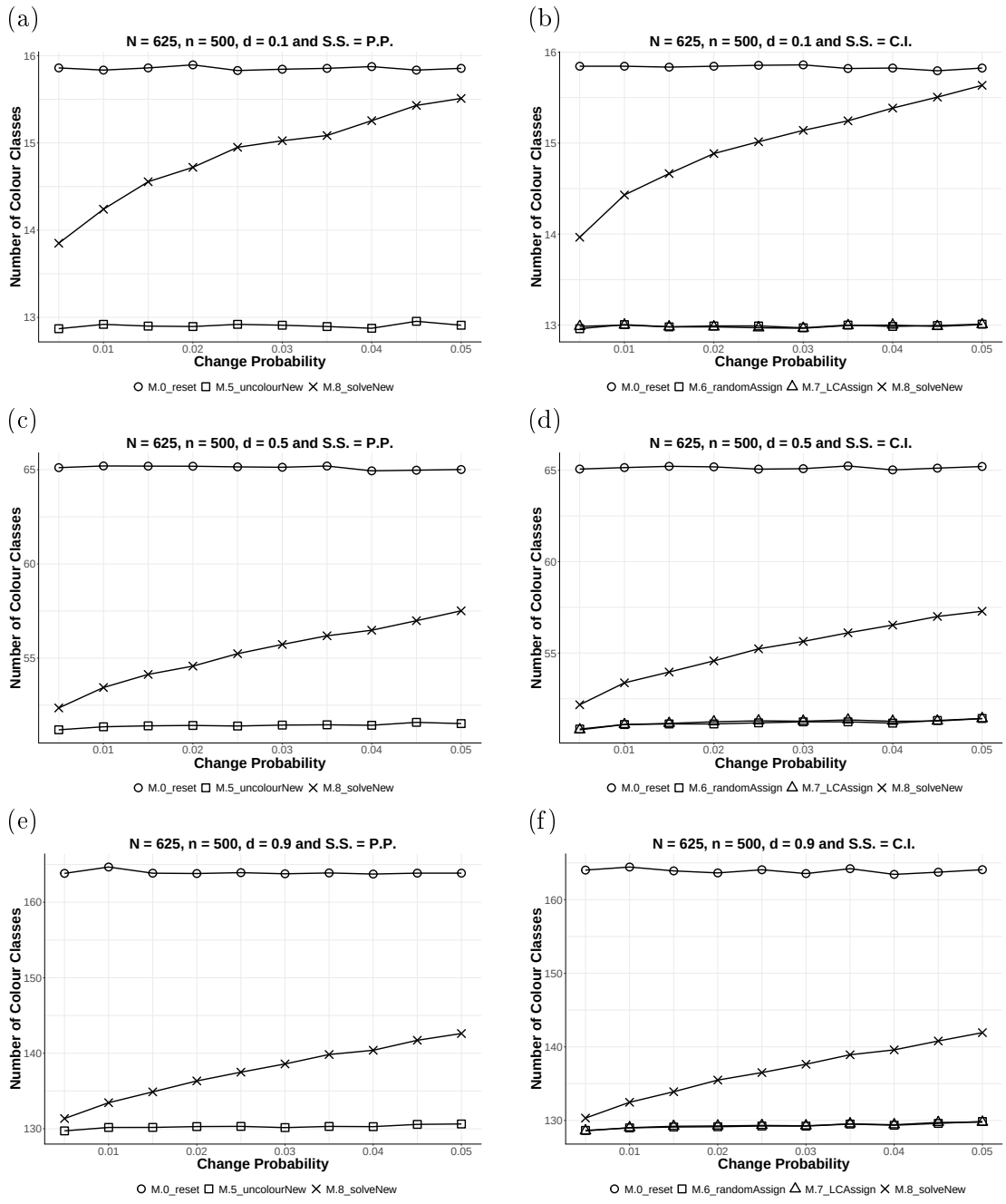

Figure 6.5: Mean number of colour classes in initial, feasible colourings for test instances with $N = 625$. Graphs (a), (c) and (e) represent methods that operate in the partial, proper solution space, graphs (b), (d) and (f) represent methods that operate in the complete, improper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

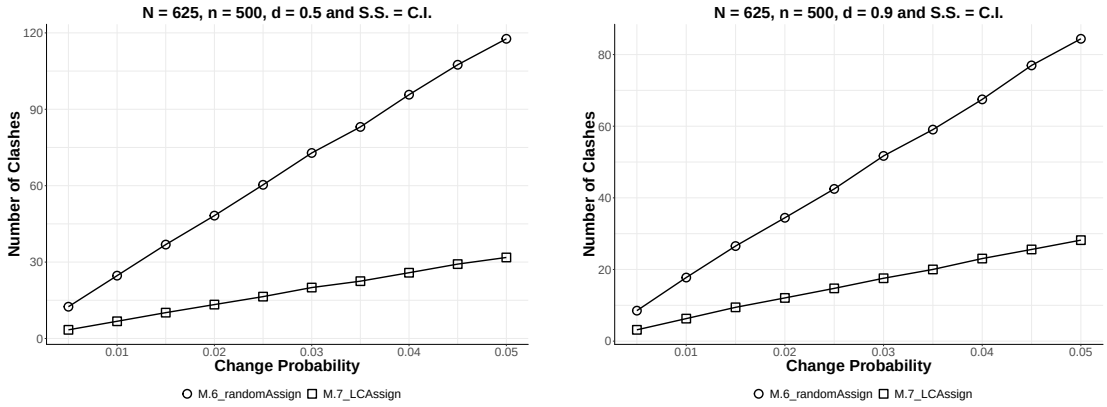

Figure 6.6: Mean number of clashes in colourings produced by Methods 6 and 7 for test instances with $N = 625$ and $d = \{0.5, 0.9\}$.

Figure 6.6. Although the relevant information is not available for comparison, it is believed that this result also holds true for the experiments in Chapter 5 on test instances with $n = 500$ because the modification operators are unchanged.

As expected, all of the relationships presented above are unaffected by the size of the vertex pool N. With regards to the number of colour classes in the initial feasible colourings achieved, this is clearly illustrated in Figure 6.7 for methods operating in the partial, proper solution space on test instances with $d = 0.5$. With regards to computational effort, no significant correlation at the $\alpha = 0.01$ level can be observed between N and the time required to identify an initial feasible colouring. As such, Methods 5 to 7 require significantly more time than Method 0 to identify an initial feasible colouring, and Method 8 requires significantly less time, all regardless of the value of N.

Final Colourings

In general, the relationships observed between the different modification operators are again similar to those observed in Chapter 5 for test instances with $n = 500$. To summarise:

- Final feasible colourings with significantly fewer colour classes are achieved under Method 5 compared to those achieved under Methods 0 and 8 for test instances with high values of d and/or low values of p.
- Final feasible colourings with both significantly fewer and significantly more colour classes are achieved under Methods 6 to 8 compared to those achieved under Method 0. However, there is no obvious relationship between the results observed and the test instance parameters. This relationship also holds between the final feasible colourings achieved under Methods 6 and 7 when compared against those achieved under Method 8.

When final feasible colourings with the same number of colour classes are achieved:

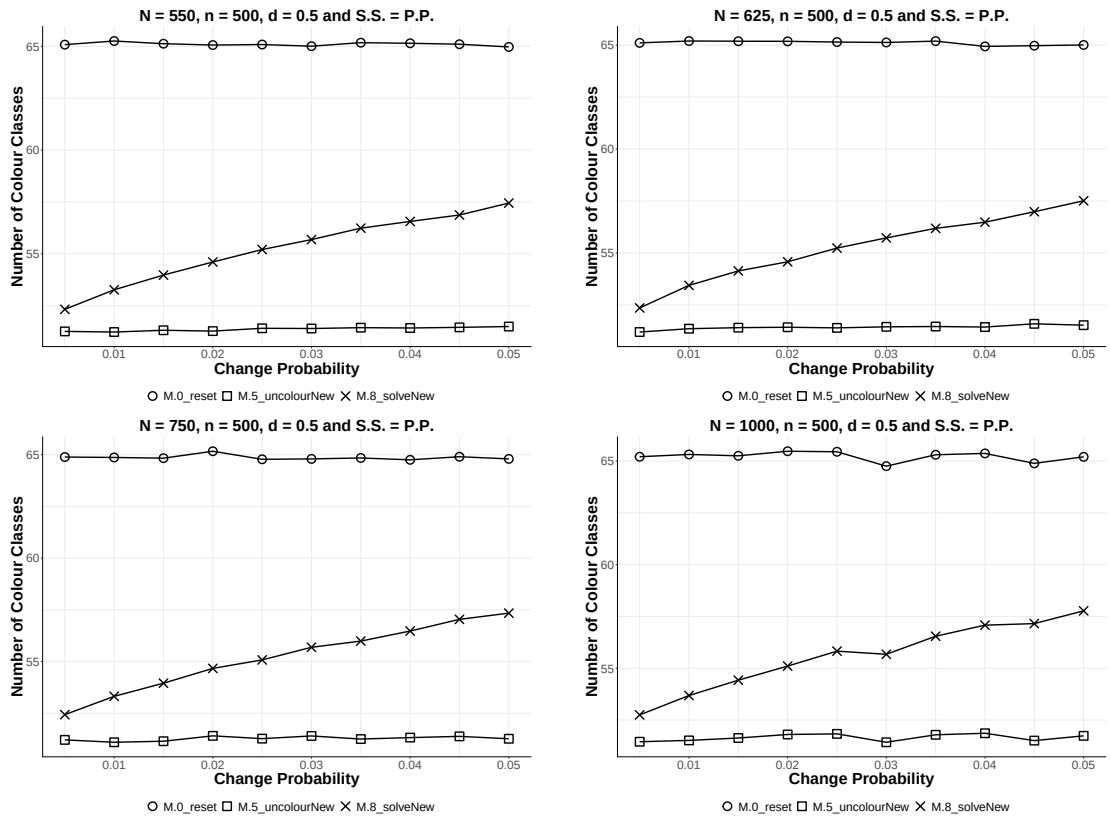

Figure 6.7: Mean number of colour classes in initial, feasible colourings achieved in the partial, proper solution space (in clockwise order, starting from the top-left, graphs represent test instances with $d = 0.5$ and $N = 550, 625, 750$ and 1000, respectively).

- Methods 5 to 7 require significantly less time than Method 0 for test instances with low values of d and/or low values of p.
- Method 8 requires significantly less time than Method 0 for most test instances, especially those with lower of values p.
- Method 8 also requires significantly less time than Methods 5 to 7 for most test instances with $d \in \{0.5, 0.9\}$, especially those with higher values of p.

However, unlike when considering initial feasible colourings, here the size of the vertex pool N *does* appear to have an affect on the results observed, at least in some cases. Therefore the observations above are to be considered with some potential variation based on N. For example, Method 5 achieves final feasible colourings with significantly more colour classes than those achieved under Methods 0 and 8 on some test instances with $d \in \{0.1, 0.5\}$ when $N \in \{550, 625\}$ but not when $N \in \{750, 1000\}$. Similarly, Methods 6 and 7 require significantly less time to achieve final feasible colourings[5] compared to those achieved under Method 8 on some test instances with $d = 0.1$ when $N = 550$ but not when $N \in \{625, 750, 1000\}$.

The reason that the size of the vertex pool N affects some modification operators more than others, and produces certain effects in one solution space and not the other, is unclear. Perhaps the variation in observations between modification operators and solution spaces indicates that the underlying reason for the observations is not strictly caused by N. These observations are made more perplexing because N was shown to have no affect on the results concerning initial feasible colouring.

6.4.2 With Secondary Optimisation

We now consider the situation where the future inclusion vector P_{t+1} is known for each time-step $t \in \{0, 1, \ldots, T-1\}$. We have shown in Section 6.2 that the tabu search procedure introduced in Section 4.3 can also be used to reduce total expected future saturation (i. e. Equation (6.2)). Here we wish to investigate whether using this tabu search procedure within our two-stage approach (Algorithm 4.1) can produce robust colourings for the vertex dynamic GCP with future inclusion information.

In particular we wish to explore how our two-stage approach affects the number of colour classes in the initial feasible colourings and the computational effort required to identify them. The number of colour classes in the final feasible colourings will be naturally capped by the target number of colour classes $k^\star$, so they will not be explored here.

It might be expected that the value of N will have some effect when including secondary optimisation. Larger values of N indicate more non-active vertices during

[5] As usual, the results used for comparison regard final feasible colourings with the same number of colour classes achieved under the respective modification operators.

each time-step t, which could lead to more time required to calculate the total expected future saturation $\mathcal{F}(\mathcal{S}_t)$. Fewer iterations of the secondary optimisation may then take place because more time will be required, during each iteration, to update the data structures related to "cost" changes. However, our experiments indicate that N has very little effect on the relationships observed between the modification operators[6]. To reduce the volume of results presented, unless stated otherwise, we will only present results for test instances with $N = 625$ here.

Initial Total Saturation

As stated above, we have shown that the total expected future saturation of a colouring $\mathcal{S}_t$ can be reduced by the tabu search procedure outlined in Section 4.3 by replacing Equation (4.3) with Equation (6.2). To measure the effectiveness of this reduction we look at the total saturation of the newly active vertices at time-step $t + 1$, which is given by

$$\sum_{v \in V_{t+1}^{+}} \deg_{sat}(v, \mathcal{S}_t \backslash V_{t+1}^{-}) \tag{6.6}$$

where $\deg_{sat}(v, \mathcal{S}_t \backslash V_{t+1}^{-})$ is the saturation degree of v with respect to the colouring $\mathcal{S}_t \backslash V_{t+1}^{-} = \{S_1 \backslash V_{t+1}^{-}, \ldots, S_k \backslash V_{t+1}^{-}\}$.

With the exception of the smallest values of $k^\star$, Equation (6.6) was successfully reduced when the second stage of our approach is implemented (i. e. Lines 11-16 of Algorithm 4.1). More specifically, in most cases where $k^\star$ is set above or equal to 13, 52 and 131 for test instances with $d = 0.1, 0.5$ and 0.9 respectfully, the total saturation of the new vertices at the beginning of the time-step is significantly smaller compared to when secondary optimisation is not implemented. As $k^\star$ increases so too does the number of feasible $k^\star$-colourings, therefore increasing the chance of our tabu search procedure to identify a colouring with reduced total expected future saturation.

It is interesting to observe that if $k^\star$ tends towards higher values then our tabu search procedure naturally empties colour classes in its attempt to reduce total expected future saturation, which leads to a $k^\star$-colouring with empty colour classes. An empty colour class within a colouring $\mathcal{S}_t$ corresponds to a decreased upper bound for the saturation degree of every non-active vertex with regards to $\mathcal{S}_t$ (i. e. if $\mathcal{S}_t$ has x empty colour classes then $\deg_{sat}(v, \mathcal{S}_t) \leq k^\star - x$ for every $v \in V_{t+1}^{+}$). As can be seen from Figure 6.8, as $k^\star$ increases so too does the number of empty colour classes. This is to be expected because as $k^\star$ increases away from $\chi(G_t)$, the number of feasible k-colourings (where k satisfies $\chi(G_t) \leq k < k^\star$) also increases.

It stands to reason that if the total saturation of the new vertices at the start of a time-step $t+1$ is reduced then so too might the number of clashes in the colourings

[6]This does not contradict the results presented in Section 6.4.1, because any potential effects caused by the size of the vertex pool are limited to the results concerning final feasible colourings.

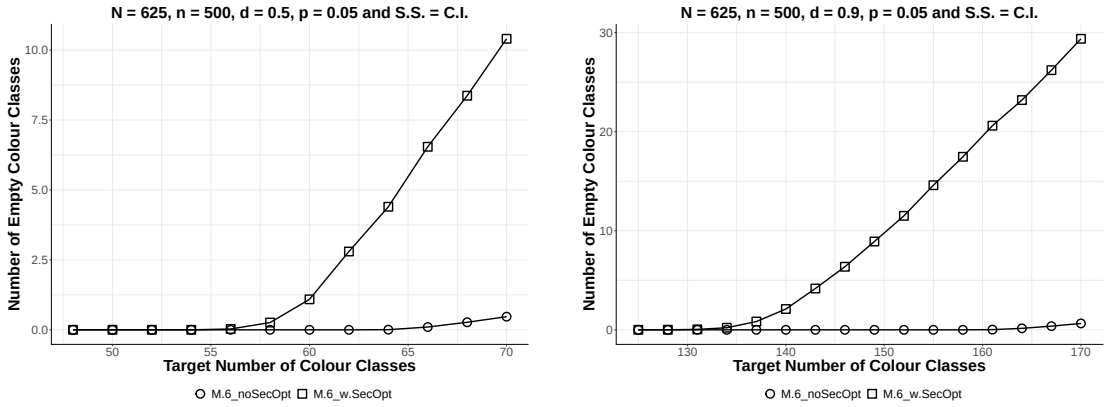

Figure 6.8: Mean number of empty colour classes at the start of a time-step against the target number of colour classes $k^\star$ for test instances with $N = 625, d = \{0.5, 0.9\}$ and $p = 0.05$ whilst implementing modification operator 6 (*randomAssign*).

produced by Methods 6 and 7. For Method 6, the likelihood of a vertex $v \in V^+_{t+1}$ being assigned to a colour class of $\mathcal{S}_t \backslash V^-_{t+1}$ such that no clashes will be introduced is increased if $\deg_{sat}(v, \mathcal{S}_t \backslash V^-_{t+1})$ is decreased. However, with the exception of 15 of the 1400 test instance parameter settings, there was no significant difference between the number of clashes at the start of a time-step. Regardless of decreasing $\deg_{sat}(v, \mathcal{S}_t \backslash V^-_{t+1})$ for a vertex $v \in V^+_{t+1}$, it is still likely that a higher proportion of the colour classes in $\mathcal{S}_t \backslash V^-_{t+1}$ will introduce clashes, if v is assigned to them. Furthermore, if $\mathcal{S}_t \backslash V^-_{t+1}$ contains empty colour classes, then the non-empty colour classes are likely to contain more vertices; therefore, if v is assigned to one of these non-empty colour class, then more clashes are likely to be introduced also.

Indeed, let us assume that $\mathcal{S}_t \backslash V^-_{t+1}$ contains x empty colour classes and satisfies $|\mathcal{S}_t \backslash V^-_{t+1}| = k$. Then, the $k - x$ non-empty colour classes in $\mathcal{S}_t \backslash V^-_{t+1}$ will contain approximately $\frac{n}{k-x}$ vertices. If $v \in V^+_{t+1}$ is assigned to a colour class of $\mathcal{S}_t \backslash V^-_{t+1}$ at random, then one would expect

$$\left(\frac{x}{k} \cdot 0\right) + \left(\frac{k-x}{k} \cdot \frac{dn}{k-x}\right) = \frac{dn}{k} \tag{6.7}$$

clashes to be introduced, where $\frac{x}{k}$ and $\frac{k-x}{k}$ are the probabilities of v being assigned to an empty colour class or not, respectively. It should be clear from Equation (6.7) that, regardless of the number of empty colour classes in $\mathcal{S}_t \backslash V^-_{t+1}$, the expected number of clashes introduced by Method 6 will remain the same (i. e. $\approx np \cdot \frac{dn}{k}$).

Method 7, on the other hand, specifically assigns a vertex $v \in V^+_{t+1}$ to the colour class of $\mathcal{S}_t \backslash V^-_{t+1}$ such that the least number of clashes are introduced. For this modification operator, the number of clashes in the initial colourings produced are significantly fewer than when secondary optimisation is omitted for most of the cases where total saturation was also significantly reduced (i. e. when $k^\star \geq 13, 52$ and 131 for test instances with $d = 0.1, 0.5$ and 0.9 respectively). This can clearly be seen in

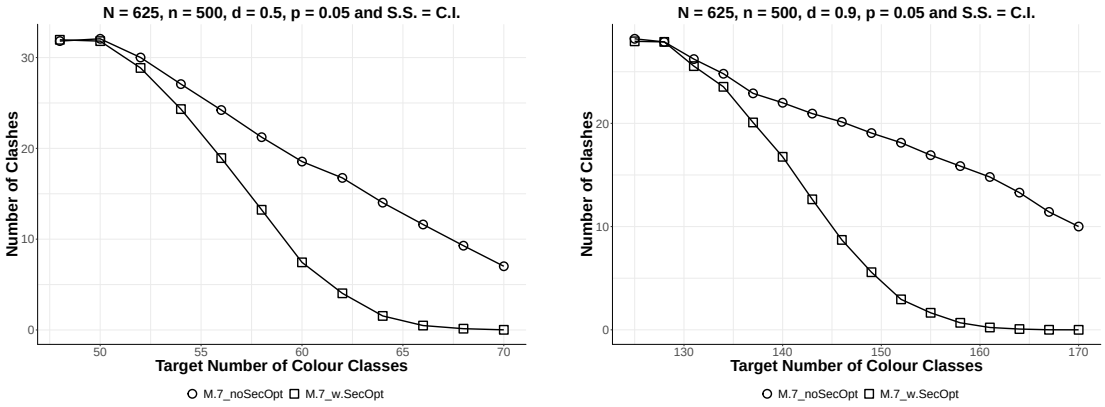

Figure 6.9: Mean number of clashes at the start of a time-step against the target number of colour classes $k^\star$ for test instances with $N = 625, d = \{0.5, 0.9\}$ and $p = 0.05$ whilst implementing modification operator 7 (*leastClashesAssign*).

Figure 6.9. In a lot of these cases, $\mathcal{S}_t \backslash V^-_{t+1}$ is known to contain empty colour classes. If $\mathcal{S}_t \backslash V^-_{t+1}$ has x empty colour classes, then at least x of the $|V^+_{t+1}|$ new vertices will be assigned without introducing clashes. In fact, if $x \geq |V^+_{t+1}|$ then Method 7 is guaranteed to produce a feasible k-colouring for G_{t+1} where $k = |\mathcal{S}_t|$. The values in Table 6.3 show the mean number of clashes in the colouring produced by Method 7, and a value of zero in this table indicates that every colouring produced for the associated test instances are feasible. It can be seen from this table that, as $k^\star$ increases, so too does the value of p for which Method 7 will always produce a feasible colouring.

We therefore conclude that reducing the total expected future saturation does reduce the total saturation of the newly active vertices in V^+_{t+1} with regards to $\mathcal{S}_t \backslash V^-_{t+1}$. In the case of Method 7 we have shown that this reduction of total saturation can lead to colourings with significantly fewer clashes compared to when the secondary optimisation is not implemented. In the best case scenarios (i. e. those with low value of p and high values of $k^\star$), Method 7 is able to assign the new vertices in V^+_{t+1} to colour classes in $\mathcal{S}_t \backslash V^-_{t+1}$ such that the resultant colouring is feasible for G_{t+1} in many more cases than when the secondary optimisation is omitted.

Initial, Feasible Colourings

Now that we know that total saturation of the newly active vertices is successfully reduced by our two-stage approach, we wish to explore whether this has any impact on the initial feasible colourings achieved after Methods 5 to 8 have been implemented. We have already shown that Method 7 can now produce feasible colourings more often than before, but how else are the initial feasible colourings effected? Is there a reduction in the number of colour classes in the initial feasible colourings? Are there any computational savings? Regardless of whether the initial colouring produced by Methods 5 to 8 are feasible, a reduced total saturation should mean that

Table 6.3: Mean number of clashes in the colourings produced by modification operator 7 (*leastClashesAssign*) for test instances with $N = 625$.

		p									
d	$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
0.1	12	2.575	5.220	8.010	10.465	12.395	15.075	17.445	19.305	22.320	24.720
	13	2.330	4.500	6.915	8.810	11.005	13.415	15.645	17.955	19.885	22.365
	14	1.280	2.885	4.270	5.200	6.370	7.900	9.185	10.525	11.765	12.700
	15	0.290	0.690	0.925	1.485	1.560	1.955	2.765	3.305	3.565	4.705
	16	0.010	0.060	0.140	0.225	0.385	0.515	0.620	0.825	1.180	1.225
	17	0	0	0.015	0.020	0.035	0.075	0.115	0.180	0.250	0.340
	18	0	0	0	0	0	0.015	0.035	0.040	0.055	0.050
0.5	48	3.305	6.925	10.235	13.625	16.655	19.665	22.540	25.805	29.285	31.955
	50	3.345	6.630	9.910	13.335	15.960	19.830	22.500	25.760	28.945	31.810
	52	3.005	6.230	9.215	12.205	14.945	17.975	20.535	23.285	26.485	28.855
	54	2.550	4.800	7.570	9.815	12.450	14.475	17.175	19.455	22.400	24.325
	56	1.725	3.510	5.610	7.345	8.975	11.145	13.055	14.775	17.380	18.930
	58	0.660	1.955	3.310	4.280	6.130	7.655	8.670	9.555	11.710	13.235
	60	0.115	0.565	0.980	1.945	2.505	3.390	4.820	5.320	6.490	7.445
	62	0.055	0.090	0.255	0.545	0.855	1.155	2.350	2.370	3.040	4.035
	64	0	0.005	0.015	0.050	0.210	0.375	0.640	0.815	1.375	1.545
	66	0	0	0	0.005	0.045	0.060	0.125	0.245	0.370	0.485
	68	0	0	0	0	0.005	0	0.020	0.055	0.120	0.140
	70	0	0	0	0	0	0	0	0.010	0.030	0.015
0.9	125	3.205	6.240	9.600	11.910	14.770	17.605	20.035	23.115	25.860	27.945
	128	3.135	6.235	9.215	11.595	14.445	17.570	19.430	22.230	25.285	27.890
	131	2.810	5.445	8.365	10.710	13.325	16.085	18.285	20.750	23.245	25.545
	134	2.170	4.645	7.155	9.685	11.640	14.270	16.600	18.670	21.550	23.540
	137	1.080	3.330	5.470	7.455	9.240	11.625	13.965	15.465	18.195	20.085
	140	0.460	1.815	3.235	4.920	6.775	8.825	10.660	12.680	14.985	16.760
	143	0.100	0.650	1.710	2.900	4.015	5.690	7.815	9.525	11.150	12.640
	146	0.020	0.155	0.585	1.175	1.980	3.320	4.760	5.745	7.860	8.710
	149	0	0.020	0.125	0.310	0.870	1.410	2.495	3.260	4.660	5.580
	152	0	0.005	0.015	0.045	0.260	0.400	1.250	1.600	2.645	2.950
	155	0	0	0	0.010	0.060	0.150	0.370	0.700	1.185	1.650
	158	0	0	0	0	0.005	0.040	0.170	0.180	0.355	0.680
	161	0	0	0	0	0	0.005	0.020	0.040	0.145	0.225
	164	0	0	0	0	0	0	0.010	0.010	0.035	0.075
	167	0	0	0	0	0	0	0	0	0.005	0.005
	170	0	0	0	0	0	0	0	0	0	0

Table 6.4: Significant differences between the time required to achieve an initial feasible colouring under Method 6 with and without secondary optimisation on test instances with $N = 625$ and $d = 0.9$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
125	-	-	-	-	-	-	-	-	-	-
128	-	-	-	-	-	-	-	-	-	-
131	-	-	-	-	-	-	-	-	-	-
134	X	X	X	-	-	-	X	-	-	-
137	X	X	X	X	X	X	X	X	X	X
140	X	X	X	X	X	X	X	X	X	X
143	X	X	X	X	X	X	X	X	X	X
146	-	-	X	X	X	X	X	X	X	X
149	-	-	-	X	X	X	X	X	X	X
152	-	-	-	X	X	X	X	X	X	X
155	-	-	-	-	X	-	X	-	-	X
158	-	-	-	-	X	-	X	-	-	X
161	-	-	-	-	-	-	-	-	-	X
164	-	-	-	-	-	-	-	-	-	-
167	-	-	-	-	-	-	-	-	-	-
170	-	-	X	-	-	-	-	-	-	-

X - indicates that initial feasible colourings are achieved in significantly less time when secondary optimisation is included.

TabuCol and PartialCol are able to identify feasible colourings "more easily".

As with Methods 1 to 3 for the edge dynamic GCP with future adjacency information, the number of colour classes in the initial feasible colourings achieved under Methods 5 to 7 is not affected by the inclusion of secondary optimisation. This is to be expected, because the initial number of colour classes k at time-step $t + 1$ is determined by $\mathcal{S}_t$ (and $k^\star$) rather than the modification operators themselves. However, these modification operators are still able to achieve initial feasible colourings with significantly fewer colour classes than Method 0 for small enough values of $k^\star$ (i. e. $k^\star \leq 15, 56$ and 164 for test instances with $d = 0.1, 0.5$ and 0.9 respectively).

Under Methods 5 and 6, initial feasible colouring are achieved in significantly less time when secondary optimisation is implemented in some test instances. In general, the pattern governing the cases where a time reduction occurs is similar to the pattern illustrated in Table 6.4 (i. e. a band of instances through the mid-range values of $k^\star$, followed by a "drop-off" as $k^\star$ continues to increase). The observations where a time reduction is observed all relate to test instances for which the total saturation is also reduced. The "drop-off" in time reductions for Methods 5 and 6 is likely due to TabuCol and PartialCol's abilities, in their own right, to identify feasible $k^\star$-colourings in very short periods of time for large value of $k^\star$.

Method 7 also identifies initial feasible colourings in significantly less time when secondary optimisation is applied, but without the "drop-off" for higher values of $k^\star$. As with Methods 5 and 6, all cases where a time reduction is observed relate to test instances for which the total saturation was also reduced. The "drop-off" is

Table 6.5: Significant differences between the time required to achieve an initial, feasible colouring when using modification operator 5 (*uncolourNew*) within our two stage approach (Algorithm 4.1) compared against Method 0 on test instances with $N = 625$ and $d = 0.9$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
125	O	O	O	O	O	O	O	O	O	O
128	O	O	O	O	O	O	O	O	O	O
131	O	O	O	O	O	O	O	O	O	O
134	-	O	O	O	O	O	O	O	O	O
137	X	X	X	X	-	-	-	-	O	O
140	X	X	X	X	X	X	X	X	X	X
143	X	X	X	X	X	X	X	X	X	X
146	X	X	X	X	X	X	X	X	X	X
149	X	X	X	X	X	X	X	X	X	X
152	X	X	X	X	X	X	X	X	X	X
155	X	X	X	X	X	X	X	X	X	X
158	X	X	X	X	X	X	X	X	X	X
161	X	X	X	X	X	X	X	X	X	X
164	X	X	X	X	X	X	X	X	X	X
167	X	X	X	X	X	X	X	X	X	X
170	X	X	X	X	X	X	X	X	X	X

X - indicates that our two stage approach with Method 5 requires significantly less time than Method 0.
O - indicates that our two stage approach with Method 5 requires significantly more time than Method 0.

probably absent here, because Method 7 is able to produce initial feasible colourings for high values of $k^\star$ and low values of p, therefore bypassing TabuCol altogether in these cases.

These time reductions lead to cases where Methods 5 to 7 are able to achieve initial feasible colourings in significantly less time than those achieved under Method 0 also. As was the case in Section 4.5.2 for the edge dynamic GCP with future adjacency, these cases are dependent on $k^\star$ and p. Here, it appears that $k^\star$ may be more influential than p, as a slight increase in $k^\star$ leads to a much larger increase in the values of p for which Methods 5 to 7 are able to outperform Method 0, with regards to computational effort. By comparing Tables 4.11 and 6.5, we can also see that the value of $k^\star$ does not have to be as large for these achievements to be realised in the vertex dynamic case. This is probably caused by the amount of "problematic" change introduced for a vertex dynamic GCP with a change probability p, which is likely to be less than that introduced for an edge dynamic GCP with the same change probability p (see Chapter 5).

We can therefore conclude that, for mid-range values of $k^\star$, using Methods 5 to 7 to modify a feasible colouring for G_t into an initial colouring for G_{t+1} is more beneficial than using Method 0, with regards to both quality (i.e. fewer colour classes) and computational effort (i.e. less time required).

Let us now consider our final modification operator. Method 8 is able to pro-

duce initial feasible colourings with significantly fewer colour classes when secondary optimisation has been implemented for large values of $k^\star$. In fact, for $k^\star \geq 14, 54$ and 134, Method 8 produces initial feasible colourings with significantly fewer colour classes for all test instances with $d = 0.1, 0.5$ and 0.9 respectively. This is yet another knock-on effect of our secondary optimisation stage, which successfully reduced the total saturation in all of the cases mentioned above.

Unfortunately, our two-stage approach results in Method 8 producing colouring with significantly more colour classes than those produced by Method 0 for mid-range values of $k^\star$ on test instances with $d = 0.1$ and high values of p (see Figure 6.10 and Table 6.6). In Figure 6.10, this is particularly illustrated where the lines for Method 0 (i. e. the line of circles) are met and then crossed for by the lines for Method 8 (i. e. the line of crosses) when $k^\star = \{14, 16\}$. It should also be noted that when $k^\star = 18$, the lines for Methods 0 and 8 lie on top of one another, illustrating that there is no significant difference between the number of colour classes in their respective initial feasible colourings.

Given a feasible colouring $\mathcal{S}_t$ for G_t, Method 8 will always produce a feasible k-colouring for G_{t+1} where $k \geq |\mathcal{S}_t| \approx k^\star$. As $k^\star$ increases, the number of additional colour classes added to $\mathcal{S}_t \backslash V_{t+1}^-$ by Method 8 is likely to cause the total number of colour classes to overtake the number of colour classes used by Method 0. This is clearly evident from the "noSecOpt" (i. e. no secondary optimisation) lines in the graphs of Figure 6.10. However, as $k^\star$ continues to increase, our secondary optimisation identifies colourings with empty classes, which can be utilised by Method 8 before it adds any new colour classes to $\mathcal{S}_t \backslash V_{t+1}^-$. In fact, Method 8 should produce initial feasible k-colourings for G_{t+1}, where $k = |\mathcal{S}_t| \approx k^\star$, for most of the same test instances that Method 7 is able to do so (see Table 6.3). Hence, for the largest values of $k^\star$, there is no significant difference in the number of colour classes of the initial feasible colourings produced by Methods 0 and 8. This holds for all test instances, including those with $d \in \{0.5, 0.9\}$.

With regards to time requirements, the inclusion of secondary optimisation has no effect on the time required by Method 8 to achieve initial feasible colourings (i. e. there is no significant difference). As such, Method 8 continues to be significantly faster than Method 0 with regards to achieving an initial feasible colouring, as was the case in Section 6.4.1.

Method 6 (*randomAssign*) vs. Method 7 (*leastClashesAssign*)

The main observable difference between Methods 6 and 7 so far, has been with regards to the number of clashes in the colourings they produce (see Figure 6.6). When considering the vertex dynamic GCP without future inclusion information, this difference has had little-to-no knock-on effect with regards to the quality of the initial feasible colourings achieved, nor the computational effort required to do so

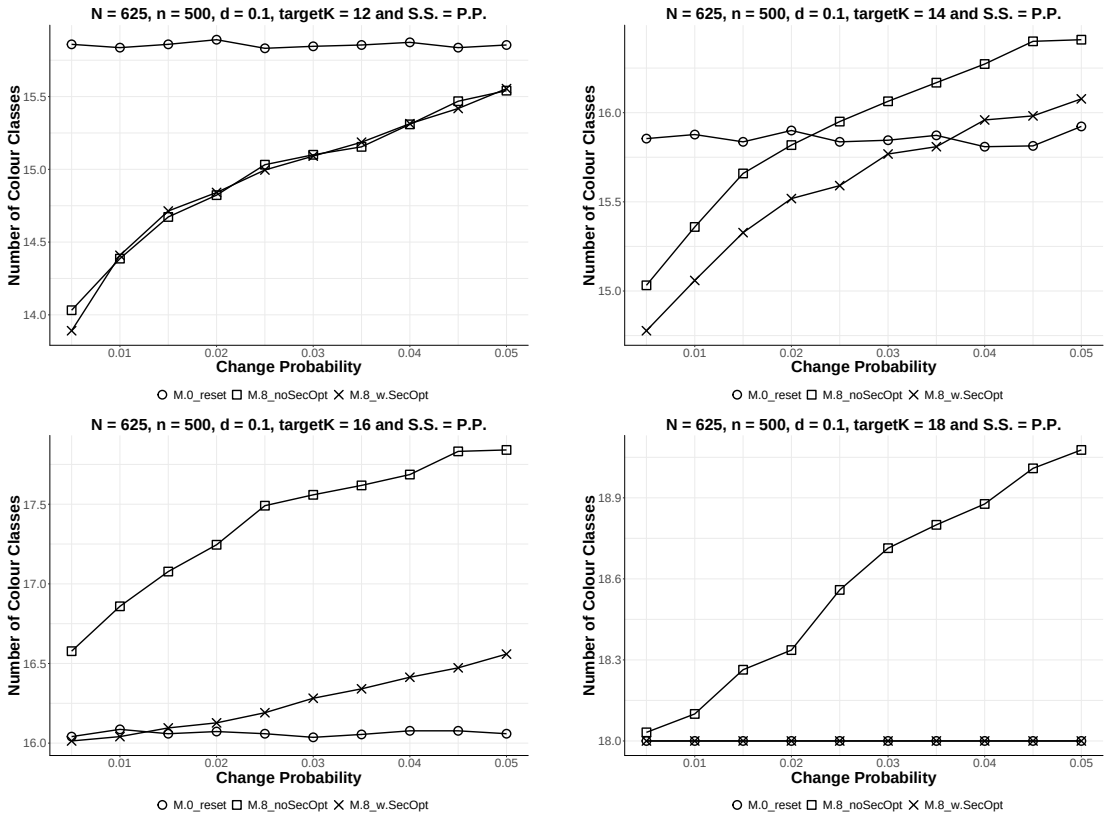

Figure 6.10: Mean number of colour classes in the initial feasible colouring produced by Methods 0 and 8 with and without secondary optimisation whilst operating in the partial, proper solutions space (in clockwise order, starting from the top-left, graphs represent trials conducted with $k^\star = 12, 14, 16$ and 18, respectively, on test instances with $N = 625$ and $d = 0.1$).

Table 6.6: Significant differences between the number of colour classes in the initial feasible colourings when using modification operator 8 (*solveNew*) within our approach (Algorithm 4.1) compared against Method 0 on test instances with $N = 625$ and $d = 0.1$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
12	X	X	X	X	X	X	X	X	X	X
13	X	X	X	X	X	X	X	X	X	X
14	X	X	X	X	X	-	-	O	O	O
15	X	X	X	X	-	O	O	O	O	O
16	-	-	-	-	O	O	O	O	O	O
17	-	-	-	-	-	-	-	-	O	O
18	-	-	-	-	-	-	-	-	-	-

X - indicates that Method 8 achieves an initial colouring with significantly fewer colour classes than Method 0.
O - indicates that Method 8 achieves an initial colouring with significantly more colour classes than Method 0.

Table 6.7: Significant differences between the time required to achieve an initial, feasible colouring when using Methods 6 and 7 within our two stage approach on test instances with $N = 625$ and $d = 0.9$.

	p									
$k^\star$	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
125	-	-	-	-	-	-	-	-	-	-
128	-	-	-	-	-	-	-	-	-	-
131	-	-	-	-	-	-	-	-	-	-
134	-	-	-	-	-	-	-	-	-	-
137	-	-	-	-	-	-	-	O	-	-
140	-	-	-	-	-	-	-	-	-	-
143	X	-	-	-	-	-	-	-	-	-
146	X	X	X	-	X	-	-	-	-	-
149	X	X	X	X	X	-	-	-	X	-
152	X	X	X	X	X	X	X	-	-	-
155	X	X	X	X	X	X	X	X	X	-
158	X	X	X	X	X	X	X	X	X	X
161	X	X	X	X	X	X	X	X	X	X
164	X	X	X	X	X	X	X	X	X	X
167	X	X	X	X	X	X	X	X	X	X
170	X	X	X	X	X	X	X	X	X	X

X - indicates that Method 7 requires significantly less time than Method 6.
O - indicates that Method 7 requires significantly more time than Method 6.

(see Sections 5.4.1 and 6.4.1).

When implementing our two stage approach, there is still no significant difference between the number of colour classes in the initial feasible colourings achieved under Methods 6 and 7. However, we can now observe cases where Method 7 requires significantly less time to identify an initial feasible colouring compared to Method 6. Indeed, for high values of $k^\star$ and low values of p, initial feasible colourings are identified in significantly less time under Method 7 than under Method 6. As $k^\star$ increases, so too does the value of p for which this computational saving is observed (see Table 6.7).

This observation is the result of a series of knock-on effects. Firstly, our two stage approach significantly reduces the total saturation. This reduced saturation then leads to significantly fewer clashes in the colouring produced by Method 7 (see Figure 6.9). Finally, this reduction in clashes significantly reduces the time required to identify an initial feasible colouring under Method 7. It should also be noted that, our two stage approach also significantly reduces the time required to identify an initial feasible colouring under Method 6, but only for mid-range values of $k^\star$ (see Table 6.5).

6.5 Conclusions

In this chapter we proposed an alternative version of vertex dynamic graphs, based on a fixed pool of available vertices and a fixed set of globally active edges between

the vertices in that pool. Using this new definition, we introduced the concept of future vertex inclusion, and proposed using the two-stage approach with the tabu search procedure previously introduced in Sections 4.2 and 4.3, respectively, to produce more robust colourings with respect to future change.

As was shown in Chapter 4 for the edge dynamic GCP, changing the constructive operator does not change the relationships observed in Chapter 5 between the modification operators for the vertex dynamic GCP, other than shifting the value of p for which the observations hold in some cases. It was also shown that the size of the vertex pool N has no effect on the relationships observed with regards to the initial, feasible colourings achieved. On the other hand, N does appear to have some effect with regards to final feasible colourings. The observable influence of N varies across different modification operators and solution spaces, so it is difficult to determine why it is present. Nevertheless, this is of little concern when considering our two-stage approach because we will only be interested in results regarding the initial feasible colourings achieved.

By using future vertex inclusion to reduce the total expected future saturation, within our two-stage approach, the total saturation of the non-active vertices was significantly reduced for the larger values of $k^\star$ used. This then lead to the number of clashes observed in the colourings produced by Method 7 to be significantly reduced in the same cases also. For large values of $k^\star$ and small values of p, Method 7 was even able to produce feasible colourings (i. e. colourings without clashes).

The relationships observed, with regards to the number of colour classes in the initial feasible colourings achieved, between Methods 5 to 8 (*uncolourNew, randomAssign, leastClashesAssign* and *solveNew*, respectively) and Method 0 (*reset*) without future inclusion information (i. e. those presented in Chapter 5 and Section 6.4.1) remain the same when using our two-stage approach with low values of $k^\star$. However, the strength of these relationships diminish as $k^\star$ increases. For example, Methods 5 to 7 cease to achieve initial feasible colourings with significantly fewer colour classes than Method 0 for the largest values of $k^\star$.

On the other hand, there is clear inverse relationship between the target number of colour classes $k^\star$ and the time required to achieve initial feasible colourings (i. e. as $k^\star$ increases, the time required decreases). For mid-range values of $k^\star$ Methods 5 to 7 are now able to achieve initial feasible colouring with significantly fewer colour classes than those achieved under Method 0, whilst also requiring significantly less time to do so.

As observed in Chapter 4, the relationship between the quality (i. e. number of colour classes, determined by $k^\star$) and "robustness" of a colouring is clearly negative (i. e. as $k^\star$ decreases, "robustness" decreases). As described in Section 4.6, this is unsurprising because, for a given graph G_t, our secondary optimisation stage will allow more feasible colourings to be considered as $k^\star$ increases away from $\chi(G_t)$.

We saw in Chapter 5 that, in general, the vertex dynamic GCP achieves more beneficial results than its edge dynamic counterpart (i.e. where the test instance parameters were equivalent). This seems to carry over here, such that when a benefit (e.g. reduced time to initial feasible colouring) is observed, it usually occurs for a smaller value of $k^\star$ than would have been required in the corresponding edge dynamic GCP. As was the case in Chapter 5, this is likely due to the amount of "problematic" change introduced for a vertex dynamic GCP with change probability p, which is less severe than the amount introduced for an edge dynamic GCP with the same change probability p.

Key Findings and Recommendations

- Using future vertex inclusion information within our two-stage approach leads to a significant decrease in the total saturation of the non-active vertices in the current time-step (i.e. increased robustness).
- By increasing the robustness of a colouring to future changes, the number of clashes introduced by Method 7 is significantly reduced, and for large values of $k^\star$ and low values of p this modification operator can even produce feasible colourings. We therefore suggest using Method 7 over Methods 5 and 6 when implementing our two-stage approach.
- As with the edge dynamic GCP with future adjacency information, a clear inverse relationship between quality and robustness has been demonstrated. Therefore, we again suggest that the user prioritises between the two objectives based on their preferences.
- In general, the benefits of using the two-stage approach to tackle the vertex dynamic GCP outweigh those observed when tackling the edge dynamic GCP, especially for decreased values of $k^\star$.

This chapter's results are the last to be presented in this thesis. Over Chapters 3 to 6 we have introduced two dynamic variants of the GCP, considered using a modification approach to "repair" colourings when future change information is unavailable, and considered using a two-stage approach to increase robustness when future change information is available. In the final chapter, we will summarise all of the key findings from these past four chapters and propose areas of potential future research.

Chapter 7

Conclusions and Future Work

In this final chapter we aim to summarise all of the findings from the work presented in Chapters 3 to 6, draw final conclusions based on the aims set out at the beginning of this thesis, and discuss areas of potential future work.

7.1 Modification Approach for Dynamic GCPs without Future Change Information

In Chapters 3 and 5 we introduced two cases of dynamic GCPs: edge dynamic and vertex dynamic. In particular, we proposed an approach which attempts to modify a feasible k-colouring for the static representation of a dynamic graph at time-step t into an initial, though not necessarily feasible, k'-colouring for the static representation at time-step $t+1$, where $k' \geq k$.

In these chapters we assumed that changes occur completely at random, i. e. no future change information is available. By doing this, the goal of our proposed approach is to "repair" an infeasible colouring (or solution), that was itself feasible before the changes occurred, in some beneficial way compared to treating each time-step as an individual static graph and starting from scratch. The main benefits we consider are: i) reduced number of colour classes (i. e. improved solution quality); and ii) reduced time requirements for achieving feasible colourings (i. e. reduced computational effort).

For both the edge and vertex dynamic GCPs, four different modification operators (or methods) were introduced and empirically compared. For the edge dynamic GCP, the modification operators are: *calculateClashes*, *uncolourClashes*, *uncolourMostClashing*, and *solveClashes*, denoted by Methods 1 to 4 respectively. For the vertex dynamic GCP, the modification operators are: *uncolourNew*, *randomAssign*, *leastClashesAssign*, and *solveNew*, denoted by Methods 5 to 8 respectively. These modification operators can broadly be split into two different types: Methods 1 to 3 and 5 to 7 modify a feasible k-colouring for G_t into a, most likely, infeasible k-colouring for G_{t+1}; and Methods 4 and 8 modify a feasible k-colouring for G_t into

a feasible k'-colouring for G_{t+1} where, in most cases, $k' > k$.

To compare our modification approach, we used *reset*, denoted by Method 0, which treats each time-step of a dynamic graph as an individual static graph (i. e. Method 0 does not consider any information regarding the colourings achieved in the previous time-step).

In comparison to Method 0, Methods 1 to 3 and 5 to 7 all achieve initial feasible colourings with significantly fewer colour classes, but require significantly more time to do so. Therefore, using these modification operators is beneficial with regards to the quality of initial feasible colourings, but detrimental with regards to computational effort.

To combat this computational increase, variants of Methods 1 to 3 and 5 to 7 were considered that add $x > 0$ empty colour classes to the colourings they produce. Therefore, if a feasible k-colouring for G_t is passed to one of these modification operators, the resultant colouring for G_{t+1} will have $k + x$ colour classes. Provided x is "large enough", Methods 1 to 3 and 5 to 7 now require significantly less time to achieve initial feasible colourings compared to Method 0. Of course, if x is "too large", then these modification operators will achieve initial feasible colourings with significantly more colour classes than Method 0. Thus, dependent on an appropriate choice of x, using this variant of Methods 1 to 3 and 5 to 7 is beneficial with regards to both the quality of initial feasible colourings and the computational effort required to achieve them.

Methods 4 and 8, on the other hand, were shown to achieve initial feasible colourings in significantly less time than Method 0 in most cases[1]. However, the initial feasible colouring produced by these modification operators have both significantly fewer and significantly more colour classes than those produced by Method 0, dependent on the change probability p. Therefore, for smaller values of p these modification methods are beneficial with regards to the quality of initial feasible colourings, as well as being beneficial with regards to computational effort. It should be noted that Method 8 was able to achieve these benefits for higher values of p than Method 4.

With regards to final feasible colourings, Methods 1 to 4 were able to achieve both quality and computational benefits in comparison to Method 0 for a handful of cases with small values of d (density) and p. In the same regard, Methods 5 to 8 were also able to achieve both quality and computational benefits over Method 0, but in far more cases. Interestingly, these benefits were observed more often on test instances with large values of d rather than small ones. As with the difference in performance between Methods 4 and 8, the differences observed here are caused by the differing behaviours of the GCPs themselves.

Consider an edge dynamic test instance, and a vertex dynamic test instance

[1] In the few cases where this does not hold, no significant difference is observed.

with the same parameters as the edge dynamic instance. We have shown, both theoretically and empirically, that a change probability of p is likely to introduce more "problematic" changes (e. g. clashes, "uncoloured" vertices) in an edge dynamic GCP than it will do in a vertex dynamic GCP in most cases[2]. It is therefore not surprising that our modification approach performs better for higher values of p on vertex dynamic GCPs than it does on edge dynamic GCPs.

Overall, our modification approach can be beneficial with regards to both quality and computational effort, especially with regards to initial feasible colourings. Methods 4 and 8 have been shown to perform very well in both senses for small values of p, and Methods 1 to 3 and 5 to 7 have been shown to perform very well (again, in both senses) but regardless of the value of p, provided they included an "appropriate" number of empty colour classes x. With regards to the final feasible colourings achieved when our modification approach is implemented, the results are less impressive, being both beneficial and detrimental depending on the test instance parameters, especially d and p.

In terms of real-world applications, this approach might be of particular interest if a "good" solution is known for the current state of a problem (i. e. its current size and/or constraints) and a small amount of "problematic" change is introduced, which must be dealt with quickly.

7.2 Two Stage Approach for Dynamic GCPs with Probabilistic Future Change Information

In Chapters 4 and 6 the work presented in Chapters 3 and 5 was extended to include future change information. For the edge dynamic GCP this information considered the probability of an non-active edge at time-step t becoming active in time-step $t+1$; for the vertex dynamic GCP this information regarded the probability of a non-active vertex at time-step t becoming active in time-step $t+1$. This future change information is called "future adjacency information" and "future vertex inclusion information" for the edge and vertex dynamic GCPs, respectively.

The main goal of these chapters was to determine whether this future change information could be utilised in the current time-step in order to produce a colouring that is currently feasible but also more robust to the changes that may be introduced. To ascertain whether this has occurred, we looked at whether the benefits previously observed in Chapters 3 and 5 (i. e. improved solution quality and reduced computational effort) were themselves improved upon, or whether they became observable in cases where they had not been before.

In order to achieve the above goal, a two stage approach was proposed. The

[2]This statement is also, to a lesser extent, dependent on the number of vertices n and the density d of a test instance.

first stage of this approach aims to identify a feasible $k^\star$-colouring for the current time-step, where $k^\star$ is user-defined, by using the modification approach from Chapters 3 and 5. The second stage of this approach attempts to optimise a secondary "robustness objective" via a tabu search procedure which utilises move operators that do not compromise the feasibility of a colouring. Therefore, if a feasible $k^\star$-colouring for the current time-step is successfully identified in the first stage, the final colouring returned after the second stage is also guaranteed to be feasible for the current time-step.

7.2.1 With Future Adjacency Information

In Chapter 4, we introduced future adjacency information for the edge dynamic GCP, which indicates the probability of an edge $\{u, v\} \notin E_t$ being included in E_{t+1}. This information can be used to reduce the expected number of clashes in the subsequent time-step.

By reducing the expected number of clashes, our two-stage approach was able to significantly reduce the number of actual clashes that are introduced between time-steps, dependent on the values of $k^\star$ and p. For relatively small values of $k^\star$, significant reductions in the number of clashes were only observed for the highest values of p used. As $k^\star$ increases however, the values of p decrease. Hence, for the largest values of $k^\star$ used, significant reductions were observable for all values of p used. The amount of reduction was also positively related to the value of $k^\star$ (i. e. as $k^\star$ increases, the reduction in clashes also increases).

This reduction in the total number of clashes at the start of a time-step suggests that the initial colourings are "closer" to feasible. Indeed, depending on $k^\star$ and p, initial feasible colourings can be achieved via tabu search in significantly less time compared to using Method 0. The relationship between $k^\star$ and p here is reversed from that discussed above: for relatively small values of $k^\star$, this computational benefit can only be observed for the smallest values of p used, but as $k^\star$ increases, so too does the values of p.

One drawback of our two-stage approach is with regard to the number of colour classes in the initial feasible colourings achieved. For the largest values of $k^\star$, Methods 1 to 3 are unable to achieve initial feasible colourings with significantly fewer colour classes than Method 0. Also, as $k^\star$ increases, Method 4 produces initial feasible colourings with significantly more colour classes than Method 0 for smaller values of p. It should be noted that these observations are caused by the capping of colour classes (i. e. by $k^\star$), rather than our two-stage approach.

Overall, our two-stage approach has illustrated a clear relationship between the quality of a colouring (i. e. the number of colour classes) and the "robustness" of a colouring to future change. For mid-range values of $k^\star$, Methods 1 to 3 are able to achieve initial feasible colourings with significantly fewer colour classes than those

produced by Method 0, and they require significantly less time to do so. Therefore, our two-stage approach is able to improve upon both quality and computational effort for "appropriate" values of $k^\star$.

It is interesting to note that this is very similar to the benefits achievable without future adjacency information in Chapter 3, when implementing Methods 1 to 3 with $x > 0$ empty colour classes. For "appropriate" values of x, benefits could be achieved with regards to both quality and computational effort, but without the need of any secondary optimisation.

7.2.2 With Future Vertex Inclusion Information

In Chapter 6, we introduced future vertex inclusion information for the vertex dynamic GCP, which indicates the probability of a vertex $v \in \mathcal{V} \backslash V_t$ being included in V_{t+1}, where $\mathcal{V}$ is a fixed pool of vertices. In order to utilise this information, we introduced a fixed set of global edges (i. e. if $u, v \in V_t$ then it is known whether $\{u, v\} \in E_t$, for all time-steps t). Using future vertex inclusion information and the set of global edges, we can then attempt to reduce the total expected saturation of the non-active vertices in $\mathcal{V} \backslash V_t$ with respect to the colouring for G_t.

By reducing the total expected saturation of the non-active vertices at time-step t, our two-stage approach was able to significantly reduce the total saturation of the new vertices in V_{t+1}^+, dependent on $k^\star$. This was mostly achieved by emptying colour classes of $\mathcal{S}_t$. There is also a positive relationship between the amount of reduction and the value of $k^\star$ (i. e. as $k^\star$ increases, the reduction in the total saturation also increases). In general, these observations are similar to those observed for the edge dynamic GCP with future adjacency information but without the dependence on p.

A reduction in the total saturation of the new vertices would suggest that it will be "easier" to identify a feasible colouring. This appears to hold true, because for large values of $k^\star$ and low value of p, Method 7 is now able to produce initial feasible colourings independently. In fact, as $k^\star$ increases, so too do the values of p for which this is achievable. In general, the reduction of the total saturation of the new vertices also leads to computational savings when implementing Methods 5 to 7, for mid-range values of $k^\star$. For the smallest and the largest values of $k^\star$, no significant time changes are observed.

As with the edge dynamic GCP with future adjacency information, we see the same drawback with regards to the number of colour classes in the initial feasible colourings achieved. Methods 5 to 7 are unable to achieve initial feasible colourings with significantly fewer colour classes than Method 0 for the largest values of $k^\star$, and Method 8 produces initial feasible colourings with significantly more colour classes than Method 0 for the mid-range values of $k^\star$ and high values of p on low density graphs. Again, this is caused by the capping of colour classes, rather than our two-stage approach directly.

Again, our two-stage approach clearly demonstrates the relationship between the quality of a colouring and its "robustness". For mid-range values of $k^\star$, Methods 5 to 7 are able to achieve initial feasible colourings with significantly fewer colour classes than Method 0, and require less time to do so. It should be noted that, the range of $k^\star$ values for which this holds is larger than the range for which similar results are achieved by Methods 1 to 3 for the edge dynamic problem. The change probability is also much less important here, indicating that the "best" quality and computational benefits achievable are dependent on "appropriate" values of $k^\star$ alone. Just like the cases where future change information is not used, we believe this difference between the edge and vertex dynamic problems is caused by the amount of "problematic" change that is introduced between time-steps (i. e. the amount of "problematic" change for a vertex dynamic graph with change probability p is likely to be less than that of an edge dynamic graph with the same change probability p).

7.3 Future Work

In this final section we will discuss ideas for future research that would extend the work presented in this thesis.

7.3.1 Different Dynamic Graphs

In this thesis we have only considered a few different types of test instances. More specifically, we have focused on random graphs that are constructed such that the density and the number of vertices remains approximately equal over all time-steps. One simple way of extending this work would be to explore how our modification approach and two-stage approach behave on different graphical topologies. In this section we will briefly discuss different test instances that might be of interest.

Firstly, we could look at random dynamic graphs where the density or number of vertices (i. e. $|E_t|$ and $|V_t|$, respectively) do not remain approximately equal over all time-steps. On-line graph colouring considers dynamic graphs where the vertex set is ever increasing, i. e. $|V_{t+1}| = |V_t| + 1$ for all time-steps $t = 0, 1, 2, \ldots, T$. One potential problem of these types of dynamic graphs is that $\chi(G_t)$ is also likely to vary over all time-steps t. Therefore, Methods 1 to 3 and 5 to 7, which modify a feasible k-colouring $\mathcal{S}_t$ for G_t into a k-colouring for G_{t+1}, may become less appropriate because k may be smaller than $\chi(G_{t+1})$.

Graphs with different structure may also be considered, such as scale-free networks. Heuristics might even be designed in order to take advantage of a graph's underlying structure, if this is known. Of course, maintaining the particular structure of a graph over all time-steps may be difficult to accomplish in practice.

Throughout this thesis, we have considered edge dynamic and vertex dynamic graphs separately. Within a vertex dynamic graph, both the edge and the vertex

sets change between each time-step. However, changes to the edge set are strictly related to associated changes in the vertex set (i.e. the adding/removal of edges incident to vertices that are added/removed). One may wish to consider a vertex dynamic problem where changes can also occur on the edge set independently of the changes to the vertex set. In order to tackle such a problem, new modification operators would need to be proposed that can deal with both types of changes (e.g. a combination of Methods 4 and 8). We would also expect large amounts of "problematic" change (i.e. clashes and "uncoloured" vertices) to be introduced between each time-step for even the smallest values of p, when compared against the edge and vertex dynamic versions studied in this thesis.

Finally, in order to motivate this work we proposed that many real-world operational research problems, which have been reformulated as static GCPs, would be better represented by dynamic GCPs (e.g. frequency assignment [Aardal et al., 2007; Dupont et al., 2009]). It would be of particular interest to see how our modification and two-stage approaches perform on real-world test instances. In order to acquire such instances, one might consider a partnership with a specific industry, such as defence or telecommunications. Regardless of whether such information currently exists, these industries may have enough insight for dummy instances to be constructed.

By studying such real-world problems, we are unlikely to encounter test instances in which the density and the number of vertices remains approximately equal between changes (i.e. time-steps). For example, consider a frequency assignment problem, its associated GCP and a feasible colouring for that GCP. If the proximity within which interference can occur is increased, then the number of edges and the density of the associated GCP will also increase. Alternatively, additional locations (or objects) could be introduced to the network, which would increase the number of vertices in the associated GCP. In both cases, constraint violations (i.e. clashes or "uncoloured" vertices) are likely to be introduced and a colouring that was previously feasible is unlikely to remain so. An initial frequency assignment problem, along with examples of these two scenarios and their effect on the associated GCP of the initial problem are illustrated in Figure 7.1.

Although not strictly related to graph topologies, we will briefly discuss future change information here. For our test instances, all probabilities related to future change were independent. However, this might not be representative of real-world problems, as a given change could potentially influence additional or future changes. For example, in a frequency assignment problem, if one location (or object) is moved within the network then other locations in its near vicinity might be more (or less) likely to be moved also, either in the same time-step or in future time-steps. Therefore, the future adjacency probabilities of the edges incident to the associated vertices in the associated edge dynamic GCP would not be independent of one another, in

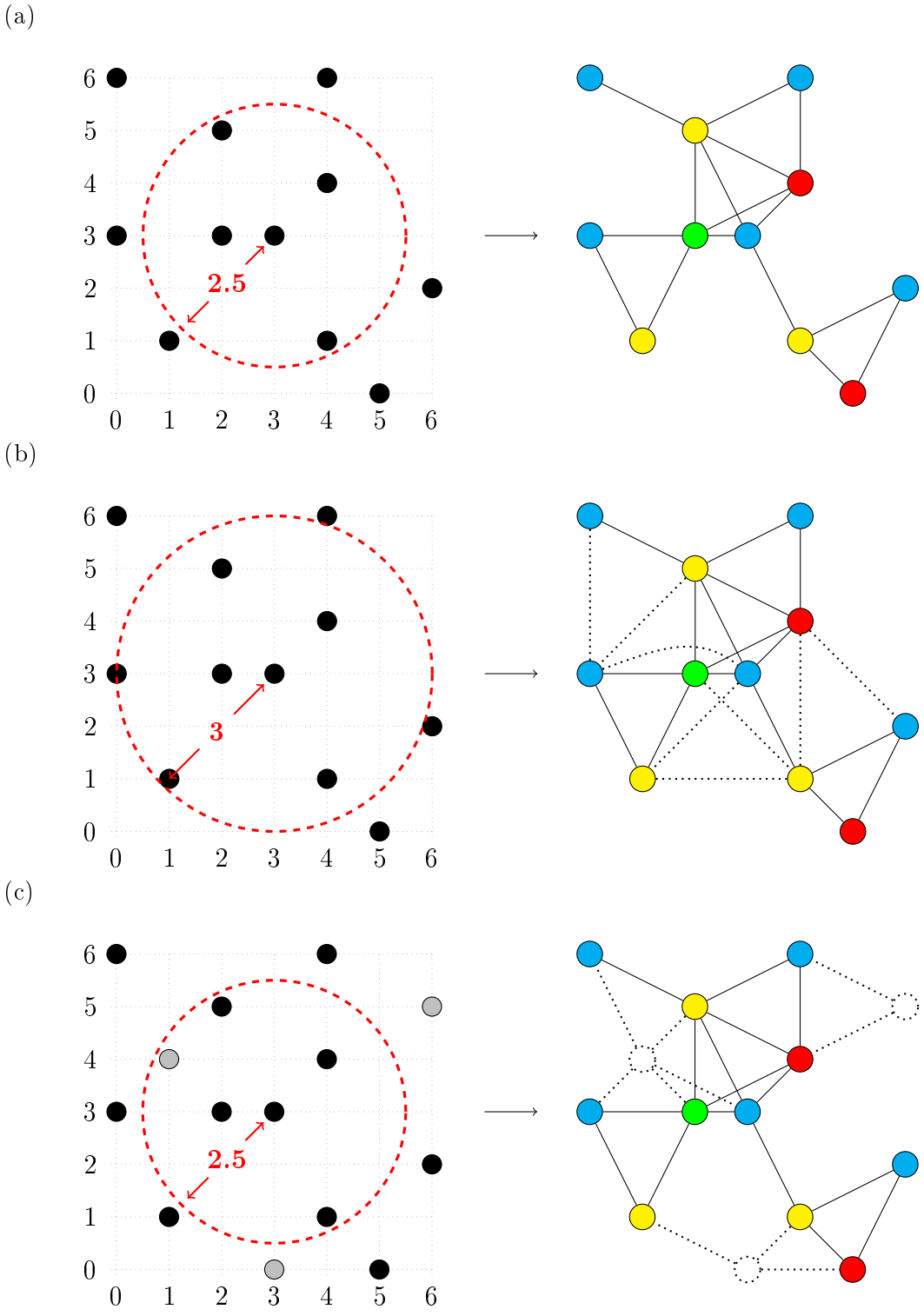

Figure 7.1: Example of a frequency assignment problem and the effects of two different changes on the associated GCP. The initial problem is presented in (a); the interference radius is increased from 2.5 to 3 in (b); and three additional locations (or objects) are introduced to the network in (c). Dotted lines and nodes represent edges and vertices that have been introduced because of the changes in the associated frequency assignment problem.

this situation.

7.3.2 "Similarity" as a Secondary Objective

Another way of extending the work of this thesis would be to introduce different secondary objectives that replace or work in conjunction with our secondary "robustness" objective.

One potential secondary objective could be the "similarity" of colourings between subsequent time-steps. The goal of this objective would be to increase the "similarity" of two colourings, or reduce the "distance" between them. Colourings that are "similar" to one another may be desirable for real-world applications, as the recolouring of vertices (or their associated real-world action) may incur some form of cost.

Indeed, one of the focuses of Dupont et al. [2009] in their work on the dynamic frequency assignment problem, is to reduce the number of frequency reassignments (i. e. the number of vertices that are recoloured) between time-steps, as there is a cost assigned to this action. However, unlike the work presented in this thesis, theirs is less concerned in reducing the number of colour classes (i. e. frequencies) used and more on reducing the number of vertices that are recoloured between time-steps. In other words, they are more concerned with solution robustness, whereas we have attempted to find a balance between multiple objectives (i. e. solution quality *and* robustness).

With regards to "similarity", there are several measures for set partitions which could be utilised. Considering two colouring $\mathcal{S}_1$ and $\mathcal{S}_2$, one potential similarity measure is the the Jaccard index, given by

$$\frac{|P_1 \cap P_2|}{|P_1 \cup P_2|} \tag{7.1}$$

where P_i is the set of vertex pairs which are in the same colour class in colouring $\mathcal{S}_i$ for $i = 1, 2$. Alternatively, we could use the Sørensen-Dice coefficient, given by

$$\frac{2|P_1 \cap P_2|}{|P_1| + |P_2|} \tag{7.2}$$

or the Rand index, given by

$$\frac{|P_1 \cap P_2| + |P_1^c \cap P_2^c|}{|P|} \tag{7.3}$$

where $P = \mathcal{E}$ is the set of all possible vertex pairs, and $P_i^c = P \backslash P_i$ for $i = 1, 2$. Each of these three measures gives a value between 0 and 1, where a value of 1 indicates that two colourings are identical (see Figure 7.2). As a secondary objective, we would aim to maximise one of Equations (7.1) to (7.3).

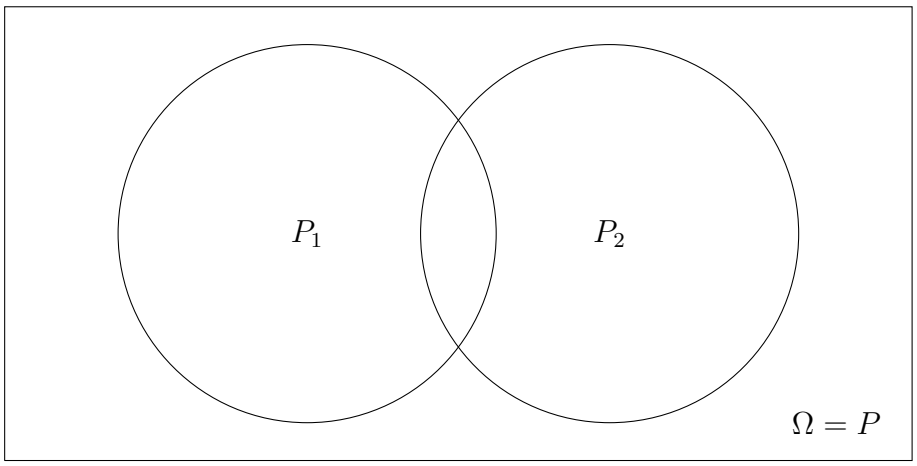

Figure 7.2: Space of all vertex pairs.

These measures can also be easily transformed into distance measures, which we would aim to minimise instead. To do this, we simply subtract our chosen similarity measure from 1. For example, the Jaccard "distance" between two colourings $\mathcal{S}_1$ and $\mathcal{S}_2$ would be given by

$$\mathcal{D}(\mathcal{S}_1, \mathcal{S}_2) = 1 - \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|} = \frac{|P_1 \cup P_2| - |P_1 \cap P_2|}{|P_1 \cup P_2|} \tag{7.4}$$

where P_i is defined as before, for $i = 1, 2$. A distance measure, defined like so, will also give a value between 0 and 1, but now a value of 0 will indicate that two colourings are identical (i. e. there is zero distance between them).

Another way of measuring distance between colourings could be to simply count how many vertices have changed colour class (e. g. as shown in Algorithm 7.1) which might be a more appropriate measure for real-world applications. One potential problem with this measure could occur in the case where two colour classes may have identical elements but are labelled differently. To counter such problems a matching algorithm, such as the Hungarian algorithm [Kuhn, 1955], could be implemented to re-label two colourings before they are compared.

Algorithm 7.1 "Distance" Calculator

Input: a graph $G = (V, E)$, two colourings $\mathcal{S}_1, \mathcal{S}_2$ for G, and their associated colouring functions $c_1 : V \to \{1, \ldots, k_1\}, c_2 : V \to \{1, \ldots, k_2\}$

Output: the number of 1-moves m required to transform $\mathcal{S}_1$ into $\mathcal{S}_2$ (or vice versa)

```
1: m ← 0
2: for i ← 1 to |V| do
3:     if c_1(v_i) ≠ c_2(v_i) then
4:         m ← m + 1
5: return m
```

Of course, this secondary distance objective $\mathcal{D}$ could easily be incorporated in conjunction with our secondary robustness objective $\mathcal{F}$: given a feasible colouring

$\mathcal{S}_t$ for G_t, our secondary objective might be to simultaneously reduce $\mathcal{F}(\mathcal{S}_{t+1})$ and $\mathcal{D}(\mathcal{S}_t, \mathcal{S}_{t+1})$. In other words, we would want to find a feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1} which is both robust to future change but not too "far away" from $\mathcal{S}_t$. If these secondary objectives were combined, then a pay-off might be observed, such that a colouring with small expected future adjacency (or low expected total saturation) will require fewer moves to become a feasible colouring in the following time-step and therefore the two colourings will also be relatively "similar".

In order to incorporate both of these secondary objectives simultaneously, a weighted function $w_{\mathcal{F}}\mathcal{F}(\mathcal{S}_{t+1}) + w_{\mathcal{D}}\mathcal{D}(\mathcal{S}_t, \mathcal{S}_{t+1})$ could be used during the second stage of Algorithm 4.1. The weights $w_{\mathcal{D}}$ and $w_{\mathcal{F}}$ could also be adjusted according to user preference.

7.4 Chapter Summary

In this thesis we have considered four different types of dynamic graph colouring problems: edge and vertex dynamic GCPs, both of which were considered with and without future change information. By applying new heuristic approaches to these problems, we have added to the very little research that currently exists in this area.

We have shown, for dynamic GCPs without future change information, that a modification approach with appropriate parameter settings can be beneficial with regards to either improving quality or reducing computational effort, when compared against treating each time-step of a dynamic graph independently. In fact, for test instances with small amounts of change between time-steps, this approach can be beneficial with regards to both quality *and* computational effort.

For dynamic GCPs with future change information, we have shown that a two-stage approach can reduce the amount of "problematic" change introduced between time-steps (i. e. this approach can improve the robustness of a colouring). However, there is a clear trade-off between the quality of a colouring and its potential robustness, such that a colouring with more colours (i. e. reduced quality) can be made more robust. By improving the robustness of a colouring, the benefits of our modification approach are then observed for parameter setting and test instances for which they were previously absent.

Finally, we have suggested areas of future research to extend the work presented in this thesis. In particular, one should consider test instances and secondary objectives for the two-stage approach that reflect real-world dynamic problems (e. g. dynamic frequency assignment).

Appendix A

Additional Pseudocode

The pseudocode and details presented in this appendix are done so to supplement the descriptions of some of the algorithms discussed in Chapter 2.

A.1 Constructive Algorithms

Algorithm A.1 GREEDY Algorithm

Input: a graph $G = (V, E)$, and an ordering of the $|V|$ vertices $\{v_1, \ldots, v_{|V|}\}$
Output: a feasible colouring $\mathcal{S}$ for G

```
1:  S_1 ← ∅
2:  S ← {S_1}
3:  for i ← 1 to |V| do
4:      success ← false
5:      for j ← 1 to |S| do
6:          if S_j ∪ {v_i} is an independent set then
7:              S_j ← S_j ∪ {v_i}
8:              success ← true
9:              break
10:     if not success then
11:         S_{|S|+1} ← {v_i}
12:         S ← S ∪ S_{|S|+1}
13: return S
```

Algorithm A.2 DSATUR Algorithm

Input: a graph $G = (V, E)$
Output: a feasible colouring $\mathcal{S}$ for G

```
 1: S_1 ← ∅
 2: S ← {S_1}
 3: X ← V
 4: while X ≠ ∅ do
 5:    choose v ∈ X
 6:    success ← false
 7:    for j ← 1 to |S| do
 8:       if S_j ∪ {v} is an independent set then
 9:          S_j ← S_j ∪ {v}
10:          success ← true
11:          break
12:    if not success then
13:       S_{|S|+1} ← {v}
14:       S ← S ∪ S_{|S|+1}
15:    X ← X\{v}
16: return S
```

For Line 4 of Algorithm A.2, the initial vertex $v \in X$ is chosen such that v has the maximum degree in G. Subsequently, $v \in X$ is chosen such that v has the highest saturation degree with regards to $\mathcal{S}$ (see Definition 2.1). Ties are broken randomly.

Algorithm A.3 RLF (Recursive Largest First) Algorithm

Input: a graph $G = (V, E)$
Output: a feasible colouring $\mathcal{S}$ for G

```
 1: S ← ∅ (i.e. S = {} and |S| = 0)
 2: X ← V
 3: Y ← ∅
 4: while X ≠ ∅ do
 5:    S_{|S|+1} ← ∅
 6:    while X ≠ ∅ do
 7:       choose v ∈ X
 8:       S_{|S|+1} ← S_{|S|+1} ∪ {v}
 9:       Y ← Y ∪ Γ_X(v)
10:       X ← X\{Y ∪ {v}}
11:    S ← S ∪ S_{|S|+1}
12:    X ← Y
13:    Y ← ∅
14: return S
```

For Line 7 of Algorithm A.3, the initial vertex $v \in X$ for each new colour class S_i is chosen such that v has the highest degree in the sub-graph induced by X. Each subsequent vertex $v \in X$ selected for S_i is chosen such that v has the highest degree in the sub-graph induced by $Y \cup \{v\}$. Ties are broken randomly.

For Line 9 of Algorithm A.3, $\Gamma_X(v) = \{u \in X : \{u, v\} \in E\}$.

A.2 Local Search Heuristics

Algorithm A.4 Simulated Annealing

Input: an initial solution $\mathcal{S}$, a starting temperature t_{start}, an objective function f (to be minimised), and a cooling function g
Output: the best solution found $\mathcal{S}_{best}$

1: $\mathcal{S}_{best} \leftarrow \mathcal{S}$
2: $t \leftarrow t_{start}$
3: **while not** stopping criterion **do**
4: choose $\mathcal{S}' \in \mathcal{N}(\mathcal{S})$
5: $\delta = |f(\mathcal{S}) - f(\mathcal{S}')|$
6: **if** $f(\mathcal{S}') < f(\mathcal{S})$ **then**
7: $\mathcal{S} \leftarrow \mathcal{S}'$
8: **if** $f(\mathcal{S}) < f(\mathcal{S}_{best})$ **then**
9: $\mathcal{S}_{best} \leftarrow \mathcal{S}$
10: **else if** $r \leq \exp(\frac{-\delta}{t})$ **then**
11: $\mathcal{S} \leftarrow \mathcal{S}'$
12: $t \leftarrow g(t)$
13: **return** $\mathcal{S}_{best}$

For Line 4 of Algorithm A.4, $\mathcal{N}(\mathcal{S})$ is the neighbourhood of solutions reachable from $\mathcal{S}$.

For Line 10 of Algorithm A.4, r is a random number sampled between 0 and 1, and is re-sampled at the beginning of each iteration.

For Line 13 of Algorithm A.4, the cooling function g satisfies $g(t) \leq t$ for all temperatures t.

Algorithm A.5 Tabu Search

Input: an initial solution $\mathcal{S}$, an objective function f (to be minimised), and update rules for tabu list $\mathcal{T}$
Output: the best solution found $\mathcal{S}_{best}$

1: $\mathcal{S}_{best} \leftarrow \mathcal{S}$
2: $\mathcal{T} \leftarrow \emptyset$
3: **while not** stopping criterion **do**
4: choose $\mathcal{S}' \in \mathcal{N}(\mathcal{S}) \backslash \mathcal{T}$
5: $\mathcal{S} \leftarrow \mathcal{S}'$
6: **if** $f(\mathcal{S}) < f(\mathcal{S}_{best})$ **then**
7: $\mathcal{S}_{best} \leftarrow \mathcal{S}$
8: update $\mathcal{T}$
9: **return** $\mathcal{S}_{best}$

For Line 4 of Algorithm A.5, $\mathcal{S}' \in \mathcal{N}(\mathcal{S}) \backslash \mathcal{T}$ is chosen such that either the most amount of reduction, or the least amount of increase to f can be realised. Note that there is no aspiration criterion here.

For Line 8 of Algorithm A.5, the update procedure both adds and removes solutions from the tabu list $\mathcal{T}$.

A.3 Independent Set Extraction

Algorithm A.6 Generic Independent Set Identifying Algorithm

Input: a graph $G = (V, E)$
Output: the largest independent set found $I_{best} \subseteq V$

1: Let I be an independent set produced by a constructive operator
2: $I_{best} \leftarrow I$
3: $l \leftarrow |I| + 1$
4: **while not** stopping criterion **do**
5: attempt to make I an independent set that satisfies $|I| = l$
6: **if** I is an independent set that satisfies $|I| = l$ **then**
7: $I_{best} \leftarrow I$
8: $l \leftarrow l + 1$
9: **return** I_{best}

For our experiments in Section 2.5, we use a "greedy" operator to produce an initial independent set I (i. e. Line 1 of Algorithm A.6). Beginning with $I \leftarrow \emptyset$, our "greedy" operator considers all vertices $v \in V$ in a random order, such that if $I \cup \{v\}$ is independent then $I \leftarrow I \cup \{v\}$.

For our experiments in Section 2.5, we use a tabu search procedure within the general framework of Algorithm A.6 (i. e. Line 5). The objective function of this tabu search procedure is

$$f(I) = |E \cap \{I \times I\}| \tag{A.1}$$

(i. e. the number of clashes in I), and if $f(I) = 0$ then I is an independent set. Here, transferring a vertex in or out of I is considered either "tabu" or not. During each iteration, I is transformed into $I' = (I \cup \{v\}) \backslash \{u\}$ such that $u \in I$ has the most adjacent vertices in I, and $v \in V \backslash I$ has least adjacent vertices in I. Of course, transferring both u and v must also be considered non-tabu for the current iteration. Once I' has been defined and $I \leftarrow I'$, the transferring of vertices u and v back in or out of I, respectively, is made "tabu" for a fixed number of subsequent iterations[1].

[1] More specifically, if u is transferred out of I, then it cannot be transferred back into I for the next $f(I') + r$ iterations, where r is a random integer from 0 to 4, and if v is transferred into I, then it cannot be transferred back out for the next $0.6 \cdot (f(I') + r)$ iterations.

Appendix B

Additional Results

The aim of this appendix is to demonstrate that the number (or desired number) of vertices for a given test instance (i. e. the value of n) has very little effect on the relationships observed between the different modification operators and their "benefits". We have chosen to present results related to the experiments covered Chapters 3 and 5 only, as we believe that this is enough to fully demonstrate our desired aim for both the edge dynamic and vertex dynamic cases, respectively.

For Chapter 3, Edge Dynamic GCP without Future Change Information

Figures B.1 and B.2, and Tables B.1 and B.2.

For Chapter 5, Vertex Dynamic GCP without Future Change Information

Figures B.3 and B.4, and Tables B.3 and B.4.

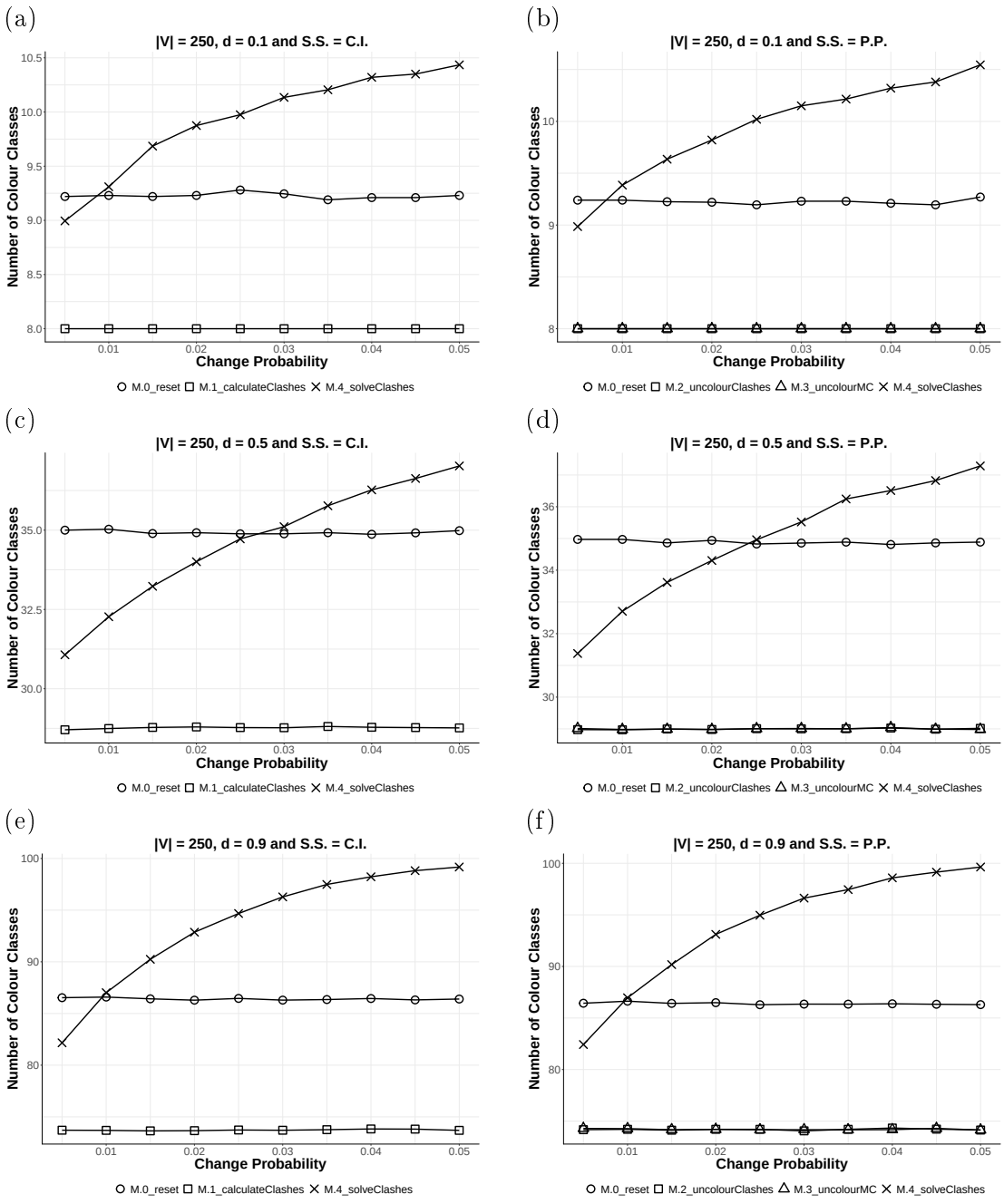

Figure B.1: Mean number of colour classes in initial, feasible colourings for the edge dynamic GCP on test instances with $|V| = 250$. Graphs (a), (c) and (e) represent methods that operate in the complete, improper solution space, graphs (b), (d) and (f) represent methods that operate in the partial, proper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

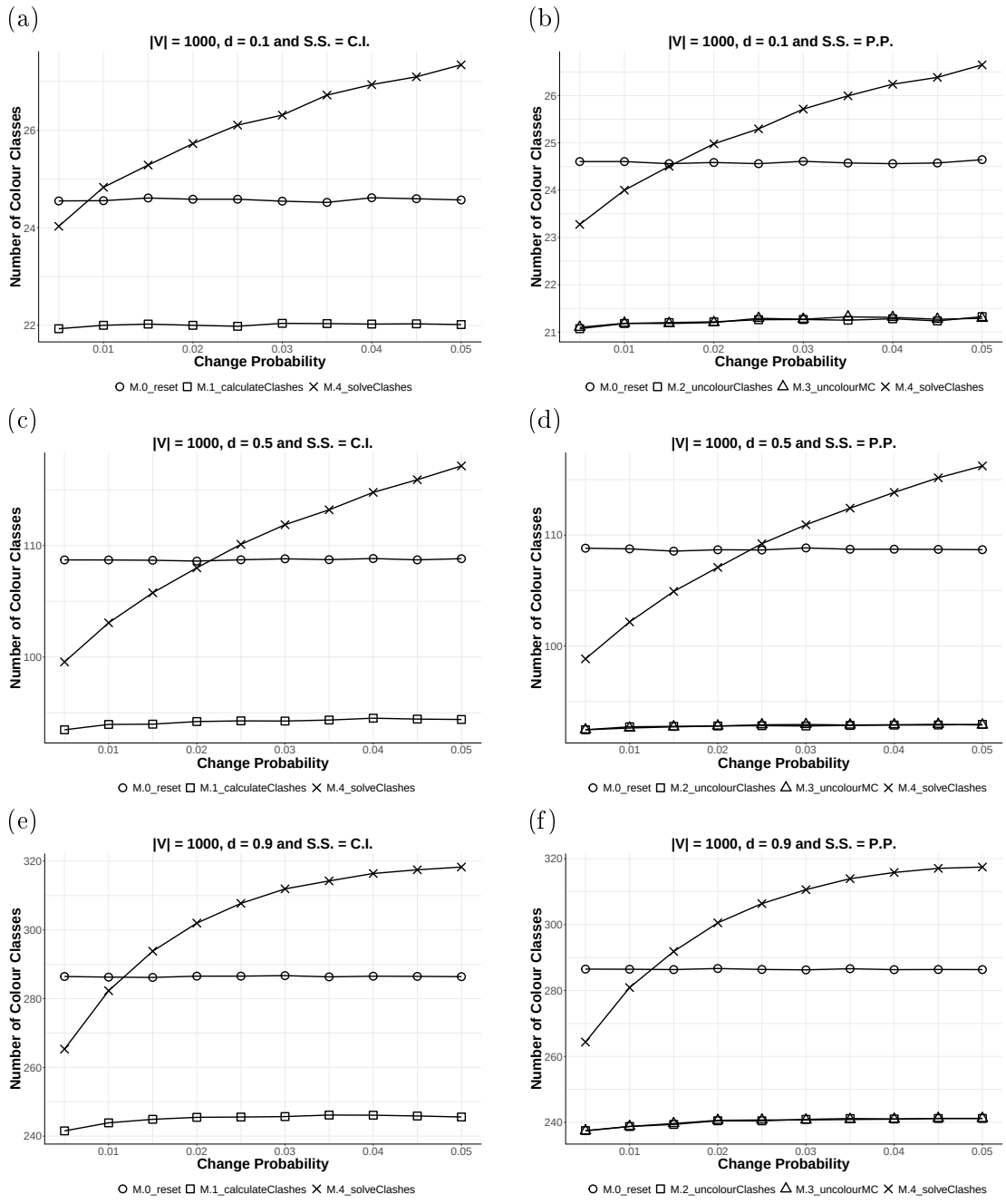

Figure B.2: Mean number of colour classes in initial, feasible colourings for the edge dynamic GCP on test instances with $|V| = 1000$. Graphs (a), (c) and (e) represent methods that operate in the complete, improper solution space, graphs (b), (d) and (f) represent methods that operate in the partial, proper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

Table B.1: Median time (in seconds) required to obtain an initial, feasible colouring for the edge dynamic GCP.

			p									
n	d	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	0	0	0	0	0	0	0	0	0	0	0
		1	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0	0.015
		4	0	0	0	0	0	0	0	0	0	0
	0.5	0	0	0	0	0	0	0	0	0	0	0
		1	0.296	0.429	0.538	0.733	0.647	1.334	0.726	1.084	0.780	0.748
		2	0.289	0.561	0.452	0.382	0.515	0.546	0.515	0.468	0.460	0.421
		3	0.445	0.452	0.390	0.406	0.476	0.390	0.531	0.515	0.492	0.468
		4	0	0	0	0	0	0	0	0	0	0
	0.9	0	0	0	0	0.015	0.015	0.015	0.015	0.015	0	0
		1	1.006	1.482	1.014	0.983	1.178	1.326	1.326	1.232	1.272	1.038
		2	0.819	0.936	0.843	1.022	0.889	1.443	1.170	1.389	1.162	1.076
		3	1.069	0.889	0.936	1.139	0.842	1.006	1.233	1.139	0.913	1.186
		4	0	0	0*	0*	0*	0*	0*	0*	0*	0*
1000	0.1	0	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031
		1	0.063	0.117	0.172	0.203	0.234	0.296	0.297	0.249	0.281	0.351
		2	0.882	1.428	1.537	2.083	2.075	2.270	2.504	2.489	2.247	2.402
		3	1.014	1.584	1.787	1.794	2.543	2.223	2.309	2.184	2.075	2.575
		4	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
	0.5	0	0.171	0.172	0.171	0.171	0.171	0.172	0.171	0.171	0.172	0.171
		1	5.007	5.007	5.007	5.008	5.008	5.008	5.008	5.008	5.008	5.008
		2	3.737	4.485	5.007	4.852	4.766	4.844	4.898	5.007	5.007	5.008
		3	3.307	3.994	4.735	4.891	5.007	5.008	5.007	5.007	5.008	5.008
		4	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
	0.9	0	0.219	0.234	0.234	0.234	0.234	0.234	0.234	0.234	0.234	0.234
		1	7.714	8.767	8.799	8.737	8.783	8.845	9.189	9.174	9.111	8.768
		2	6.630	7.472	7.628	7.815	8.050	7.956	7.941	8.081	8.206	8.065
		3	6.638	7.683	7.730	7.925	8.268	7.941	7.893	7.863	8.136	8.089
		4	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than all others for the same values of n, d and p.

Table B.2: Median time (in seconds) required to obtain final, feasible colourings with the same numbers of colour classes for the edge dynamic GCP. Missing values indicate test instances for which the colourings achieved by the associated modification operators were never *all* equal.

				p									
n	d	S.S.	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	C.I.	0	0	0	0	0	0	0	0	0	0	0
			1	0⋆	0⋆	0⋆	0⋆	0⋆	0	0	0⋆	0	0
			4	0⋆	0⋆	0⋆	0⋆	0⋆	0⋆	0	0	0	0
		P.P.	0	0	0	0	0	0	0	0	0	0	0
			2	0⋆	0⋆	0⋆	0	0	0	0	0	0	0
			3	0	0	0	0	0	0	0	0	0	0.015
			4	0⋆	0	0⋆	0	0	0	0	0	0	0
	0.5	C.I.	0	0.718	0.686	1.295	0.733	0.717	0.811	0.647	1.053	0.905	1.123
			1	0.312	0.593	1.201	1.802	1.466	2.184	1.295	2.184	2.901	1.466
			4	0.203⋆	0.390	0.748⋆	0.609	0.624	0.811	0.593	0.834	0.826	1.185
		P.P.	0	0.655	0.585	0.421	0.546	0.483	0.484	0.530	0.515	0.436	0.600
			2	0.327⋆	0.577	0.499	0.436	0.616	0.593	0.608	0.562	0.624	0.554
			3	0.500	0.507	0.436	0.452	0.586	0.452	0.577	0.593	0.639	0.522
			4	0.328⋆	0.460	0.484	0.483	0.507	0.452	0.554	0.608	0.468	0.507
	0.9	C.I.	0	0.913	1.030	0.905	0.842	0.748	0.827	0.803	0.842	0.991	0.975
			1	1.841	2.106	1.654	1.841	2.262	1.934	1.693	2.121	1.826	2.020
			4	0.905	1.014	1.170	0.983	0.967	0.951	0.843	0.842	0.944	0.921
		P.P.	0	0.913	0.826	0.866	0.897	1.123	0.967	1.022	0.952	1.170	0.827
			2	1.084	1.123	1.583	1.529	2.082	1.981	2.137	2.215	1.779	2.324
			3	1.381	1.763	1.342	1.739	1.474	1.981	1.615	1.755	1.841	2.200
			4	0.826	1.092	0.812	0.998	0.858	1.123	0.967	1.061	1.186	0.843
1000	0.1	C.I.	0	0.273	0.265	0.234	0.266	0.281	0.250	0.218	0.265	0.280	0.265
			1	0.063⋆	0.125⋆	0.172	0.203	0.234	0.289	0.296	0.250	0.281	0.343
			4	0.094⋆	0.156⋆	0.187	0.234	0.234	0.281	0.250	0.289	0.297	0.359
		P.P.	0	2.855	2.761	2.730	2.325	2.823	2.574	2.870	2.676	2.496	2.512
			2	0.858⋆	1.428⋆	1.677⋆	2.121⋆	2.433	2.168	2.434	2.426	2.153	2.403
			3	1.123⋆	1.631⋆	1.856⋆	1.716⋆	2.527	2.231⋆	2.418	2.215	2.106	2.465
			4	0.936⋆	1.568⋆	2.122⋆	1.950⋆	2.230⋆	2.090	2.293⋆	2.403	2.387	2.918
	0.5	C.I.	0	5.896	6.264	4.673	6.061	6.638	5.741	5.710	4.805	4.524	5.483
			1	5.023	5.585	5.390	6.381	6.864	5.881	6.077	6.833	6.974	6.131
			4	3.182⋆	4.025⋆	4.751	5.242	5.429	4.923	4.696	4.805	5.390	5.382
		P.P.	0	5.382	6.942	5.241	5.764	6.482	6.505	4.976	6.115	5.601	6.084
			2	4.914	4.547	6.232	6.248	5.585	5.522	7.004	6.505	6.450	5.928
			3	3.510⋆	4.727⋆	5.156	5.148	5.257	5.819	5.725	5.835	5.897	6.271
			4	3.744⋆	4.329⋆	5.764	5.202	5.561	5.507	5.648	5.835	5.023	4.758
	0.9	C.I.	0	9.017	9.337	8.744	8.377	9.540	8.159	6.006	5.553	6.677	7.659
			1	9.516	7.161	9.688	9.391	9.665	9.922	8.143	9.765	9.493	8.892
			4	6.225⋆	7.082	7.184	7.691	6.958	5.241	8.214	9.609	8.549	9.828
		P.P.	0	7.831	8.924	8.128	-	9.438	9.462	-	8.151	-	-
			2	8.284	7.964	8.331	-	7.863	9.134	-	8.853	-	-
			3	8.673	9.766	8.673	-	8.175	9.617	-	6.248	-	-
			4	6.974	6.272	7.472	-	9.032	9.275	-	7.839	-	-

0 represents a time less than 10^{-3} seconds.

⋆ indicates a time that is significantly less than Method 0 for the same values of n, d and p whilst operating in the same solution space.

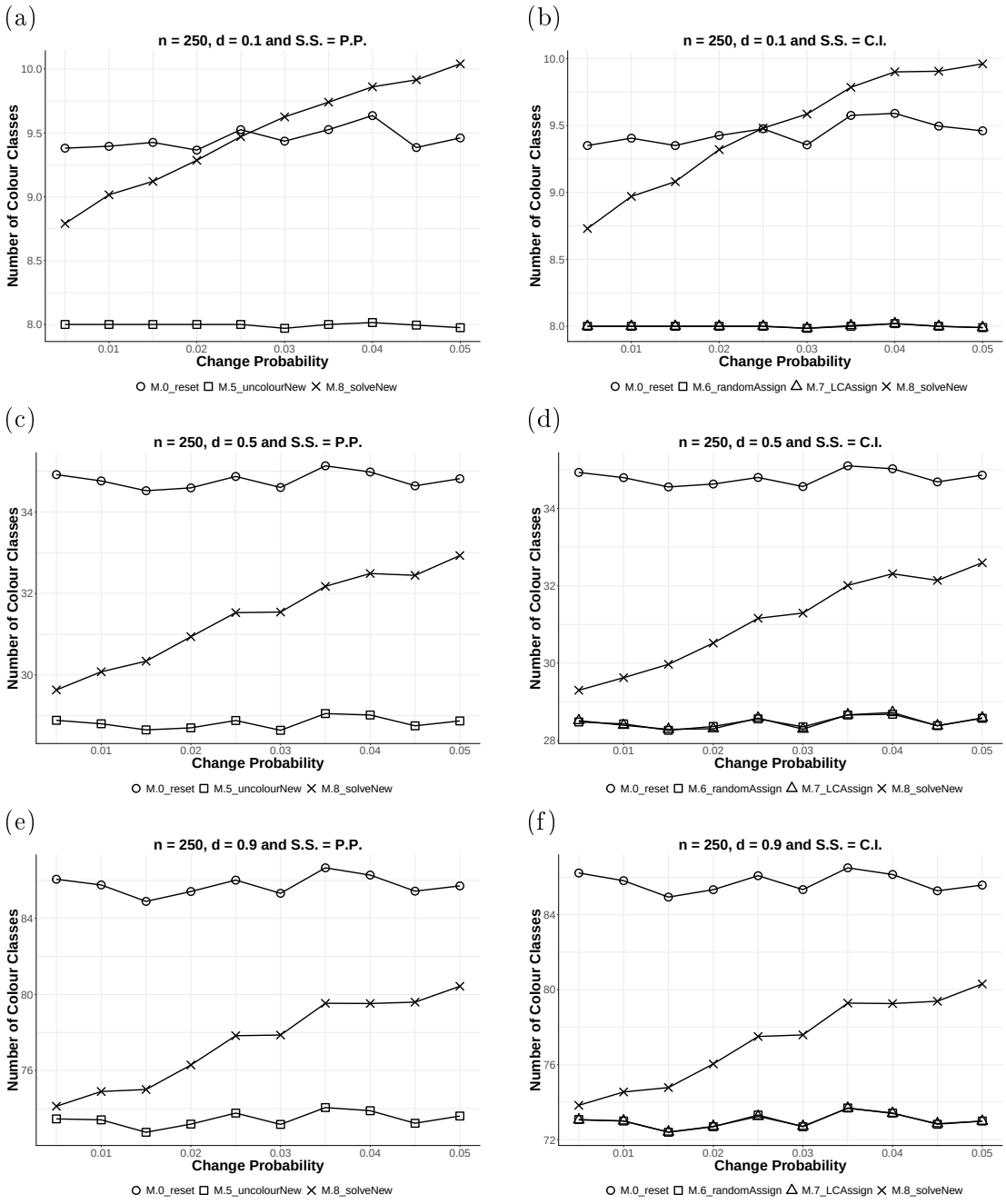

Figure B.3: Mean number of colour classes in initial, feasible colourings for the vertex dynamic GCP on test instances with $|V| = 250$. Graphs (a), (c) and (e) represent methods that operate in the partial, proper solution space, graphs (b), (d) and (f) represent methods that operate in the complete, improper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

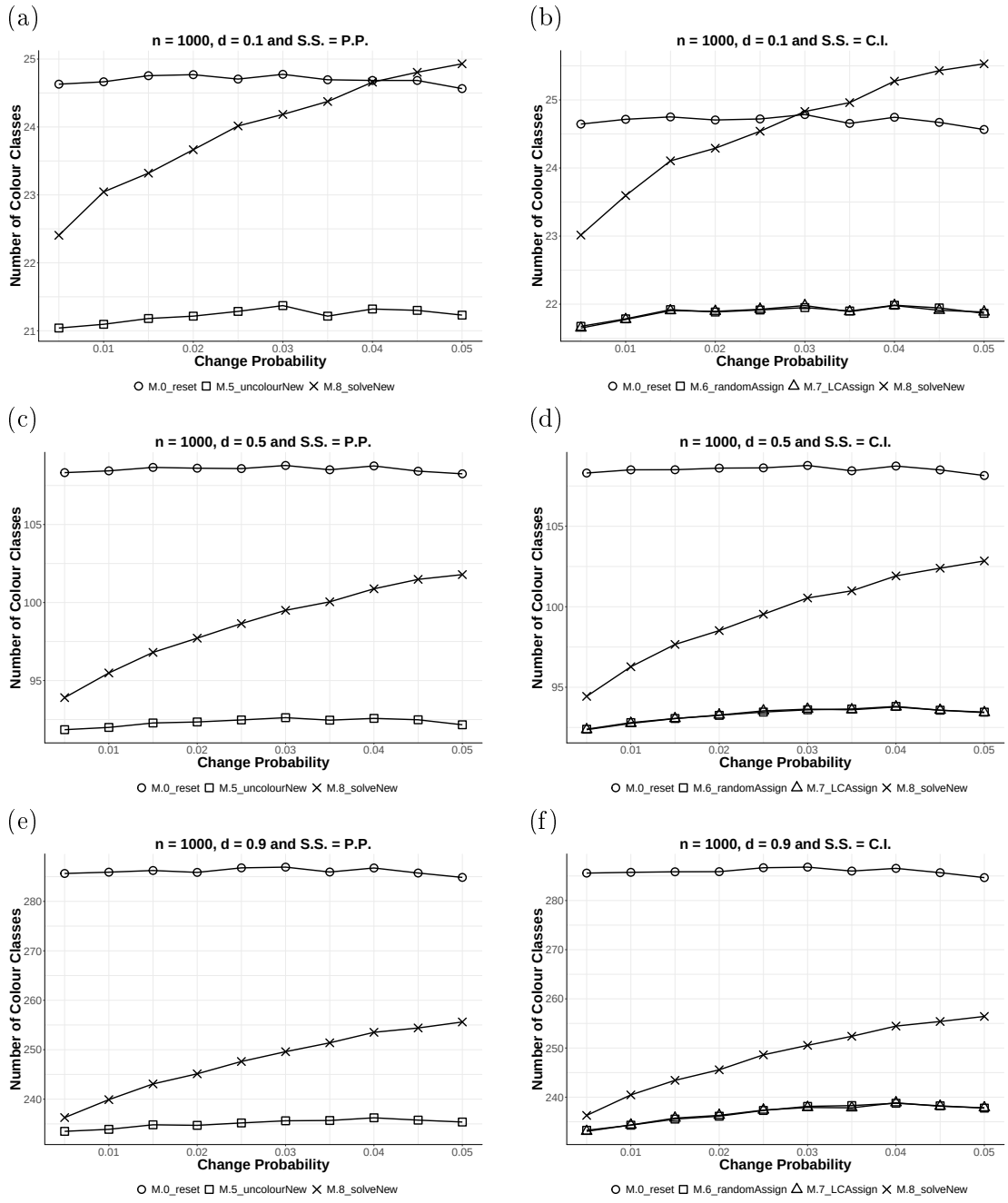

Figure B.4: Mean number of colour classes in initial, feasible colourings for the vertex dynamic GCP on test instances with $|V| = 1000$. Graphs (a), (c) and (e) represent methods that operate in the partial, proper solution space, graphs (b), (d) and (f) represent methods that operate in the complete, improper solution space, and the rows from top to bottom represent test instances with $d = 0.1, 0.5$ and 0.9, respectively.

Table B.3: Median time (in seconds) required to obtain an initial, feasible colouring for the vertex dynamic GCP on test instances with $n \in \{250, 1000\}$.

			p									
n	d	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	0	0	0	0	0	0	0	0	0	0	0
		5	0	0	0	0	0	0	0	0	0	0
		6	0	0	0	0	0	0	0	0	0	0
		7	0	0	0	0	0	0	0	0	0	0
		8	0	0	0	0	0	0	0	0	0	0
	0.5	0	0	0	0	0	0	0	0	0	0	0
		5	0.031	0.218	0.140	0.343	0.328	0.320	0.718	0.710	0.445	0.398
		6	0.062	0.164	0.187	0.281	0.359	0.374	0.515	0.694	0.608	0.460
		7	0.031	0.195	0.148	0.437	0.530	0.375	0.562	0.600	0.827	0.492
		8	0	0	0	0	0	0	0*	0*	0*	0*
	0.9	0	0	0	0.015	0.015	0	0	0	0	0	0
		5	0.203	0.273	0.280	0.234	0.655	0.531	0.632	0.647	0.546	0.515
		6	0.078	0.187	0.164	0.328	0.694	0.312	0.578	0.640	0.561	0.687
		7	0.062	0.133	0.172	0.258	0.718	0.406	0.656	0.600	0.577	0.632
		8	0	0*	0*	0*	0*	0*	0*	0*	0*	0*
1000	0.1	0	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031	0.031
		5	0.187	0.546	0.851	0.952	0.913	1.584	1.108	0.874	0.889	0.881
		6	0.047	0.063	0.094	0.140	0.296	0.265	0.273	0.312	0.289	0.281
		7	0.047	0.062	0.109	0.187	0.281	0.218	0.297	0.328	0.305	0.390
		8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
	0.5	0	0.171	0.171	0.172	0.172	0.171	0.172	0.172	0.172	0.172	0.172
		5	2.068	2.559	2.902	3.611	3.705	3.963	3.370	4.259	4.470	4.228
		6	1.428	2.418	3.167	4.009	4.236	5.007	5.007	5.007	5.007	5.007
		7	1.030	2.637	2.964	4.103	4.727	5.007	5.007	5.008	5.007	5.007
		8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*
	0.9	0	0.234	0.249	0.234	0.234	0.249	0.249	0.234	0.234	0.218	0.218
		5	2.707	3.721	4.509	4.634	5.039	5.265	5.202	5.562	5.695	5.250
		6	3.019	4.688	5.016	5.156	5.967	6.303	6.575	7.005	7.293	7.519
		7	3.198	4.751	5.156	5.335	5.554	6.669	5.842	6.786	7.129	7.387
		8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than all others for the same values of n, d and p.

Table B.4: Median time (in seconds) required to obtain final, feasible colourings with the same numbers of colour classes for the vertex dynamic GCP on test instances with $n \in \{250, 1000\}$. Missing values indicate test instances for which the colourings achieved by the associated modification operators were never *all* equal.

				p									
n	d	S.S.	M.	0.005	0.010	0.015	0.020	0.025	0.030	0.035	0.040	0.045	0.050
250	0.1	P.P.	0	0	0	0	0	0	0	0	0	0	0
			5	0*	0*	0	0*	0*	0*	0*	0	0*	0*
			8	0*	0*	0*	0*	0	0*	0	0	0*	0
		C.I.	0	0	0	0	0	0	0	0	0	0	0
			6	0*	0*	0*	0*	0*	0*	0*	0*	0*	0
			7	0*	0*	0*	0*	0*	0*	0*	0*	0*	0
			8	0*	0*	0*	0*	0*	0*	0*	0*	0*	0
	0.5	P.P.	0	0.749	0.499	0.624	0.561	0.717	0.795	0.780	0.764	0.780	0.694
			5	0.070*	0.359*	0.327	0.554	0.499	0.624	0.811	1.279	0.951	0.702
			8	0.078*	0.140*	0.203*	0.351*	0.452*	0.468*	0.609*	0.577*	0.640	0.678
		C.I.	0	1.076	1.201	1.014	1.310	1.123	1.006	1.092	0.983	1.303	1.014
			6	0.109*	0.265*	0.405*	0.483*	0.795	0.679	0.796	1.014	0.952	0.764
			7	0.094*	0.304*	0.421*	0.686*	0.827	0.710	0.881	0.998	1.178	1.014
			8	0.078*	0.211*	0.265*	0.468*	0.655*	0.570*	0.600*	0.577*	0.882*	0.686*
	0.9	P.P.	0	0.998	0.827	0.928	0.842	0.889	0.897	0.850	0.904	0.827	1.419
			5	0.374	0.702	0.624	0.975	1.124	1.459	1.451	1.623	2.060	1.404
			8	0.218*	0.328*	0.359*	0.523*	0.748	0.710	0.858	0.656	0.749	0.904*
		C.I.	0	0.952	0.920	0.804	0.834	0.998	0.951	0.999	0.913	1.030	0.733
			6	0.187*	0.405	0.398*	0.921	1.061	0.873	1.358	1.162	1.568	1.700
			7	0.195*	0.437*	0.530	0.921	1.186	1.154	1.279	1.342	1.607	1.544
			8	0.140*	0.296*	0.390*	0.554*	0.500*	0.592	0.562	0.640	0.820	0.733
1000	0.1	P.P.	0	2.044	1.904	2.075	1.841	1.622	1.966	1.482	1.474	1.318	1.178
			5	0.187*	0.531*	0.757*	0.889*	0.873*	1.584	1.092*	0.952*	0.889*	0.881
			8	0.250*	0.640*	1.007*	1.030*	0.812*	1.381*	1.061*	0.952*	1.178*	1.045*
		C.I.	0	0.406	0.421	0.328	0.468	0.624	0.359	0.406	0.406	0.437	0.499
			6	0.032*	0.086*	0.117*	0.172*	0.343	0.266	0.312	0.312	0.383	0.515
			7	0.047*	0.062*	0.109†	0.187*	0.297*	0.234*	0.359	0.344	0.351	0.483
			8	0.047*	0.125*	0.164*	0.203*	0.234*	0.289*	0.281*	0.327	0.296	0.351
	0.5	P.P.	0	5.709	6.123	6.459	5.913	5.553	5.678	5.312	5.780	6.146	5.616
			5	3.525*	5.032*	5.007*	5.094*	4.384*	5.008	5.335	4.844	5.008*	5.179
			8	2.324*	2.949*	3.354*	3.237*	3.681*	4.454*	3.970	4.844*	3.416*	3.854*
		C.I.	0	7.434	5.725	5.850	5.865	5.975	6.490	5.819	5.148	6.747	5.741
			6	2.363*	2.652*	3.073*	5.663*	5.008	5.491	5.546	5.881	7.169	5.460
			7	1.007*	2.948*	3.401*	5.008*	5.008*	5.335	5.897	5.429	6.108	5.944
			8	1.779*	1.662*	2.465*	2.777*	3.229*	2.979*	2.793*	3.089*	5.445	4.852
	0.9	P.P.	0	8.596	7.472	7.769	8.752	9.516	8.370	7.940	7.379	8.159	7.660
			5	5.850	4.071	5.491	2.902	8.534	8.159	6.069	8.205*	7.184	7.940
			8	6.880	7.862	8.314*	6.396	6.381*	6.084*	5.117*	5.522*	5.593*	5.195*
		C.I.	0	-	-	-	9.376	-	-	7.690	8.518	-	9.501
			6	-	-	-	7.160	-	-	8.596	8.393	-	9.329
			7	-	-	-	7.309	-	-	7.519	8.144	-	6.396
			8	-	-	-	4.906	-	-	5.226*	5.203*	-	9.454*

0 represents a time less than 10^{-3} seconds.
* indicates a time that is significantly less than Method 0 for the same values of n, d and p whilst operating in the same solution space.
† indicates a time that is significantly less than Methods 0 and 8 for the same values of n, d and p whilst operating in the same solution space.

References

Aardal, K. I., van Hoesel, S. P. M., Koster, A. M. C. A., Mannino, C., and Sassano, A. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, 2007.

Avanthay, C., Hertz, A., and Zufferey, N. A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151(2):379–388, 2003.

Barba, L., Cardinal, J., Korman, M., Langerman, S., van Renssen, A., Roeloffzen, M., and Verdonschot, S. Dynamic graph coloring. In *Workshop on Algorithms and Data Structures*, pages 97–108. Springer, 2017.

Bhattacharya, S., Chakrabarty, D., Henzinger, M., and Nanongkai, D. Dynamic algorithms for graph coloring. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–20. SIAM, 2018.

Blöchliger, I. and Zufferey, N. A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research*, 35(3):960–975, 2008.

Brélaz, D. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

Brooks, R. L. On colouring the nodes of a network. *Mathematical Proceedings of the Cambridge Philosophical Society*, 37(2):194–197, 1941.

Chaitin, G. J. Register allocation & spilling via graph coloring. *ACM Sigplan Notices*, 17(6):98–101, 1982.

Chams, M., Hertz, A., and de Werra, D. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32(2):260–266, 1987.

Chiarandini, M. and Stützle, T. An application of iterated local search to graph coloring problem. In *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations*, pages 112–125, 2002.

Costa, D. An evolutionary tabu search algorithm and the NHL scheduling problem. *Infor-Information Systems and Operational Research*, 33(3):161–178, 1995.

Costa, D. and Hertz, A. Ants can colour graphs. *Journal of the operational research society*, 48(3):295–305, 1997.

Costa, D., Hertz, A., and Dubuis, C. Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs. *Journal of Heuristics*, 1 (1):105–128, 1995.

Dorigo, M., Birattari, M., and Stutzle, T. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

Dorne, R. and Hao, J.-K. A new genetic local search algorithm for graph coloring. In *International Conference on Parallel Problem Solving from Nature*, pages 745–754. Springer, 1998.

Dowsland, K. A. and Thompson, J. M. An improved ant colony optimisation heuristic for graph colouring. *Discrete Applied Mathematics*, 156(3):313–324, 2008.

Dupont, A., Linhares, A. C., Artigues, C., Feillet, D., Michelon, P., and Vasquez, M. The dynamic frequency assignment problem. *European Journal of Operational Research*, 195(1):75–88, 2009.

Dutot, A., Guinand, F., Olivier, D., and Pigné, Y. On the decentralized dynamic graph coloring problem. In *European Simulation and Modelling Conference*, pages 259–261. EUROSIS, 2007.

Erben, W. A grouping genetic algorithm for graph colouring and exam timetabling. In *Practice and Theory of Automated Timetabling III*, pages 132–156. Springer, 2001.

Fleurent, C. and Ferland, J. A. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63(3):437–461, 1996.

Galinier, P. and Hao, J.-K. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, 3(4):379–397, 1999.

Galinier, P. and Hertz, A. A survey of local search methods for graph coloring. *Computers & Operations Research*, 33(9):2547–2562, 2006.

Galinier, P., Hertz, A., and Zufferey, N. *An Adaptive Memory Algorithm for the k-colouring problem.* Groupe d'études et de recherche en analyse des décisions, HEC Montréal, 2004.

Galinier, P., Hamiez, J.-P., Hao, J.-K., and Porumbel, D. Recent advances in graph vertex coloring. In *Handbook of optimization*, pages 505–528. Springer, 2013.

Garey, M. R. and Johnson, D. S. The complexity of near-optimal graph coloring. *Journal of the ACM (JACM)*, 23(1):43–49, 1976.

Garey, M. R. and Johnson, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co., San Francisco, 1979.

Geller, D. P. and Schmeichel, E. F. 5713. *The American Mathematical Monthly*, 78 (2):205–206, 1971.

Glass, C. A. and Prügel-Bennett, A. Genetic algorithm for graph coloring: exploration of galinier and hao's algorithm. *Journal of Combinatorial Optimization*, 7 (3):229–236, 2003.

Glover, F. Tabu search - part I. *ORSA Journal on computing*, 1(3):190–206, 1989.

Glover, F. Tabu search - part II. *ORSA Journal on computing*, 2(1):4–32, 1990.

Gyárfás, A. and Lehel, J. On-line and first fit colorings of graphs. *Journal of Graph theory*, 12(2):217–227, 1988.

Hansen, P., Labbé, M., and Schindl, D. Set covering and packing formulations of graph coloring: algorithms and first polyhedral results. *Discrete Optimization*, 6 (2):135–147, 2009.

Hao, J.-K. and Wu, Q. Improving the extraction and expansion method for large graph coloring. *Discrete Applied Mathematics*, 160(16):2397–2407, 2012.

Harary, F. and Gupta, G. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79–87, 1997.

Hell, P. and Nešetril, J. *Graphs and Homomorphisms (Volume 28 of Oxford Lecture Series in Mathematics and its Applications)*. Oxford University Press, 2004.

Hertz, A. and de Werra, D. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

Hertz, A., Plumettaz, M., and Zufferey, N. Variable space search for graph coloring. *Discrete Applied Mathematics*, 156(13):2551–2560, 2008.

Jagota, A. An adaptive, multiple restarts neural network algorithm for graph coloring. *European Journal of Operational Research*, 93(2):257–270, 1996.

Johnson, D. S., Aragon, C. R., McGeoch, L. A., and Schevon, C. Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. *Operations research*, 39(3):378–406, 1991.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

Korman, S. M. The graph-colouring problem. *Combinatorial optimization*, pages 211–235, 1979.

Kubale, M. and Jackowski, B. A generalized implicit enumeration algorithm for graph coloring. *Communications of the ACM*, 28(4):412–418, 1985.

Kuhn, H. W. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

Laguna, M. and Martí, R. A grasp for coloring sparse graphs. *Computational optimization and applications*, 19(2):165–178, 2001.

Leighton, F. T. A graph coloring algorithm for large scheduling problems. *Journal of research of the national bureau of standards*, 84(6):489–506, 1979.

Lewis, R. A general-purpose hill-climbing method for order independent minimum grouping problems: A case study in graph colouring and bin packing. *Computers & Operations Research*, 36(7):2295–2310, 2009.

Lewis, R. *A Guide to Graph Colouring*. Springer, 2015.

Lewis, R. and Carroll, F. Creating seating plans: a practical application. *Journal of the Operational Research Society*, 67(11):1353–1362, 2016.

Lewis, R., Thompson, J., Mumford, C., and Gillard, J. A wide-ranging computational comparison of high-performance graph colouring algorithms. *Computers & Operations Research*, 39(9):1933–1950, 2012.

Lovász, L., Saks, M., and Trotter, W. T. An on-line graph coloring algorithm with sublinear performance ratio. *Annals of Discrete Mathematics*, 43:319–325, 1989.

Lü, Z. and Hao, J.-K. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(1):241–250, 2010.

Malaguti, E., Monaci, M., and Toth, P. A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, 20(2):302–316, 2008.

Malaguti, E., Monaci, M., and Toth, P. An exact approach for the vertex coloring problem. *Discrete Optimization*, 8(2):174–190, 2011.

Mehrotra, A. and Trick, M. A. A column generation approach for graph coloring. *informs Journal on Computing*, 8(4):344–354, 1996.

Mladenović, N. and Hansen, P. Variable neighborhood search. *Computers & operations research*, 24(11):1097–1100, 1997.

Monical, C. and Stonedahl, F. Static vs. dynamic populations in genetic algorithms for coloring a dynamic graph. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 469–476. ACM, 2014.

Morgenstern, C. Distributed coloration neighborhood search. *Discrete Mathematics and Theoretical Computer Science*, 26:335–358, 1996.

Morgenstern, C. and Shapiro, H. Coloration neighborhood structures for general graph coloring. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 226–235. Society for Industrial and Applied Mathematics, 1990.

Ouerfelli, L. and Bouziri, H. Greedy algorithms for dynamic graph coloring. In *Communications, Computing and Control Applications (CCCA), 2011 International Conference on*, pages 1–5. IEEE, 2011.

Preuveneers, D. and Berbers, Y. ACODYGRA: an agent algorithm for coloring dynamic graphs. *Symbolic and Numeric Algorithms for Scientific Computing (September 2004)*, 6:381–390, 2004.

Qu, R., Burke, E. K., and McCollum, B. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, 198(2):392–404, 2009.

Tantipathananandh, C., Berger-Wolf, T., and Kempe, D. A framework for community identification in dynamic social networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 717–726. ACM, 2007.

Thompson, J. M. and Dowsland, K. A. A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7):637–648, 1998.

Titiloye, O. and Crispin, A. Quantum annealing of the graph coloring problem. *Discrete Optimization*, 8(2):376–384, 2011.

Welsh, D. J. A. and Powell, M. B. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10 (1):85–86, 1967.

Wu, Q. and Hao, J.-K. Coloring large graphs based on independent set extraction. *Computers & Operations Research*, 39(2):283–290, 2012.

Wu, Q. and Hao, J.-K. An extraction and expansion approach for graph coloring. *Asia-Pacific Journal of Operational Research*, 30(05), 2013.