J Heuristics
DOI 10.1007/s10732-017-9327-z

Tackling the edge dynamic graph colouring problem with and without future adjacency information

Bradley Hardy[1] · **Rhyd Lewis**[1] · **Jonathan Thompson**[1]

Received: 28 July 2016 / Accepted: 20 March 2017

Abstract Many real world operational research problems, such as frequency assignment and exam timetabling, can be reformulated as graph colouring problems (GCPs). Most algorithms for the GCP operate under the assumption that its constraints are fixed, allowing us to model the problem using a static graph. However, in many real-world cases this does not hold and it is more appropriate to model problems with constraints that change over time using an edge dynamic graph. Although exploring methods for colouring dynamic graphs has been identified as an area of interest with many real-world applications, to date, very little literature exists regarding such methods. In this paper we present several heuristic methods for modifying a feasible colouring at time-step t into an initial, but not necessarily feasible, colouring for a "similar" graph at time-step $t + 1$. We will discuss two cases; (1) where changes occur at random, and (2) where probabilistic information about future changes is provided. Experimental results are also presented and the benefits of applying these particular modification methods are investigated.

Keywords Graph colouring · Dynamic graphs · Heuristics

This work was funded by an EPSRC scholarship (Project Code: 1376020). Information about the data that underpins the results presented here, including how to access them, can be found in the Cardiff University data catalogue at http://doi.org/10.17035/d.2017.0033322283.

✉ Bradley Hardy
hardyB@cardiff.ac.uk

Rhyd Lewis
lewisR9@cardiff.ac.uk

Jonathan Thompson
thompsonJM1@cardiff.ac.uk

[1] School of Mathematics, Cardiff University, Cardiff CF24 4AG, UK

Published online: 04 April 2017

1 Introduction

Given a graph $G = (V, E)$ with vertex set V and edge set E, the NP-hard graph colouring problem (GCP) aims to colour each vertex 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 called the chromatic number of G, denoted by $\chi(G)$.

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), one can reformulate many real world problems into a GCP. A prominent example is frequency assignment (Aardal et al. 2007) where we wish to assign communication frequencies (e.g. radio wavelengths) to a set of physical locations such that there is no interference. Here, each location is represented by a vertex, an edge exists between two vertices if their respective locations are within a certain proximity of one another (which could lead to interference) and colours represent the communication frequencies. Other examples include register allocation (Chaitin 1982), tournament scheduling (Costa 1995), exam timetabling (Erben 2001; Qu et al. 2009), designing seating plans (Lewis and Carroll 2016) and grouping people in social networks (Tantipathananandh et al. 2007).

Most GCP methods suggested in the literature have only been applied to static graphs and can therefore only be applied to real world problems under the assumption that the size and constraints of a given problem are fixed (i.e. V and E are fixed in the associated graph $G = (V, E)$). However, in areas such as the frequency assignment problem (Dupont et al. 2009) this is not always appropriate because physical locations can be added or removed from the communication network, or relocated within the communication network.

On-line graph colouring, is one example of dynamic graph colouring that has received some attention in the literature. In this case, one vertex and its associated edges are revealed at each time-step. Once revealed, a vertex must be coloured using only the information available up to and including that time-step and may not be "recoloured" in subsequent time-steps. Research concerning on-line graph colouring mainly consists of worst-case behaviour analysis of simple constructive algorithms (Gyárfás and Lehel 1988; Lovász et al. 1989).

The aim of this particular research is to explore graph colouring on edge dynamic graphs where the edge set E changes over time. More specifically, we wish to look at methods that are able to modify a feasible colouring for one graph into an initial colouring for a new graph in a subsequent time-step. In doing so, we wish to explore whether this modification approach leads to any advantages with regards to colouring quality and/or time requirements. We also wish to explore how information about the likelihood of future changes can be used to produce more robust colourings. To the best of our knowledge, the problem of colouring edge dynamic graphs has received very little attention in the literature to date.

This paper has the following structure: Sect. 2 will formally define edge dynamic graphs and the associated graph colouring problem. Section 3 will then introduce ways of representing a colouring (or solution) and the various solution spaces in which graph colouring heuristics generally operate. In Sect. 4, the idea of future adjacency will be

discussed along with move operators that can maintain the feasibility of a colouring. Section 5 will then outline a general approach for solving the edge dynamic GCP (both with and without future adjacency information) and define the different modification operators used. Sections 6 and 7 then contain the experimentation details and results respectively. Finally, Sect. 8 summarises the findings of the experiments and discusses opportunities for future work.

2 Edge dynamic graphs

The importance of studying dynamic graphs and their associated problems is highlighted by Harary and Gupta (1997) who have described many practical application areas, especially in the area of computer science, and postulate that techniques applied to static graphs should be extended for dynamic graphs. Despite this, there has been very little research regarding methods designed explicitly for colouring dynamic graphs.

For the purpose of this particular work, 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 Sect. 3 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 can be considered as two separate cases: edge dynamic graphs and vertex dynamic graphs. In this paper we focus solely on the former case. For an edge dynamic graph, changes can only occur on the edge set E_t; therefore $V_t = V$ for all time-steps $t \in \{0, 1, \ldots, T\}$. Given an edge dynamic graph $\mathcal{G}$, consider the graph $G_t = (V, E_t)$ for 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 Fig. 1. As mentioned, the vertex set V remains fixed between time-steps.

3 Colourings and solution spaces

In this section we introduce ways of representing a colouring (or solution) and the various solution spaces that are commonly used with graph colouring heuristics.

A feasible colouring for a graph $G = (V, E)$ is a partition of the vertex set V into k disjoint (i.e. non-overlapping) subsets $\mathcal{S} = \{S_1, \ldots, S_k\}$ such that adjacent vertices are always in different subsets. By partitioning V in this way, each $S_i \in \mathcal{S}$ is an independent set. We call $\mathcal{S}$ a k-colouring of G and S_i the ith colour class of $\mathcal{S}$. For convenience, we can also use the colouring function $c : V \to \{1, \ldots, k\}$ which is defined such that $c(v) = i$ for all $v \in S_i$. If adjacent vertices are assigned to the same

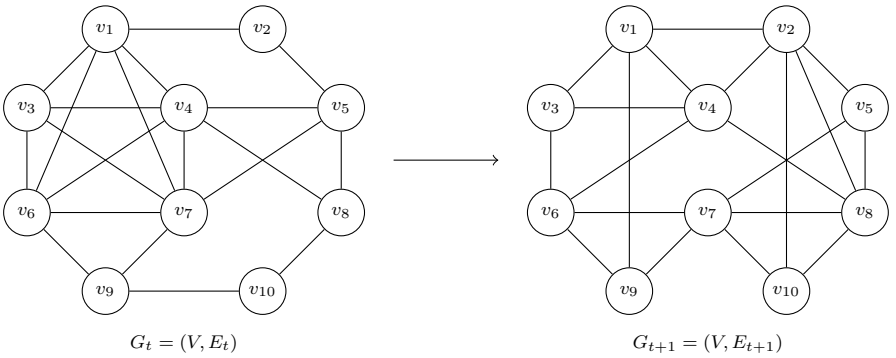

Fig. 1 Edge dynamic graph with $E^-_{t+1} = \{\{v_1, v_6\}, \{v_1, v_7\}, \{v_3, v_7\}, \{v_4, v_5\}, \{v_4, v_7\}, \{v_9, v_{10}\}\}$ and $E^+_{t+1} = \{\{v_1, v_9\}, \{v_2, v_4\}, \{v_2, v_8\}, \{v_2, v_{10}\}, \{v_7, v_8\}, \{v_7, v_{10}\}\}$

colour class, such that $c(u) = c(v)$ and $\{u, v\} \in E$, then this is called a *clash*. By definition, feasible colourings have no clashes.

Another way of representing feasible colourings, not used here, involves graph homomorphisms. In this representation, non-adjacent vertices are contracted to produce a complete graph with k hyper-vertices such that each hyper-vertex represents a colour class (Hell and Nešetril 2004).

When considering heuristic methods for solving the static GCP, Hertz et al. (2008), Lewis (2015), and Lewis et al. (2012) suggest three main solution spaces: (1) *feasible only*, where every vertex is coloured, there are no clashes, and the number of colour classes is allowed to vary; (2) *complete, improper*, where every vertex is coloured but clashes are permitted; and (3) *partial, proper*, where no clashes occur but there may be "uncoloured" vertices. The number of colour classes k is generally fixed when operating in the latter two cases, though it is often reduced once a feasible k-colouring has been obtained.

The feasible only solution space is rarely used for the static GCP. This is due to the difficulty in determining which of two feasible k-colourings is "closer" to becoming a colouring with $k - 1$ colour classes. One exception is a simulated annealing approach proposed by Johnson et al. (1991). As we will see in Sect. 4, this solution space is perhaps the most useful when considering the edge dynamic problem with probabilistic future adjacency information. This is because working within this solution space allows a colouring to be optimised with regards to a secondary objective without violating the constraints of the GCP.

As mentioned, in the complete, improper solution space a colour class $S_i \in \mathcal{S}$ is allowed to contain adjacent vertices. Therefore, an appropriate objective function for this solution space is simply

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

which is equivalent to the number of clashes in the colouring. If $f(\mathcal{S}) = 0$ then $\mathcal{S}$ is a feasible k-colouring.

To move from one colouring $\mathcal{S}$ to a neighbouring colouring $\mathcal{S}'$ within the complete, improper solution space, one popular strategy is to transfer a vertex v from its current colour class $S_{c(v)}$ to a different colour class S_j where $j \neq c(v)$. The vertex v to be moved can also be chosen exclusively from the set of currently clashing vertices (i.e. we can transfer $v \in S_i$ if and only if $\exists u \in S_i$ such that $u \neq v$ and $\{u, v\} \in E$). The tabu search algorithm TabuCol (Hertz and Werra 1987) works in this solution space and uses this particular move operator. Other examples can be found in Galinier and Hao (1999), Johnson et al. (1991) and Lü and Hao (2010).

When considering a colouring $\mathcal{S}$ in the partial, proper solution space, the accompanying set of "uncoloured" vertices is defined as $U = V \backslash (\bigcup_{i=1}^{k} S_i)$, within which clashes are permitted. In this solution space, a suitable objective function is

$$f(\mathcal{S}) = |U| = |V| - \sum_{i=1}^{k} |S_i| \tag{2}$$

which is simply the number of "uncoloured" vertices. As with the previous objective function, $f(\mathcal{S}) = 0$ indicates that $\mathcal{S}$ is a feasible k-colouring.

A common strategy for moving from one colouring $\mathcal{S}$ to a neighbouring colouring $\mathcal{S}'$ within the partial, proper solution space, is to transfer an "uncoloured" vertex $v \in U$ to a colour class S_i and then transfer all vertices adjacent to v in S_i into U. The tabu search algorithm PartialCol (Blöchliger and Zufferey 2008), which is a modification of TabuCol, works in this solution space and uses this move operator.

A discussion of alternative solution spaces, objective functions and neighbourhood move operators for local search methods for the GCP can be found in Galinier and Hertz (2006).

4 Future adjacency

In this section we discuss what is meant by future adjacency with regards to edge dynamic graphs and introduce two neighbourhood operators that work in the feasible only solution space.

For a graph $G_t = (V, E_t) \in \mathcal{G}$, let $p_{t+1}(u, v)$ 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 $p_{t+1}(u, v)$.

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) in time-step $t + 1$ as

$$\mathcal{F}(\mathcal{S}_t) = \mathbb{E}(p_{t+1}) \cdot \sum_{i=1}^{k} \binom{|S_i|}{2} \tag{3}$$

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 Eq. (3) can be simplified to

$$\mathcal{F}(\mathcal{S}_t) = \mathbb{E}(p_{t+1}) \cdot \frac{n(n-k)}{2k}. \tag{4}$$

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

$$\mathcal{F}(\mathcal{S}_t) = \sum_{i=1}^{k} \sum_{u \in S_i} \sum_{\substack{v \in S_i \\ v \neq u}} p_{t+1}(u, v) \tag{5}$$

which is more useful. Vertices in $\mathcal{S}_t$ can now be "recoloured" in an attempt to reduce Eq. (5), 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.

4.1 Move operators that maintain feasibility

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 Sect. 3, neighbourhood move operators designed to work in the feasible only solution space are useful here. Outlined below are two suitable move operators.

4.1.1 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.

To illustrate, the graph in Fig. 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, e.g. $\textsc{Kempe}(v_1, 1, 2)$ can also be identified as $\textsc{Kempe}(v_3, 1, 2)$, $\textsc{Kempe}(v_6, 2, 1)$ or $\textsc{Kempe}(v_7, 2, 1)$.

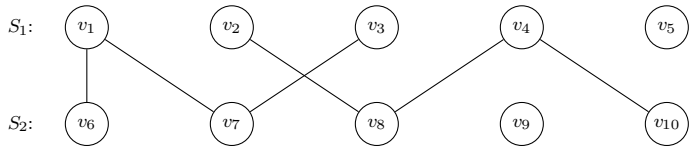

Fig. 2 A feasible 2-colouring with the vertices in colour class S_1 displayed in the *row above* those in colour class S_2

4.1.2 Pair swap

An additional move operator for the feasible only solution space is the pair swap. This operator 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. It should be noted that a pair swap is the simultaneous application of 2 Kempe-chain interchanges on $\text{KEMPE}(v, c(v), c(u))$ and $\text{KEMPE}(u, c(u), c(v))$ such that $|\text{KEMPE}(v, c(v), c(u))| = |\text{KEMPE}(u, c(u), c(v))| = 1$.

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. There is only one pair swap in Fig. 2 which transfers vertex v_5 from S_1 to S_2 and v_9 from S_2 to S_1.

5 Methods

The NP-hard nature of the static GCP (Garey and Johnson 1979), together with the fact that no approximation algorithm exists with an approximation ratio of less than two (Garey and Johnson 1976), means that heuristic methods are usually the go-to solution approach for finding colourings with good approximations of $\chi(G)$ for a given graph G. Previously, local search, evolutionary and hybrid heuristics have all been applied to solving the GCP, all of which have been shown to produce competitive results on publicly available benchmark instances.[1] Due to the time constraints we choose to enforce during our experimentation, we will focus on local search methods (Galinier and Hertz 2006) here, which are usually quicker at locating local optima (Lewis 2015).

The most common approach for solving the static GCP via heuristic methods is to solve a series of k-GCPs for G such that k is decreasing. The goal of a k-GCP is to determine whether a graph G can be feasibly coloured using k colours. In practice, an initial value for k can be calculated by a constructive operator that is guaranteed to produce a feasible colouring for G, such as the well known greedy (Welsh and Powell 1967) or DSATUR (Brélaz 1979) algorithms. 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 can be made, and so on, until some stopping criteria (e.g. a time or computational limit) is reached.

[1] These can be found at http://mat.gsia.cmu.edu/COLOR/instances.html.

For the edge dynamic GCP, we can replace the constructive operator with a modification operator that takes a feasible colouring $\mathcal{S}_t$ for G_t and modifies it for use as an initial (though not necessarily feasible) colouring $\mathcal{S}_{t+1}$ for G_{t+1}. From here, we can then attempt to change $\mathcal{S}_{t+1}$ into a feasible k-colouring for G_{t+1} where, at least for the initial feasible colouring, $k \geq |\mathcal{S}_t|$. Our strategy for the latter is shown in Algorithm 1.

Algorithm 1 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: $\mathcal{S}_{\text{best}} \leftarrow \emptyset$
2: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t$ modified in some way (see Sect. 5.1)
3: $k \leftarrow |\mathcal{S}_{t+1}|$
4: **while not** stopping criterion **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 computation limit is reached **then**
10: $k \leftarrow k + 1$
11: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{\text{best}}$
12: **return** $\mathcal{S}_{t+1}$

A problem arises, however, if $\mathcal{S}_t$ is modified into a k-colouring for G_{t+1} such that $k < \chi(G_{t+1})$. If this happens then it will be impossible to find a feasible k-colouring for G_{t+1}. To combat this, we allow the value of k to be increased as shown on Lines 9 and 10 of Algorithm 1.

As mentioned previously, one of the main objectives of this research is to explore the different methods for modifying a feasible colouring for time-step t into an initial colouring for time-step $t + 1$ (i.e. Line 2 of Algorithm 1). The essential question to be answered is: can a feasible colouring for one graph G_t be used in some advantageous way to help find a feasible colouring for a similar but currently unknown graph G_{t+1}?

At this point it is worth noting that the edge dynamic GCP has previously been considered by Preuveneers and Berbers (2004) who proposed an agent-based approach for "repairing" colourings between time-steps; however, their method differs from our approach as it is only concerned with the quality of the initial colourings achieved. As such, it does not include any local optimisation between time-steps.

5.1 Modification operators

The final feasible colouring $\mathcal{S}_t$ for G_t is likely to be a complete, improper colouring for G_{t+1}. This holds because every vertex $v \in V$ will be coloured but the new edges E^+_{t+1} could potentially cause clashes where none existed previously. With this knowledge, we propose applying one of the following modification methods to find an initial (but not necessarily feasible) colouring $\mathcal{S}_{t+1}$ for G_{t+1}. The effectiveness of these modification operators will be explored and assessed in Sect. 7.

Method 0 (*reset*): Ignore $\mathcal{S}_t$ completely and use a constructive operator to produce an initial colouring $\mathcal{S}_{t+1}$. This is the approach used for G_0 and can be used for a base-line comparison against the remaining methods for time-steps $t = 1, \ldots, T$.

Method 1 (*calculateClashes*): Simply set $\mathcal{S}_{t+1} = \mathcal{S}_t$, calculate the number of clashes, and then pass $\mathcal{S}_{t+1}$ directly to a tabu search operator that operates in the complete, improper solution space and attempts to remove all clashes to achieve a feasible k-colouring for G_{t+1} (where $k = |\mathcal{S}_t|$).

Method 2 (*uncolourClashes*): Identify pairs of clashing vertices in $\mathcal{S}_t$ and then transfer one vertex from each of these pairs to a set of "uncoloured" vertices U. The resultant colouring $\mathcal{S}_{t+1}$ is a partial, proper colouring for G_{t+1} which, along with U, can then be passed to a tabu search operator that operates in the partial, proper solution space and attempts to feasibly colour all "uncoloured" vertices, to achieve a feasible k-colouring for G_{t+1} (where $k = |\mathcal{S}_t|$).

Method 3 (*solveClashes*): First apply Method 2, to obtain a partial, proper colouring $\mathcal{S}_{t+1}$ for G_{t+1} and a set of "uncoloured" vertices U. Then attempt to re-insert each "uncoloured" vertex $v \in U$ into a colour class of $\mathcal{S}_{t+1}$ such that no clashes are incurred. The residual graph $\tilde{G}$ induced by the remaining "uncoloured" vertices in U is then passed to a constructive operator which produces a feasible $\tilde{k}$-colouring $\tilde{\mathcal{S}}$ for $\tilde{G}$. Finally, combine $\mathcal{S}_{t+1}$ and $\tilde{\mathcal{S}}$ to produce a feasible colouring for G_{t+1} with $k + \tilde{k}$ colour classes where $k = |\mathcal{S}_t|$ and $\tilde{k} = |\tilde{\mathcal{S}}|$.

The details regarding the constructive and tabu search operators used within these modification operators are discussed in Sect. 6.2.

5.2 Future adjacency reduction

As noted in Sect. 4, 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 Eq. (5). The approach outlined in Algorithm 2 is a two-stage approach that first attempts to find a feasible colouring for the current time-step with a target number of colour classes (Lines 1–10). It then attempts to reduce the estimated number of clashes in the following time-step (Lines 11–16).

Stage 1 (Lines 1–10) of this approach is almost identical to Algorithm 1 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, identifies and executes the best Kempe-chain interchange or pair swap, and then makes all inverse moves "tabu" for a given number of subsequent iterations (see Sect. 6.2 for specific parameter settings). With regards to the example given in Sect. 4.1 for Fig. 2, the inverse moves of the Kempe-chain interchange applied to KEMPE$(v_1, 1, 2)$ is the Kempe-chain interchange applied to KEMPE$(v_1, 2, 1)$, KEMPE$(v_3, 2, 1)$, KEMPE$(v_6, 1, 2)$ or KEMPE$(v_7, 1, 2)$.

Algorithm 2 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 adjacency matrix P_{t+2}
Output: A feasible colouring $\mathcal{S}_{t+1}$ for G_{t+1} such that $|\mathcal{S}_{t+1}| \geq k^\star$
$\mathcal{S}_{\text{best}} \leftarrow \emptyset$
$\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t$ modified in some way (see Sect. 5.1)
$k \leftarrow |\mathcal{S}_{t+1}|$
while not stopping criterion **and** $|\mathcal{S}_{\text{best}}| \neq k^\star$ **do**
  Attempt to make $\mathcal{S}_{t+1}$ a feasible k-colouring for G_{t+1}
  if $\mathcal{S}_{t+1}$ is a feasible k-colouring for G_{t+1} **then**
    $\mathcal{S}_{\text{best}} \leftarrow \mathcal{S}_{t+1}$
    $k \leftarrow k - 1$
  if $\mathcal{S}_{\text{best}} = \emptyset$ **and** a computational limit is reached **then**
    $k \leftarrow k + 1$
$\mathcal{F}_{\text{best}} \leftarrow \mathcal{F}(\mathcal{S}_{\text{best}})$ (see Eq. (5) in Sect. 4)
while not stopping criterion **do**
  Attempt to reduce $\mathcal{F}(\mathcal{S}_{t+1})$
  if $\mathcal{F}(\mathcal{S}_{t+1}) < \mathcal{F}_{\text{best}}$ **then**
    $\mathcal{S}_{\text{best}} \leftarrow \mathcal{S}_{t+1}$
    $\mathcal{F}_{\text{best}} \leftarrow \mathcal{F}(\mathcal{S}_{t+1})$
$\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{\text{best}}$
return $\mathcal{S}_{t+1}$

6 Experimentation details

6.1 Test instances

In our experiments we consider 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. In our case we use $n = 500$,[2] $d \in \{0.1, 0.5, 0.9\}$, $p \in \{0.005, 0.01, \ldots, 0.05\}$ 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 Sect. 5 and Footnote 1). 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 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 $\mathcal{E} \setminus E_t$ 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 $\mathcal{E} \setminus E_t$ 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.

[2] Trials were also conducted on test instances with $n = 250$ and 1000 and the relationships observed between the different modification operators were similar regardless of the value of n. To reduce the volume of results presented, we have opted to present results regarding test instances with $n = 500$ only.

6.1.1 Legal parameter combinations

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 $|E_t| \approx d|\mathcal{E}|$ for all t and the above equation 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 $\mathcal{E} \setminus E_t$ 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 clearly illustrates the legal combinations of p and d (represented by the shaded area).

6.2 Algorithm parameters

For our experiments we used DSATUR (Brélaz 1979) as our constructive operator. In the case of G_0, this is used to find an initial colouring (i.e. DSATUR replaces Line

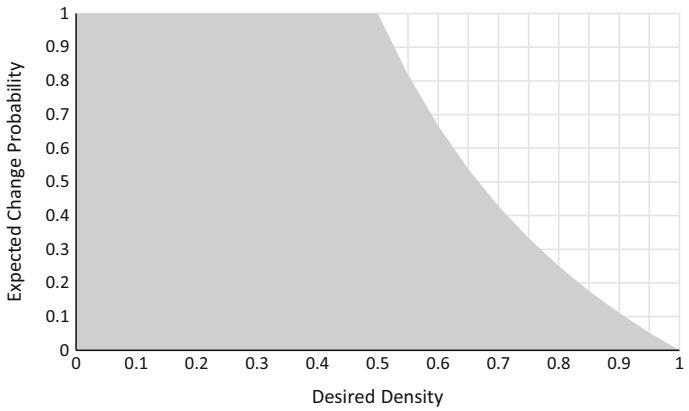

Fig. 3 Legal combination of expected change probability p and desired density d that maintain approximately equal density of an edge dynamic graph over all time-steps

2 in Algorithms 1 and 2 for G_0). We used a time limit of 10 seconds[3] per time-step (i.e. Line 4 in Algorithm 1, and Lines 4 and 12 in Algorithm 2). If this time limit had been set much longer, say hours, then the advantage of modifying colourings between time-steps obviously diminishes.

As mentioned in Sect. 5, local search heuristics lend themselves more appropriately to this short time limit in comparison to more elaborate evolutionary or hybrid methods. Therefore, we exclusively use tabu search operators to solve each k-GCP (i.e. Line 5 in Algorithms 1 and 2). More specifically, we use the TabuCol (Hertz and Werra 1987) and PartialCol (Blöchliger and Zufferey 2008) algorithms in the complete, improper and partial, proper solution spaces, respectively. These algorithms use the neighbourhood move operators and objective functions outlined in Sect. 3.

At each iteration of these algorithms, the best "non-tabu" move is selected and executed. The inverse moves are then made "tabu" for $0.6 \times f(\mathcal{S}') + r$ iterations, where f is the objective function given in Eqs. (1) and (2) respectively, $\mathcal{S}'$ is the resultant colouring after the neighbourhood move, and r is a random integer from the set $\{0, 1, \ldots, 9\}$ (this tabu tenure is recommended in Blöchliger and Zufferey 2008; Hertz and Werra 1987).

We implement TabuCol such that only the moves that involve currently clashing vertices are considered. By doing so, the algorithm only examines the problematic region of the neighbourhood of all possible moves. By design, PartialCol acts in a similar manner by only considering the moves that involve currently "uncoloured" vertices.

Our implementations of TabuCol and PartialCol also include an aspiration criterion that allows "tabu" moves to be selected and executed if they lead to a colouring which has fewer clashes or "uncoloured" vertices, respectively, than the best colouring observed up until that iteration.

During execution, the target number of colour classes 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 Algorithms 1 and 2). This adjustment is particularly useful if the initial value of k satisfies $k < \chi(G_{t+1})$ (see Sect. 5).

Note that Method 1 operates exclusively in the complete, improper solution space, and Method 2 operates exclusively in the partial, proper solution space. On the other hand, Methods 0 and 3 can operate in either solution space as required. Because of this, only comparisons between methods operating in the same solution space will be compared (i.e. Methods 1 and 2 will not be compared against one another).

The tabu search operator described in Sect. 5.2 for reducing $\mathcal{F}(\mathcal{S}_t)$ (i.e. Line 13 of Algorithm 2) uses a tabu tenure of $\frac{|V|}{2}$ iterations, which for our test instances is 250. When applying Algorithm 2 to test instances with $d = 0.1, 0.5$ and 0.9, we have found that appropriate values of $k^\star$ lie in the ranges of 12–18, 48–70 and 125–170

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

respectively. If the initial, feasible colouring achieved has less than $k^\star$ colour classes then empty colour classes are added until $k^\star$ is reached.

7 Results

In this section we present 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 statistical comparisons are based on the Wilcoxon signed rank test with significance level $\alpha = 0.01$.

7.1 Without future adjacency information

We first consider the situation where we have no information regarding the likelihood of changes between time-steps (i.e. changes to the edge set occur at random). In this case our goal is to use the available time limit to find feasible colourings for each $G_t \in \mathcal{G}$ with the fewest number of colour classes using the approach described in Algorithm 1.

7.1.1 Initial, feasible colourings

We start by comparing the initial, feasible colourings achieved when applying different modification operators. For all values of d and p tested, Methods 1 and 2 were found to achieve initial, feasible colourings with significantly fewer colour classes than Methods 0 and 3 but there is a trade off as they also required significantly more time to do so. These observations can be seen in Fig. 4 and Table 1, respectively.

A main contributing factor to the time difference may be found in the nature of the different methods: Methods 0 and 3 both start from feasible colourings, whereas Methods 1 and 2 do not and therefore require more time to move to a feasible region of the solution space. In fact, observations in Table 1 with values greater than 5 seconds indicate that, for at least half of the test instances, with the associated parameters settings the initial value of k has been increased at least once.

The number of colour classes in the initial, feasible colourings achieved by Method 3 was found to be significantly and positively correlated to p for all values of d tested. This leads to observations in which Method 3 achieves initial, feasible colourings with both significantly fewer and significantly more colour classes than Method 0 (see Fig. 4).

Considering computational effort, we found that the time required by Method 3 to achieve initial, feasible colourings was significantly less compared to Method 0 for all values of d and p. Both Methods 0 and 3 employ the same constructive operator (DSATUR); however, Method 0 applies it to the whole graph G_t at each time-step t, whereas Method 3 only applies it to a residual graph of G_t, which is subsequently "smaller" than G_t. We conclude, therefore, that applying Method 3 to dynamic graphs with low values of p is advantageous with regards to both the number of colour classes in the initial, feasible colourings achieved and the time required to obtain them.

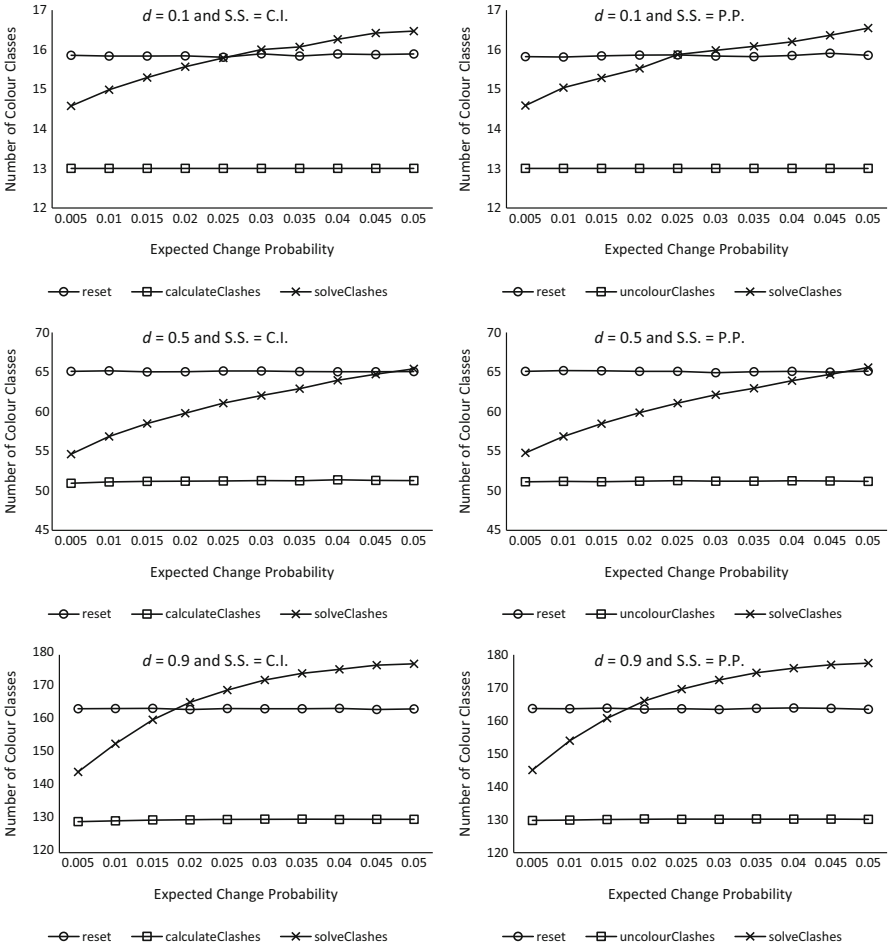

Fig. 4 Mean number of colour classes in initial, feasible colourings (*LHS* represent methods in the complete, improper solution space, *RHS* represent methods in the partial, proper solution space, and *rows from top to bottom* represent graphs with $d = 0.1$, 0.5 and 0.9, respectively)

7.1.2 Final, feasible colourings

We now look at the final (or "best"), feasible colourings achieved after applying different modification operators. 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 $d = 0.1$. This continues to hold for Methods 0, 2 and 3 operating in the partial, proper solution space on test instances with $d = 0.5$.

In comparison to Method 0, Method 1 was found to produce final, feasible colourings with significantly fewer colour classes, for $d = 0.5$ with small values of p. For small values of p, it is likely that an algorithm that uses Method 0 will require more

Table 1 Median time (in seconds) required to obtain an initial, feasible colouring

d	M.	p									
		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	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	0*	0*	0*	0*	0*	0*	0*	0*	0*	0*

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

time to achieve a feasible colouring with the same number of colour classes as the initial, feasible colourings achieved by Method 1, which is often also the final, feasible colouring.

On the other hand, Methods 1 and 2 achieve final, feasible colourings with significantly more colour classes than Method 0 for $d = 0.9$ with some values of p. This observation is likely due to the relatively large amount of time required by Methods 1 and 2 to find an initial, feasible colouring compared to Method 0 for $d = 0.9$ with all values of p (see Table 1). This "wasted" time then translates to time not being allocated to finding feasible colourings with fewer colour classes. For the same reasons, Method 3 was observed to achieve final, feasible colourings with significantly fewer colour classes than those achieved when using Methods 0 to 2 for $d \in \{0.5, 0.9\}$ with some values of p.

The following time comparisons correspond to trials where employing the corresponding modification operators achieved final, feasible colourings with an equal number of colour classes. These results are displayed in Table 2.

When using Methods 1 and 2, the time required to achieve a final, feasible colouring was found to be significantly less for $d = 0.1$ with all values of p compared to Method 0. Both of these methods were also able to reach final, feasible colourings significantly faster than Method 3 for $d = 0.1$ with some values of p. These observations are again likely due to the fact that the initial, feasible colourings achieved by Methods 1 and 2 are also the final, feasible colourings achieved for $d \in \{0.1, 0.5\}$ with low values of p. The opposite was found to hold for $d = 0.9$ and most values of p. This is probably the result of the "wasted" time mentioned previously with regards to finding an initial, feasible solution for $d \in \{0.5, 0.9\}$ and high values of p.

Unlike Methods 1 and 2, Method 3 did not require significantly more time than Method 0 for any combination of d and p to achieve final, feasible colourings. Moreover, for $d = 0.1$ with all values of p, and $d = 0.5$ with some low values of p, it required significantly less time. It should be highlighted that for low values of p,

Table 2 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

d	S.S.	M.	p									
			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
		3	0.874*	1.622	2.348	2.325	3.526	2.801	2.987	3.151	3.166	3.105
	P.P.	0	3.136	3.198	2.855	2.668	2.996	2.800	2.481	2.980	3.237	2.933
		2	2.247	2.200	2.589	3.299	3.869	3.121	2.402	2.964	3.620	2.823
		3	1.669	2.324	2.324	2.207*	2.746	2.714	2.901	2.652	2.769	2.948
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
		3	6.224	3.993*	4.711	5.507	4.150	5.616	6.139	5.070	5.132	4.851
	P.P.	0	4.274	4.851	6.100	5.218	5.117	3.947	4.532	4.095	4.226	4.789
		2	4.852	5.460	5.257	5.640	5.882	6.380	6.217	5.881	6.396	6.021
		3	4.259	4.477	4.337	5.117	5.226	3.885	4.228	4.454	5.070	4.789

As with Table 1, results for test instances with $d = 0.1$ are omitted as they were deemed to be insufficiently accurate to present

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

Method 3 is able to produce initial, feasible colourings with significantly fewer colour classes than Method 0 and requires less time to do so. This has a knock-on effect which allows the algorithm to attempt to find feasible colourings with fewer colour classes from an earlier point in the allotted time limit.

7.2 With probabilistic future adjacency information

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 2 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 are not be explored, because altering the value of $k^\star$ will naturally cap the number of colour classes in these colourings.

7.2.1 Initial clashes

The secondary objective of the approach outlined in Algorithm 2 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

Table 3 Significant differences between the number of clashes at the start of each time-step when using modification operator 3 (*solveClashes*) within our approach (Algorithm 2) compared against an algorithm without any secondary optimisation on test instances with $d = 0.1$

$k^\star$	p									
	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" indicate that our approach achieves colourings with significantly fewer clashes, and "–" indicates no significant difference

algorithm that does not include a secondary optimisation stage (i.e. against Algorithm 2 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 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, the values of p decrease for which we first observe these significant reductions.

For example, consider test instances with $d = 0.1$ where it is likely that $\chi(G_t) \approx 12$ or 13 for all $t \in \{0, 1, \ldots, T\}$ (illustrated in Table 3). When using Method 3 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 Fig. 5 where the two lines (corresponding to the omission and inclusion of secondary optimisation) become further apart as the value of $k^\star$ increases.

7.2.2 Initial, feasible colourings

Here, we assume that if a colouring $\mathcal{S}_t$ has fewer clashes for the following time-step then our modification operators should require less time to achieve 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 and 2 for most instances with $d = 0.9, k^\star \geq 143$ and $p \geq 0.01$ and 0.02, respectively. For these test instances, we also observed a high level of reduction with regards to

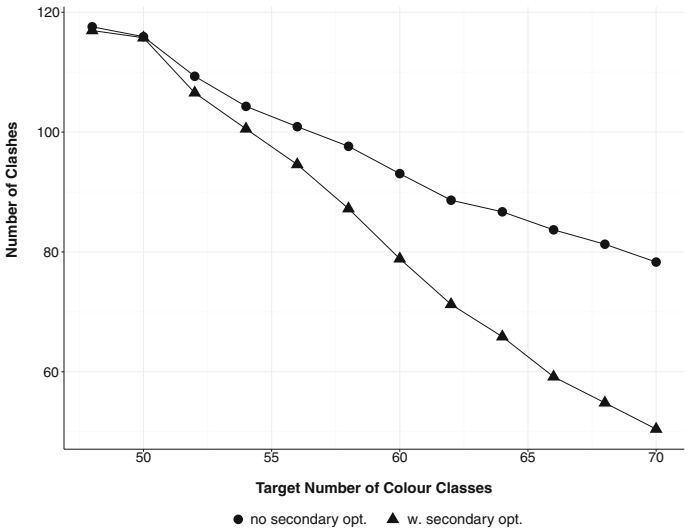

Fig. 5 Mean number of clashes in $\mathcal{S}_t$ for time-step $t+1$ against the target number of colour classes $k^\star$ for test instances with $d = 0.5$ and $p = 0.05$ whilst applying modification operator 1 (*calculateClashes*)

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 and 2.

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. Therefore, our secondary optimisation when used in conjunction with Methods 1 and 2 can produce initial, feasible colourings with fewer colour classes and require less time to do so in comparison to Method 0 for low values of $k^\star$ and p.

For Method 3 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 3 requires significantly less time to achieve initial, feasible colourings for the same reasons outlined in Sect. 7.1.

With regards to the number of colour class in the initial, feasible colourings achieved by Method 3, any significant differences when compared against omitting 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 (an example of which is illustrated in Fig. 6). 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

Table 4 Significant differences between the time required to achieve an initial, feasible colouring when using modification operator 1 (*calculateClashes*) within our approach (Algorithm 2) compared against Method 0 on test instances with $d = 0.9$

$k^\star$	p									
	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" indicate that our approach requires significantly less time, "O" indicates the opposite, and "–" indicates no significant difference

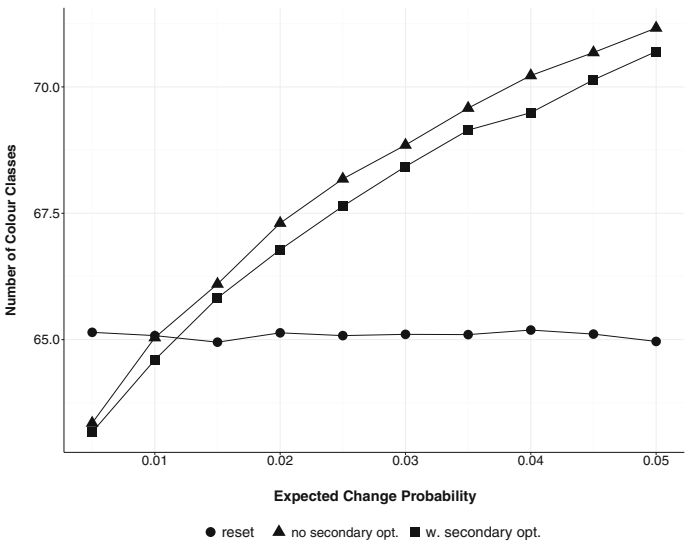

Fig. 6 Mean number of colour classes in initial, feasible colourings for test instances with $d = 0.5$ and $k^\star = 60$ whilst applying modification operator 3 (*solveClashes*) in the complete, improper solution space

Table 5 Significant differences between the number of colour classes in the initial, feasible colourings achieved when using modification operator 3 (*solveClashes*) within our approach (Algorithm 2) compared against Method 0 on test instances with $d = 0.5$

$k^\star$	p									
	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" indicate that our approach achieves colourings with significantly fewer colour classes, "O" indicates the opposite, and "–" indicates no significant difference

is more likely to find an initial, feasible k-colouring for G such that $k < k^\star$ and, therefore, a number of empty colour classes will be added to this colouring (see Sect. 6.2). During secondary optimisation, it is likely that vertices are being 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 5.

8 Conclusions and future work

This paper has introduced a number of methods for modifying colourings for graphs whose edge sets are subject to change over time. We have also introduced a tabu search method 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, thus producing more robust colourings.

Our experiments have shown that, for edge dynamic graphs without future adjacency information, initial colourings with significantly fewer colour classes can be achieved by using Methods 1 and 2 (*calculateClashes* and *uncolourClashes*, respectively), which both 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. These operators were also found to achieve final, feasible colourings with both significantly fewer or more colour classes depending on p. The time required to achieve comparable final colourings via these methods also appears to be dependent on both d and p.

It has also been shown, again without future adjacency information, that Method 3 (*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, this modification operator also results in final, feasible colourings with the same or significantly fewer colour classes and requires significantly less time to do so for low values of d and p.

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 high 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 previous conclusions against Method 0 (*reset*) without future adjacency information continue to hold here but become dependent on the value of $k^\star$ also, with the strength of the statements diminishing as the value of $k^\star$ increases.

Future issues of interest may include the addition of a cost associated with altering the colour of a vertex between time-steps, in a similar fashion to the dynamic frequency assignment problem in Dupont et al. (2009). If a colouring is no longer feasible in the subsequent time-step, then it is desirable to achieve feasibility by "recolouring" as few vertices as possible. We hypothesise that reducing the estimated number of clashes in the following time-step will also reduce the number of colour changes required between time-steps.

Moving away from the edge dynamic GCP, some of our previous work has introduced and explored the effects of modification operators for the vertex dynamic GCP without future change information (see Hardy et al. 2016). We plan to extend this work, as we have done here, to explore the situation where information regarding the likelihood of future changes is provided. More specifically, we wish to investigate whether this information can again be used in some advantageous way.

The main foreseeable problem with such work is the difficulty in formulating a "future cost" function analogous to Eq. (5) for the vertex dynamic problem. When a new vertex is introduced, it is not determined beforehand which colour class it will be placed in, which makes the question "how likely is it that we can feasibly colour the new vertex?" a very difficult one to answer in practice.

References

Aardal, K.I., Van Hoesel, S.P., Koster, A.M., Mannino, C., Sassano, A.: Models and solution techniques for frequency assignment problems. Ann. Oper. Res. **153**(1), 79–129 (2007)

Blöchliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Comput. Oper. Res. **35**(3), 960–975 (2008)

Brélaz, D.: New methods to color the vertices of a graph. Commun. ACM **22**(4), 251–256 (1979)

Chaitin, G.J.: Register allocation & spilling via graph coloring. ACM Sigplan Not. **17**(6), 98–101 (1982)

Costa, D.: An evolutionary tabu search algorithm and the NHL scheduling problem. INFOR Inf. Syst. Oper. Res. **33**(3), 161–178 (1995)

Dupont, A., Linhares, A.C., Artigues, C., Feillet, D., Michelon, P., Vasquez, M.: The dynamic frequency assignment problem. Eur. J. Oper. Res. **195**(1), 75–88 (2009)

Erben, W.: A grouping genetic algorithm for graph colouring and exam timetabling. In: Burke, E., Erben, W. (eds.) Practice and Theory of Automated Timetabling III. PATAT. Lecture Notes in Computer Science, vol 2079. Springer, Berlin, Heidelberg (2000)

Galinier, P., Hao, J.K.: Hybrid evolutionary algorithms for graph coloring. J. Comb. Optim. **3**(4), 379–397 (1999)

Galinier, P., Hertz, A.: A survey of local search methods for graph coloring. Comput. Oper. Res. **33**(9), 2547–2562 (2006)

Garey, M.R., Johnson, D.S.: The complexity of near-optimal graph coloring. JACM **23**(1), 43–49 (1976)

Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman & Co., San Francisco (1979)

Gyárfás, A., Lehel, J.: On-line and first fit colorings of graphs. J. Graph Theory **12**(2), 217–227 (1988)

Harary, F., Gupta, G.: Dynamic graph models. Math. Comput. Model. **25**(7), 79–87 (1997)

Hardy, B., Lewis, R., Thompson, J.: Modifying colourings between time-steps to tackle changes in dynamic random graphs. In: Chicano, F., Hu, B., García-Sánchez, P. (eds.) Evolutionary Computation in Combinatorial Optimization. EvoCOP. Lecture Notes in Computer Science, vol 9595. Springer, Cham (2016)

Hell, P., Nešetril, J.: Graphs and Homomorphisms (Volume 28 of Oxford Lecture Series in Mathematics and its Applications). Oxford University Press, Oxford (2004)

Hertz, A., Plumettaz, M., Zufferey, N.: Variable space search for graph coloring. Discrete Appl. Math. **156**(13), 2551–2560 (2008)

Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing **39**(4), 345–351 (1987)

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Oper. Res. **39**(3), 378–406 (1991)

Lewis, R.: A Guide to Graph Colouring. Springer, Berlin (2015)

Lewis, R., Carroll, F.: Creating seating plans: a practical application. J. Oper. Res. Soc. **67**(11), 1353–1362 (2016)

Lewis, R., Thompson, J., Mumford, C., Gillard, J.: A wide-ranging computational comparison of high-performance graph colouring algorithms. Comput. Oper. Res. **39**(9), 1933–1950 (2012)

Lovász, L., Saks, M., Trotter, W.T.: An on-line graph coloring algorithm with sublinear performance ratio. Ann. Discret. Math. **43**, 319–325 (1989)

Lü, Z., Hao, J.K.: A memetic algorithm for graph coloring. Eur. J. Oper. Res. **203**(1), 241–250 (2010)

Preuveneers, D., Berbers, Y.: ACODYGRA: an agent algorithm for coloring dynamic graphs. Symb. Numer. Algorithms Sci. Comput. **6**, 381–390 (2004)

Qu, R., Burke, E.K., McCollum, B.: Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. Eur. J. Oper. Res. **198**(2), 392–404 (2009)

Tantipathananandh, C., Berger-Wolf, T., 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, pp. 717–726. ACM (2007)

Welsh, D.J., Powell, M.B.: An upper bound for the chromatic number of a graph and its application to timetabling problems. Comput. J. **10**(1), 85–86 (1967)