Recombinative Approaches for the Maximum Happy Vertices Problem

Dhananjay Thiruvady[a], Rhyd Lewis[b]

[a]*School of Information Technology, Deakin University, Geelong, Australia*
[b]*School of Mathematics, Cardiff University, Cardiff, Wales.*

Abstract

The maximum happy vertices problem involves taking a simple graph in which several vertices are pre-coloured. The aim is to colour the remaining vertices to maximise the number *happy* vertices, where a vertex is considered happy if and only if all of its neighbours are assigned the same colour as the vertex itself. Previous studies for this problem have investigated integer programming, tabu search and the construct, solve, merge & adapt (CMSA) matheuristic. In this study, we develop three different evolutionary approaches based on tabu search and also develop a CMSA-tabu search (CMSA-TS) hybrid. We conduct experiments on a wide range of d-regular and scale-free graphs and find that, overall, CMSA-TS is the most effective approach, nearly always finding the best-observed solutions. However, in special cases where problems have large numbers of vertices and low vertex degrees, the evolutionary algorithms have a distinct advantage.

Keywords: Tabu search, Evolutionary Algorithms, Construct, Solve, Merge & Adapt, Graph Colouring, Maximum Happy Vertices Problem

1. Introduction

In the past five years, studies of the *maximum happy vertices* (MHV) problem have been of growing interest in the literature [Li and Zhang, 2015, Agrawal, 2018a, Lewis et al., 2019, Bliznets and Sagunov, 2019, Thiruvady et al., 2020b]. This problem involves taking a partially coloured graph and colouring the remaining vertices such that the number of *happy* vertices is maximised. Happy vertices are those that are assigned the same colour as all of their neighbours.

The MHV problem can be seen as a variant of the well-known graph colouring problem. The latter is a classical problem in combinatorial optimisation that requires an assignment of colours to vertices such that no two adjacent vertices have the same colour while minimising the number of colours used. Graph colouring has applications in many real-world problems such as resource allocation, timetabling, sports scheduling, and frequency assignment [Carter et al., 1996, Lewis, 2021, Lewis and Thompson, 2015, McCollum et al., 2010]. In contrast, the MHV problem involves trying to assign adjacent vertices to the *same* colour and is, therefore, more suited to areas where group relationships and homophily are considered important [Li and Zhang, 2015]. Like graph colouring, several practical situations can be modelled using the concept of vertex happiness. One example is provided by Li and Zhang [2015], where they consider the problem of identifying "connectivity" between academic papers. The network they devise consists of papers (vertices) with citations between papers signified by edges. The papers' titles and abstracts are known for all cases, but the keywords are only listed in about 5% of papers. By representing keywords by colours and then maximising the number of happy vertices in the graph, the subject areas of papers can be predicted with an accuracy of 69%. Lewis et al. [2019] have also given several applications of the MHV problem

by considering social networks (vertices represent people and links friendships) and seating allocations for events (where people are represented by vertices and links define relationships between them: see also the work of Lewis and Carroll [2016]). Consider, for example, a wedding party where guests are to be placed at tables. Assuming some members of the wedding party have already been allocated to tables, the remaining guests can now be assigned to tables where they are familiar with other guests. Solving this problem as the MHV allows finding a solution where acquaintances are maximised. The MHV problem can also be applied to problems where clusters need to be found. More specifically, if some data points within clusters are known, the remaining data points can be assigned to clusters in an optimal fashion [Everitt et al., 2011].

To date, there have been relatively few approaches proposed for tackling the MHV problem. The first study was due to Li and Zhang [2015], who provided two constructive approximation algorithms and also proved the NP-hardness of the problem. This was followed by the work of Zhang et al. [2018], who showed how an improved approximation algorithm can be achieved via a rounding heuristic based on a linear programming relaxation of the problem. Agrawal [2018a] have shown that the MHV problem is W[1]-hard when considering a parameter that is the minimum bound on the potential number of happy vertices. Other studies have also noted that this problem is fixed-parameter tractable when parameterised by the treewidth of the graph and the number of different colours used in the initial problem [Agrawal, 2018b, Aravind et al., 2016, Misra and Vinod Reddy, 2018, Li and Zhang, 2015]. Although the problem of determining the treewidth in an arbitrary graph is itself NP-hard, this still brings useful results in cases where it is known, such as trees, cycles, and Halin graphs, allowing their optimal solutions to be achieved in polynomial time. Lewis et al. [2019], meanwhile, have proposed algorithms for generating upper and lower bounds on the optimal number of happy vertices in a graph. They also show how graphs can sometimes be broken up into smaller subgraphs, and compare the performance of two integer programming models with the construct, solve, merge & adapt (CMSA) matheuristic of Blum et al. [2016]. Most recently, Thiruvady et al. [2020b] have developed a tabu search method and an approach for finding efficient upper bounds. Their results show that tabu search is effective at finding high-quality solutions in short time frames.

As noted, previous studies with this problem have demonstrated the effectiveness of simple metaheuristics (tabu search), integer programming (IP) and a CMSA-based matheuristic. While these approaches have been seen to be effective on certain graphs, they do not generalise across different types of graphs (including d-regular and scale-free graphs) and do not scale gracefully with problem size. Metaheuristics can typically find good solutions to small and medium-sized problems quickly but often get stuck in local optima. IP is very effective on certain classes of problems (where it often proves optimality), but it fails to scale well with problem size. CMSA has also previously shown to be very effective in combining IP with heuristics [Blum and Blesa, 2016, Blum, 2016] and metaheuristics [Thiruvady et al., 2019, Polyakovskiy et al., 2020, Thiruvady et al., 2020a], though, the full potential of this approach is still to be explored. In this present study, we aim to improve on existing approaches for tackling the MHV so that higher-quality solutions can be found within reasonable time frames. Moreover, the approaches we propose are also designed to scale efficiently with problem size, a factor that is vital in large networks. The contributions of this study are (a) an evolutionary approach underpinned by the tabu search of Thiruvady et al. [2020b], (b) a CMSA matheuristic also enhanced by tabu search, and (c) a demonstration of the efficacy of these approaches on a wide range of problem instances. In addition, we also generalise a theoretical result of Lewis et al. [2019] by showing further ways in which problem instances can be subdivided.

The remainder of this paper is organised as follows. In Section 2, we formally define the MHV problem. Section 3 then provides details of previously proposed preprocessing procedures, upper bounding methods and constructive heuristics. These techniques are all used within our proposed approaches. Section 4 gives a new result on the subdivision of MHV problem instances, while Section 5 describes the methods proposed

in this study. Here, the details of tabu search are first provided, followed by our evolutionary and CSMA approaches. Section 6 gives details of the experiments, including problem instance generation, while Section 7 discusses the results. Section 8 concludes the paper.

2. Problem Definition

For the MHV problem we are given a simple graph $G = (V, E)$ comprising n vertices and m edges. We denote the neighbourhood of a vertex v by $\Gamma(v)$, and use the function $c : V \to \{1, \ldots, k\}$ to denote a complete colouring of the vertices. If $c(v) = c(u)\ \forall u \in \Gamma(v)$, then vertex v is considered *happy*, else it is considered *unhappy*. In other words, a vertex v is happy if and only if all of its neighbours have the same colour as the vertex itself.

The objective of the MHV problem is to maximise the number of happy vertices once a complete colouring has been formed. That is, starting with a partial colouring $c' : V' \to \{1, \ldots, k\}$ (where $V' \subseteq V$), the function c' is extended to form a complete colouring $c : V \to \{1, \ldots, k\}$, in such a way as to maximise the number of happy vertices. If a vertex is not assigned to a colour in the initial problem instance, it is referred to as a *free* vertex. The maximum (optimal) number of happy vertices for a particular instance is denoted by $H(G)^*$.

The following IP formulation of the MHV problem was originally proposed by Lewis et al. [2019]. It uses the variables $x_j \in \{1, \ldots, k\}$ and $y_j \in \{0, 1\}$ for all vertices $v_j \in V$, where x_j represents the colour of vertex v_j, and $y_j = 1$ if and only if v_j is happy. The full formulation is:

$$\text{Maximise} \quad n - \sum_{j=1}^{n} y_j \tag{1}$$

$$\text{subject to:} \quad x_j = c(v_j) \qquad \forall v_j \in V' \tag{2}$$

$$y_j \geq \frac{|x_j - x_i|}{n} \qquad \forall v_i \in \Gamma(v_j), \forall v_j \in V. \tag{3}$$

Here Equation (1) is the objective function, which seeks to maximise the number of happy vertices. Constraint (2) assigns precoloured vertices to their predefined colours, while Constraint (3) assigns y_j to 1 if and only if all of v_j's neighbours have the same colour as v_j. An example MHV problem instance and solution are shown in Figure 1.

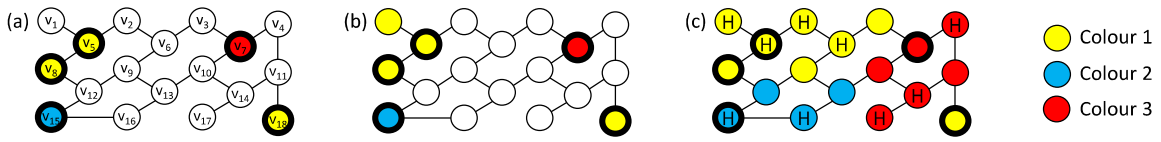


Figure 1: (a) An example MHV problem instance involving $n = 18$ vertices, $m = 21$ edges, $k = 3$ colours, and $|V'| = 5$ precoloured vertices, indicated by bold outlines; (b) the same problem instance with one additional precoloured vertex, determined by the AddPrecol procedure (see Section 3); (c) a complete colouring with nine happy vertices, labelled with H's.

3. Preprocessing, Upper Bounds, and Constructive Heuristics

In this section, we give a more detailed description of previously proposed techniques for the MHV problem. We consider these techniques because they are used and built upon in our current methods.

First, Thiruvady et al. [2020b] previously proposed a preprocessing routine called Add-Precol, which fixes the colours of certain free vertices in a graph, thereby reducing the size of the solution space while not changing the optimal number of happy vertices $H(G)^*$. Their method operates in two stages. In the first stage, the subgraph induced from the set of free vertices $(V - V')$ is taken. Suppose that this subgraph comprises the following components $C_1, \ldots, C_l$. Each component C_i is now examined in turn and, if the neighbouring precoloured vertices of C_i in G are all seen to have the same colour j, then all vertices in C_i can also be precoloured with j and will be guaranteed to be happy in any solution. Similarly, if the vertices of C_i are found to have no neighbouring precoloured vertices in G, then the vertices of C_i can all be assigned to the same arbitrarily chosen colour, also making them happy. Note that this ensures that all singletons in the original graph are precoloured.

In the second stage of Add-Precol, further vertices are then also coloured by identifying free vertices v whose neighbours are all precoloured and guaranteed to be unhappy, but where v also has at least two neighbours that are precoloured differently. In this case, v is also guaranteed to be unhappy, and can therefore be permanently labelled with an arbitrary colour.

In the work of Thiruvady et al. [2020b], the Add-Precol process, which operates in $O(m)$ time, is shown to have the largest effects when graphs are either very sparse (because the lack of edges means that many free vertices will be coloured by the first stage), or very dense (because a large number of edges will increase the number of unhappy vertices in the graph, making free vertices more likely to meet the conditions given in the second stage). These patterns are also seen to be more pronounced when the proportion of precoloured vertices in the initial problem instance is higher. In general, this procedure has been observed to have the smallest effects in graphs with average degrees of approximately four to ten.

As mentioned, Lewis et al. [2019] have also proposed a method for calculating an upper bound $\bar{H}(G)$ on the optimal value $H(G)^*$. This is based on the idea of *unhappy paths*. In a partial colouring of G, unhappy paths are simple u-v-paths whose terminal vertices are precoloured differently (i.e., $c(u) \neq c(v)$) and whose internal vertices, if any, are free and uncoloured. It is obvious that an unhappy path implies the presence of at least one unhappy vertex somewhere along the path, so their approach operates by repeatedly identifying and removing unhappy paths from a graph while keeping count of the minimum possible number of unhappy vertices x in the graph. At the end of this process, the upper bound $\bar{H}(G)$ is set to $n - x$.

Finally, the two constructive algorithms of Li and Zhang [2015] are also used in this paper for producing initial solutions with our methods. The first algorithm, named Greedy-MHV, operates by assigning all free vertices to the first colour and calculating the resultant number of happy vertices. These actions are then repeated using colours $2, \ldots, k$, and the best of these k solutions is then chosen. The second heuristic, Growth-MHV is slightly more involved but essentially involves colouring free vertices one by one, so that any vertices that have the potential of being happy are coloured first. Full details are described by Thiruvady et al. [2020b].

4. Problem Subdivision

Previously, Lewis et al. [2019] proposed methods by which instances of the MHV problem can be broken into smaller sub-problems that can each be tackled independently. Their methods are based on the identification of vertex separating sets; that is, subsets of vertices that, when removed from the graph, will increase the number of connected components. To work correctly with the MHV, each vertex v in the separating set must have one of two properties: (a) v is precoloured and guaranteed to be happy (because all of its neighbours are precoloured with the same colour), or (b) v is precoloured but guaranteed to be unhappy (because it has at least one neighbour that is precoloured differently). The removal of separating sets that meet these conditions gives multiple sub-problems that can each be solved separately. The resultant

sub-solutions can then be merged into a final, full solution. If all of the sub-solutions are optimal, then the final solution will also be optimal.

Here, we extend the results of Lewis et al. [2019] by noting how certain edges in an MHV problem instance can be classified as *redundant* and therefore removed from the graph.

Definition 1. *In an MHV problem instance, an edge* $\{u, v\}$ *is considered* redundant *when both its endpoints are precoloured with the same colour (i.e.,* $u, v \in V'$ *and* $c(u) = c(v)$*).*

The identification of a graph's redundant edges (which can be easily achieved in $O(m)$ time) leads to the following theorem.

Theorem 1. *Given a complete colouring of a graph* G*, let* G' *be a copy of* G *with the same colour assignments, but with some or all of the redundant edges removed. Then* G *and* G' *both feature the same number of happy vertices.*

Proof. Let v be an endpoint of a redundant edge, with $c(v) = i$. In a complete colouring, each vertex in a graph is either happy or unhappy. It is therefore sufficient to show that v is happy in G if and only if it is also happy in G'. If v is happy in G then, by definition, all neighbours of v are also assigned to colour i. The removal of an edge incident to v will therefore maintain v's happiness. Conversely, if v is happy in G', the addition of the redundant edge will maintain v's happiness because this edge's other endpoint is also precoloured with colour i. □

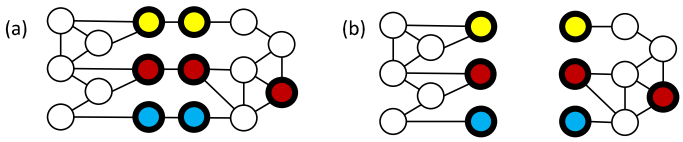


Figure 2: (a) An example MHV problem instance with three redundant edges; (b) the subgraphs resulting from the removal of these redundant edges.

Theorem 1 allows the removal of all redundant edges in a graph G without affecting the quality of the solutions available. It also brings the possibility of G being broken into separate components that can then be considered separately. An example of this is shown in Figure 2. Observe that this process generalises the methods given in Lewis et al. [2019] since the removal of a vertex v meeting property (a) above implies that all edges incident to v must be redundant. The removal of redundant edges, therefore, brings additional opportunities for problem subdivision.

5. Methods for the MHV Problem

Previous approaches, including IP, tabu search and CMSA have shown to be reasonably effective with the MHV problem [Lewis et al., 2019, Thiruvady et al., 2020b]; however, they all have some inherent limitations. IP, as an exact approach, has the benefits of providing a certificate of optimality (or infeasibility if no solution exists) [Wolsey, 1998]. In the context of the MHV problem, it is known to be effective with small problem instances, where it is often possible to produce provably optimal solutions in short time frames; however, it can struggle with medium and large problem instances. On the other hand, while tabu search can produce high-quality solutions, it also tends to quickly get stuck at local optima with many instances. CMSA, on the other hand, aims to make use of the relative advantages of both exact methods

and heuristics. In this study we use IP for the exact element of CMSA. This has two distinct advantages. First, it has strong diversification properties; second, the use of a restricted IP model (as opposed to the full model given in Section 2) brings greater opportunities for efficient solving. Nonetheless, the IP seems to rely heavily on the production of a good starting solution, which is not always possible with the heuristics used by Lewis et al. [2019]. Hence, tabu search serves to provide high quality starting solutions for the IP.

Hence, our objective is to develop approaches that combine the power of local search with suitable diversification mechanisms, placing a particular emphasis on operators that seek to combine features of previously observed high-quality solutions. The first option we consider in this regard is evolutionary algorithms, which can be devised in multiple ways. As an alternative, CMSA might also be viewed as a special type of evolutionary algorithm in that it allows features from previously observed solutions to be combined into new solutions, in this case via IP.

In the following subsection, we describe a tabu search approach for the MHV problem. Section 5.2 then describes three bespoke recombination operators that can be used within an evolutionary-based approach. Section 5.3 then describes a tight integration of CMSA and tabu search. We note that before starting the main components of our algorithms (tabu search followed by the evolutionary algorithms or CMSA), we always apply the Add-Precol procedure and compute the upper bound $\bar{H}(G)$ (see Section 3). The upper bound provides useful information for initialising tabu search, the details of which are discussed presently.

5.1. Tabu search

The tabu search method used here was originally proposed by Thiruvady et al. [2020b] and is similar in style to those used with other types of graph partitioning problems [Lewis, 2021, Blöchliger and Zufferey, 2008].[1] A key feature of this method is its ability to quickly scan and evaluate all solutions that neighbour the incumbent, as described below. The algorithm starts with an initial complete colouring and makes a series of neighbourhood moves until a terminating criterion is satisfied. Through the course of execution, a tabu list is maintained that ensures recently visited solutions are not visited again. This is vital in prohibiting cycling and encourages the algorithm to explore new regions of the solution space where possible. For this method, a candidate solution is represented by a full partition of the vertices into k subsets (colour classes). In our case this is encoded using a sequence (array) $\mathcal{S}$ of length k in which the ith element, $\mathcal{S}(i)$, is a set comprising all vertices labeled with colour i (that is, $v \in \mathcal{S}(i)$ if and only if $c(v) = i$). In this form, the solution shown in Figure 2(c), for example, is written $(\{v_1, v_2, v_3, v_5, v_6, v_8, v_9\}, \{v_{12}, v_{13}, v_{15}, v_{16}\}, \{v_4, v_7, v_{10}, v_{11}, v_{14}, v_{17}\})$.

The neighbourhood operator in this method takes an unhappy free vertex v and moves it from its current colour class $\mathcal{S}(i)$ into a new colour class $\mathcal{S}(j)$. Precolourings are therefore always respected. Starting with a solution $\mathcal{S}$, the search progresses as follows. In each iteration, all possible neighbourhood moves are evaluated, and the non-tabu move with the largest improvement (or smallest degradation) in quality is applied, breaking any ties randomly. A tabu move is also permitted if the resultant solution improves on the best solution seen in the run so far (an aspiration criterion). In the situation that all neighbourhood solutions are tabu, a single move is applied at random.

In this approach, the tabu list is encoded using a matrix $\mathbf{T}_{n\times k}$. If, in iteration l of the algorithm, a vertex v is moved from colour class $\mathcal{S}(i)$ to $\mathcal{S}(j)$, then element T_{vi} is set to $l + t$. This signifies that all moves that would result in v being reassigned to colour i are classed as tabu for the next t iterations. A second matrix $\mathbf{C}_{n\times k}$ is also used to speed up the evaluation of a solution's neighbourhood where, given the current solution $\mathcal{S}$, element C_{vj} denotes the change in the number of happy vertices that would result if vertex v were to be moved to colour j. This means that the objective value of all solutions neighbouring $\mathcal{S}$ can be determined by

[1] The tabu search source code is available at [Lewis, 2019].

simply scanning the rows of $\mathbf{C}$ corresponding to free unhappy vertices. Once a move has been performed by moving v to j, only rows corresponding to free vertices within a distance of two from v need to be updated in $\mathbf{C}$. All other rows of the matrix are not affected.

Finally, in our approach the tabu tenure t is set to be a random variable based on the quality of the current solution. In essence, when solution quality is perceived to be poor, high values for t are used, therefore encouraging diversification into new parts of the search space. Conversely, when solution quality is high, smaller values for t are used. Here we follow the scheme of Thiruvady et al. [2020b] and use $t = r + \tau(\bar{H}(G) - f(\mathcal{S}))$ where r is randomly selected from the set $\{1, 2 \ldots, 9\}$ allowing for a little randomisation. The parameter $\tau \in \mathbb{R}^+$ is user-defined, and $f(\mathcal{S})$ gives the objective value (number of happy vertices) in the current solution $\mathcal{S}$.

5.2. Evolutionary Algorithm Operators

Evolutionary algorithms (EAs) are a type of metaheuristic inspired by biological evolution and natural selection. EAs operate by maintaining a population of candidate solutions that represent a sample of the solution space. During a run, efforts are made to improve the quality of this population using recombination (crossover), mutation, and evolutionary pressure. Recombination seeks to create new "offspring" solutions by combining different parts of existing population members (the "parents"). Mutation, on the other hand, makes random changes to a candidate solution to allow new regions of the solution space to be explored. Evolutionary pressure then seeks to exhibit some bias towards keeping good candidate solutions in the population and rejecting bad ones.

The evolution of an EA's population takes place with the repeated application of the above operators; however, it is often necessary to design specialised recombination and mutation operators that can suitably exploit the underlying structures of the problem at hand. To date, we are not aware of any previously suggested evolutionary methods for the MHV problem, though research into other types of partitioning problem suggest that effective recombination operators should seek to preserve the "groupings" in solutions wherever possible, thereby allowing the appropriate substructures to be propagated through the population [Brown and Sumichrast, 2005]. Such schemes have been successfully applied to a variety of partitioning problems including bin packing [Quiroz-Castellanos et al., 2015], graph colouring [Galinier and Hao, 1999, Lewis, 2021], truss cutting [Lewis and Holborn, 2017], and load balancing [Falkenauer, 1998]. For the MHV problem, the "groupings" that we seek to propagate are the sets of vertices belonging to each colour class. As with the examples just mentioned, problem-specific features must also then be added in order to repair solutions and help preserve groupings where needed.

Algorithm 1 Random Grouping Crossover (RGX)

Let $\mathcal{S}_1$ be a copy of the first parent and $\mathcal{S}_2$ be a copy of the second parent
$P \leftarrow \{1, \ldots, k\}$
$\mathcal{S}(i) \leftarrow \emptyset \; \forall i \in \{1, \ldots, k\}$
for all $i \in \{1, \ldots, k\}$ **do**
    With 50% probability $j \leftarrow 1$, else $j \leftarrow 2$
    Randomly select and remove an element (colour) $l \in P$
    for all $v \in \mathcal{S}_j(l)$ **do**
        Remove v from $\mathcal{S}_1$
        Remove v from $\mathcal{S}_2$
        Insert v into $\mathcal{S}(l)$
OUTPUT: A (partial) offspring solution $\mathcal{S}$

Our recombination operators use the same solution encoding scheme as the tabu search algorithm. Our first operator, which we call *random grouping crossover* (RGX), starts by taking copies of two selected parent solutions. We call these copies $\mathcal{S}_1$ and $\mathcal{S}_2$. In each step, one of these copies, $\mathcal{S}_j$, is randomly selected together with a random colour class l. Next, all vertices assigned to the colour class $\mathcal{S}_j(l)$ are copied into colour class l in the offspring solution. In addition, all vertices in $\mathcal{S}_j(l)$ are removed from both $\mathcal{S}_1$ and $\mathcal{S}_2$. This process repeats for k steps, ensuring that all colours are considered once. A full pseudocode description of this operator is given in Algorithm 1. An example application is also shown in Figure 3.

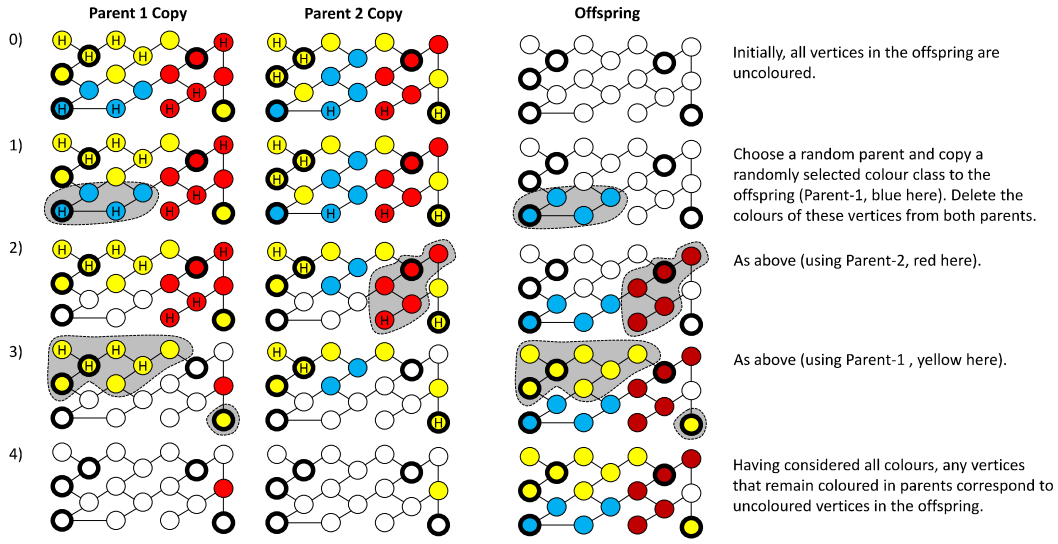


Figure 3: Example application of the RGX recombination operator. Happy vertices are marked by H's, and precoloured vertices by heavy outlines.

Note that the RGX operator exhibits the following three features.

1. If a vertex v is assigned to the same colour in both $\mathcal{S}_1$ and $\mathcal{S}_2$, then v will also assume this colour in the offspring. Hence, precolourings are always respected.
2. If, when being copied to the offspring, a vertex $v \in \mathcal{S}_j(l)$ is happy, then v will also be happy in the offspring $\mathcal{S}$. (Note, however, that due to the actions of Lines 8 and 9 in Algorithm 1, $\mathcal{S}_j(l)$ may be a subset of the ith colour class in the original parent.)
3. After k steps, some free vertices in the offspring may remain uncoloured.

If the latter feature occurs, a repair operator is required to restore solution feasibility. In our case, this is carried out by simply assigning each uncoloured vertex to a randomly selected colour. In all cases, tabu search (Section 5.1) is then applied. The application of this local search procedure serves as the mutation operator for our EA—a strategy employed by many high-performance EAs [Lewis and Holborn, 2017, Yan et al., 2020, Huang et al., 2020].

Considering Feature 2 above, a natural variant of the RGX operator arises if, at Line 6 of Algorithm 1, we introduce some bias into the selection of colour classes. Here, our *biased grouping crossover* (BGX) operates in the same way as RGX except that, when choosing a colour class l, the one with the highest number of happy vertices in the selected parent $\mathcal{S}_j$ is returned. The intention here it to produce offspring

with an increased number of happy vertices. In Step 1) of Figure 3, for example, the yellow colour class in Parent 1 will be selected, as it has four happy vertices (marked by H's).

Our third and final suggested recombination operator uses the basic uniform crossover (UX): that is, each vertex in the offspring is considered in turn and inherits the colour of the corresponding vertex in Parent 1 with a 50% probability, else it assumes the colour from Parent 2. Note that this operator satisfies the first feature above, but not the second and third. On completion of this procedure, tabu search is also used as a mutation operator.

5.3. CMSA and Tabu Search

The motivation for integrating tabu search with CMSA is to combine the complementary advantages of each method. As mentioned, the tabu search approach of Thiruvady et al. [2020b] usually converges quickly. As such, the method is effective at intensification but seems to lack sufficient diversification characteristics for escaping local optima. On the other hand, the CMSA approach of Lewis et al. [2019] has an effective diversification mechanism but seems to be reliant on a good initial solution being provided.

The general CMSA methodology has similarities to evolutionary algorithms. As a substitute for a population of solutions, a set of *solution components* is maintained, derived from promising solutions observed earlier in the run. This set of components is then used to construct new solutions, typically using a MIP formulation, which allows the production of optimal solutions with respect to the current component set. In this sense, the construction procedure can be seen as a special type of recombination in which solution components are optimally combined from multiple sources to produce new solutions.

For this application of CMSA, a complete set of solution components is defined by the set $C = V \times \{1, \ldots, k\}$. Each element of C, therefore, corresponds to a vertex/colour assignment. A valid solution is then represented by a subset of C in which each vertex is assigned to exactly one colour and all precolourings are obeyed. During the execution of CMSA, a subset $C' \subseteq C$ of components is maintained and MIP methods are used to construct solutions that only use components from this subset. The aim is for the contents of C' to be adapted so that high-quality solutions can be produced. In particular, the promising components in C' are kept, while others are removed through the use of an ageing mechanism, as explained below.

Our overall CMSA procedure is presented in Algorithm 2. The parameters used by the algorithm are (1) n_{sols}: the number of solutions produced per iteration; (2) t_{max}: the overall time limit; (3) t_{mip}: a time limit for each application of the MIP solver; (4) t_{ts}: a time limit for each application of tabu search; and (5) age_{max}: a maximum age limit.

As shown in the pseudocode, the algorithm begins by running tabu search for half the allotted time. This is to ensure significant intensification, leading to a high-quality starting solution. This is then passed to the main CMSA process, which iterates until the time limit is reached. In the first part of each CMSA iteration (Lines 6 to 10), the aim is to generate several different solutions. In our case, this is achieved by mutating the best-observed solution $\mathcal{S}_{\text{bsf}}$ using the procedure Gen-Solution. The key to the success of CMSA is that the solutions produced by this procedure are of sufficient diversity. This is achieved by producing each individual solution from $\mathcal{S}_{\text{bsf}}$ in such a way that precoloured vertices remain unchanged. To do this, let v be a free vertex. Then v's colour is reassigned to colour $j \in \{1, \ldots, k\}$ with probability:

$$P(v, j) = \frac{\phi(v, j)}{\deg(v)} \tag{4}$$

where $\phi(v, j)$ is the number of neighbours of v that are assigned the colour j. This function allows the colour of a vertex to change and biases the selection of this colour towards those appearing most frequently among the neighbours. In Lines 8 to 10, the components of these new solutions are then added to C', with new additions being assigned an "age" of zero.

Algorithm 2 CMSA

1: INPUT: n_{sols}, t_{max}, t_{mip}, t_{ts}, age_{max}
2: $C' \leftarrow \emptyset$,
3: $\text{age}_c \leftarrow 0$, $\forall c \in C = (V \times \{1, \ldots, k\})$
4: $\mathcal{S}_{\text{bsf}} \leftarrow$ TABU-SEARCH($\emptyset$, $t_{\text{max}}/2$)
5: **while** t_{max} is not exceeded **do**
6: **for** $i = 1, 2, \ldots, n_{\text{sols}}$ **do**
7: $\mathcal{S} \leftarrow$ GEN-SOLUTION($\mathcal{S}_{\text{bsf}}$)
8: **for** $c \in \mathcal{S} : c \notin C'$ **do**
9: $\text{age}_c \leftarrow 0$
10: $C' \leftarrow C' \cup \{c\}$
11: $\mathcal{S}_{\text{mip}} \leftarrow$ SOLVE-MIP($C', \mathcal{S}_{\text{bsf}}, t_{\text{mip}}$)
12: $\mathcal{S}_{\text{mip}} \leftarrow$ TABU-SEARCH($\mathcal{S}_{\text{mip}}$, t_{ts})
13: **if** $\mathcal{S}_{\text{mip}}$ is better than $\mathcal{S}_{\text{bsf}}$ **then** $\mathcal{S}_{\text{bsf}} \leftarrow \mathcal{S}_{\text{mip}}$
14: ADAPT(C', $\mathcal{S}_{\text{mip}}$, age_{max})
15: OUTPUT: $\mathcal{S}_{\text{bsf}}$

In Line 11 of Algorithm 2, the SOLVE-MIP procedure is used to solve the MHV problem with respect to C'; that is, the procedure seeks the optimal solution that only uses vertex-colour assignments appearing in C'. In our case, we use the MIP formulation proposed by Lewis et al. [2019] as the basis of our restricted MIP, which is provided in Section 2. In addition, let $V'' \subset V$ include all vertices whose colour remains unchanged in all solutions $\mathcal{S}$ produced in the loop defined at Line 6 of Algorithm 2. The following constraint is also added to ensure that all vertices in V'' are fixed to the corresponding colour:

$$x_j = c(v'_j) \qquad \forall v'_j \in V'' \tag{5}$$

Note that if $C' = C$ then, given excess time, the SOLVE-MIP procedure will always return the globally optimum solution for the MHV problem. However, this is unlikely to be possible in reasonable time with non-trivial cases. The use of $C' \subset C$ leads to a more manageable search space for the MIP method. That said, applications of SOLVE-MIP may still not halt in reasonable time, particularly if C' has many elements, hence the time limit t_{mip} is also used in each application. In our case SOLVE-MIP is also warm-started with the best known solution $\mathcal{S}_{\text{bsf}}$. This ensures that, at worst, its output will be the best-known solution. In Line (12), the output of SOLVE-MIP is then passed to tabu search to seek further improvements, if possible.

The ADAPT procedure in Step (14) of Algorithm 2 is used to periodically remove certain elements (components) from C', helping to keep it at a reasonable size. Specifically, the age variables are updated in such a way that (a) the ages of components in C' not appearing in $\mathcal{S}_{\text{mip}}$ are first incremented by one, and (b) the components whose age values exceed age_{max} are removed. The final output of the algorithm is the best-observed solution across the entire run: $\mathcal{S}_{\text{bsf}}$.

Finally, for robustness, we consider a further variation on the CMSA implementation of Lewis et al. [2019]. Note that their original CMSA algorithm allowed the colours of all free vertices to be changed except those with a degree of one. The reason for this was to build a relatively compact MIP with as few free variables as possible; however, this restriction is inappropriate for graphs where such vertices exist (such as scale-free graphs for which $q = 1$) as it may prevent the discovery of global optima. Consequently, we propose a modified implementation to allow single-degree vertices to change colour (Line 7 in Algorithm 2). If they do, then these vertices will then have corresponding free variables when initialising the restricted MIP in Line 11 of Algorithm 2.

6. Experimental Setting

We consider two types of graph topology in our experiments: d-regular and scale-free graphs. The former are randomly generated graphs in which $\deg(v) = d$ for all $v \in V$. The parameter d can assume any value in the range $\{0, \ldots, n-1\}$. In our case, these graphs were generated using the method of Steger and Wormald [1999].

In contrast to d-regular graphs, scale-free graphs are constructed by adding vertices one at a time while showing a preference for linking these new vertices to existing vertices of a high degree. The resulting graphs therefore have a degree distribution that follows a power law, and the structure of the graphs is such that most vertices will have low degrees but a few vertices will have very high degrees [Barabási and Põsfai, 2016].

To construct scale-free graphs, we use the Barabási-Albert model using a parameter $q \in \{0, 1, \ldots, n\}$. We start with a complete graph G comprising q vertices and $\binom{q}{2}$ edges. In each step a new vertex v is then added to G together with q edges that connect v to vertices already in G. This is done by performing q roulette-wheel trials where, in each case, the probability $P(u, v)$ of adding the edge $\{u, v\}$ to E is calculated as

$$P(u,v) = \begin{cases} \dfrac{\deg(u)}{\sum_{w \in (V - \Gamma(v))} \deg(w)} & \text{if } \{u,v\} \notin E \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Here, $(V - \Gamma(v))$ denotes the set of vertices in G that are not yet adjacent to v. Vertices are added in this way until a graph with n vertices and $m = \binom{q}{2} + q(n-q)$ edges is formed.

Finally, on the production of the desired graph, k colours are assigned to a user-specified proportion of randomly chosen vertices, ensuring that each colour is used at least once. This completes the problem instance construction process.

For our trials, all algorithms were implemented in C++ and compiled with GCC-5.4.0. The MIP component of CMSA was implemented using Gurobi Optimiser Version 9.0.1 with a limit of 20 GB of memory. All experiments were conducted on Monash university's MonARCH cluster.[2] In all trials the algorithms were executed for $t_{\max} = 600$ seconds per instance. The parameters of CMSA were set using the work of Lewis et al. [2019] as a guide, giving $n_{\text{sols}} = 10$ (ensuring sufficient diversity); $t_{\text{mip}} = 60$ seconds; $t_{\text{ts}} = 5$ seconds; and $\text{age}_{\text{mip}} = 3$. As advised by [Thiruvady et al., 2020b], a value of $\tau = 4$ was used for updating the tabu list.

As discussed, the CMSA approach investigated here uses the first half of the allotted run-time for tabu search and the remainder for CMSA. Moreover, we refer to the modified CMSA-TS, allowing single-degree vertices to be modified, as CMSA-TS-F. To provide parity with CMSA, in trials with the EAs the tabu search procedure was also executed for the first half of all runs. An initial population for the EA was then formed using the output of this procedure, together with several solutions produced using randomised versions of the constructive algorithms described in Section 3. Each member of the initial population was then improved by executing tabu search for $t_{\text{ts}} = 5$ seconds. In each iteration of the EA, two parent solutions were selected at random from the population and the chosen recombination operator was used to create a single offspring. This was then mutated using tabu search for $t_{\text{ts}} = 5$ seconds before replacing the least fit of the two parents. In preliminary tests, we experimented with several different selection and replacement policies but found that this strategy, whose evolutionary pressure only exists due to the replacement of the weaker parent, gave the most consistent results. We also experimented with a range of population sizes and found that the best

[2]Each machine in this cluster has 24 cores and 256 GB RAM. Each physical core consists of two hyper-threaded cores with Intel Xeon E5-2680 v3 2.5GHz, 30M Cache, 9.60GT/s QPI, Turbo, HT, 12C/24T (120W).

results were achieved using a relatively small population of size ten. This suggests that good solutions are derived from repeated applications of the EA's operators on a small pool of solutions as opposed to a wide sampling of the solution space. These findings are consistent with those of Quiroz-Castellanos et al. [2015], Galinier and Hao [1999], Lewis and Holborn [2017], and Falkenauer [1998], who have previously applied EAs to other types of partitioning problems.

7. Results and Discussion

In the tables that follow, each presented value is a mean across ten problem instances. For d-regular graphs, we generated problem instances using numbers of vertices $n \in \{1000, 2500, 5000, 7500, 10000, 15000, 20000\}$, colors $k \in \{10, 50, 100\}$, vertex degrees $d \in \{2, 5, 8\}$, and two proportions of precoloured vertices, 25% and 50%. This led to 1260 different problem instances. Scale-free graphs were generated using the same values for n and k, $q \in \{1, 2, 4, 5, 8\}$, and 50% of precoloured vertices. This gave a further 1050 problem instances.

7.1. d-regular Graphs

Table 1 compares the results of all five algorithms on d-regular graphs with 25% precoloured vertices. The results reported are the percentage difference of each method to the best found in terms of the proportion of happy vertices. They show that, overall, CMSA-TS is consistently the best performing method. The differences between the remaining four algorithms are generally small, though pockets do seem to exist where the EA with the (very basic) UX operator seems to produce favourable results (compared to the other EAs), particularly with the sparsest graphs.

The reasons for UX's favourable performance against the other recombination operators are explored in Figure 4. Here we see that, on its own, UX produces poorer-quality offspring compared to RGX; however, once tabu search (mutation) is applied to these offspring the reverse is true, with UX's solutions nearly always featuring more happy vertices. In these cases, it seems that UX is more suited to the problem in that, by producing low-quality disrupted offspring, tabu search can start at a wider diversity of points within the solution space, ultimately producing higher-quality solutions.

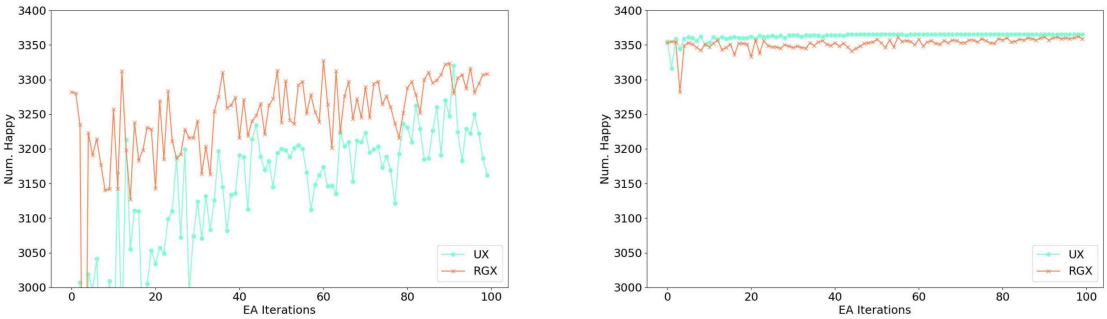


Figure 4: Number of happy vertices in each successive offspring produced by the EA using the UX and RGX operators. The left figure shows the number of happy vertices occurring after recombination is applied. The right figure shows the number once recombination *and* tabu search (mutation) is applied. These results were produced using the EA with a 2-regular graph with $n = 5000$ vertices, $k = 10$ colours and 25% of vertices precoloured.

Table 2 shows the results for d-regular graphs in which 50% of vertices are precoloured. Here, most of the values are substantially larger than those in Table 2. This is because a higher proportion of (randomly

Table 1: Comparison of the proportion of happy vertices for d-regular graphs with 25% of the vertices precoloured. The results show percentage differences of each method to the best achieved across all the methods. The best result for each comparison is highlighted in bold, while statistically significant results (pair-wise Wilcoxon ranked-sum test at a confidence interval of 95%) are marked with a '*'.

	Colours k	10			50			100		
Vertices	Degree d	2	5	8	2	5	8	2	5	8
1000										
	TS	0.25	0.04	0.00	0.62	0.63	0.00	0.65	0.86	0.36
	RGX	0.01	**0.00**	0.00	0.15	0.63	0.00	0.50	0.86	0.36
	BGX	0.01	0.04	0.00	0.11	0.63	0.00	0.36	0.86	0.36
	UX	0.01	0.04	0.00	0.05	0.63	0.00	0.22	0.86	0.36
	CMSA-TS	**0.00**	0.04	0.00	**0.00**	***0.00**	0.00	***0.00**	***0.00**	***0.00**
2500										
	TS	0.37	0.00	0.04	0.51	0.47	0.43	0.59	0.65	0.40
	RGX	0.15	0.00	0.04	0.51	0.47	0.43	0.64	0.65	0.40
	BGX	0.15	0.00	0.04	0.37	0.43	0.43	0.65	0.65	0.40
	UX	0.03	0.00	0.04	0.22	0.47	0.43	0.58	0.65	0.40
	CMSA-TS	***0.00**	**0.00**	**0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
5000										
	TS	0.37	0.01	0.00	0.56	0.59	0.23	0.85	0.68	0.27
	RGX	0.24	0.01	0.00	0.60	0.59	0.23	0.99	0.68	0.27
	BGX	0.21	0.01	0.00	0.59	0.59	0.23	1.00	0.68	0.27
	UX	0.04	0.01	0.00	0.57	0.59	0.23	0.98	0.68	0.27
	CMSA-TS	***0.00**	**0.00**	0.00	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
7500										
	TS	0.41	0.00	0.03	0.65	0.73	0.37	1.12	0.78	0.53
	RGX	0.26	0.01	0.03	0.72	0.73	0.37	1.23	0.78	0.53
	BGX	0.25	**0.00**	0.03	0.71	0.71	0.37	1.25	0.78	0.53
	UX	0.04	0.01	0.03	0.70	0.73	0.37	1.25	0.78	0.53
	CMSA-TS	***0.00**	0.01	**0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
10000										
	TS	0.40	**0.00**	0.01	0.71	0.55	0.21	1.11	0.61	0.57
	RGX	0.27	0.03	0.01	0.85	0.55	0.21	1.24	0.61	0.57
	BGX	0.26	0.01	0.01	0.83	0.55	0.21	1.22	0.61	0.57
	UX	0.04	0.03	0.01	0.83	0.55	0.21	1.23	0.61	0.57
	CMSA-TS	***0.00**	**0.00**	**0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
15000										
	TS	0.40	0.04	0.01	0.88	0.29	0.53	1.35	0.54	0.50
	RGX	0.32	0.09	0.01	1.02	0.29	0.53	1.49	0.54	0.50
	BGX	0.31	0.08	0.01	1.02	0.29	0.53	1.49	0.54	0.50
	UX	0.06	0.09	0.01	1.01	0.29	0.53	1.50	0.54	0.50
	CMSA-TS	***0.00**	***0.00**	**0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
20000										
	TS	0.34	0.11	0.00	1.01	0.00	0.00	1.28	0.00	0.04
	RGX	0.33	0.16	0.00	1.21	0.00	0.00	1.48	0.00	0.04
	BGX	0.33	0.14	0.00	1.16	0.00	0.00	1.44	0.00	0.04
	UX	0.05	0.14	0.00	1.20	0.00	0.00	1.46	0.00	0.04
	CMSA-TS	***0.00**	***0.00**	**0.00**	***0.00**	0.00	0.00	***0.00**	0.00	**0.00**

allocated) precoloured vertices leads to fewer happy vertices being possible in the graphs. We again see that CMSA-TS is the best performing method here. The only places where it performs relatively poorly are for

the largest, densest problem instances ($n = 20000$, $d = 8$) where the difficulty of the underlying MIPs stops the algorithm from being able to perform a sufficient number of iterations within the time limit.

Comparing the EAs to tabu search, we see that the EAs outperform tabu search across most problem instances. In a few cases—for example, large problem instances with 10 colours and a degree of 2—we see that tabu search is comparable to RGX and BGX. But even in these cases, UX outperforms the other EAs and tabu search.

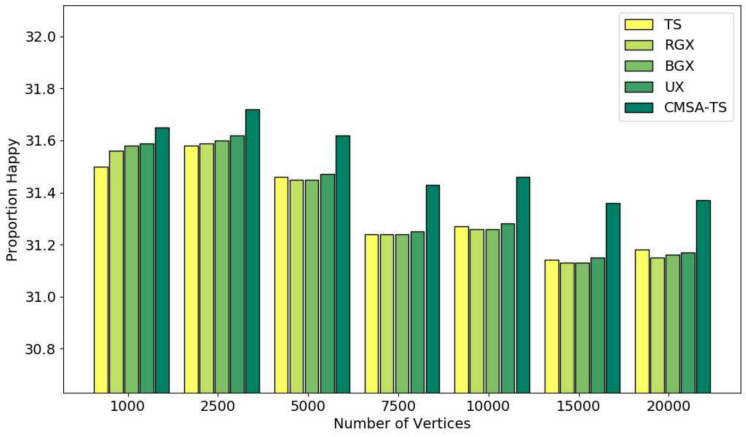


Figure 5: Proportion of happy vertices in our d-regular graphs with 25% of precoloured vertices, split by instance size.

We now examine algorithm performance in terms of the proportion of happy vertices achieved. Figure 5 considers our d-regular graphs with 25% precoloured vertices and groups the results by the number of vertices n. We see that CMSA-TS is easily the best performing method with these instances. Of the remaining four algorithms, the EA using the UX operator is again the most favourable generally, though, for the largest problem instances ($n = 20000$), tabu search produces slightly better solutions because of the insufficient number of EA iterations taking place within the time limit. Similar results are also shown in Figure 6, where 50% of vertices are precoloured.

In Figures 7 and 8 these same results are partitioned according to the number of colours k and degrees d. Figure 7 shows the results for graphs with 25% precoloured vertices. Here, we see that the largest differences occur with low degrees and/or larger numbers of colours. This could suggest that when problem instances have relatively few colours and high degrees, then the various algorithms are all able to find solutions close to the optimal.

Figure 8 shows similar patterns though with some important differences. First, we see that across these problem instances, CMSA-TS consistently produces the best solutions. Second, these differences do not seem to diminish according to increases in degree as with the previous case. Increasing the number of colours does slightly enhance the differences between the algorithms, with $k = 100$ showing the largest differences. Thus, having a large proportion of precoloured vertices seems to allow a larger number of solutions when problems have a medium level of degree, and the diversity achieved by CMSA-TS proves to be vital here.

Overall, the results presented for d-regular graphs show that combining the relative advantages of tabu search intensification and CMSA diversification leads to better outcomes than our alternative algorithms. However, there are certain small pockets of the instance space where the EAs are sometimes more effective

Table 2: Comparison of the proportion of happy vertices for d-regular graphs with 50% of the vertices precoloured. The results show percentage differences of each method to the best achieved across all the methods. The best result for each comparison is highlighted in bold, while statistically significant results (pair-wise Wilcoxon ranked-sum test at a confidence interval of 95%) are marked with a '*'.

	Colours k	10			50			100		
Vertices	Degree d	2	5	8	2	5	8	2	5	8
1000										
	TS	0.24	0.23	8.00	0.23	13.22	6.67	0.32	14.67	6.90
	RGX	0.00	0.03	3.33	0.03	8.19	2.50	0.12	9.71	4.31
	BGX	0.00	0.03	2.67	0.03	9.50	2.50	0.18	8.95	4.31
	UX	0.00	0.00	2.00	0.00	9.68	2.50	0.06	9.90	4.31
	CMSA-TS	0.00	0.00	***0.00**	0.00	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
2500										
	TS	0.14	0.21	9.31	0.21	15.18	7.92	0.32	14.54	7.27
	RGX	0.00	0.22	6.86	0.22	12.13	3.63	0.37	12.00	3.46
	BGX	0.01	0.09	5.88	0.09	12.20	2.97	0.35	11.69	2.77
	UX	0.00	0.05	5.64	0.05	12.35	3.63	0.15	12.69	3.46
	CMSA-TS	0.00	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
5000										
	TS	0.12	0.26	11.69	0.26	15.25	8.94	0.64	14.15	8.58
	RGX	0.03	0.30	7.64	0.30	12.02	6.36	0.75	12.30	6.58
	BGX	0.04	0.27	7.64	0.27	12.69	5.45	0.73	11.80	5.82
	UX	0.00	0.20	8.00	0.20	12.02	5.91	0.47	12.26	6.43
	CMSA-TS	0.00	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
7500										
	TS	0.14	0.39	10.75	0.39	14.59	9.06	0.95	14.74	8.24
	RGX	0.05	0.56	6.92	0.56	12.85	7.38	1.04	12.69	5.93
	BGX	0.05	0.51	7.74	0.51	12.62	7.48	1.03	12.82	6.70
	UX	0.00	0.44	6.68	0.44	12.92	6.85	0.76	12.92	5.93
	CMSA-TS	0.00	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
10000										
	TS	0.15	0.49	9.93	0.49	15.15	7.86	1.03	14.32	8.47
	RGX	0.07	0.63	8.24	0.63	12.94	6.45	1.09	12.90	6.62
	BGX	0.07	0.61	8.00	0.61	12.73	6.84	1.10	12.75	6.78
	UX	0.00	0.55	8.18	0.55	13.36	6.53	0.98	12.84	6.38
	CMSA-TS	0.00	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
15000										
	TS	0.17	0.74	9.64	0.74	13.43	9.31	1.10	11.92	9.97
	RGX	0.16	0.85	7.75	0.85	12.36	7.22	1.17	10.88	7.78
	BGX	0.16	0.86	8.20	0.86	12.14	6.90	1.15	10.84	6.96
	UX	0.00	0.80	8.16	0.80	12.24	7.12	1.17	10.96	7.94
	CMSA-TS	**0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**	***0.00**
20000										
	TS	0.16	5.81	1.57	0.99	8.18	2.39	1.10	2.60	2.60
	RGX	0.16	0.00	0.13	1.08	6.80	0.30	1.21	0.44	0.44
	BGX	0.16	0.00	***0.00**	1.08	6.88	0.85	1.19	0.26	0.26
	UX	0.02	0.00	0.13	1.10	7.06	***0.00**	1.20	***0.00**	***0.00**
	CMSA-TS	***0.00**	0.00	1.57	***0.00**	***0.00**	0.13	***0.00**	0.53	0.53

and, as we will see in the following, this is attributable to the perturbations afforded by the recombination

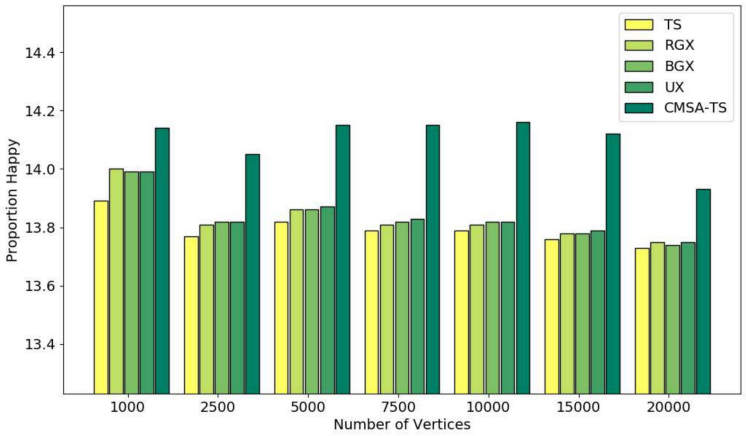
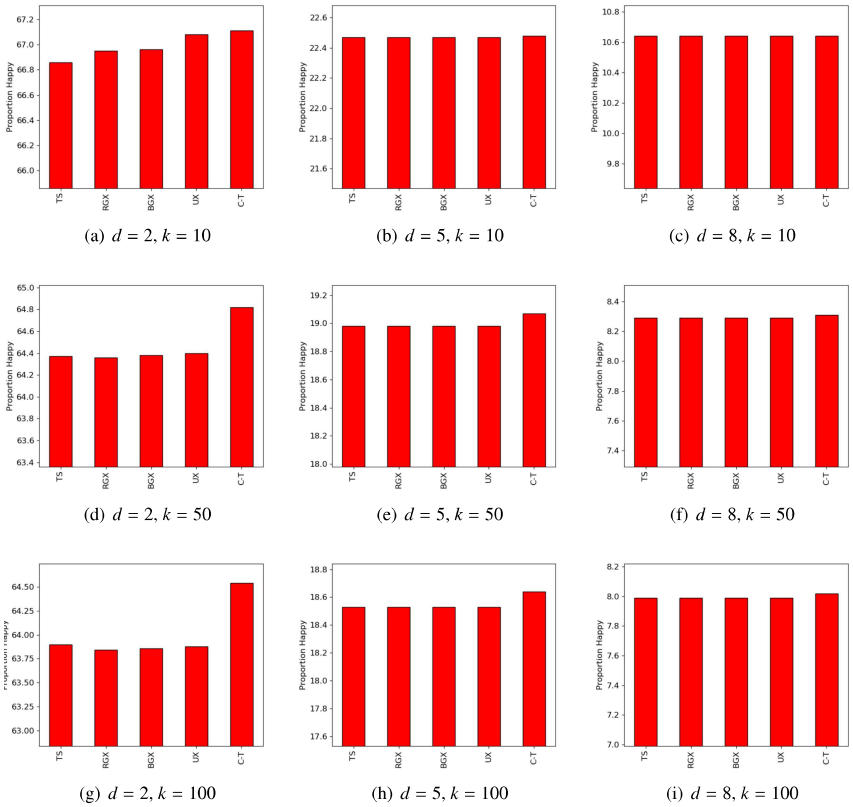


Figure 6: Proportion of happy vertices in our d-regular graphs with 50% of precoloured vertices, split by instance size.



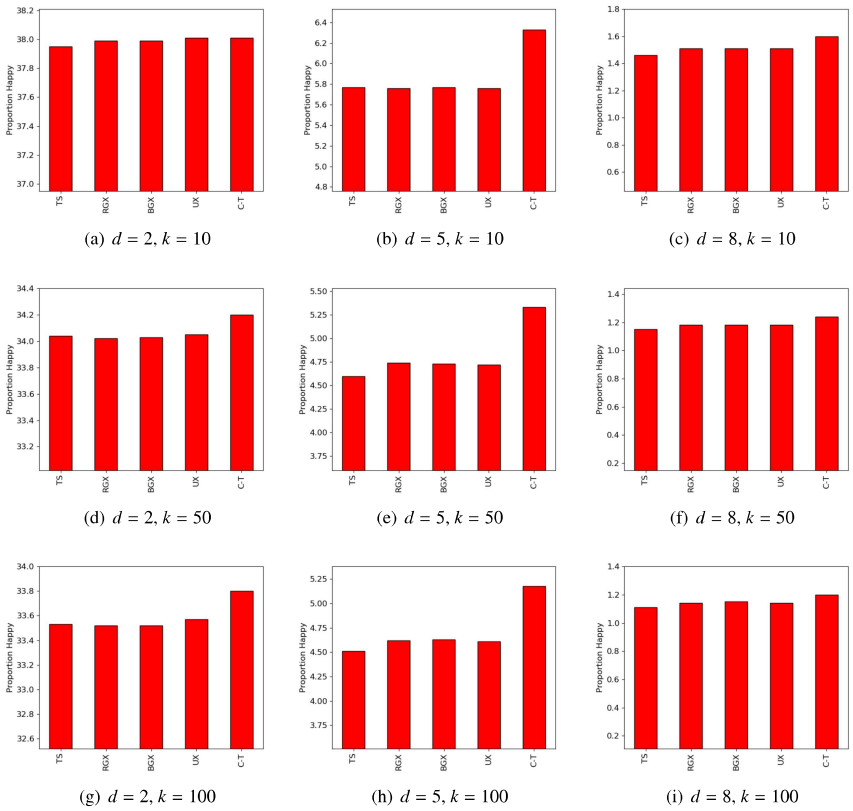
(a) $d = 2, k = 10$ (b) $d = 5, k = 10$ (c) $d = 8, k = 10$

(d) $d = 2, k = 50$ (e) $d = 5, k = 50$ (f) $d = 8, k = 50$

(g) $d = 2, k = 100$ (h) $d = 5, k = 100$ (i) $d = 8, k = 100$

Figure 7: Average proportion of happy vertices for our d-regular graphs with 25% of precoloured vertices.

operators combined with the inability of the restricted MIP to be efficiently initialised on occasion.



(a) $d = 2, k = 10$ (b) $d = 5, k = 10$ (c) $d = 8, k = 10$

(d) $d = 2, k = 50$ (e) $d = 5, k = 50$ (f) $d = 8, k = 50$

(g) $d = 2, k = 100$ (h) $d = 5, k = 100$ (i) $d = 8, k = 100$

Figure 8: Average proportion of happy vertices for our d-regular graphs with 50% of precoloured vertices.

7.2. Scale-free Graphs

We now investigate algorithm performance with scale-free graphs. Table 3 shows our results with these problem instances using the same format as previous tables.

We see again that CMSA-TS is very effective at finding the best solutions across most problem instances, though the EA using the UX operator is again superior for the sparsest graphs (lowest values for q). In addition, CMSA-TS also shows inferior performance with the largest, densest graphs where, as noted previously, the difficulty of constructing and solving the restricted MIPs prevents a sufficiently large number of algorithm iterations from taking place within the time limit.

The reasons why CMSA-TS is outperformed by the EAs in these cases, and in particular by UX, can be attributed to the single-degree vertices in the scale-free graphs and specific settings within the solution generation mechanism of CMSA. The scale-free graphs with $q = 1$ consist of many single-degree vertices and, as discussed in Section 5.3, CMSA-TS will not allow the colours of these vertices to change from what is seen in the current best solution. Hence, it is necessary to use the modified implementation CMSA-TS-F. Note that this modification does not affect the results gained with d-regular graphs as none of these contain vertices of degree one.

Table 4 shows a comparison of UX, CMSA-TS and CMSA-TS-F by reporting the proportion of happy vertices found. It is clear that allowing single-degree vertices to change colours (as would be seen in the optimal solution) leads to significant improvements, and hence better solutions are found by CMSA-TS-F

Table 3: Comparison of the proportion of happy vertices for scale-free graphs with 50% of the vertices precoloured. The figures reported are the percentage differences of each method to the best achieved across all the methods. The best result for each comparison is highlighted in bold, while statistically significant results (pair-wise Wilcoxon ranked-sum test at a confidence interval of 95%) are marked with a '*'.

	Colours k	10					50					100				
Vertices	q	1	2	4	5	8	1	2	4	5	8	1	2	4	5	8
	TS	1.25	1.00	5.49	4.80	2.94	0.96	3.54	18.59	11.26	0.00	0.79	5.78	22.25	11.89	0.00
	RGX	0.04	0.47	2.44	5.17	2.94	0.12	4.43	14.32	12.12	0.00	0.10	6.49	15.86	9.25	0.00
	BGX	0.17	0.74	2.64	5.90	2.94	0.12	4.17	13.82	12.55	0.00	***0.00**	5.97	14.58	9.25	0.00
	UX	**0.00**	0.26	2.64	3.69	2.94	***0.00**	3.04	14.07	11.26	0.00	0.06	4.74	14.07	10.57	0.00
	CMSA-TS	1.25	***0.00**	***0.00**	***0.00**	***0.00**	0.96	***0.00**	***0.00**	***0.00**	0.00	0.79	***0.00**	***0.00**	***0.00**	0.00
2500																
	TS	0.97	1.42	5.24	11.32	2.90	0.80	7.78	21.93	14.57	2.02	0.67	9.92	21.72	15.71	1.08
	RGX	0.28	1.14	5.97	12.65	2.90	0.10	7.93	15.63	14.21	2.02	0.02	12.68	15.86	14.60	1.08
	BGX	0.34	1.48	5.32	12.06	2.90	0.04	7.88	15.54	14.75	2.02	**0.00**	10.20	15.25	15.71	1.08
	UX	***0.00**	0.82	6.22	12.65	2.90	**0.00**	6.03	15.44	14.93	2.02	0.06	8.54	15.05	14.23	1.08
	CMSA-TS	0.97	***0.00**	***0.00**	***0.00**	***0.00**	0.80	***0.00**	***0.00**	***0.00**	***0.00**	0.67	***0.00**	***0.00**	***0.00**	**0.00**
5000																
	TS	0.92	2.35	11.91	13.93	4.67	0.56	10.38	21.40	15.41	5.35	0.53	12.48	22.41	14.75	5.88
	RGX	0.31	1.90	13.05	14.37	6.23	0.07	11.47	16.77	13.86	3.21	0.02	12.97	16.74	14.29	6.42
	BGX	0.32	2.29	13.31	14.67	5.06	0.00	9.63	16.26	14.68	4.28	0.00	12.25	16.54	14.56	5.35
	UX	***0.00**	0.96	13.39	14.81	5.06	**0.00**	8.54	15.81	14.49	3.21	**0.00**	11.98	16.85	15.21	5.88
	CMSA-TS	0.92	***0.00**	***0.00**	***0.00**	***0.00**	0.56	***0.00**	***0.00**	***0.00**	***0.00**	0.54	***0.00**	***0.00**	***0.00**	***0.00**
7500																
	TS	0.87	2.82	11.21	15.42	6.94	0.49	12.35	21.20	14.10	4.48	0.48	12.31	21.35	14.30	4.66
	RGX	0.38	2.63	13.20	14.92	7.18	0.05	12.29	16.25	15.21	5.07	0.02	12.46	16.28	14.74	4.97
	BGX	0.38	2.75	13.11	14.92	7.64	0.03	12.25	16.18	14.53	3.88	0.00	12.31	16.60	14.11	4.04
	UX	***0.00**	1.79	13.25	14.92	9.03	**0.00**	9.80	16.69	14.96	3.88	0.00	11.51	16.11	14.68	3.73
	CMSA-TS	0.87	***0.00**	***0.00**	***0.00**	***0.00**	0.49	***0.00**	***0.00**	***0.00**	***0.00**	0.48	***0.00**	***0.00**	***0.00**	***0.00**
10000																
	TS	0.78	3.35	13.26	15.74	5.54	0.43	12.51	20.85	14.72	5.37	0.41	12.60	20.29	14.49	4.88
	RGX	0.30	3.46	14.82	15.93	5.88	0.01	12.72	15.49	15.13	4.25	0.00	12.62	15.64	14.21	3.02
	BGX	0.26	3.43	15.11	15.81	5.88	0.02	12.55	15.62	14.90	4.25	0.02	12.51	15.27	15.10	4.42
	UX	***0.00**	2.25	15.28	15.85	5.54	**0.00**	11.49	15.47	14.72	4.03	0.00	12.10	15.27	14.49	3.95
	CMSA-TS	0.77	***0.00**	***0.00**	***0.00**	***0.00**	0.43	***0.00**	***0.00**	***0.00**	***0.00**	0.41	***0.00**	***0.00**	***0.00**	***0.00**
15000																
	TS	0.79	3.78	15.12	11.29	***0.00**	0.50	12.67	18.55	3.61	***0.00**	0.49	12.75	18.32	0.26	0.16
	RGX	0.30	3.85	14.80	10.17	0.77	0.01	12.53	13.79	3.37	0.94	0.01	12.84	13.86	0.33	***0.00**
	BGX	0.33	3.74	14.92	10.25	0.26	0.03	12.70	13.82	3.26	0.47	**0.00**	12.88	13.41	1.00	0.32
	UX	***0.00**	2.76	14.87	10.92	0.77	**0.00**	12.43	13.75	3.51	0.78	0.01	12.52	13.58	***0.00**	0.48
	CMSA-TS	0.79	***0.00**	***0.00**	***0.00**	5.36	0.49	***0.00**	***0.00**	***0.00**	5.49	0.48	***0.00**	***0.00**	1.14	5.63
20000																
	TS	0.63	2.09	9.68	0.20	0.48	0.47	3.09	5.59	0.11	0.99	0.46	7.56	5.46	0.58	0.00
	RGX	0.20	2.19	6.94	0.11	0.57	**0.00**	3.02	0.12	**0.00**	0.74	0.00	7.60	***0.00**	0.38	0.64
	BGX	0.18	2.23	7.16	***0.00**	***0.00**	0.03	3.07	***0.00**	0.03	0.62	0.00	7.57	0.14	0.16	***0.00**
	UX	***0.00**	2.05	6.84	0.18	0.76	0.02	3.02	0.14	0.05	***0.00**	0.00	7.31	0.24	***0.00**	0.90
	CMSA-TS	0.63	***0.00**	***0.00**	4.20	3.91	0.47	***0.00**	0.96	5.17	5.81	0.46	***0.00**	0.83	5.34	6.39

compared to UX. This confirms the hypothesis that CMSA-TS can be limited if degree-one vertices are not allowed to change colours.

Table 4: The proportion of happy vertices for scale-free graphs with $q = 1$. CMSA-TS-F allows single degree vertices to be modified in the solution construction process. The best result for each comparison is highlighted in bold, while statistically significant results (pair-wise Wilcoxon ranked-sum test at a confidence interval of 95%) are marked with a '*'.

	Colours k	10	50	100
Vertices				
	UX	0.530	0.499	0.492
	CMSA-TS	0.524	0.494	0.489
	CMSA-TS-F	***0.532**	***0.500**	***0.494**
2500				
	UX	0.532	0.497	0.492
	CMSA-TS	0.527	0.493	0.489
	CMSA-TS-F	***0.535**	***0.499**	***0.493**
5000				
	UX	0.528	0.496	0.492
	CMSA-TS	0.523	0.493	0.489
	CMSA-TS-F	***0.532**	***0.498**	***0.493**
7500				
	UX	0.528	0.497	0.491
	CMSA-TS	0.524	0.494	0.489
	CMSA-TS-F	***0.532**	***0.499**	***0.493**
10000				
	UX	0.525	0.494	0.491
	CMSA-TS	0.521	0.492	0.489
	CMSA-TS-F	***0.529**	***0.496**	***0.493**
15000				
	UX	0.527	0.496	0.491
	CMSA-TS	0.523	0.493	0.489
	CMSA-TS-F	***0.531**	***0.498**	***0.493**
20000				
	UX	0.525	0.495	0.491
	CMSA-TS	0.522	0.492	0.489
	CMSA-TS-F	***0.530**	***0.497**	***0.493**

7.3. Investigating the Contributions of the Different Algorithms

In this section, we aim to identify the contributions of tabu search, the EAs, and CMSA across different runs. For this purpose, we select specific problem instances (d-regular and scale-free graphs) and examine the progress of each algorithm over 600 seconds. As mentioned, we ensure that the time is split equally between tabu search and the EAs or CMSA-TS. That is, tabu search is run for the first 300 seconds followed by one of the other algorithms for the next 300 seconds. Results are presented in Figures 9, 10 and 11. The y-axes in these figures show the number of happy vertices found by each algorithm, while the x-axes show time in seconds. In particular, we have chosen instances where we can observe the specific improvements (or lack thereof) seen by each algorithm.

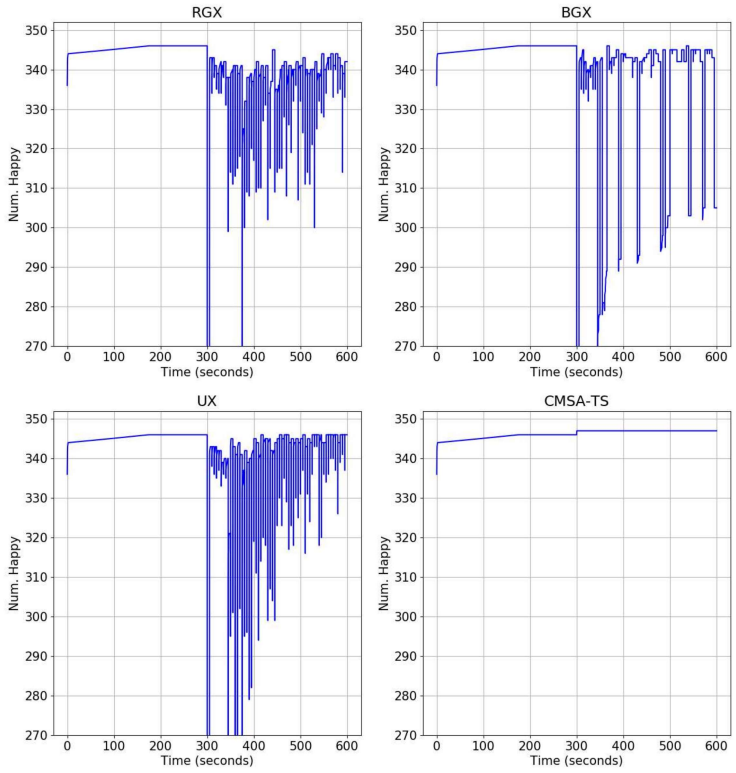


Figure 9: A single run for each algorithm on a d-regular graph with 1000 vertices, 50% precoloured vertices, 100 colours and degree of 2.

Figure 9 shows the results on a d-regular graph with 1000 vertices, a precoloured proportion of 50%, 100 colours and a vertex degree of two. We see that tabu search finds reasonable gains at the beginning (approximately forty additional happy vertices) with a slight subsequent improvement over the next 300 seconds. The randomisation of the EAs means that they typically produce low-quality solutions once they commence. There are improvements across a run, though none of the EAs improves on the solution found by tabu search. On the other hand, CMSA-TS find a small improvement as soon as it commences but solutions do not improve any further in the remainder of the run.

Figures 10 and 11 show results on scale-free graphs. We see that for the small problem instance with 2500 vertices (Figure 10), the EAs quickly find improvements after they start, though no further improvements occur across the remainder of the run. This is consistent across RGX, BGX and UX. CMSA-TS, on the other hand, finds a very large improvement shortly after the halfway mark but does not find any improvements thereafter.

Finally, for a large problem instance with 20,000 vertices (Figure 11), we see that tabu search does not really find any improvements over the initial solution. Again, the EAs find improvements at the early stages of the execution of the evolutionary component but do not improve further. CMSA-TS on the other hand

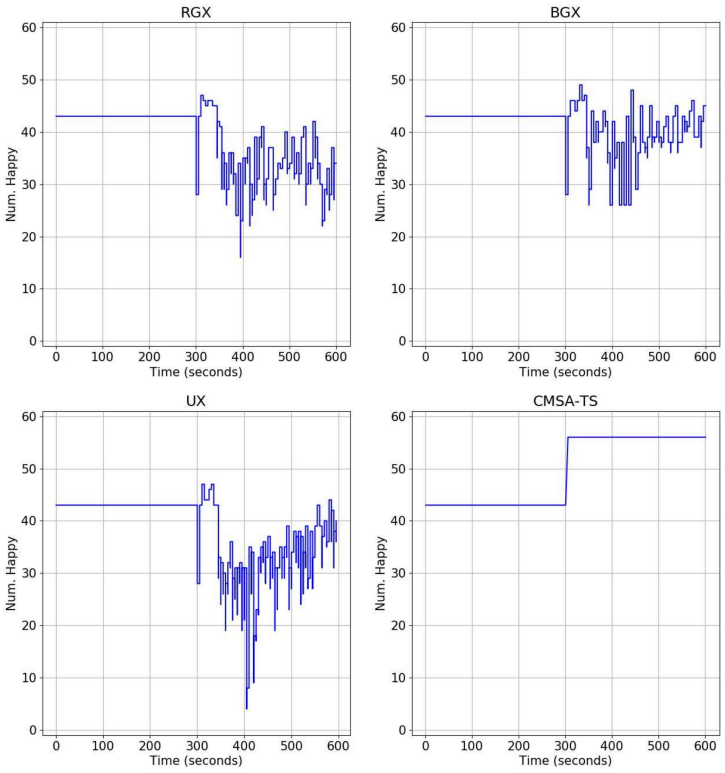


Figure 10: A single run for each algorithm on a scale-free graph with 2500 vertices, 50% precoloured vertices, 50 colours and $q = 5$.

does not find any improvements over the initial solution found by tabu search.

In summary, we see that tabu search is typically effective on small problem instances where there are a large number of happy vertices available. In this situation, the EA operators provide little assistance. Conversely, for more constrained problems with relatively few happy vertices, tabu search is less effective and the EAs do provide reasonable gains. CMSA-TS is less affected by the level of "tightness" of a problem instance, but rather by the size of the instance. We see in Figure 11, for example, that CMSA cannot improve at all. This can be attributed to the construction of the restricted MIP which consumes nearly all of the remaining execution time. As a result, the branch & bound search component of the MIP barely commences and no new solutions can be found.

8. Conclusion

This study has investigated evolutionary and matheuristic approaches for the maximum happy vertices problem. Each of these has also been augmented by an effective tabu search scheme. To investigate the performance of these algorithms, we considered various d-regular and scale-free graphs. Overall our CMSA-TS

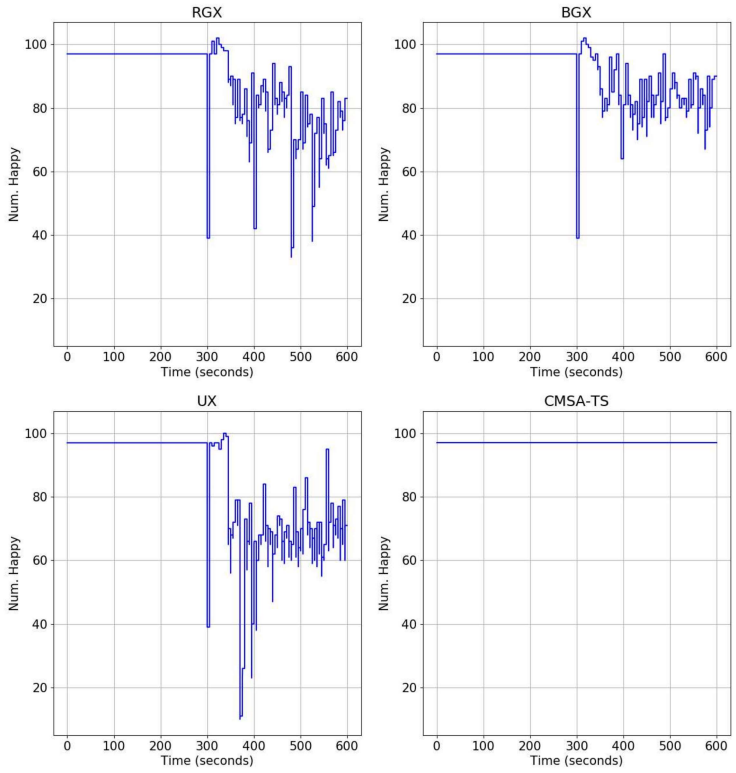


Figure 11: A single run for each algorithm on a scale-free graph with 20,000 vertices, 50% precoloured vertices, 10 colours and $q = 3$

approach produced the best performance, though are certain problem instances where the EAs are superior, particularly for very large problem instances with low vertex degrees.

There are several possible directions for future work. While CMSA-TS has been seen to be effective, we have found that the restricted MIP can require large amounts of computational resources, particularly for larger problems. Consequently, there is a need to develop alternate MIP formulations that are resource-efficient but also solve quickly. Other possibilities include constraint programming models, which allow increased flexibility in modelling with complicated constraints [Apt, 2003]. Given the complexity of the problem, parallel solution construction mechanisms might also prove useful in speeding up the algorithms. Finally, we conjecture that variants of the MHV problem, such as the maximum happy induced subgraph problem [Lewis et al., 2021] might also be effectively tackled with the CMSA-TS and EA approaches proposed in this study.

References

Agrawal, A., 2018a. On the Parameterized Complexity of Happy Vertex Coloring, in: Brankovic, L., Ryan, J., Smyth, W.F. (Eds.), Combinatorial Algorithms, Springer International Publishing, Cham. pp. 103–115.

Agrawal, A., 2018b. On the parameterized complexity of happy vertex coloring, in: Brankovic, L., Ryan, J., Smyth, W. (Eds.), Combinatorial Algorithms. IWOCA 2017. Springer, Cham.. volume 10765 of *Lecture Notes in Computer Science*, pp. 103–115.

Apt, K., 2003. Principles of Constraint Programming. Cambridge University Press. doi:10.1017/CBO9780511615320.

Aravind, N., Kalyanasundaram, S., Kare, A., 2016. Linear time algorithms for happy vertex coloring problems for trees, in: Mäkinen, V., Puglisi, S., Salmela, L. (Eds.), Combinatorial Algorithms: IWOCA 2016. Springer Cham.. volume 9843 of *Lecture Notes in Computer Science*, pp. 281–292.

Barabási, A.L., Põsfai, M., 2016. Network Science. Cambridge University Press.

Bliznets, I., Sagunov, D., 2019. Lower Bounds for the Happy Coloring Problems, in: Du, D.Z., Duan, Z., Tian, C. (Eds.), Computing and Combinatorics, Springer International Publishing, Cham. pp. 490–502.

Blöchliger, I., Zufferey, N., 2008. A Graph Coloring Heuristic using Partial Solutions and a Reactive Tabu Scheme. Computers & Operations Research 35, 960 – 975. Part Special Issue: New Trends in Locational Analysis.

Blum, C., 2016. Construct, Merge, Solve and Adapt: Application to Unbalanced Minimum Common String Partition, in: Blesa, M.J., Blum, C., Cangelosi, A., Cutello, V., Di Nuovo, A.G., Pavone, M., Talbi, E.G. (Eds.), Proceedings of HM 2016 – 10th International Workshop on Hybrid Metaheuristics, Springer - Berlin, Heidelberg. pp. 17–31.

Blum, C., Blesa, M.J., 2016. Construct, Merge, Solve & Adapt: Application to the Repetition-free Longest Common Subsequence Problem, in: Chicano, F., Hu, B. (Eds.), Proceedings of EvoCOP 2016 – 16th European Conference on Evolutionary Computation in Combinatorial Optimization, Springer - Berlin, Heidelberg. pp. 46–57.

Blum, C., Pinacho, P., López-Ibáñez, M., Lozano, J.A., 2016. Construct, Merge, Solve & Adapt A New General Algorithm for Combinatorial Optimization. Computers & Operations Research 68, 75 – 88.

Brown, E., Sumichrast, R., 2005. Evaluating performance advantages of grouping genetic algorithms. Engineering Applications of Artificial Intelligence 18, 1–12.

Carter, M.W., Laporte, G., Lee, S.Y., 1996. Examination Timetabling: Algorithmic Strategies and Applications. The Journal of the Operational Research Society 47, 373–383.

Everitt, B., Landau, S., Leese, M., Stahl, D., 2011. Cluster Analysis. John Wiley and Sons.

Falkenauer, E., 1998. Genetic Algorithms and Grouping Problems. John Wiley and Sons.

Galinier, P., Hao, J.K., 1999. Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 3, 379–397.

Huang, T., Gong, Y., Kwong, S., Wang, H., Zhang, J., 2020. A niching memetic algorithm for multi-solution traveling salesman problem. IEEE Transactions on Evolutionary Computation 24, 508–522.

Lewis, R., 2019. Tabu search source code. http://www.rhydlewis.eu/resources/happytabu.zip. Accessed: 2019-05-20.

Lewis, R., 2021. A Guide to Graph Colouring: Algorithms and Applications. 2nd ed., Springer International.

Lewis, R., Carroll, F., 2016. Creating seating plans: a practical application. Journal of the Operational Research Society 67, 1353–1362.

Lewis, R., Holborn, P., 2017. How to Pack Trapezoids: Exact and Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation 21, 463–476.

Lewis, R., Thiruvady, D., Morgan, K., 2019. Finding happiness: An analysis of the maximum happy vertices problem. Computers & Operations Research 103, 265–276.

Lewis, R., Thiruvady, D., Morgan, K., 2021. The Maximum Happy Induced Subgraph Problem: Bounds and Algorithms. Computers & Operations Research 126, 105114. doi:https://doi.org/10.1016/j.cor.2020.105114.

Lewis, R., Thompson, J., 2015. Analysing the Effects of Solution Space Connectivity with an Effective Metaheuristic for the Course Timetabling Problem. European Journal of Operational Research 240, 637 – 648.

Li, A., Zhang, P., 2015. Algorithmic Aspects of Homophyly of Networks. Theoretical Computer Science 593, 117 – 131.

McCollum, B., Schaerf, A., Paechter, B., McMullan, P., Lewis, R., Parkes, A.J., Gaspero, L., Qu, R., Burke, E.K., 2010. Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition. INFORMS J. on Computing 22, 120–130.

Misra, N., Vinod Reddy, I., 2018. The parameterized complexity of happy colorings, in: Brankovic, L., Ryan, J., Smyth, W. (Eds.), Combinatorial Algorithms. IWOCA 2017. Springer, Cham.. volume 10765 of *Lecture Notes in Computer Science*, pp. 142–153.

Polyakovskiy, S., Thiruvady, D., M'Hallah, R., 2020. Just-in-Time Batch Scheduling Subject to Batch Size, in: Proceeding of the Genetic and Evolutionary Computing Conference, Association for Computing Machinery, New York, NY, USA. pp. 228–235. URL: https://doi.org/10.1145/3377930.3390207, doi:10.1145/3377930.3390207.

Quiroz-Castellanos, M., Cruz-Reyes, L., Torres-Jimenez, J., Gomez, C., Fraire Huacuja, H., Alvim, A., 2015. A grouping genetic algorithm with controlled gene transmission for the bin packing problem. Computers and Operations Research 55, 52–64.

Steger, A., Wormald, N.C., 1999. Generating Random Regular Graphs Quickly. Combinatorics, Probability and Computing 8, 377—-396. doi:10.1017/S0963548399003867.

Thiruvady, D., Blum, C., Ernst, A.T., 2019. Maximising the Net Present Value of Project Schedules Using CMSA and Parallel ACO, in: Blesa Aguilera, M.J., Blum, C., Gambini Santos, H., Pinacho-Davidson, P., Godoy del Campo, J. (Eds.), Hybrid

Metaheuristics, Springer International Publishing, Cham. pp. 16–30.
Thiruvady, D., Blum, C., Ernst, A.T., 2020a. Solution Merging in Matheuristics for Resource Constrained Job Scheduling. Algorithms 13. doi:10.3390/a13100256.
Thiruvady, D., Lewis, R., Morgan, K., 2020b. Tackling the Maximum Happy Vertices Problem in Large Networks. 4OR , 1–21.
Wolsey, L., 1998. Integer Programming. Wiley Series in Discrete Mathematics and Optimization, Wiley.
Yan, X., Huang, H., Hao, Z., Wang, J., 2020. A graph-based fuzzy evolutionary algorithm for solving two-echelon vehicle routing problems. IEEE Transactions on Evolutionary Computation 24, 129–141.
Zhang, P., Xu, Y., Jiang, T., Li, A., Lin, G., Miyano, E., 2018. Improved Approximation Algorithms for the Maximum Happy Vertices and Edges Problems. Algorithmica 80, 1412–1438.