Quotient and Power methods for the Graph Colouring Problem

Author: *István Juhos*

Ph.D. candidate

Department of Computer Algorithms and Artificial Intelligence

Advisor: *Prof. János Csirik*

Head of the
Department of Computer Algorithms and Artificial Intelligence

Szeged 2009

Summary of the dissertation submitted for the degree of doctor of philosophy of the University of Szeged

University of Szeged
Ph.D. School in Computer Science

Introduction

This booklet summarises the scientific results of *the author* of the Ph.D. dissertation entitled "Quotient and Power methods for the Graph Colouring Problem". *The author* developed a general framework for graph colouring methods, where the traditional colouring scheme is defined via special graph homomorphisms motivated by the Zykov theorem [64; 65]. These special homomorphisms proved useful in the design of algorithms ([35; 37–43]). This summary is structured in a similar way to the thesis itself. The results can be separated into different groups according to the parts of the graph colouring framework. *The author* defined the problem via certain graph homomorphisms using quotient and power graphs. *The author* called these *Quotient and Power methods*. Then he described these graphs and homomorphisms by matrix representations with suitable operations, resulting in his *Merge Models* with his nomenclature [37; 40; 41]. Merge Models provide a novel description of the colouring problem. The operations (i.e. the *Merge Operations*) subsequently change the state of the model and direct it to a possible solution of the original graph colouring problem. *The author* developed strategies in the model called *Merge Strategies* [35; 38; 42; 43], which define possible directions to a solution. Furthermore, *the author* constructed general frameworks (*Merge Frameworks*) in which strategies can be embedded [38; 40]. These frameworks are generalisations of the traditional sequential colouring schemes, hence existing algorithm strategies can be embedded into one of them. Such an embedding may considerably decrease the computational efforts. Moreover, the embedding supports the structural analysis of the algorithms in a common way and makes available a natural extension of them, which may result in an increase in their performance. Such algorithms generate a sequence of model operations according to the strategy. The end of the sequence is a candidate solution for the original graph colouring problem. The author provided several novel algorithms in [35; 37–43]. These algorithms proved useful in an experimental analysis and theoretical study.

Graph Colouring Problem

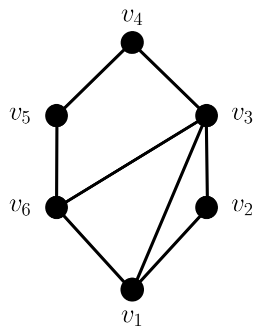
A graph is a pair $G = (V, E)$ of disjoint finite sets, where $E \subseteq V \times V$. The elements of V are the vertices of the graph G, the elements of E are its edges. Put briefly, *graph vertex k-colouring* (or simply *graph k-colouring*) is an assignment of colours from a colour set C for each vertex, where the number of the colours in the colour set C is k. The problem occurs in the colouring process when we consider edges as constraints.

Definition 1 (Proper graph vertex k-colouring) *A proper graph vertex k-colouring of $G = (V, E)$, if it exists, is a k-colouring where adjacent vertices are assigned different colours:*

$$c : V \xrightarrow{sur} C \quad , \quad v_i \mapsto c(v_i) \quad , \quad \forall (v_i, v_j) \in E \Rightarrow c(v_i) \neq c(v_j) \quad , \quad |C| = k$$

Definition 2 (Graph minimum vertex colouring) *Graph minimum vertex χ-colouring is a proper χ-colouring, where χ is the smallest integer needed to get a proper colouring.*

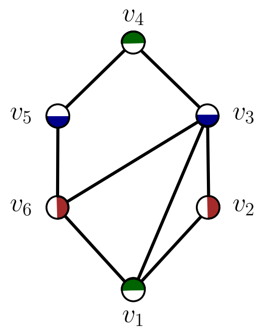
Here the Graph Colouring Problem is the graph minimum vertex colouring problem. An example for this can be found in Figure 1. The smallest number of colours that can properly colour vertices is called the **chromatic number** of a graph and will be denoted by χ. Figure 1(a) shows a



(a) A graph

	v_1	v_2	v_3	v_4	v_5	v_6
v_1	0	1	1	0	·	1
v_2	1	·	1	·	·	·
v_3	1	1	·	1	·	1
v_4	0	·	1	0	1	·
v_5	·	·	·	1	·	1
v_6	1	·	1	·	1	·

(b) Adjacency matrix



(c) A proper colouring

Figure 1: A graph and a proper 3-colouring, which is the minimum.

drawing of a graph, while the corresponding adjacency matrix of the graph [1] is illustrated in Figure 1(b), which describes the edge relation between the vertices and Figure 1(a). Figure 1(c) shows a proper colouring of the graph, which is a minimum as well, where sets $\{v_1, v_4\}$, $\{v_2, v_6\}$ and $\{v_3, v_5\}$ are the colour classes. Colour classes must form independent sets[2] in order to get a proper colouring. The 0-s in Figure 1(b) represent an independent set.

Lots of algorithms have been created and studied to solve the graph minimum vertex colouring problem. Actually, these algorithms come in two main types: the exact algorithms where finding of a solution is guaranteed, but the time involved may be considerable due to the complexity of the problem (- which is NP-complete [45]); and the non-exact algorithms, that is, the approximation algorithms where a solution is not guaranteed but one may find a solution or a good approximation of it in a reasonable time. The latter methods may have stochastic components. Some recent surveys of these methods can be found in [23; 32; 47; 63] The graph colouring problem can be solved exactly by an exhaustive search, i.e. systematically exploring a search space [15; 16; 34]. Unfortunately, when the size of the instances grows the running time for an exhaustive search soon become prohibitively large, even for instances of fairly small search space. To improve the efficiency of the search, several heuristics were developed to generate a 'good' starting candidate solution which may be close to an optimal solution [4; 17; 26–28; 46; 49; 55; 58; 60–62]. Then starting the exploration process with the generated candidate solution, a systematic search can considerably improve the performance. Usually, the exploration is based on an examination of the local environment of the generated solution and it assumes that a neighbourhood relation is defined on the elements of the search space. This approach led to the development of local search methods [1; 8; 10; 23; 29; 31]. These methods usually apply some heuristic to generate a new candidate solution from an existing one in its local environment. But though a heuristic can considerably improve a solution they do not always provide an optimal solution, hence these

[1] The 0-s have been replaced by dots for the sake of clarity.
[2] There is no edge between the vertices.

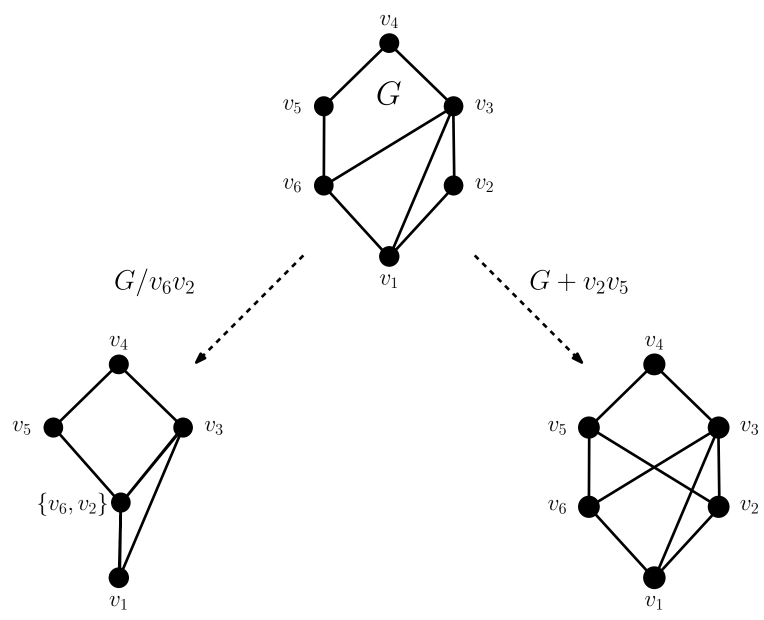


Figure 2: Connection and contraction steps in a Zykov-tree. Assigning the same colour for v_2 and v_5 must be avoided, because this always results in a non-optimal colouring, hence v_2v_5 edge addition is reasonable, but the same colour assignment for v_2 and v_6 supports the minimum colouring, therefore they can be contracted.

methods belong to the class of approximate algorithms. Many algorithms studied today employ a stochastic process in the local search to guide a candidate solution to a suboptimal solution or, hopefully, to an optimal solution. Several of these approaches maintain a population of candidate solutions. Examples of such methods include tabu-search [3; 31], simulated annealing [9; 33] and ant colony optimisation [5; 12]. One popular approach for dealing with graph colouring is evolutionary computation [2; 13; 14; 19; 20; 22; 25; 30; 48; 56; 59]. In the development of algorithms for graph colouring, various integer programming formulations of the problem could be used. Several such formulations, usually involving binary variables, have been proposed. These variables can identify different structures: e.g. independent sets [50]; a variable for each possible colour and vertex [11; 51; 53]; acyclic orientations of a graph [21]. In several formulations an optimal solution can be represented as a binary vector of the variables. These binary vectors constitute a polytope, a colouring polytope. These polytopes are the central topics of the analysis of the problem based on integer programming approaches [6; 24]. Several relaxed versions of these integer programmes have been developed to approximate a face of a colouring polytope [18; 44; 50; 52; 57]. Different techniques may improve the efficiency of these methods e.g. column generation with branch-and-bound [7; 50; 57] or branch-and-cut [53]. Actually the branch-and-bound technique implicitly uses Zykov's idea (see [57]). In the middle of the last century Zykov came up with the idea of applying an edge addition or vertex contraction instead of a colour assignment in the colouring problem (see Figure 2). During these operations new graphs are created from the original one which may inherit the parent graph's properties.

In the thesis we generalise Zykov's approach by introducing different models (Merge Models). We will demonstrate the efficiency of these novel models via a theoretical analysis and experimental study. Merge Models reformulate the original problem. In this reformulated environment three different general frameworks will be introduced to describe an abstraction for algorithms based on the Merge Models. They provide a uniform and compact way in which algorithms can be defined. Embedding algorithms in the same common framework supports both their structural and performance comparison, which can be anyway problematic. Traditional colouring schemes can be identified in one of the frameworks and extended schemes may be provided. The framework itself generalises the formal sequential colouring approach. With this generalisation an algorithm can be extended in a natural way, which may result in new algorithms. The novel aspect of the Merge Models implies the development of novel colouring strategies, i.e. Merge Strategies. The Merge Models describe special graph homomorphisms, hence their analysis may reveal connections between strategies and different graph properties. Many novel efficient Merge Strategies will be provided which outperform several standard benchmark algorithms. In addition a general strategy design is discussed, which allows the application of machine learning techniques in the algorithm design.

Quotient and Power Methods

The author defined graph colouring processes as a series of homomorphisms using quotient or power graphs and multigraphs, where the vertices which get the same colour will be 'glued' or 'grouped' together to form new vertex sets (see Juhos et al. [37; 41]), as illustrated in Figure 3. *The author* called the new colouring methods which are based on these principles Quotient and Power methods. The goal of a Quotient/Power method is to find a suitable homomorphism which maps the original graph [3] into a complete graph or an appropriate graph which is homomorphic with a complete graph. The homomorphism obtained defines a colouring of the original graph. In order to support the design of sequential colouring algorithms a homomorphism is created as a composition of a series of intermediate homomorphisms. These homomorphisms produce helpful intermediate graph structures which may be exploited for an efficient colouring and also help to provide a deeper insight into the colouring procedure. Moreover, they allow us to design efficient new algorithms or redesign existing graph colouring algorithms in a framework supported by quotient or power graphs (see Juhos et al. [37–43]).

Merge Models

The relation between the original graph and a quotient or power graph/multigraph is defined by a graph homomorphism. *The author* introduced four kinds of matrix operations, called Merge Operations (or 'merges' for short) to map the adjacency matrix of the original graph to its four different homomorphic images: called Binary/Integer Merge Square ($A/\mathbb{A}$) and Binary/Integer Merge Table ($T/\mathbb{T}$) matrices [37; 41]. In general they are referred to as a Merge Matrix (M).

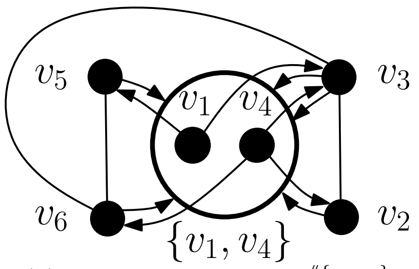
[3] Or an equivalent reformulation of the original graph.

Subsequent Merge Operations will produce a composition of the homomorphisms until no further Merge Operation is possible. The merge condition is defined by $M_{ij} = 0$; that is, in this case M_i and M_j rows (and columns) are mergeable, as can be seen in Figure 1(b). The merge results in a $M_{/ij}$ Merge Matrix. With the last possible merge, the last homomorphic image defines a candidate solution for the colouring, where the merged rows and the corresponding vertices determine the colour classes. Figure 3 shows an example for each Merge Operation, while Table 1 gives an exact description of them via row and matrix-based formulations. Moreover, Table 1 shows a relation between the Binary Merge Matrices (A and T) and their Integer counterparts $\mathbb{A}$ and $\mathbb{T}$. Merge Tables characterise a relation between the original vertices and the neighbouring

Row-based formula		Matrix-based formula
$\mathbb{T}_i^{[t+1]} = \mathbf{a} + \mathbf{b}$	$\mathbb{T}_j^{[t+1]} = \mathbf{0}$	$\mathbb{T}^{[t+1]} = (I + W)\mathbb{T}^{[t]}$
$T_i^{[t+1]} = \mathbf{a} \vee \mathbf{b}$ $T_i^{[t+1]} = \mathbb{T}_i^{[t+1]} - \mathbf{a} \circ \mathbf{b}$	$T_j^{[t+1]} = \mathbf{0}^T$ $T_j^{[t+1]} = \mathbf{0}^T$	$T^{[t+1]} = T^{[t]} \vee PT^{[t]} - MT^{[t]}$ $T^{[t+1]} = \mathbb{T}^{[t+1]} - \sum_j (\mathbf{a} \otimes \mathbf{b})(I_j \otimes I_i)$
$\mathbb{A}_i^{[t+1]} = \mathbf{a} + \mathbf{b}$ $\mathbb{A}_{\_i}^{[t+1]} = \mathbf{a}^T + \mathbf{b}^T$	$\mathbb{A}_j^{[t+1]} = \mathbf{0}^T$ $\mathbb{A}_{\_j}^{[t+1]} = \mathbf{0}$	$\mathbb{A}^{[t+1]} = (I + W)\mathbb{A}^{[t]}(I + W)^T$
$A_i^{[t+1]} = \mathbf{a} \vee \mathbf{b}$ $A_i^{[t+1]} = \mathbb{A}_i^{[t+1]} - \mathbf{a} \circ \mathbf{b}$ $A_{\_i}^{[t+1]} = (A_i^{[t+1]})^T$	$A_j^{[t+1]} = \mathbf{0}^T$ $A_j^{[t+1]} = \mathbf{0}^T$ $A_{\_j}^{[t+1]} = \mathbf{0}$	$A^{[t+1]} = A^{[t]} \vee (PA^{[t]}P^T) - (MA^{[t]}M^T)$

Table 1: Description of different merge operations when the $\mathbf{a}$ and $\mathbf{b}$, the i-th and j-th mergeable rows of a Merge Matrix are merged. The superscript $^{[t]}$ defines the t-th merge step and $P = I_i \otimes I_j$, $R = I_j \otimes I_j$, $W = P - R$, where I_i is the i-th row of the identity matrix I. M_i stands for the i-th row of a matrix M, while $M_{\_i}$ denotes its the column. The operation $\circ$ is called the Hadamard-Schur product, while $\otimes$ is called the dyadic product.

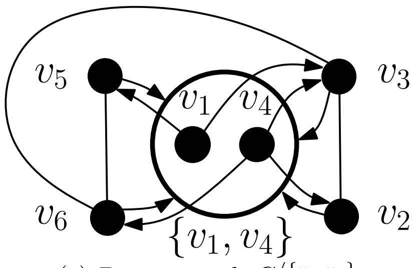
colour classes. We may associate rows of an adjacency matrix with colour classes or power vertices and columns with vertices of the original graph. As previously mentioned, there are two subtypes, namely a weighted type (Integer for power multigraphs) and an unweighted type (Binary for power graphs), based on whether the number of the multiple edges are taken into account in the merging process. Hence, there are two basic row operations the addition and the piecewise binary OR operations. When they are applied on the rows only, we arrive at power multigraphs/graphs, i.e. Integer/Binary Merge Tables, but applying them on the rows and on the relevant columns as well, we arrive at a quotient multigraph/graph, i.e. Integer/Binary Merge Squares. In Merge Squares, the rows and the columns of the matrix correspond to colour classes, and their edges define a relation between the colour classes. The representations and the operations form new colouring models, called Merge Models. Each row of a Merge Matrix corresponds to an independent set in the original graph. Recall that colour classes are independent sets and each vertex constitutes a one-element independent set in the original graph. Actually, a Merge Operation creates the union of two independent sets in the traditional sense. Figure 4 shows how the structures of different Merge Matrices are related to the appropriate graphs. These models support parallel software and hardware implementations. All the models have their own strong points, and they can assist each other in different ways. *The author* obtained significant improvements, both theoretically



(a) Power multigraph $G^{/\!\!/\{v_1,v_4\}}$

	v_1	v_2	v_3	v_4	v_5	v_6
$\{r_1, r_4\}$	0	1	2	0	1	1
r_2	1	·	1	·	·	·
r_3	1	1	·	1	·	1
r_5	·	·	·	1	·	1
r_6	1	·	1	·	1	·

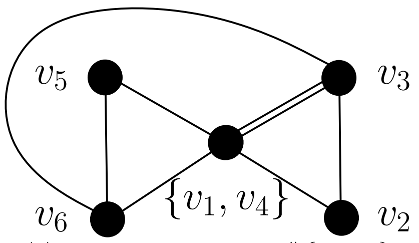
(b) Integer Merge Table. Addition is performed on the r_1 and r_2 rows.



(c) Power graph $G^{/\{v_1,v_4\}}$

	v_1	v_2	v_3	v_4	v_5	v_6
$\{r_1, r_4\}$	0	1	1	0	1	1
r_2	1	·	1	·	·	·
r_3	1	1	·	1	·	1
r_5	·	·	·	1	·	1
r_6	1	·	1	·	1	·

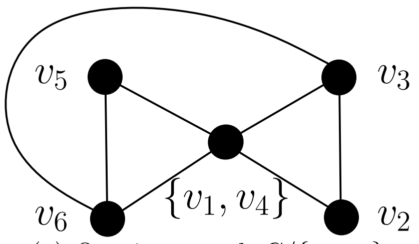
(d) Binary Merge Table. Piecewise OR operation is performed on the r_1 and r_2 rows.



(e) Quotient graph $G /\!\!/ \{v_1, v_4\}$

	$\{v_1, v_4\}$	v_2	v_3	v_5	v_6
$\{r_1, r_4\}$	0	1	2	1	1
r_2	1	·	1	·	·
r_3	2	1	·	·	1
r_5	1	·	·	·	1
r_6	1	·	1	1	·

(f) Integer Merge Square. Additions are performed on the r_1 and r_2 rows and v_1 and v_4 columns.

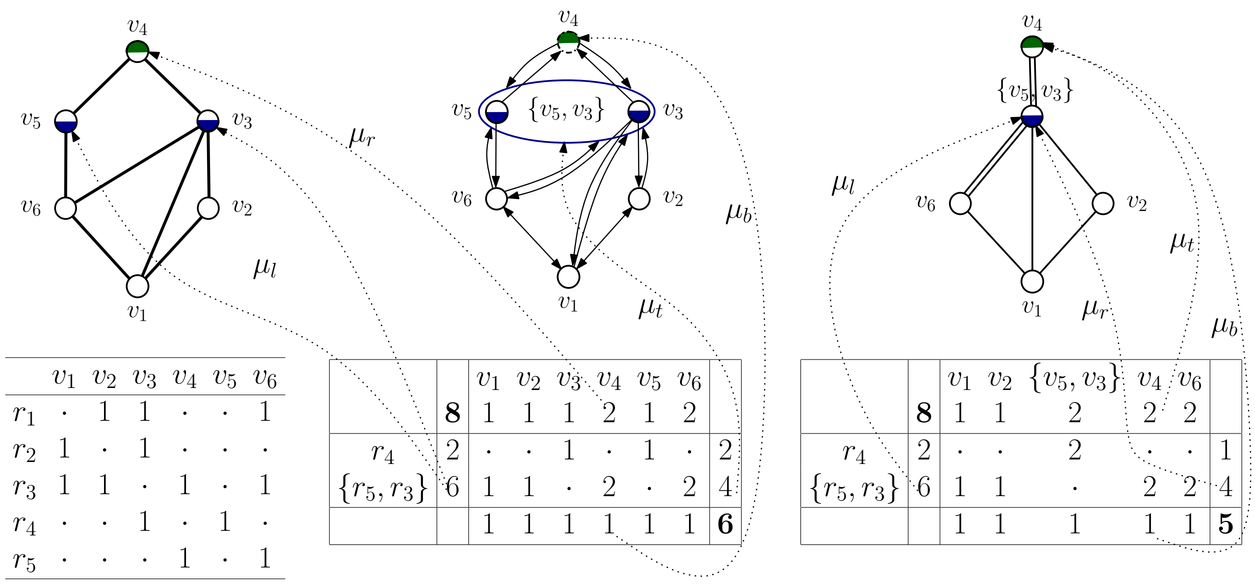


(g) Quotient graph $G/\{v_1, v_4\}$

	$\{v_1, v_4\}$	v_2	v_3	v_5	v_6
$\{r_1, r_4\}$	0	1	1	1	1
r_2	1	·	1	·	·
r_3	1	1	·	·	1
r_5	1	·	·	·	1
r_6	1	·	1	1	·

(h) Binary Merge Square. Piecewise OR operations are performed on the r_1 and r_2 rows and v_1 and v_4 columns.

Figure 3: Results of different merge operations.



	v_1	v_2	v_3	v_4	v_5	v_6
r_1	·	1	1	·	·	1
r_2	1	·	1	·	·	·
r_3	1	1	·	1	·	1
r_4	·	·	1	·	1	·
r_5	·	·	·	1	·	1

		v_1	v_2	v_3	v_4	v_5	v_6	
	8	1	1	1	2	1	2	
r_4	2	·	·	1	·	1	·	2
$\{r_5, r_3\}$	6	1	1	·	2	·	2	4
		1	1	1	1	1	1	**6**

		v_1	v_2	$\{v_5, v_3\}$	v_4	v_6	
	8	1	1	2	2	2	
r_4	2	·	·	2	·	·	1
$\{r_5, r_3\}$	6	1	1	·	2	2	4
		1	1	1	1	1	**5**

Figure 4: The original graph, a sub-Integer Merge Table and then a sub-Integer Merge Square of coloured vertices when colouring is in progress. Co-structures occur on the sides of the sub-matrices and they sum and count the non-zero elements. Here μ_l denotes the sum of the degree of the vertices in a colour class, μ_r denotes the number of adjacent vertices of a colour class, μ_t denotes the number of adjacent coloured vertices, and μ_b stands for the number of adjacent colour classes.

and via experiment, when an algorithm applied one of these models [40]. Exploiting their benefits, *the author* used the models to design powerful graph colouring algorithms in [35; 38–40; 42; 43].

Merge Frameworks

Merge Models provide a model for the graph colouring problem via matrix representations and operations. *The author* introduced three general frameworks for graph colouring algorithms supported by Merge Models in [41; 42]. These are generalisations of the traditional sequential colouring schemes. Merge Models replace the colour assignment operation with a Merge Operation, and this eliminates the difference between the colour selection and the vertex selection strategies. Merge Models define these different selection strategies in a common way as a common row selection strategy. Therefore, a general row selection strategy can operate as a coloured or uncoloured row selection when we would like to model the traditional selection strategies. Here the colours only indicate whether a row has already been taken into account in the merge process. Depending on the order of the selection of the different (coloured/uncoloured) state rows, two general frameworks can be defined: either we choose an **u**ncoloured row first and then choose a suitable **c**oloured one (UC Merge Framework) or, conversely, we can choose a **c**oloured first and then find an appropriate **u**ncoloured row for the merge (CU Merge Framework) [41]. The UC and the CU frameworks provide a generalisation of the sequential colouring schemes (see Figure 5). The choose-unc and choose-col functions/strategies are not defined precisely here. They can be replaced by different concrete choice strategies which operate on coloured (M^{col}) and un-

```
UC Merge Framework($A$ adjacency matrix )
1  $M \leftarrow A$
2  repeat
3          $u \leftarrow \arg \text{choose-unc}_i\{M_i^{unc}\}$ //Choose an uncoloured row index
4          $c \leftarrow \arg \text{choose-col}_i\{M_i^{col}\}$ //Choose a coloured row index,[a] where $M_{uc} = 0$
5          $M \leftarrow \text{merge}(M, \{u, c\})$ //Merge $u$ and $c$ rows/columns [b]
6    until $M^{unc}$ is empty
7  return $M$
```

```
CU Merge Framework($A$ adjacency matrix )
1  $M \leftarrow A$
2  repeat
3          $c \leftarrow \arg \text{choose-col}_i\{M_i^{col}\}$ //Choose a coloured row index
4          $u \leftarrow \arg \text{choose-unc}_i\{M_i^{unc}\}$ //Choose an uncoloured row index[c], where $M_{cu} = 0$
5          $M \leftarrow \text{merge}(M, \{u, c\})$ //Merge $u$ and $c$ rows/columns
6    until $M^{unc}$ is empty
7  return $M$
```

[a] $M_{uc} = M_{cu} = 0$ is the merge condition, i.e. there is no edge.
[b] For Merge Squares, columns are also affected in a Merge Operation.
[c] $M_{cu} = M_{uc} = 0$ is the merge condition, i.e. there is no edge.

Figure 5: The UC and CU Merge Frameworks

coloured (M^{unc}) sub-merge-matrices, respectively. These sub-merge-matrices consist of coloured (M_i^{col}) and uncoloured (M_j^{unc}) rows of the original merge matrix. Figure 4 shows examples for coloured sub-merge-matrices. The choose-unc function selects an uncoloured row/vertex, while choose-col selects a coloured row/'colour class' or allocates a new empty row in the coloured sub-merge-matrix, in order to support the one-operand Merge. In fact, there is no need to distinguish between the coloured or uncoloured states of the rows; just take the set of rows and apply a common choose strategy suitable for all of them. After, select an arbitrary row-pair from the Merge Matrix by a strategy and merge them. This approach is formulated in the CC Merge Framework [37], as shown in Figure 6. The rows of the Merge Matrix correspond to

```
CC Merge Framework($A$ adjacency matrix )
1  $M \leftarrow A$
2  repeat
3          $\{i, j\} \leftarrow \arg \text{choose}_{\{i,j\}}\{M_i, M_j : i \neq j\}$ //Choose two row indices[a], where $M_{ij} = 0$
4          $M \leftarrow \text{merge}(M, \{i, j\})$ //Merge $i$ and $j$ rows/columns
5    until $M$ is not mergeable
6  return $M$
```

[a] $M_{ij} = M_{ji} = 0$ is the merge condition, i.e. there is no edge.

Figure 6: The CC Merge Frameworks

colour classes, i.e. independent sets. An algorithm in a CC Merge Framework selects two colour classes/independent sets and creates the union of them in the traditional sense. The CC Merge Framework is the most general. Even though it covers the UC and CU Merge Frameworks, it is worth defining them separately so as to have the possibility of categorising the algorithms later Moreover, it is useful in the identification of the traditional schemes. These general frameworks with the new Merge Models support a common structural analysis of the existing and novel graph colouring methods, as shown in [38; 40; 42; 43]. *The author* demonstrated improvements in the performance of an algorithm after embedding it into a suitable Merge Model. Without any change in the algorithm steps, the representation of the problem in a Merge Model leads to a reduction in the computational cost. In [19; 59], Eiben and van Hemert et al. pointed out that the number of constraint checks is a key factor in the computational cost in most of the colouring algorithms. In traditional schemes, the adjacency matrix representation plays the key role in the GCP[4]. We have two choices when colouring a vertex for constraint checking; either along the already coloured vertices ($\mathcal{A}_{col}$), or along all the neighbours of the vertex considered ($\mathcal{A}_{neigh}$). In the following we will show how to markedly reduce the number of constraint checks by applying our proposed Merge Models ($\mathcal{A}_{mm}$).

Corollary 1 ([40]) *Given a random graph $G_{n,p}$ with fixed p edge probability and given a colouring algorithm $\mathcal{A}$, then the following performance is expected on average based on counting constraint checks $\#(.)$:*

1. *Checking the coloured vertices: $\#(\mathcal{A}_{col}) = \mathcal{O}(n^2)$*
2. *Checking the neighbours: $\#(\mathcal{A}_{neigh}) = \mathcal{O}(n^2)$*
3. *Checking the merged-vertices/colour classes: $\#(\mathcal{A}_{mm}) \leq \mathcal{O}\left(\frac{n^2}{\log n}\right)$*

As the theorem above tells us the asymptotic performance of the algorithms, we can check the worst case performance of a colouring algorithm using these different approaches.

Corollary 2 ([40]) *Let G be an arbitrary graph, then the following relations hold*

1. $\#(\mathcal{A}_{mm}) \leq \#(\mathcal{A}_{col})$
2. $\#(\mathcal{A}_{mm}) \leq \#(\mathcal{A}_{neigh})$

All of these frameworks are defined in a unified manner using the Merge Model scheme. An algorithm in one of these frameworks applies a subsequent selection of rows of the merge matrices and merges them to achieve a colouring. None of these frameworks has a concrete strategy for the choice of rows for merging. A framework with a concrete choice strategy, i.e. Merge Strategy, forms a particular algorithm.

Merge Strategies

In order to get a colouring algorithm, the algorithm steps must be defined; that is, a sequence of the Merge Operations. A Merge Operation takes two rows/columns of a Merge Matrix and

[4]List based or incidence matrix representations require more operations for graph colouring.

produces a new Merge Matrix if the merge condition allows it. By repeating Merge Operations we will end up with a final Merge Matrix where a Merge Operation is no longer possible. The sequence of the Merge Operations is crucial. It determines the quality of the solution, i.e. the number of colours used in the colouring of the original graph. *The author* described various Merge Strategies in order to generate efficient merge sequences, as described in [35; 37–43]. These strategies proved useful in the theoretical and experimental parts of our analysis. The novel description of the colouring process provides new aspects which can be exploited in the design and analysis of Merge Strategies, as described in the following. This strategies assume Binary Merge Models, but their integer extensions are also available. The importance of the Integer Models are discussed separately. They support the algorithm design, e.g. backtracking or tie breaking, as shown in [40]. The following strategies define row-pair selection strategies; that is, they support the most general strategy, the CC framework choose function. Hence they are suitable for the UC and CU frameworks as well. Let $\hat{X}$ be the basis of the row-pair selection, i.e. the choose function. An $\hat{X}_{ij}$ element of the matrix is proportional to the probability of the selection of row i and j for a merge in the next algorithm step [5]. The following strategies will define $\hat{X}$ values. The choose function applies the following selection $\{i, j\} = \arg\max_{ij} \hat{X}_{ij}$.

The longest merge sequence. Since the Merge Matrix rows (M_i-s) represent colour classes, the main aim is to reduce the number of rows by consecutive merges. The longest merge sequence produces the fewest rows. *The author* in [38] introduced two novel strategies to generate the longest merge sequence. The Dot Product Strategy focuses on the evolution of the number of non-zero elements during successive merges and attempts to keep them as low as possible, using

$$\hat{X}_{ij} = \langle M_i, M_j \rangle \, [M_{ij} = 0] \tag{1}$$

where $[M_{ij} = 0]$ is the Kronecker delta function, where $[x = x] := 1$, otherwise it is 0 (this encodes the merge condition) M_i and M_j are the i-th and j-th rows of a Binary Merge Matrix [6]. For a Binary Merge Square A, it can be defined by

$$\hat{X} = AA^T \circ \bar{A} \tag{2}$$

Though the non-zero elements in a Merge Matrix frustrate the merges, the number of zeros assist them. Hence the Cosine Strategy takes the number of non-zero elements into account, but also considers the number of zeros present, using

$$\arg\max_{i,j} \hat{X}_{ij} = \arg\max_{i,j} \frac{\langle M_i, M_j \rangle}{|M_i| \, |M_j|} [M_{ij} = 0] = \arg\max_{i,j} \frac{\langle M_i, M_j \rangle}{||M_i|| \, ||M_j||} [M_{ij} = 0] \tag{3}$$

Parallel rows. The Cosine strategy favours the parallel rows in the Merge Matrices. It is reasonable because the rows of the adjacency matrix which correspond to the same coloured vertices in an optimal solution are almost parallel. Their parallel behaviour becomes clearer with

[5] $\hat{X}$ may change during the steps

[6] For Binary Merge Matrices $[M_{ij} = 0] \equiv (1 - M_{ij})$.

each successive merge. For the Merge Square Model, there is a certain modification of the Merge Matrices based on a semi-definite optimisation by Karger et al. [44], which further supports the Cosine strategy. Exploiting this fact, *the author* in [35; 43] defined the Zykov-tree and Lovász-theta strategy.

Colour similarities Actually, a 'Zykov-tree and Lovász-theta' strategy is based on the estimation of the colour similarities of the vertices of the quotient graphs. The adjacency matrix describes an *exact* colour dissimilarity relation, where the vertices in a relation defined by the edges cannot get the same colour. The opposite approach is the colour similarity relation. A particular colouring can be defined via a colour similarity relation between the vertices, where only the same coloured vertices are included in the relation. This relation can be represented by a $\{0,1\}$-matrix, namely a colouring matrix. It describes whether two vertices are coloured with the same or different colours. Although the optimal solutions can be represented in this form, they are unknown because they are the solutions of the problem. Despite this, their average can be approximated by a solution of a semi-definite program (see Karger et al. [44]), which provides the Lovász-theta $\bar{\theta}$:

$$\bar{\theta} = \min_t \{t : Z \succeq 0, z_{ii} = t-1, z_e = -1 \; \forall e \in E\} \tag{4}$$

Hence a *non-exact* (an approximated) colour similarity relation becomes available between the vertices. This can be described by a real-valued matrix Z_{opt}, which is a solution of Eq. 4 and we can define a matrix $\hat{X}$ by

$$\hat{X} = (Z_{opt} + 1) \circ (1 - I)$$

where I is the identity matrix. The largest and the smallest values of $\hat{X}$ contain valuable information. Using this information and Zykov's work in [64; 65], *the author* created the 'Zykov-tree and Lovász-theta' strategy in [35; 43], where quotient graph vertices are connected ($\arg\min_{i,j}\{\hat{X}_{ij} : \hat{X}_{ij} < 0\}$) or merged ($\arg\max_{i,j} \hat{X}_{ij}$) according to their approximated similarities (see Figure 2). The approximation becomes more exact with each subsequent merge, supporting more confident decisions of this strategy. To speed-up the algorithm, multiple edge additions ($\hat{X}_{ij} < 0$) or merges ($\hat{X}_{ij} > 0.5\bar{\theta}$) can be performed.

Norm minimisation in the resulting state. The Dot Product Strategy selects two rows M_r and M_s which produce the maximum dot product $\{r,s\} = \arg\max_{i,j} \langle M_i, M_j \rangle$, then merges them. This introduces a minimisation in the entrywise 1-norm[7] in the resulting Merge Matrix $|M_{/rs}| = |M| - \max_{i,j} \langle M_i, M_j \rangle = |M| - \langle M_r, M_s \rangle$, thus

$$\arg\left(|M| - \max_{i,j} \langle M_i, M_j \rangle\right) = \arg\min_{i,j} \left(|M| - \langle M_i, M_j \rangle\right) = \arg\min_{i,j} |M_{/ij}| \tag{5}$$

A final Merge Matrix which corresponds to an optimal solution has the smallest entrywise norm among the possible merge matrices (homomorphic images). Hence, the entrywise norm minimisation approach is reasonable. In addition such a Merge Matrix has minimal induced norms

[7]This is valid for all entrywise norms.

as well. This observation led us to apply the steepest descent norm minimisation strategy, in particular the steepest descent Spectral Norm Strategy (induced 2-norm), which was introduced by *the author* in [42] and was found to be an efficient strategy with

$$\hat{X}_{ij} = \begin{cases} \frac{1}{||M_{/ij}||_2} [M_{ij} = 0] & i \neq j \\ 0 & i = j \end{cases} \tag{6}$$

The Spectral Norm Strategy must first make several trial merges. With the resulting trial merge matrices $M_{/ij}$, this strategy makes spectral norm calculations $||M_{/ij}||_2$ to create a selection of a row-pair for merging. Calculating the spectral norm is computationally expensive, but Merikoski and Kumar once introduced an efficient spectral norm approximation in [54]. Let $M = A$ be a Binary Merge Square then

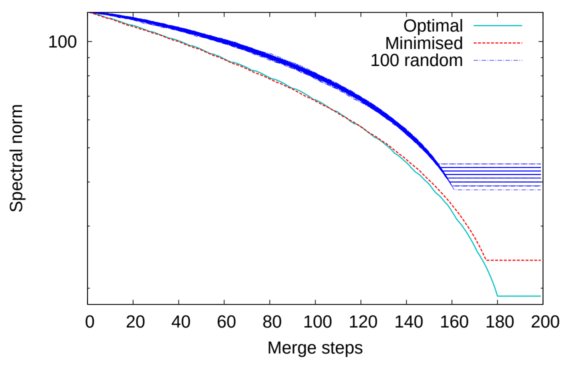
$$||A_{/ij}||_2 \approx \sqrt{\frac{\sum_{r=1}^{l} \left\langle (A_{/ij})_r, \mathbf{e} \right\rangle^2}{l}} \tag{7}$$

where l is the number of rows of the 'trial' Merge Matrix $A_{/ij}$. Based on Merikoski and Kumar's results, *the author* adapted his Spectral Norm Strategy to an approximated spectral norm strategy [42]. Owing to this, this strategy can exploit an update mechanism where an investigation of the resulting Merge Matrices is no longer needed as it is just based on the current Merge Matrix

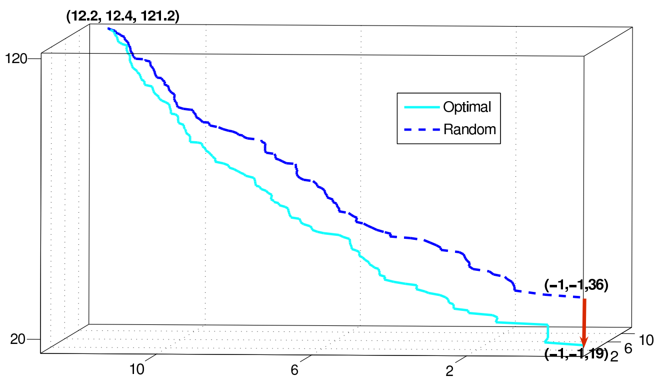
$$\begin{aligned} \left\langle (A_{/ij})_i, \mathbf{e} \right\rangle &= \langle A_i, \mathbf{e} \rangle + \langle A_j, \mathbf{e} \rangle - \langle A_i, A_j \rangle \\ \left\langle (A_{/ij})_j, \mathbf{e} \right\rangle &= 0 \\ \left\langle (A_{/ij})_r, \mathbf{e} \right\rangle &= \langle A_r, \mathbf{e} \rangle - 1 \quad r \in \mathcal{I} \\ \left\langle (A_{/ij})_r, \mathbf{e} \right\rangle &= \langle A_r, \mathbf{e} \rangle \quad r \notin \mathcal{I} \cup \{i, j\} \end{aligned} \tag{8}$$

where $\mathcal{I}$ is an index set, the set of the common one positions of the rows A_i and A_j. Hence Eq. 7 can be directly calculated from the Merge Matrix values without any trial merges. In addition, this reformulation revealed a connection with the Dot Product strategy. Eq. 7 provides an efficient strategy with a Merge Table T as well, but in a direct calculation the third line of Eq. 8 must be replaced with $\langle (T_{/ij})_r, \mathbf{e} \rangle = \langle T_r, \mathbf{e} \rangle$. However, in order to get the original form of the approximated Spectral Norm Strategy for Binary Merge Tables, Eq. 7 must be applied to the $T_{/ij} T_{/ij}^T$ (symmetric) matrix, which provides an approximation for the square of the spectral norm of $T_{/ij}$. In this case a similar direct calculation is available.

Matrix properties – Merge Paths The author introduced the notion of Merge Paths [42]. Certain graph properties like matrix norms may be evaluated during the selection of two rows for a Merge Operation. Gathering these graph properties into a vector (e.g. eigenvalues), they form the basis of the decision. The changes of the property vector with each successive merge describe a path called the Merge Path. This path is responsible for determining the colouring, and the end of the path defines the quality of the colouring (see Figure 7(a)). Unfortunately, the ideal path (which results in an optimal solution) is of course unknown; the task of colouring is to find this



(a) Spectral norm steepest descent minimisation. The ends of the curves are extended in order to have clearer comparison (horizontal lines).



(b) An example 3D Merge Path of the three largest eigenvalues of a graph during the merges.

Figure 7: Evolution of the eigenvalues along a merge sequence. The graph is a 20−chromatic, equi-partite graph having 200 vertices with a 0.64 edge density from the peak of the phase transition. The spectral norm value of the final Binary Merge Square is $\chi - 1 = 19$ in the optimal case, otherwise bigger.

path. *The author* introduced a general strategy which approximates an optimal Merge Path [42]. The start and the end points of the path are usually known and the curve of the path may be estimated by using preliminary knowledge. In order to build the knowledge base the Merge Path approach can be combined with artificial intelligence methods, such as instance-based learning or clustering, in accordance with the results described in [36].

Enhanced heuristics and meta-heuristics A non-merge based colour strategy can be extended and enhanced by reformulating the strategy in a Merge Model. A Binary Merge Square[8] is the adjacency matrix of a quotient graph. Consequently, if a strategy can operate on the adjacency matrix of the original graph, then the same strategy can cooperate with a merged adjacency matrix (an intermediate Merge Square) as well. It introduces a dynamic reconsideration process where previous decisions of a strategy can be revised after each Merge Operation by exploiting the additional information contained in the intermediate matrices. *The author* in [38] showed the efficiency of such an extension.

The author in [37] applied the structural properties of the Merge Table Models in the meta-heuristics design. *The author* introduced a better granular fitness function than the traditional one for the evolutionary solvers of the colouring problem. This resulted in a smoother landscape of the objective function, which increased the efficiency of the optimisation process. If $\zeta_{M(\pi)}$ denotes the number of non-zeros in a final Merge Table (see Figure 4), then the ζ-fitness function is $f(\pi) = (k_{M(\pi)} - \chi)\zeta_{M(\pi)}$, where $M(\pi)$ is the final Merge Table corresponding to the π permutation and a greedy merge/colouring scheme. This approach follows the entrywise norm optimisation of a Binary Merge Table (see the Dot Product strategy). Moreover, *the author* defined a mutation which forces the difficult vertices by a Merge Table Model (for which the colouring is problematic) in advance in the merge/colour assignment.

[8]Usually the extension can be applied on the other Merge Models as well.

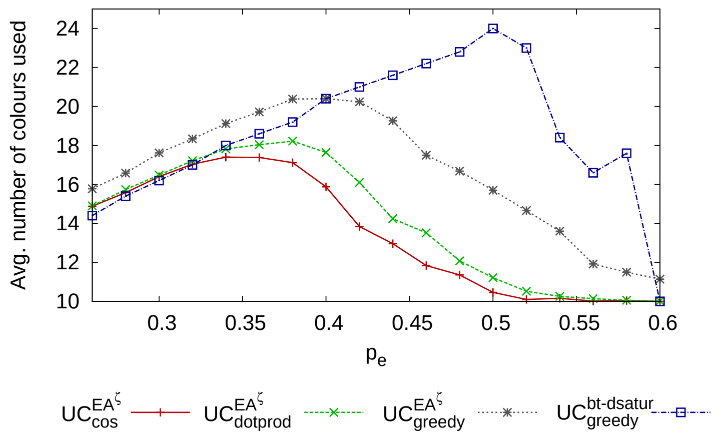
Merge Algorithms

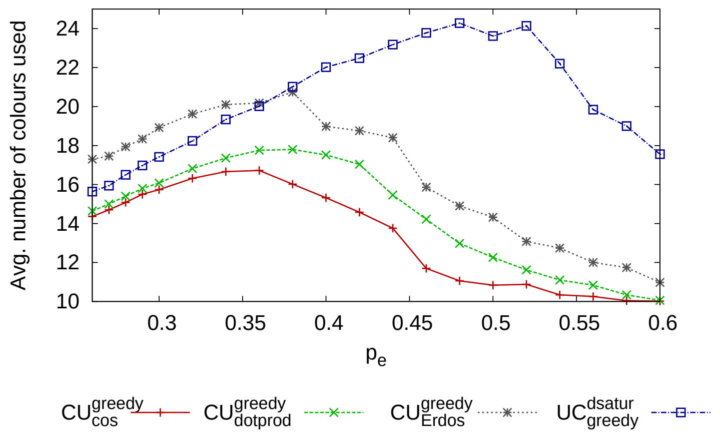
The author in [35; 37–43] combined various novel Merge Strategies with different Merge Frameworks and analysed their performance. The algorithms were compared with standard benchmark algorithms on various benchmark graphs. The experimental analysis showed that the novel Merge Algorithms perform well in the comparison. They generally outperformed the benchmark algorithms especially in the phase transition region where the problems become hard. Some results of an extensive study can be found in Figure 8, where some of the novel Merge Strategies are embedded into different Merge Frameworks and compared with benchmark algorithms and each other. In order to denote an algorithm in the UC Merge Framework we introduced the following notation: $UC_{choose-unc}^{choose-col}$, where the $choose - unc$ denotes the uncoloured row choice strategy, while the $choose - col$ denotes the coloured one. We did likewise with the CU Merge Framework using the $CU_{choose-col}^{choose-unc}$ denotation. In the CC Merge Framework $CC - \textsc{choose}$, the $\textsc{choose}$ denotes the only row-pair choice strategy. The choose functions and the Merge Strategies introduced by *the author* were denoted by: 'dotprod' - Dot Product and 'cos' - Cosine; '$\tilde{\sigma}$' - approximated spectral norm; 'Zykov$_{\bar{\theta}}$' - Zykov-tree and Lovász-theta[9] and 'EA^{ζ}' - evolutionary algorithm with the ζ-fitness , while some of the selected benchmark algorithms are denoted by: 'dsatur' - DSatur heuristic and 'Erdős' - Erdős heuristic. Furthermore the 'greedy' stands for the greedy strategy. In order to get a fair comparison each 'benchmark' algorithm was embedded into a suitable Merge Framework applying a suitable Merge Model.

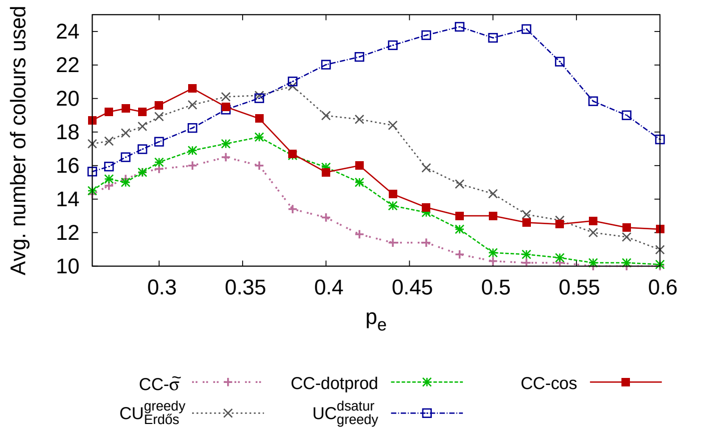
Conclusions

The new colouring approach presented in this thesis demonstrates that graph colouring can be effectively modelled by quotient or power graphs. It provides a potential reduction in computational cost, as well as a uniform and compact way in which algorithms can be defined. Embedding algorithms in the same common framework supports both their structural and performance comparison, as making a comparison is sometimes problematic. The framework itself generalises a formal colouring approach. With this generalisation an algorithm can be naturally extended, which may result in new algorithms. The novel problem description yields novel information that can help us to extract and support a new scheme of the colouring process.

[9] Superscript + will denote multiple edge additions.







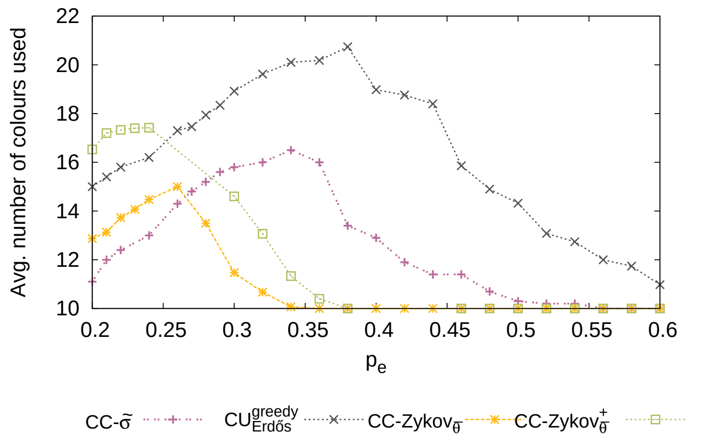


Figure 8: Results of the average number of colours used during the phase transition. The 10-chromatic random equi-partite graph set consists of groups with edge probabilities defined by $0.1 \leq p_e \leq 0.9$. Each group has ten graph instances generated by using the same p_e, but different random seeds $\{1, 2, \ldots, 10\}$ in the generation process.

THESIS 1 *The author*, applying certain graph homomorphisms, defined two general concepts to redefine the graph colouring problem, namely the Quotient and Power methods [37; 40; 41]. He provided a concrete description of the general methods using matrix representations and Merge Operation of the rows or columns. He called these descriptions Merge Models. Based on the Merge Models the original problem undergoes an evolution and produces homomorphic graph images. These models can be a basis of novel and existing algorithms too. Embedding an algorithm into a Merge Model may considerably decrease its computational efforts. Moreover, such an embedding supports the structural analysis of the algorithms in a common way and makes available a natural extension of them, which may result in an increase in their performance. Traditional colouring schemes distinguish between the colours and the vertices of the graph. Merge Models integrate them into one single object. This anticipates a uniform algorithm design, where colour choices do not differ from the vertex choices.

THESIS 2 Based on the Merge Models of the colouring, *the author* unified and generalised the formal sequential colouring model in three different Merge Frameworks [41; 42]. These frameworks provide a uniform and compact description in which algorithms can be defined and analysed in the same systematic way. Furthermore, exploiting the uniform description, he sketched some explanations of how the structure of algorithms can have an influence on the overall performance. Existing sequential colouring algorithms fit into one of the Merge Frameworks, and the frameworks provide novel approaches for algorithm design.

THESIS 3 *The author* provided a way to reduce the computational cost of colouring algorithms after embedding them into a Merge Framework [38; 40]. This improvement was demonstrated and analysed via experiments as well. In the experiments he analysed the phase transitions of different algorithms implemented in different Merge Frameworks. Furthermore, *the author* provided a natural extension of sequential colouring algorithms in the Merge Framework, which results in an increase in their efficiency.

THESIS 4 In each Merge Model the colouring operation is replaced by a Merge Operation. Several Merge Strategies were developed by *the author*. Since the models use matrix representations, he was able to define some of his strategies by applying special matrix row operations as well as matrix norms. The novel strategies of *the author* are listed below:

– Extended Hajnal; Extended Welsh-Powell (∞–norm) [38]

– Spectral norm[42]

– Spectral norm approximations [42]

– Dot product (entrywise norms) [38]

– Cosine [38]

– Zykov-tree and Lovász-theta [35; 43]

These strategies can be combined with different Merge Models and Merge Frameworks to form different algorithms. The performance analysis of these strategies are given. The novel algorithms are compared with several well-known benchmark algorithms. The novel algorithms outperformed the well-known algorithms in a standard benchmark set of graph instances. Moreover, their efficiency revealed in a more difficult-to-solve graph instance set, where the graphs are generated during the phase transition region, where finding a solution becomes really hard. In this case, the comparison is fair; that is, it cannot be manipulated by a good choice of the benchmark instances since the generated instances represent well all instances from difficult-to-solve graph classes.

THESIS 5 *The author* introduced the notion of a *Merge Path* in [42]. A Merge Path arises from the properties of the dynamically changing model during its evolution. Elements of such a path are associated with colouring steps. He was able to describe an abstract graph colouring approach based on Merge Paths, which allows the application of artificial intelligence methods in graph colouring e.g.:

– Using a training set of known graphs, a supervised learning algorithm [36] can learn certain optimal Merge Paths that are associated with optimal colouring steps. Then using the learnt knowledge, colouring steps for an unknown graph instance can be predicted.

– In an unsupervised learning task optimal Merge Paths of known graphs are clustered. Then unknown graphs, which are not involved in the clustering, can be classified in order to predict their properties such as their chromatic number.

THESIS 6 He embedded his colouring strategies into a meta heuristic, an evolutionary algorithm and created the following evolutionary operators for colouring [37–39; 42] :

– A mutation operator by acquiring difficult vertices in a candidate solution and forcing their early colouring

– A fitness function which solves the fitness granularity problem of the colouring

These novel meta heuristic algorithms performed well in an experimental comparison with different benchmark algorithms, on different benchmark graphs and difficult-to-solve generated problem sets as well.

Bibliography

[1] Avanthay C, Hertz A, Zufferey N: **A variable neighborhood search for graph coloring**. *European Journal of Operational Research* 2003, **151**:379–388.

[2] Bäck T: *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York 1996.

[3] Blochliger I, Zufferey N: **A graph coloring heuristic using partial solutions and a reactive tabu scheme.** *Computers & Operations Research* 2008, **35**:960–975.

[4] Brèlaz D: **New methods to color the vertices of a graph**. *Communications of the ACM* 1979, **22**:251–256.

[5] Bui TN, Patel CM: **An ant system algorithm for coloring graphs**. In *Proceedings of the Computational Symposium on Graph Coloring and its Generalizations* 2002:83–91.

[6] Campelo M, Correa R, Frota Y: **Cliques, holes and the vertex coloring polytope**. *Information Processing Letters* 2004, **89**:159–164.

[7] Caramia M, Dell'Olmo P: **Bounding vertex coloring by truncated multistage branch and bound**. *Networks* 2004, **44**:231–242.

[8] Caramia M, Dell'Olmo P: **Coloring graphs by iterated local search traversing feasible and infeasible solutions**. *Discrete Applied Mathematics* 2008, **156**:201–217.

[9] Chams M, Hertz A, de Werra D: **Some experiments with simulated annealing for coloring graphs**. *European Journal of Operational Research* 1987, **32**:260–266.

[10] Chiarandini M, Dumitrescu I, Stützle T: **Very large-scale neighborhood search: Overview and case studies on coloring problems**. In *European Journal of Operational Research* 2008.

[11] Coll P, Marenco J, Méndez Díaz I, P Z: **Facets of the graph coloring polytope**. *Annals of Operations Research* 2002, **116**:79–90.

[12] Costa D, Hertz A: **Ants Can Colour Graphs**. *Journal of the Operations Research Society* 1997, **48**:295–305.

[13] Costa D, Hertz A, Dubuis C: **Embedding a sequential procedure within an evolutionary algorithm for coloring problems in graphs.** *Journal of Heuristics* 1995, **1**:105–128.

[14] Craenen B, Eiben A, van Hemert J: **Comparing Evolutionary Algorithms on Binary Constraint Satisfaction Problems**. *IEEE Transactions on Evolutionary Computation* 2003, **7**(5):424–444.

[15] Culberson J, Gent I: **Frozen development in graph coloring**. *Theor. Comput. Sci.* 2001, **265**(1–2):227–264.

[16] Culberson J: **Iterated Greedy Graph Coloring and the Difficulty Landscape**. Tech. Rep. TR 92-07 1992.

[17] de Werra D: **Heuristics for Graph Coloring**. *Computational Graph Theory* 1990, **Comput. Suppl. 7**:191–208.

[18] Dukanovic I, Rendl F: **Semidefinite programming relaxations for graph coloring and maximal clique problems**. *Mathematical Programming, Serie B* 2007, **109**:345–365.

[19] Eiben AE, van Der Hauw JK, van Hemert JI: **Graph Coloring with Adaptive Evolutionary Algorithms**. *Journal of Heuristics* 1998, **4**:25–46.

[20] Falkenauer E: *Genetic Algorithms and Grouping Problems*. John Wiley 1998.

[21] Figueiredo R, Barbosa V, Maculan N, de Souza C: **New $0-1$ integer formulations of the graph coloring problem**. In *Proceedings of XI CLAIO* 2002.

[22] Galinier P, Hao JK: **Hybrid evolutionary algorithms for graph coloring**. *Journal of Combinatorial Optimization* 1999, **3**:379–397.

[23] Galinier P, Hertz A: **A survey of local search methods for graph coloring**. *Computers & Operations Research* 2006, **33**:2547–2562.

[24] Gintaras P: **On the Graph Coloring Polytope**. *Information technology and control* 2008, **37**:7–11.

[25] Glass CA, Prügel-Bennett A: **Genetic algorithms for graph colouring: Exploration of Galinier and Hao's algorithm.** *Journal of Combinatorial Optimization* 2003, **7**:229–236.

[26] Graham R, Grötschel M, Lovász L (Eds): *Handbook of Combinatorics.* North-Holland 2005.

[27] Hajnal P: **Eigenvalues of graphs: Graph colouring and the Perron-Frobenius eigenvector**. *Péter Hajnal's Combinatorics seminars. University of Szeged.* 2004.

[28] Halldórsson MM: **A Still Better Performance Guarantee for Approximate Graph Coloring**. *Inf. Process. Lett.* 1993, **45**:19–23.

[29] Hamiez JP, Hao JK: **Scatter search for graph coloring**. In *Artificial Evolution*, *Volume 2310* 2001:168–179.

[30] Harmanani H, Abas H: **A method for the minimum coloring problem using genetic algorithms**. In *MS'06: Proceedings of the 17th IASTED international conference on Modelling and simulation*, Anaheim, CA, USA: ACTA Press 2006:487–492.

[31] Hertz A, de Werra D: **Using tabu search techniques for graph coloring**. *Computing* 1987, **39**:345–351.

[32] Johnson DS, Mehrotra A, Trick MA: **Special issue on computational methods for graph coloring and its generalizations**. *Discrete Applied Mathematics* 2008, **156**:145–146.

[33] Johnson D, Aragon C, McGeoch L, Schevon C: **Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning**. *Operations Research* 1991, **39**:378–406.

[34] Johnson D, Trick M: *Cliques, Coloring, and Satisfiability*. American Mathematical Society, DIMACS 1996.

[35] Juhos I: **Graph Colouring through Clustering**. *Seminar. University of Edinburgh.* 2009.

[36] Juhos I, Szarvas G: **Intelligent Forecast with Dimension Reduction**. In *Applied Soft Computing Technologies: The Challenge of Complexity*, *Volume 34 of* Advances in Soft Computing. Edited by Abraham A, de Baets B, Köppen M, Nickolay B, Springer 2006:279–292.

[37] Juhos I, Tóth A, van Hemert J: **Binary Merge Model Representation of the Graph Colouring Problem**. In *Evolutionary Computation in Combinatorial Optimization*, *Volume 3004 of* Lecture Notes in Computer Science. Edited by Gottlieb J, Raidl, Raidl GR, Springer 2004:124–134.

[38] Juhos I, Tóth A, van Hemert J: **Heuristic Colour Assignment Strategies for Merge Models in Graph Colouring**. In *Evolutionary Computation in Combinatorial Optimization*, *Volume 3448 of* Lecture Notes in Computer Science. Edited by Gottlieb J, Raidl, Raidl GR, Springer 2005:132–143.

[39] Juhos I, van Hemert J: **Improving graph colouring algorithms and heuristics using a novel representation**. In *Evolutionary Computation in Combinatorial Optimization*, *Volume 3906 of* Lecture Notes in Computer Science. Edited by Gottlieb J, Raidl, Raidl GR, Springer 2006:123–134.

[40] Juhos I, van Hemert J: **Increasing the efficiency of graph colouring algorithms with a representation based on vector operations.** *Journal of Software* 2006, **1**(2):24–33.

[41] Juhos I, van Hemert J: *Contraction-based heuristics to improve the efficiency of algorithms solving the graph colouring problem*, Springer, *Volume 153 of* Studies in Computational Intelligence 2008 chap. III, :175–192.

[42] Juhos I, van Hemert J: **Graph Colouring Heuristics Guided by Higher Order Graph Properties**. In *Evolutionary Computation in Combinatorial Optimization*, *Volume 4972 of* Lecture Notes in Computer Science. Edited by van Hemert J, Cotta C, Springer 2008:97–109.

[43] Juhos I, van Hemert JI: **Graph colouring through clustering by Zykov-tree and Lovász-theta**. *Submitted* 2009.

[44] Karger D, Motwani R, Sudan M: **Approximate graph coloring by semidefinite programming**. *Journal of the ACM* 1998, **45**(2):246–265.

[45] Karp RM: *Complexity of Computer Computations: Reducibility Among Combinatorial Problems*, New York: Plenum 1972 :85–103.

[46] Leighton F: **A graph coloring algorithm for large scheduling problems**. *Journal of Reasearch of the National Bureau of Standards* 1979, **84**:489–506.

[47] Malaguti E, Toth P: **A survey on vertex coloring problems**. *International Transactions in Operational Research* 2009, :1–34.

[48] Marino A, Damper R: **Breaking the Symmetry of the Graph Colouring Problem with Genetic Algorithms**. In *Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference*. Edited by Whitley D 2000:240–245.

[49] Matula D, Marble G, J I: *Graph coloring algorithms*, Academic Press 1972 :109–122.

[50] Mehrotra A, Trick MA: **A column generation approach for graph coloring**. INFORMS *Journal on Computing* 1996, **8**:344–354.

[51] Méndez-Díaz I, Zabala P: **A polyhedral approach for graph coloring**. *Electronic Notes in Discrete Mathematics* 2001, **7**:178–181.

[52] Méndez-Díaz IP Zabala: **A cutting plane algorithm for graph coloring**. *Discrete Applied Mathematics* 2008, **156**:159–179.

[53] Méndez-Díaz I, Zabala P: **A branch-and-cut algorithm for graph coloring**. *Discrete Appl. Math.* 2006, **154**(5):826–847.

[54] Merikoski JK, Kumar R: **Lower Bounds for the Spectral Norm**. *Journal of Inequalities in Pure and Applied Mathematics* 2005, **6**(3).

[55] Palubeckis G: **On the recursive largest first algorithm for graph colouring**. *Int. J. Comput. Math.* 2008, **85**(2):191–200.

[56] Porumbel D, Hao JK, Kuntz P: **Diversity control and multi-parent recombination for evolutionary graph coloring algorithms**. In *LNCS 5482*, Springer 2009.

[57] Schindl D: **Graph coloring and linear programming**. In *LNCS 5482*, Presentation at First Joint Operations Research Days, Ecole Polytechnique Fédérale de Lausanne (EPFL), Available on line 2003.

[58] Sewell E: **An improved algorithm for exact graph coloring**. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 2006, **26**:359–373.

[59] van Hemert J: **Application of Evolutionary Computation to Constraint Satisfaction and Data Mining**. *PhD thesis*, Leiden University 2002.

[60] Welsh DJA, Powell MB: **An upper bound for the chromatic number of a graph and its application to timetabling problems**. *The Computer Journal* 1967, **10**:85–86.

[61] Wigderson A: **Improving the performance for approximate graph coloring**. *Journal of the ACM* 1983, **30**:729–735.

[62] Wilf HS: **The eigenvalues of a graph and its chromatic number**. *Journal of London Math. Soc.* 1967, :330–332.

[63] Woeginger G: **Exact Algorithms for NP-Hard Problems: A Survey**. In *Journal of London Math. Soc.* 2003:185–207.

[64] Zykov AA: **On some properties of linear complexes (in Russian)**. *Math. Sbornik* 1949, **24**:163–188.

[65] Zykov AA: **On some properties of linear complexes**. *American Mathematical Society Translations* 1952, **79**:81.