Modifying Colourings between Time-steps to Tackle Changes in Dynamic Random Graphs

Bradley Hardy, Rhyd Lewis and Jonathan Thompson

Cardiff University, School of Mathematics
Cardiff CF24 4AG, UK
{hardyB,lewisR9,thompsonJM1}@cardiff.ac.uk

Abstract Many real world operational research problems can be formulated as graph colouring problems. Algorithms for this problem usually operate under the assumption that the size and constraints of a problem are fixed, allowing us to model the problem using a static graph. For many problems however, this is not the case and it would be more appropriate to model such problems using dynamic graphs. In this paper we will explore whether feasible colourings for one graph at time-step t can be modified into a colouring for a similar graph at time-step $t + 1$ in some beneficial manner.

Keywords: Graph Colouring; Dynamic Graphs; Heuristics

1 Introduction

The graph colouring problem (GCP) aims to colour each vertex of a graph $G = (V, E)$ such that no adjacent vertices have the same colour 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. One example is frequency assignment [1] where each geographical site is represented by a vertex, an edge exists between two vertices if their respective sites are within a certain proximity of one another, and colours represent communication frequencies (e. g. radio frequencies). Other examples include exam timetabling [5,15], register allocation [3], designing seating plans [11] and grouping people in social networks [16].

Most GCP methods can only be applied to such problems under the assumption that the size and constraints of a 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 [4] this is not always appropriate as sites can be added or removed from the communication network, or the location of sites can themselves move. The aim of this particular research, therefore, is to explore graph colouring on dynamic graphs. More specifically, we wish to look at methods which modify a feasible colouring for one graph into a colouring for a "similar" graph.

The rest of the paper will be structured as follows: Section 2 will formerly define dynamic graphs and their associated problems and Section 3 will then discuss the various search spaces for graph colouring problems. Section 4 will then outline a general approach and define the different modification methods used, Section 5 will contain the experimentation details, and in Section 6 we will present the results. Finally, Section 7 will summarise the findings of the experiments and discuss future work.

2 Dynamic Graph Colouring Problems

The importance of studying dynamic graphs and their associated problems has been highlighted by Harary and Gupta [7] who outlined many applications, especially in the area of computer science, and postulated that techniques applied to static graphs should be extended for dynamic graphs. However, there has been very little research regarding methods designed explicitly for dynamic graphs.

Two methods for finding colourings for dynamic graphs are given in [13] and [14]. The first of these proposes a genetic algorithm that uses the same population of colourings between time-steps for vertex dynamic graphs and the second proposes an agent-based approach for repairing colourings between time-steps for edge dynamic graphs. Both of these methods are only concerned with the quality of initial colourings, whereas this research will presume that optimisation can take place between time-steps.

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 graph defined for time-step $t \in \{0, 1, \ldots, T\}$. At every time-step, the objective in analogous to the static GCP. In terms of methodology, this means using heuristic methods to find a feasible k_t-colouring for each time-step t, where k_t is a good approximation of $\chi(G_t)$. Objectively, this is an attempt to minimise $\sum_{t=0}^{T} k_t$.

In this work we choose to split the concept of dynamic graphs into two cases: edge dynamic graphs and vertex dynamic graphs. In the edge dynamic graph colouring problem, changes can only occur on the edge set E_t; therefore $V_0 = V_1 = \ldots = V_T = V$ for all time-steps. For 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 must 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 possible edges between vertices in V. The edge set for time-step $t + 1$ is then defined as $E_{t+1} = (E_t \backslash E^-_{t+1}) \cup E^+_{t+1}$.

In the vertex dynamic graph colouring problem, 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 vertices will also require the addition of new edges unless the new vertex is intended to be isolated. For a vertex dynamic graph $\mathcal{G}$, consider the graph $G_t = (V_t, E_t)$ for 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}. Once these are defined, the set of deleted edges $E^-_{t+1} \subseteq E_t$ is defined to be the set of all edges incident to the deleted vertices (i. e. E^-_{t+1} contains all the edges $\{u, v\} \in E_t$ such that either $u \in V^-_{t+1}$ or $v \in V^-_{t+1}$). The set of new edges E^+_{t+1} is a set of connecting edges from the set of new vertices to any of the vertices in V_{t+1} (i. e. E^+_{t+1} contains edges $\{u, v\} \in \mathcal{E}_{t+1}$ where $\mathcal{E}_{t+1}$ is the set of all possible edges between vertices in V_{t+1} and either $u \in V^+_{t+1}$ or $v \in V^+_{t+1}$). The vertex and edge sets for time-step $t + 1$ are then defined as $V_{t+1} = (V_t \backslash V^-_{t+1}) \cup V^+_{t+1}$ and $E_{t+1} = (E_t \backslash E^-_{t+1}) \cup E^+_{t+1}$ respectively.

In fact, edge dynamic graphs can be considered as a special case of vertex dynamic graphs where $|V^-_t| = |V^+_t| = |V_{t-1}|$ and $E^-_t = E_{t-1}$, $\forall t \in \{1, \ldots, T\}$. Another special case is on-line graph colouring, where exactly one vertex is added at each time-step (i. e. $V^-_t = \emptyset$ and $|V^+_t| = 1$, $\forall t \in \{1, \ldots, T\}$). On-line graph colouring has the additional constraint that, once coloured, a vertex cannot be transferred to a different colour class. Research concerning on-line graph colouring mainly consists of worst case behaviour analysis of algorithms [6,12].

3 Search Spaces of the GCP

In this paper we will approach dynamic graph colouring problems by adapting methods for the static problem. In general, the literature suggest three main search spaces for the static GCP: (i) *feasible colourings only*, where every vertex is coloured, there are no clashes (i. e. all adjacent vertices are coloured differently) and the number of colour classes is allowed to vary; (ii) *complete, improper colourings*, where every vertex is coloured but clashes are permitted; and (iii) *partial, proper colourings*, where no clashes occur but there may be "uncoloured" vertices.

The search space of feasible colourings only is rarely used in the literature as it is often difficult to determine which of two k-colourings is closer to becoming a colouring with $k - 1$ colour classes. One example of a heuristic method in this search space is a simulated annealing approach outlined in [9].

In the complete, improper search space a colouring $\mathcal{S} = \{S_1, \ldots, S_k\}$ is a partition of V into k disjoint subsets (i. e. $V = \bigcup_{i=1}^{k} S_i$ and $S_i \cap S_j = \emptyset$, $\forall i, j \in \{1, \ldots, k\}$ and $i \neq j$). S_i is called the ith colour class of the colouring $\mathcal{S}$ and the colouring function $c : V \to \{1, \ldots, k\}$ is defined such that $c(v) = i$ for all $v \in S_i$. One well-known algorithm that operates in this search space is TabuCol [8]. In this algorithm, to move from one colouring $\mathcal{S}$ to a neighbouring colouring $\mathcal{S}'$, a vertex v is transferred from its current colour class S_i to a different colour class S_j where $i \neq j$. Then $\mathcal{S}$ becomes $\mathcal{S}' = \{S'_1, \ldots, S'_k\}$ with $S'_i = S_i \backslash \{v\}$, $S'_j = S_j \cup \{v\}$ and $S'_l = S_l$, $\forall l \in \{1, \ldots, k\} \backslash \{i, j\}$. The vertex v to be moved can also be chosen exclusively from the set of currently clashing vertices (i. e. we can move $v \in S_i$ if and only if $\exists u \in S_i$ such that $u \neq v$ and $\{u, v\} \in E$). For a given solution $\mathcal{S}$, the associated cost function in this algorithm is given by

$$f(\mathcal{S}) = \sum_{i=1}^{k} |E_{(i)}| \qquad (1)$$

where $E_{(i)}$ is the set of edges with both end points in S_i. This cost function is equivalent to the number clashes in the colouring. If $f(\mathcal{S}) = 0$ then the colouring $\mathcal{S}$ has no clashes and is therefore a feasible k-colouring.

In the partial, proper search space a colouring $\mathcal{S} = \{S_1, \ldots, S_k, S_{k+1}\}$ is defined by a partition of V into $k+1$ disjoint subsets. The first k subsets are independent sets (i. e. $E_{(i)} = \emptyset$, $\forall i \in \{1, \ldots, k\}$) and the remaining vertices $v \in V \backslash (\bigcup_{i=1}^{k} S_i)$ are placed in the additional subset S_{k+1} of "uncoloured" vertices, in which clashes are also permitted.

PartialCol [2] (a modification of TabuCol) is an example of an algorithm that operates in this search space. In this algorithm, to move from one colouring $\mathcal{S}$ to a neighbouring colouring $\mathcal{S}'$, we transfer an uncoloured vertex $v \in S_{k+1}$ to a colour class S_i where $i \leq k$ and move the set of vertices adjacent to v, $U_i \subseteq S_i$, to S_{k+1}. Then $\mathcal{S}$ becomes $\mathcal{S}' = \{S'_1, \ldots, S'_k, S'_{k+1}\}$ with $S'_i = (S_i \backslash U_i) \cup \{v\}$, $S'_{k+1} = (S_{k+1} \cup U_i) \backslash \{v\}$ and $S'_l = S_l$, $\forall l \in \{1, \ldots, k\} \backslash \{i\}$.

For a given solution $\mathcal{S}$, the associated cost function in this algorithm is given by

$$f(\mathcal{S}) = |S_{k+1}| \tag{2}$$

which is equivalent to the number of uncoloured vertices. An alternative cost function is

$$f(\mathcal{S}) = \sum_{v \in S_{k+1}} \deg(v) \tag{3}$$

where $\deg(v)$ is the degree of vertex v. If the vertices in S_{k+1} have low degrees then, in theory, they will be easier to move into colour classes without causing clashes. For both of these cost functions, if $f(\mathcal{S}) = 0$ then there are no uncoloured vertices and $\mathcal{S}$ is therefore a feasible k-colouring.

4 Methods

Our approach for solving a dynamic graph $\mathcal{G} = \{G_0, G_1, \ldots, G_T\}$ will follow the process outlined in Algorithm 1. Notice that for G_0 a method for the static GCP needs to be applied.

Algorithm 1 Generic DGCP 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 colourings for $G_t \in \mathcal{G}$
1: $\mathcal{S}_0 \leftarrow$ Static GCP Algorithm ($G_0$)
2: for $t = 1$ to $T$ do
3:     $\mathcal{S}_t \leftarrow$ Dynamic GCP Time-step Algorithm ($G_t, \mathcal{S}_{t-1}$) (i. e. Algorithm 2)
4: return $\mathbf{S} = \{\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_T\}$
```

For each time-step t, suppose a feasible colouring $\mathcal{S}_t$ for G_t has been found; that is, a colouring where all vertices are coloured and no clashes occur. This colouring might then be saved and possibly modified in some way to be used as a colouring for G_{t+1}. Using this modified colouring with $k \geq |\mathcal{S}_t|$ colour classes as a starting point, we then wish to find a feasible k-colouring for G_{t+1}. If we succeed, then we search for a feasible colouring with one fewer colour class and so on until some stopping criteria (e.g. a time or iteration limit) is reached. It may of course be impossible to find a feasible k-colouring for G_{t+1}. In order to accommodate this eventuality, if some timing criteria is met and a feasible colouring (of any size) has not be found, then we increase k by 1, we allow the target number of colour classes to be increased indefinitely until a feasible colouring is found or the algorithm's stopping criteria is met. This process is outlined in Algorithm 2.

The focus of this particular piece of research is to explore the different methods for modifying a feasible colouring achieved in time-step t into an initial colouring for time-step $t + 1$ (i. e. line 2 of Algorithm 2). The essential question to be answered is: can a feasible colouring for one graph G_t be used in some advantageous way to find a feasible colouring for a similar graph G_{t+1}?

Algorithm 2 Generic DGCP 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: $\mathcal{S}_{best} \leftarrow \emptyset$
 2: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_t$ modified in some way (see Sections 4.1 and 4.2)
 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:         $\mathcal{S}_{best} \leftarrow \mathcal{S}_{t+1}$
 8:         $k \leftarrow k - 1$
 9:     if $\mathcal{S}_{best} = \emptyset$ and a computation limit is reached then
10:         $k \leftarrow k + 1$
11: $\mathcal{S}_{t+1} \leftarrow \mathcal{S}_{best}$
12: return $\mathcal{S}_{t+1}$
```

4.1 Modification for Edge Dynamic Graphs

For all of the following methods, the final feasible colouring $\mathcal{S}_t$ for G_t can be considered as a complete, improper colouring for G_{t+1} with $k = |\mathcal{S}_t|$ colour classes. We can do this because every vertex $v \in V$ will be coloured but the new edges E^+_{t+1} are likely to cause clashes. With this knowledge we can then apply one of the following modification methods.

(1) *Calculate the number of clashes*: Calculate the initial number of clashes and then pass $\mathcal{S}_t$ directly to the tabu search operator which will attempt to find a feasible k-colouring for G_{t+1}.

(2) *Uncolour clashing vertices*: By identifying pairs of clashing vertices in $\mathcal{S}_t$ and transferring one of the vertices in each of these pairs to a set of uncoloured vertices, one produces a partial, proper colouring $\tilde{\mathcal{S}}_{t+1}$ for G_{t+1}. $\tilde{\mathcal{S}}_{t+1}$ along with the set of uncoloured vertices can now be passed to the tabu search operator which will attempt to find a feasible k-colouring for G_{t+1}.

(3) *Solve clashing vertices*: In a similar manner to Method (2), clashing vertices are "uncoloured" to produce a partial, proper colouring $\tilde{\mathcal{S}}_{t+1}$ for G_{t+1}. An attempt is then made to re-insert each of these uncoloured vertices into a colour class in $\tilde{\mathcal{S}}_{t+1}$ such that no clashes are incurred. The remaining uncoloured vertices and any appropriate edges are then considered as a residual graph G'_{t+1} of G_{t+1}. This residual graph is passed to the constructive operator (specifically, the recursive largest first (RLF) algorithm [10]) which produces a feasible k'-colouring for G'_{t+1}. The feasible colouring for G'_{t+1} is then combined with $\tilde{\mathcal{S}}_{t+1}$ to produce a feasible colouring for G_{t+1} with $k + k'$ colour classes. The tabu search operator will then attempt to find a feasible $(k + k' - 1)$-colouring for G_{t+1}.

4.2 Modification for Vertex Dynamic Graphs

The final feasible colouring $\mathcal{S}_t$ achieved for G_t will be neither a complete, improper colouring or a partial, proper colouring for G_{t+1} as it will include the deleted vertices V^-_{t+1} and won't include the new vertices V^+_{t+1}. For each of the following methods, every deleted vertex $v \in V^-_{t+1}$ must first be removed from $\mathcal{S}_t$ in order to produce a partial, proper colouring $\tilde{\mathcal{S}}_{t+1}$ for G_{t+1} with $k = |\mathcal{S}_t|$ colour classes. We can then apply one of the following modification methods.

(4) *Randomly assign new vertices*: Each new vertex $v \in V^+_{t+1}$ is randomly assigned to a colour class in $\tilde{\mathcal{S}}_{t+1}$ to produce a complete, improper colouring for G_{t+1}. This can then be passed to the tabu search operator which will attempt to find a feasible k-colouring for G_{t+1}.

(5) *Uncolour new vertices*: Unlike Method (4), the new vertices V^+_{t+1} are not assigned to colour classes in $\tilde{\mathcal{S}}_{t+1}$. Instead the new vertices V^+_{t+1} are considered as a set of uncoloured vertices. Along with $\tilde{\mathcal{S}}_{t+1}$, this set of uncoloured vertices is passed to the tabu search operator which attempts to find a feasible k-colouring for G_{t+1}.

(6) *Solve new vertices*: An attempt is made to insert each of the new vertex $v \in V^+_{t+1}$ into an a colour class in $\tilde{\mathcal{S}}_{t+1}$ such that no clashes are incurred. The remaining new vertices and any appropriate edges are then considered as a residual graph G'_{t+1} of G_{t+1}. This residual graph is passed to the constructive operator (again, RLF) which produces a feasible k'-colouring for G'_{t+1}. The feasible colouring for G'_{t+1} is then combined with $\tilde{\mathcal{S}}_{t+1}$ to produce a feasible colouring for G_{t+1} with $k + k'$ colour classes. The tabu search operator will then attempt to find a feasible $(k + k' - 1)$-colouring for G_{t+1}.

5 Experimentation Details

In our experiments we considered dynamic random graphs. For each dynamic random graph we specify an initial number of vertices n, a desired density d, a change probability p and a number of time-steps T. To construct a sequence of graphs $\mathcal{G}$ we use the following methods.

For an edge dynamic graph consider the graph $G_t = (V, E_t)$. To construct G_{t+1}, every edge $\{u, v\} \in E_t$ is copied to the set of deleted edges E^-_{t+1} with probability p and every currently non-existent edge $\{u, v\} \in \mathcal{E} \backslash E_t$ is copied to the set of new edges E^+_{t+1} with probability $\frac{dp}{1-d}$.

For a vertex dynamic graph, consider the graph $G_t = (V_t, E_t)$. To construct G_{t+1}, every vertex $v \in V_t$ is copied to the set of deleted vertices V^-_{t+1} with probability p and the set of new vertices is constructed such that $|V^+_{t+1}|$ is an integer between $np(1-p)$ and $np(1+p)$. Every edge $\{u, v\} \in \mathcal{E}_{t+1}$ with $u \in V_{t+1}$, $v \in V^+_{t+1}$ and $u \neq v$ is then added to the set of new edges E^+_{t+1} with probability d.

For both the edge and vertex dynamic graphs, the following parameters were used: $n = 500$, $d \in \{0.1, 0.5, 0.9\}$, $p \in \{0.005, 0.01, \ldots, 0.05\}$ and $T = 10$, and for each combination of these parameters, 20 graphs were produced. The RLF algorithm [10] was applied to obtain an initial colouring for G_0. Note that all results corresponding to these initial graphs are ignored; however, the colourings they produced were used in the modification methods for G_1.

In our case, each time-step was given a time limit of 10 seconds[1] (i. e. line 4 of Algorithm 2). If this time limit had been set much longer, say hours, then the advantage of modifying colourings between time-steps would obviously diminish.

TabuCol [8] and PartialCol [2] were used to find feasible colourings in the complete, improper search space and partial, proper search space respectively (i. e. line 5 of Algorithm 2). These algorithms use the neighbourhood moves outlined in Section 3 and, upon performing a move, the inverse moves are made "tabu" for $0.6 \times f(\mathcal{S}') + r$ iterations, where f is the cost function given in Equations (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 has been used in both [8] and [2].

During execution, the target number of colour classes is adjusted in the following way. Let k be the target number of colour classes, initially defined by the modification method being implemented. If a feasible k-colouring 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 this remaining time limit then k is again increased by 1, and so on (i. e. lines 9 and 10 of Algorithm 2). For example, say the target number of colour classes for G_t is initially set as $k = 23$, if a feasible 23-colouring cannot be found within 5 seconds then the tabu search operator attempts to find a feasible 24-colouring for G_t, if this cannot be found within a further 2.5 seconds then the tabu search attempts to find a feasible 25-colouring for G_t, and so on.

For a base-line comparison, the following control method was also implemented:

(0) *Reset*: The static graph $G_t \in \mathcal{G}$ for each time-step $t \in \{1, \ldots, T\}$ is considered without any information about colourings achieved in the previous time-steps. As with G_0, the RLF algorithm is applied to obtain an initial colouring for G_t (i. e., RLF replaces line 2 of Algorithm 2). Tabu search is then applied iteratively in an attempt to find colourings with fewer colour classes. The number

[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.

of colour classes in the final, feasible colouring achieved and the time required to obtain this colouring is then recorded.

Note that Methods (1) and (4) operate exclusively in the complete, improper search space, Methods (2) and (5) operate exclusively in the partial, proper search space, and Methods (0), (3) and (6) can operate in either search space as required. Because of this, only comparisons between methods designed for the same problem and operating in the same search space are compared. For example, for the edge dynamic GCP operating in the complete, improper search space only Methods (0), (1) and (3) are compared against one another.

In all of our results, unless otherwise stated, all statistical tests are Wilcoxon signed rank tests with significance level $\alpha = 0.05$.

6 Results

6.1 Initial Colourings for the Edge Dynamic GCP

Let us first consider the initial feasible colourings produced for the edge dynamic GCP. For all densities d and change probabilities p, Methods (1) and (2) were found to produce initial, feasible colourings with significantly fewer colour classes than both Methods (0) and (3). This is clearly illustrated in Figure 1.

We have observed a significant increase in the time required by Methods (1) and (2) to achieve their initial, feasible colourings compared to Methods (0) and (3) for all values of d and p, as seen in Table 1. A main contributing factor to this 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 search space. For similar reasons, as p increases so too does the time required by Methods (1) and (2) 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 (3) 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 1. Hence we can conclude that for these high levels of p, modifying feasible colourings for G_t is of no benefit when attempting to achieve initial, feasible colourings for G_{t+1}.

Considering computational effort, we have found that the time required by Method (3) 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 (3) 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 (3) which only applies it to a residual graph $G'_t = (V', E'_t)$ of G_t where $V' \subseteq V$ (which implies $|V'| \leq |V|$). We therefore see that applying Method (3) 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.

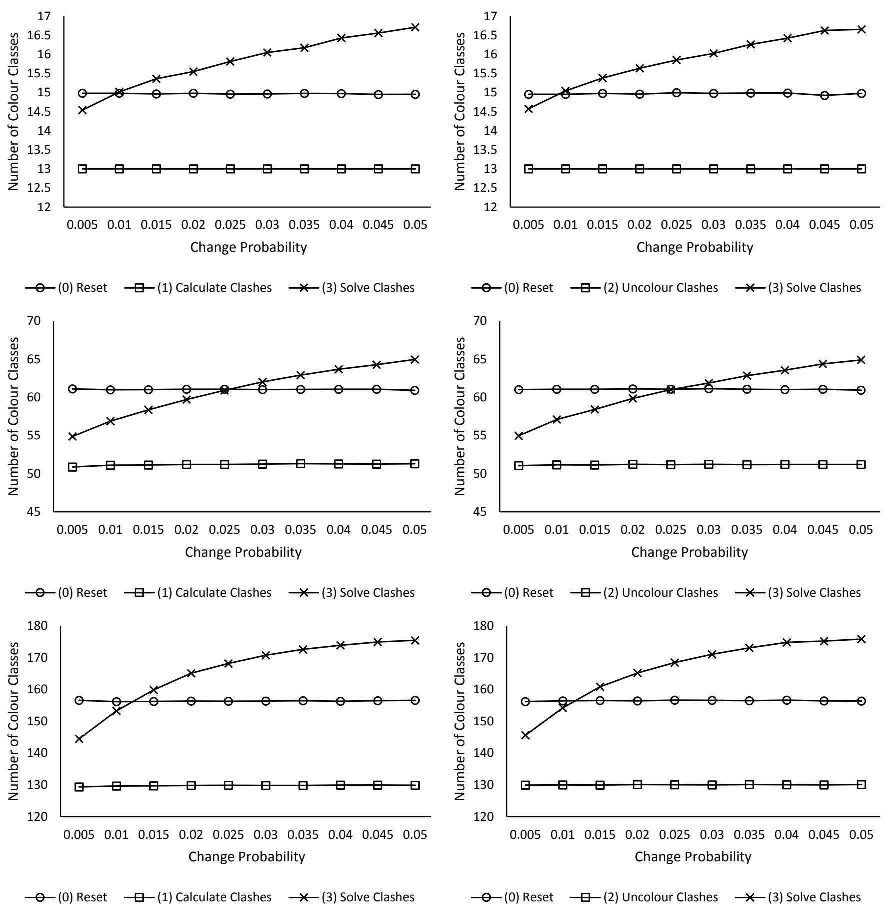

Figure 1. Mean initial, feasible colourings for the edge dynamic GCP. Graphs on the left represents results from trials in the complete, improper search space and those on the right for trials in the partial, proper search space. From top to bottom, rows represent $d = 0.1, 0.5$, and 0.9 respectively.

Table 1. Median time (in seconds) required to obtain an initial, feasible colouring for the edge dynamic GCP (a 0^* entry implies that the recorded time is less than 10^{-3} seconds).

		p(%)									
d	Method	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
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^*	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)	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
	(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)	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*

6.2 Initial Colourings for the Vertex Dynamic GCP

Let us now consider initial colourings for the vertex dynamic GCP. It is first worth mentioning that a small change to the edge set of a graph will affect more vertices than a comparable change to its vertex set. It is therefore not surprising that the following results are similar to those presented in Section 6.1 but for higher values of p.

Comparable to Methods (1) and (2) for the edge dynamic problem, the initial, feasible colourings achieved by Methods (4) and (5) have significantly fewer colour classes than Methods (0) and (6) but require significantly more time to obtain them. The time required by Methods (4) and (5) also has a positive relationship with the change probability p. These observations can be seen in Figure 2 and Table 2. The reasons for this behaviour are the same as those given for Methods (1) and (2) in Section 6.1.

Table 2. Median time (in seconds) required to obtain an initial, feasible colouring for the vertex dynamic GCP (a 0^* entry implies that the recorded time is less than 10^{-3} seconds).

		p(%)									
d	Method	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
0.1	(0)	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0.015	0.015
	(4)	0^*	0^*	0.015	0.015	0.016	0.031	0.031	0.031	0.031	0.046
	(5)	0^*	0^*	0.015	0.015	0.015	0.016	0.015	0.016	0.016	0.016
	(6)	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
	(4)	0.320	1.131	1.240	1.724	1.482	1.724	1.935	2.411	2.114	2.785
	(5)	0.663	0.983	1.537	1.529	1.630	1.537	1.973	1.794	1.841	2.340
	(6)	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
	(4)	1.069	1.997	2.941	3.830	3.565	4.189	4.820	4.938	4.852	5.007
	(5)	1.163	1.731	2.644	3.222	2.387	3.416	3.339	3.424	2.816	3.346
	(6)	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*	0^*

Again, as with Method (3) for the edge dynamic problem, Method (6) produces initial, feasible colourings with both significantly fewer and significantly more colour classes than Method (0) depending on the change probability p. However, Method (6) only produces initial, feasible colourings with significantly more colour classes for $d = 0.1$ with $p \geq 0.035$. In fact, for $d = 0.1$ with $p \leq 0.02$, and $d \in \{0.5, 0.9\}$ with all values of p, Method (6) achieves initial, feasible colourings with significantly fewer colour classes. This is clearly illustrated in Figure 2.

As with Method (3), Method (6) requires significantly less time than Method (0) in all instances except for $d = 0.1$ with $p \leq 0.03$ (as seen in Table 2). This is again likely because Method (6) applies RLF to a smaller graph G'_t with $|V'_t| = |V_t^+| \approx np$ as opposed to applying it to G_t with $|V_t| \approx n$.

6.3 Final Colourings for the Edge Dynamic GCP

Next let us consider final colourings for the edge dynamic GCP. The Friedman test with $\alpha = 0.05$ shows that for $d = 0.1$ there is no significant difference between the number of colour classes in the final, feasible colourings achieved when applying Methods (0), (1), (2) and (3). However, Methods (1) and (2) both achieve final, feasible colourings with significantly more colour classes than those achieved by Method (0) for $d = 0.9$ with $p \geq 0.01$ and $p \geq 0.035$ respectively. Methods (1) and (2) also achieve final, feasible colourings with significantly more colour classes than those achieved by Method (3) 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 Methods (0) and (3) (see Section 6.1 and Table 1). This "wasted" time then translates to time not being allocated to finding feasible colourings with fewer colour classes.

For $d = 0.5$ and some values of p, Method (3) was found to achieve final, feasible colourings with significantly fewer colour classes than Method (0). However, for $d = 0.9$ with $p \geq 0.04$ the opposite holds which is unsurprising as Method (3) produces initial, feasible colourings with significantly more colour classes under these parameter settings.

The following time comparisons correspond only to trials where the number of colour classes in the final, feasible colourings achieved by the compared methods were equal to one another. This will also be the case in Section 6.4.

Method (1) was found to reach final, feasible colourings significantly faster than Method (0) for $d = 0.1$ with $p \leq 0.035$, and $d = 0.5$ with $p \leq 0.01$ as seen in Table 3. Similarly, Method (2) also achieves final, feasible colourings in significantly less time than Method (0) for $d = 0.1$ with all values of p, and $d = 0.5$ with $p = 0.005$. 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 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.

On the other hand, Method (1) was found to require significantly more time than Method (0) to achieve final, feasible colourings for $d = 0.5$ with $p \geq 0.035$, and $d = 0.9$ with all values of p. The same was also found for Method (2) for

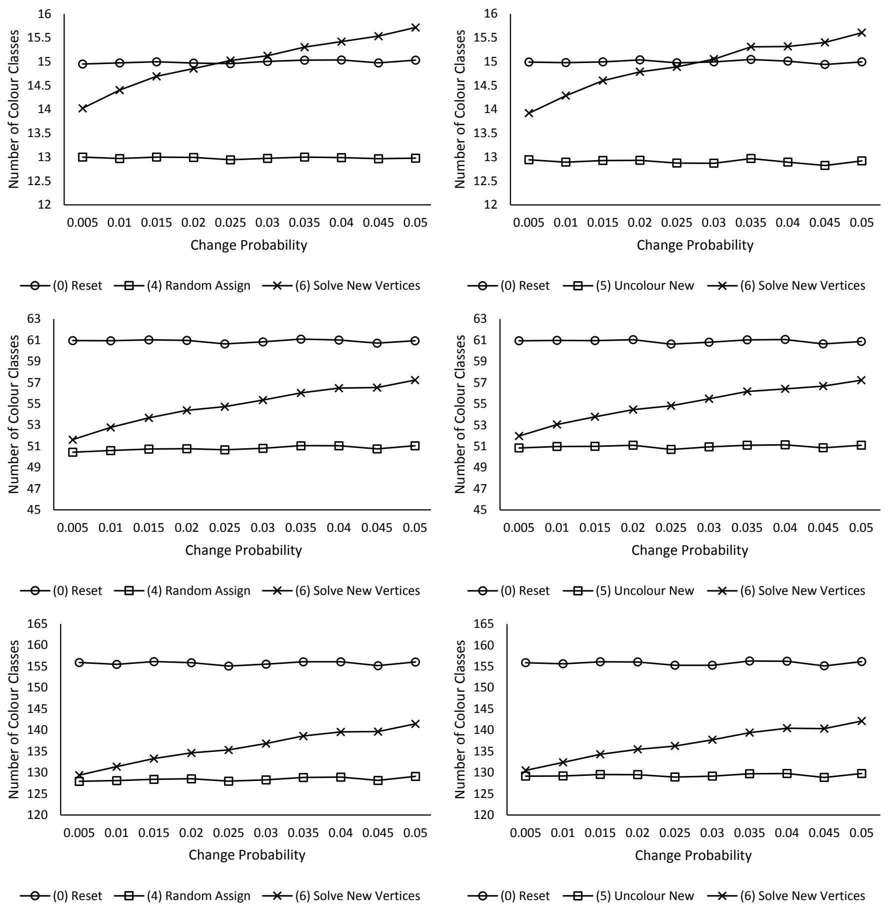

Figure 2. Mean initial, feasible colourings for the vertex dynamic GCP. Graphs on the left represents results from trials in the complete, improper search space and those on the right for trials in the partial, proper search space. From top to bottom, rows represent $d = 0.1, 0.5$, and 0.9 respectively.

$d = 0.9$ with most values of p. In a similar fashion, these two methods require significantly more time to achieve final, feasible colourings than Method (3) for $d \in \{0.5, 09\}$ with 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 $d = 0.9$.

Unlike Methods (1) and (2), Method (3) was not found to require significantly more time than Method (0) for any parameter settings. On the contrary, for $d = 0.1$ with $p \leq 0.035$, and $d = 0.5$ with $p \leq 0.02$, Method (3) requires significantly less time to achieve final, feasible colourings. It should be highlighted that these are similar parameter settings for which Method (3) is able to produce initial, feasible colourings with significantly fewer colour classes than Method (0).

Table 3. Median time (in seconds) required to obtain final, feasible colourings with the same numbers of colour classes for the edge dynamic GCP (a 0^* entry implies that the recorded time is less than 10^{-3} seconds).

			$p(\%)$									
d	S.S.	Method	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
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^*	0.015	0.016	0.031	0.031	0.031	0.031	0.047	0.047	0.047
		(3)	0.015	0.016	0.031	0.031	0.046	0.031	0.047	0.047	0.047	0.047
	P.P.	(0)	0.016	0.016	0.031	0.016	0.031	0.031	0.031	0.031	0.031	0.031
		(2)	0^*	0^*	0.015	0.015	0.015	0.015	0.015	0.016	0.016	0.016
		(3)	0^*	0.015	0.015	0.015	0.016	0.016	0.016	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	2.371	3.190	3.097	3.424	3.417	3.869	4.259	4.321	4.275
		(3)	1.077	1.794	2.130	2.683	2.239	2.652	2.754	2.465	3.284	2.762
	P.P.	(0)	2.933	2.910	3.081	2.870	2.278	2.730	2.636	2.559	2.309	2.676
		(2)	1.872	1.888	2.356	3.783	2.356	2.722	3.058	2.847	2.746	3.331
		(3)	1.435	2.160	2.060	2.356	2.246	2.699	2.169	2.442	2.168	2.598
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
		(3)	4.181	4.906	4.415	4.353	5.694	5.195	4.649	5.234	5.039	4.882
	P.P.	(0)	4.181	5.242	5.179	4.166	5.273	4.914	3.681	4.602	4.212	4.633
		(2)	5.141	5.616	5.975	6.365	6.365	5.741	6.038	5.506	5.452	5.866
		(3)	3.877	4.649	5.070	4.275	4.352	4.025	4.196	4.618	4.688	4.977

6.4 Final Colourings for the Vertex Dynamic GCP

Finally, let us consider final colourings for the vertex dynamic GCP. As mentioned in Section 6.2, a small change to the edge set will usually affect more vertices than a comparable change to its vertex set.

Method (4) was found to achieve final, feasible colourings with significantly fewer colour classes than Method (0) for $d = 0.5$ with most values of p, and $d = 0.9$ with $p \leq 0.03$. Similarly, Method (5) was also found to achieve final, feasible colourings with significantly fewer colour classes than Method (0) for $d \in \{0.5, 0.9\}$ with some values of p. On the other hand, Method (4) achieves final, feasible colourings with significantly more colour classes than Method (6) for $d = 0.9$ with $p \geq 0.025$. Although Methods (4) and (5) require significantly more time to produce initial, feasible colourings (see Section 6.2 and Table 1) it is likely that Methods (0) and (6) still require more time to reach a feasible colouring with equivalent numbers of colour classes for low levels of p. This would imply that Methods (4) and (5) attempt to find feasible colourings with fewer colour classes earlier than Methods (0) and (6). Further analysis should be conducted in order to investigate the validity of this proposition.

Unlike Method (3) for the edge dynamic problem, Method (6) was only found to reach final, feasible colourings with the same or significantly fewer colour classes than Method (0). Both Methods (0) and (6) start each time-step from a feasible colouring; however, Method (6) achieves initial colouring with significantly fewer colour classes than Method (0) for most combinations of d and p (see Section 6.2 and Figure 2). Method (6) will therefore attempt to find feasible colourings with fewer colour classes earlier than Method (0).

It was found that Methods (4) and (5) achieve final, feasible colourings in significantly less time than Method (0) for $d = 0.1$ with all values of p, and $d \in \{0.5, 0.9\}$ with $p \leq 0.01$. Additionally, Method (4) was found to achieve final, feasible colourings in significantly less time for $d = 0.5$ with $p \leq 0.04$, and $d = 0.9$ with $p \leq 0.025$ also. This can be seen in Table 4. 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.

On the contrary, Methods (4) and (5) require significantly more time to achieve final, feasible colourings than Method (6) for $d \in \{0.5, 0.9\}$ with most values of p. In comparison to Method (0), Method (6) starts from a feasible colouring with significantly fewer colour classes for $d \in \{0.5, 0.9\}$ with all values of p (see Section 6.2 and Figure 2). This will likely translate to Method (6) attempting to find feasible colourings with fewer colour classes before Methods (4) and (5) are able to produce initial, feasible colourings.

Method (6) was also found to require significantly less time to achieve final, feasible colourings than Method (0) for all values of d with most values of p (again, see Table 4). This is probably due to the same argument given earlier with regards to the number of colour classes in the final, feasible colourings achieved by Method (6) compared to Method (0).

7 Conclusions and Future Work

In this paper we have presented several methods for modifying feasible colourings from one time-step of a dynamic random graph in order to help find a feasible colouring for the next time-step.

Our experiments have shown that, for both edge and vertex dynamic graphs, initial colourings with significantly fewer colour classes can be achieved by initially modifying a feasible k-colouring for G_t into an infeasible k-colouring for G_{t+1} and then passing this directly to the 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 methods are applied. These methods were

Table 4. Median time (in seconds) required to obtain final, feasible colourings with the same numbers of colour classes for the vertex dynamic GCP (a 0^* entry implies that the recorded time is less than 10^{-3} seconds).

			$p(\%)$									
d	S.S.	Method	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
0.1	C.I.	(0)	0.046	0.046	0.047	0.047	0.047	0.047	0.047	0.047	0.047	0.062
		(4)	0^*	0^*	0.015	0.015	0.016	0.031	0.031	0.031	0.031	0.046
		(6)	0^*	0^*	0.015	0.016	0.031	0.031	0.031	0.046	0.031	0.047
	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
		(6)	0^*	0.015	0.015	0.015	0.016	0.016	0.016	0.031	0.031	0.031
0.5	C.I.	(0)	5.141	3.682	4.321	4.610	4.212	3.619	3.884	3.713	3.666	3.612
		(4)	0.515	1.224	1.996	1.747	2.738	2.699	3.713	2.551	4.103	4.470
		(6)	0.328	1.084	1.045	1.303	2.028	1.740	1.981	2.020	2.013	2.247
	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
		(6)	0.203	1.068	1.092	1.529	1.544	1.420	1.381	2.192	2.387	2.293
0.9	C.I.	(0)	5.702	6.069	6.318	6.146	6.053	6.021	6.209	5.843	5.881	6.186
		(4)	1.428	5.007	5.007	4.399	5.460	5.148	5.507	6.069	6.381	6.459
		(6)	1.786	2.090	3.019	3.307	4.033	3.681	3.667	4.227	3.791	4.142
	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
		(6)	1.373	2.013	2.566	1.981	3.261	3.330	3.416	2.745	3.884	4.189

also found to achieve final, feasible colourings with the significantly more colour classes for some edge dynamic problems but significantly fewer colour classes for some vertex dynamic problems. The time required to achieve comparable final colourings via these methods is dependent on p.

It has also been shown that reducing a feasible colouring for G_t into a partial, proper colouring for G_{t+1} and then applying a constructive algorithm to the residual graph induced by the "uncoloured" vertices can also achieve initial, feasible colourings with significantly fewer colour classes when p is small enough. These modification methods were also shown to produce initial, feasible colourings in significantly less time for $d \in \{0.5, 0.9\}$. Finally, these methods also resulted in final, feasible colourings with the same or significantly fewer colour classes and require equal or significantly less time to do so.

Note that in this piece of work, all changes between time-steps of dynamic graphs have occurred completely at random; however, for some real world applications there may be some level of predictability. More specifically, we might have some knowledge of how edges and vertices are likely to change in the future. We wish to extend this research and explore how this sort of information can be used to our advantage in order to produce more robust colourings.

References

1. Aardal, K.I., Van Hoesel, S.P., Koster, A.M., Mannino, C., Sassano, A.: Models and solution techniques for frequency assignment problems. Annals of Operations Research 153(1), 79–129 (2007)
2. Blöchliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers & Operations Research 35(3), 960–975 (2008)

3. Chaitin, G.J.: Register allocation & spilling via graph coloring. ACM Sigplan Notices 17(6), 98–101 (1982)
4. Dupont, A., Linhares, A.C., Artigues, C., Feillet, D., Michelon, P., Vasquez, M.: The dynamic frequency assignment problem. European Journal of Operational Research 195(1), 75–88 (2009)
5. Erben, W.: A grouping genetic algorithm for graph colouring and exam timetabling. In: Practice and Theory of Automated Timetabling III, pp. 132–156. Springer (2001)
6. Gyárfás, A., Lehel, J.: On-line and first fit colorings of graphs. Journal of Graph theory 12(2), 217–227 (1988)
7. Harary, F., Gupta, G.: Dynamic graph models. Mathematical and Computer Modelling 25(7), 79–87 (1997)
8. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4), 345–351 (1987)
9. 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. Operations research 39(3), 378–406 (1991)
10. 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)
11. Lewis, R.: Constructing wedding seating plans: A tabu subject. In: Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM). p. 1. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp) (2013)
12. Lovász, L., Saks, M., Trotter, W.T.: An on-line graph coloring algorithm with sublinear performance ratio. Annals of Discrete Mathematics 43, 319–325 (1989)
13. Monical, C., 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. pp. 469–476. ACM (2014)
14. Preuveneers, D., Berbers, Y.: ACODYGRA: an agent algorithm for coloring dynamic graphs. Symbolic and Numeric Algorithms for Scientific Computing (September 2004) 6, 381–390 (2004)
15. Qu, R., Burke, E.K., 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)
16. 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)