IIMB-WP No. 665/2022

WORKING PAPER NO: 665

A Constructive Matheuristic Approach For The Vertex Colouring Problem

Reshma Chirayil Chandrasekharan
Assistant Professor of Decision Sciences
Indian Institute of Management Bangalore
Bannerghatta Road, Bangalore – 5600 76

KU Leuven Department of Computer Science, CODes,
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
reshma.chirayil1@iimb.ac.in

Tony Wauters
KU Leuven Department of Computer Science, CODes,
Gebroeders De Smetstraat 1, 9000 Gent, Belgium
tony.wauters@kuleuven.be

Year of Publication – May 2022

Abstract The vertex colouring problem is one of the most widely studied and popular problems in graph theory. In light of the recent interest in hybrid methods involving mathematical programming, this paper presents an attempt to design a matheuristic approach for the problem. A decomposition-based approach is introduced which utilizes an integer programming formulation to solve subproblems to optimality. A detailed study of two different decomposition strategies, vertex-based and colour-based, is discussed. The impact of algorithm design parameters on the decompositions used and their influence on final solution quality is explored. In addition to integer programming, a constraint programming is also employed to solve the subproblems.

Keywords Vertex colouring · matheuristic · decomposition

Acknowledgements This research received funding from the Flemish Government under the *"Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen"* programme. Editorial consultation provided by Luke Conolly (KU Leuven).

1 Introduction

The vertex colouring problem (VCP) seeks to assign colours to vertices of a graph such that no two adjacent vertices are assigned the same colour. Initially studied as a problem on planar graphs, the problem has been generalized over general graphs and represents a large share of the graph theory literature given its widespread applications. Problems which can be modeled as the assignment of conflicting elements of a set to distinct subsets such as scheduling (Leighton, 1979), timetabling (Babaei, Karimpour, and Hadidi, 2015), frequency assignment (Aardal, Van Hoesel, Koster, Mannino, and Sassano, 2007) and register allocation (Chow and Hennessy, 1990), are some of the major areas where there exist practical applications of the VCP.

The VCP is an example of a problem which is easy to define, yet difficult to solve. Determining the smallest number of colours required to colour a graph is an NP-hard problem (Garey and Johnson, 1979). This inherent difficulty of the problem means that only certain kinds of graph are capable of being solved by the best mathematical models formulated for the VCP (Méndez-Díaz and Zabala (2006), Méndez-Díaz and Zabala (2008), Malaguti, Monaci, and Toth (2011)). These exact methods cease to work for random graphs of more than 100 vertices. However, most of the real world applications require colouring graphs on thousands of vertices, thereby motivating the need for efficient heuristic strategies. Decades of research have contributed multiple models and performance guarantees for the VCP. Despite these achievements, the problem continues to fascinate researchers in this area due to its theoretical complexity and the constantly growing number of practical applications that demand colouring larger graphs.

Aside from the exact techniques available, greedy constructive heuristics, local search heuristics and metaheuristics constitute much of the literature in vertex colouring. A bibliography is available at `https://imada.sdu.dk/~marco/gcp/`. Malaguti and Toth (2010) provide a useful survey on the various exact and heuristic algorithms developed for the VCP. Most of the high performing algorithms for the VCP such as Malaguti, Monaci, and Toth (2008), Funabiki and Higashino (2000), Galinier and Hao (1999) are metaheuristic in nature. Most of them operate by searching for a colouring utilizing a given number of colours. These algorithms are interesting for the multiple search strategies that are used in combination to design an efficient algorithm. This suggest that hybrid methods are efficient in colouring some of the very large random graphs. However, it must be noted that metaheuristic algorithms require much longer runtimes compared to local search and simple greedy constructive heuristics, but generates solutions of better quality. Faster algorithms are still relevant as vertex colouring appear as a subroutine in various practical applications which need to be executed within a short runtime. As implied, the trade off between algorithm runtime and solution quality often decides the choice of vertex colouring algorithm to be used for a particular problem.

The major contribution of this paper is that it presents one of the first attempts to design and test a matheuristic approach for the VCP. Matheuristics are methods which hybridize mathematical programming and heuristics. The recent success of matheuristic strategies in scheduling applications which are, at their most fundamental level, graph colouring problems has motivated this study. By combining mathematical programming and heuristics, the paper aims at designing a vertex colouring heuristic that can generate high quality solutions within a reasonable runtime. The heuristic part tested in the present work extends the idea of general greedy

constructive heuristics available for vertex colouring such as the smallest last colouring, recursive largest first heuristic and DSATUR (Matula, Marble, and Isaacson (1972), Matula et al. (1972), Leighton (1979)). It is well known that greedy colouring heuristics are highly sensitive towards the initial order in which vertices are coloured or the order in which colours are utilized. Therefore, extensive experimentation has been conducted to study the influence of these properties on the proposed matheuristic for the VCP.

The outline of the paper is as follows. Section 2 briefly introduces the problem and the terminology. The matheuristic strategy proposed for the VCP is introduced in Section 3, while the related experiments are summarized in Section 4. Section **??** presents some coefficient matrix illustrations. Section 5 then ends this paper by concluding and offering future research possibilities.

2 The vertex colouring problem

Let $G = (V, E)$ denote a graph on a finite vertex set V and edge set E, whose cardinalities are denoted by n and m respectively. In this paper, E is assumed to be the collection of unordered pairs $E = \{\{v, v'\} | v, v' \in V, v \neq v'\}$, thereby limiting the problem to finite simple graphs (no loops or multiple edges). Any two vertices v and v' said to be *adjacent* if there exists an edge $\{v, v'\} \in E$. A k-colouring of G is the assignment of k colours to elements of V such that no two adjacent vertices share the same colour. The smallest k for which a k-colouring exists for G is defined to be the *chromatic number* of G, denoted by χ_G.

A set $V' \subseteq V$ is a *clique* of G if all vertices in V' are adjacent to each other. Clearly, any two vertices belonging to a clique cannot be assigned the same colour. A *maximal clique* of G is a clique that is not a proper subset of any other clique of G. Thus, the size of a maximal clique is a valid lower bound for the chromatic number of G. A set $V' \subseteq V$ is said to be an *independent set* if no vertices in V' are adjacent. It is easy to see that the set of vertices that are assigned the same colour form an independent set of G. Independent sets corresponding to each colour are referred to as *colour classes*. Given a full or partial colouring of G, the saturation degree of a vertex is defined as the number of colours to which it is adjacent.

3 Constructive Matheuristics

The present work applies a decomposition-based approach which utilizes powerful exact techniques to solve subproblems to optimality and thus can be called a matheuristic (Maniezzo, Stützle, and Voß, 2010). More specifically, this approach adapts the constructive matheuristic (CMH) strategy introduced by Smet, Wauters, Mihaylov, and Berghe (2014) and extends it. The approach draws motivation from the method-based heuristics discussed in **?**. Given a well defined problem P and an exact method that can solve the problem if the problem size were small, the present approach designs a heuristic directly based on the capability of the exact method available and the complexity of the problem. Similar to local search heuristics, a finite sequence of easier problems $(P_1, P_2, ..., P_N)$ are solved to generate a corresponding sequence of solutions $(s_0, s_1, ..., s_N)$ and a feasible solution s for P is computed as a function of these solutions. Depending on the algorithm design, problems P_i in

$\mathbb{P} = (P_1, P_2, ..., P_N)$ can be defined as a function of the decision space X of the problem P or its subspaces or both. P_i may also be defined as a function of other problems in $\mathbb{P}$.

Given a large scale problem P, the proposed CMH approach utilizes a decomposition strategy to arrive at a sequence of simpler problems $\mathbb{P}$, called *subproblems* or *blocks* of the CMH. While the overall CMH approach is general in nature, the present approach is designed to tackle very large mathematical programming formulation models, especially integer programming models. Given the integer programming formulation of a large problem P, denoted by $\mathcal{F}$, the problem P_i solved in block i, is arrived at by decomposing its variables. In each block, an integer programming problem that involves the variables associated to the block is solved. The precise problem that is solved in each block may be defined based on the decomposition strategy utilized, the nature of the variables in the block and the characteristics of the original problem at hand. Being a constructive method, the CMH also requires an order for solving blocks to operate.

Without loss of generality, let us assume that the blocks are solved in the order: $1, 2, ..., N$. In each block i, the CMH algorithm solves problem P_i and fixes the assignments of its variables. However, for the CMH to be able to produce feasible solutions for the original problem P, problem P_{i+1} in block i is solved such that the assignments do not violate constraints in the original problem P involving variables of which assignments are fixed in blocks $1, 2, ..., i$. In other words, in each block $i+1$, CMH generates a modified problem P'_{i+1} from problem P_{i+1} by adding additional constraints such that the assignments fixed in block $i+1$ do not conflict the assignments made in blocks $1, 2, ..., i$. Thus, a new sequence subproblems $\mathbb{P}' = (P'_1 = P_1, P'_2, ..., P'_N)$ and a corresponding sequence of solutions $s' = (s_1, s'_2, ..., s'_N)$ can be obtained such that the solution of blocks are not conflicting. Proceeding in this manner, CMH constructively fixes assignments of all the variables in such a way that none of the constraints in P are violated.

The major challenge while designing the blocks is to ensure that the assignment fixings made in steps $1, ..., i$ do not result in block P_{i+1} being infeasible. Therefore, the decomposition strategy and the order of solving blocks are some of the key factors that affect the efficiency of the CMH approach. In order to navigate the CMH, the following CMH design parameters introduced in Chandrasekharan, Toffolo, and Wauters (2019) are utilized.

1. Block size (η): This parameter defines the size of subproblems/blocks and often significantly influences algorithmic runtime.
2. Overlap (θ): This parameter allows blocks to share some variables instead of being completely disjoint. θ denotes the extent of overlap between consecutive blocks.

A CMH configuration is therefore represented by the tuple (η, θ). Figure 1 illustrates the overall CMH strategy utilized in this paper and the design parameters. Depending on the solutions of previously solved subproblems, additional constraints may need to be added to the blocks to ensure feasibility of all subproblems. The objective functions utilized in the blocks may also be modified in order to navigate the CMH to produce feasible solutions for the full problem P and to improve CMH solution quality. Given a block i, its definition and the precise optimization problem it solves is realized by means of its *block objective function* $Z_i(\eta, \theta)$.

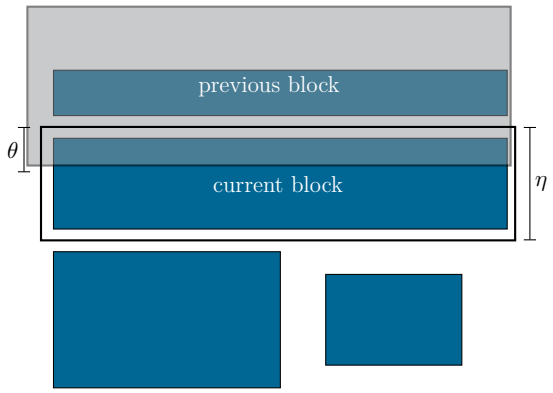


Fig. 1: An overview of the general CMH strategy. Blue rectangles represent subproblems (blocks) in the CMH strategy and the solid window represents current block. The gray rectangle represents the block previously solved of which assignments are already fixed.

The present CMH approach for the VCP utilizes the simple assignment-based IP formulation (VCP-ASS) of the VCP. Let H denote the set of colours. The algorithm starts with a total of n colours, $|H| = n$. Variables x_{ih} decide whether colour h is assigned to vertex i while variables y_h decide whether colour $h \in H$ is utilized or not.

$$x_{ih} = \begin{cases} 1 & \text{if vertex } i \in V \text{ is assigned to colour } h \in H \\ 0 & \text{otherwise} \end{cases} \tag{1a}$$

$$y_h = \begin{cases} 1 & \text{if colour } h \in H \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{1b}$$

The model can then be formulated as follows:

$$\text{minimize:} \quad \sum_{h=1}^{n} y_h \tag{1c}$$

$$\text{subject to:} \quad \sum_{h=1}^{n} x_{ih} = 1 \quad \forall i \in V \tag{1d}$$

$$x_{ih} + x_{jh} \leq y_h \quad \forall (i,j) \in E, h = 1, \dots, n \tag{1e}$$

$$x_{ih} \leq y_h \quad \forall i \in V, h = 1, \dots, n \tag{1f}$$

$$y_{h+1} \leq y_h \quad \forall i = 0, \dots, n-1 \tag{1g}$$

$$x_{ih} \in \{0,1\} \quad \forall i \in V, h = 1, \dots, n \tag{1h}$$

$$y_h \in \{0,1\} \quad \forall h = 1, \dots, n \tag{1i}$$

Constraints 1d ensure that vertices are assigned exactly one colour, while Constraints 1e prevent adjacent vertices from being assigned the same colour. Constraints 1f ensure that the value of x_{ih} is bounded even when the graph has isolated nodes. Constraints 1g are introduced to break the symmetry between used and un-used colours.

Starting with the VCP-ASS formulation, one could arrive at numerous CMH approaches for the VCP based on the decomposition strategy employed. The following subsections introduce two different CMH approaches for the VCP, the colour-based CMH and the vertex-based CMH. The colour-based CMH aims at colouring the graph with a colour or group colours in each iteration and stops when the graph is fully coloured. In contrast, the vertex-based CMH aims at colouring parts of the graph until the graph is fully coloured. The approaches are inspired from common decomposition strategies applied in real-world problems which are vertex colouring problems in its fundamental nature. For example, in an employee-task scheduling problem, employees correspond to colours while tasks correspond to vertices. In such a problem, a constructive approach that proceeds by making optimal assignments for an employee or group of employees at a time corresponds to a colour-based CMH approach. Whereas, a constructive approach that assigns a task or group of tasks optimally to employees in its iterations corresponds to a vertex-based CMH. Similar comparisons can be made across many such problems. An important class of problems to consider are time-based scheduling problems. In such problems, a common decomposition approach employed is to split the time-horizon of the problem such that the subproblems contain variables concerning one time or a period of time. When viewed as a vertex colouring problem, note here that the decomposition is happening in the vertices of the graph and therefore corresponds to a vertex-based CMH approach. In this paper, we aim to explore how both the approaches compare and thereby make general conclusions on which strategy is more suitable for designing CMH algorithms for vertex colouring and vertex colouring-like problems. This study is inspired by a similar comparison of employee-based and time-based decompositions employed in a CMH approach developed for a task scheduling problem (Chandrasekharan, Smet, and Wauters, 2020).

3.1 Colour-based CMH

The colour-based CMH (CBC) employs a decomposition on the y_h variables of the VCP-ASS formulation to arrive at its blocks. Associated with each block is a set of variables of type y_h. In other words, a partition $b_1, b_2, ..., b_N$, $\sqcup_{i=1}^{N} b_i = H$ is utilized to define the blocks of CBC. Each P_i solves for the maximal subgraph that can be coloured by variables of type y_h in the block. The following is the MIP formulation defining block P_k:

$$x_{ih} = \begin{cases} 1 & \text{if vertex } i \in V \text{ is assigned to colour } h \in b_k \\ 0 & \text{otherwise} \end{cases} \tag{2a}$$

$$y_h = \begin{cases} 1 & \text{if colour } h \in b_k \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{2b}$$

The model can then be formulated as:

$$\text{maximize:} \quad \sum_{h \in b_k} \sum_{i \in V} x_{ih} \tag{2c}$$

$$\text{subject to:} \quad \sum_{h \in b_k} x_{ih} \leq 1 \quad \forall i \in V \tag{2d}$$

$$x_{ih} + x_{jh} \leq y_h \quad \forall (i,j) \in E, h \in b_k \tag{2e}$$

$$x_{ih} \leq y_h \quad \forall i \in V, h \in b_k \tag{2f}$$

$$y_{h+1} \leq y_h \quad \forall h \in \{b_{k_1}, \ldots, b_{k_{|b_k|}}\} \tag{2g}$$

$$x_{ih} \in \{0,1\} \quad \forall i \in V, h \in b_k \tag{2h}$$

$$y_h \in \{0,1\} \quad \forall h \in b_k \tag{2i}$$

After solving the first block, assignment of variables $y_h, h \in b_1$ and the assignments of variables $x_{ih}, h \in b_1$ such that $x_{ih} = 1$ are fixed. Now, the problem to be solved in the next block P_2 is arrived at by adding constraints of type 1e to ensure that the new assignments do not conflict the variables fixed in the previous block. Similarly, problems P_i's to be solved in each block are generated. The final block is re-optimized with the original objective of minimizing the number of colours. The CMH stops when all vertices are coloured.

3.2 Vertex-based CMH

The vertex-based CMH (VBC) defines blocks by way of groups of vertices. Let $b_1, b_2, ..., b_N$, $\sqcup_{i=1}^{N} b_i = V$ be a partition of the vertices of the graph $G = (E, V)$. Based on this decomposition, block b_k colours the subgraph induced by vertices in b_k to optimality. Let b_k denote the current block and E_{b_k} denote the edges of the induced subgraph of vertices in blocks $b_1, \ldots, b_k$. The MIP formulation P_k solved in block b_k can now be formulated as follows.

$$x_{ih} = \begin{cases} 1 & \text{if vertex } i \in b_k \text{ is assigned to colour } h \in H \\ 0 & \text{otherwise} \end{cases} \tag{3a}$$

$$y_h = \begin{cases} 1 & \text{if colour } h \in H \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{3b}$$

$$\text{minimize:} \quad \sum_{h=1}^{n} y_h \tag{3c}$$

$$\text{subject to:} \quad \sum_{h=1}^{n} x_{ih} = 1 \quad \forall i \in b_k \tag{3d}$$

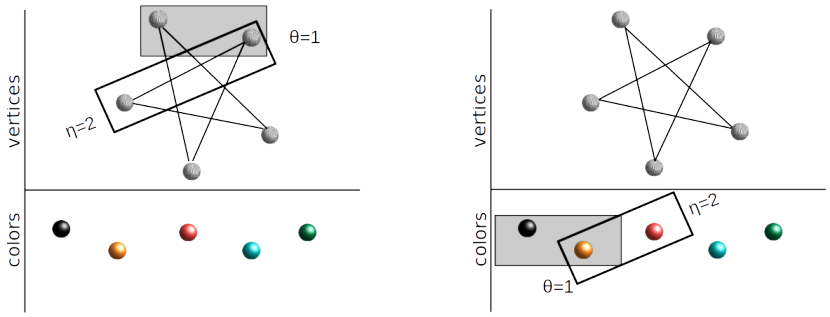
$$x_{ih} + x_{jh} \leq y_h \quad \forall (i,j) \in E_{b_k}, h = 1, \ldots, n \tag{3e}$$

$$x_{ih} \leq y_h \quad \forall i \in b_k, h = 1, \ldots, n \tag{3f}$$

$$y_{h+1} \leq y_h \quad \forall h = 1, \ldots, n \tag{3g}$$

$$x_{ih} \in \{0,1\} \quad \forall i \in b_k, h = 1, \ldots, n \tag{3h}$$

$$y_h \in \{0,1\} \quad \forall h = 1, \ldots, n \tag{3i}$$



(a) Vertex-based CMH (b) Colour-based CMH

Fig. 2: The CBC and VBC strategies. Solid rectangles represent blocks and gray windows correspond to blocks previously solved.

Once block i is solved, the values assumed by variables of type x_{ih} and non-zero assignments of variables of type y_h in the block are fixed. The subproblem P_{i+1} is defined by adding Constraints of type 1e to ensure that the solution do not conflict the assignments fixed in previous blocks. Figure 2 illustrates the CBC and the VBC strategies.

A closer look at the VBC approach reveals that the VBC relaxes one or more constraints of the VCP-ASS formulation for the VCP in order to arrive at subproblems. Each subproblem P_i is arrived at by relaxing Constraints of type 1e and 1d which involves variables which are not from the block i. Hence, problems in $\mathbb{P}$ are relaxations of problem P and, as a result, are guaranteed not to be any harder than P. If $\mathbb{S} = \{S_1, S_2, ..., S_N\}$ denotes the set of feasible solutions of subproblems in $\mathbb{P}$, it is clear that the set of feasible solutions of P, denoted by S, is contained in all $S_i \in \mathbb{S}$ (see Figure 3). Much of the research presented in this paper concerns arriving at a suitable decomposition strategy and in navigating the CMH such that the solution s constructed for P is as close to the global optimum as possible.

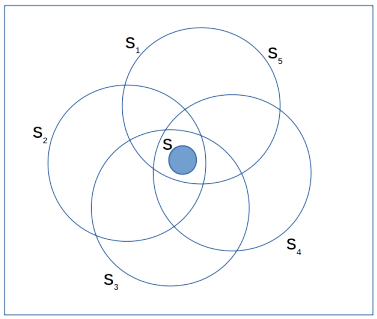


Fig. 3: Solution space of subproblems of the VBC

Given a subproblem P_i a new subproblem P'_i is defined such that the solution of P'_i does not conflict the assignments fixed after solving $P'_1, ..., P'_{i-1}$. If

$\mathbb{S}' = \{S'_1, S'_2, ..., S'_N\}$ denote sets of feasible solutions of subproblems in $\mathbb{P}'$, it is clear that $S'_i \subseteq S_i \cap S_{i-1} \cap S_{i-2} \cap \cdots \cap S_1$ as shown in Fig 4. Note here that if P'_N is feasible, the solution of the last subproblem s'_N must be a feasible solution for the problem P.

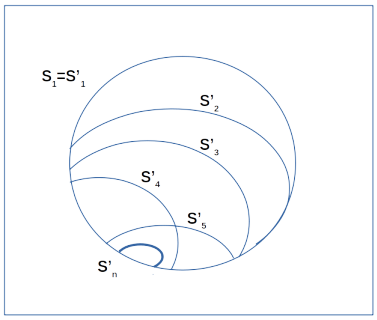


Fig. 4: Solution space of subproblems of the VBC

The proposed general CMH strategy falls in the category of successive augmentation techniques discussed in Johnson, Aragon, McGeoch, and Schevon (1991). Such techniques begin with a feasible partial colouring of the graph and then progressively extend it, examples of which include greedy colouring heuristics such as DSATUR (Brélaz, 1979) and recursive largest first or RLF (Leighton, 1979). DSATUR colours vertices one by one whereas RLF iteratively construct colour classes - groups of vertices that can be coloured by the same colour. The general CMH strategy employed in this paper can be interpreted as a generalization of these classical colouring heuristics. Rather than colouring single vertices or isolating a single colour class, vertex-based CMH optimally colours subgraphs whereas the colour-based CMH isolates a block of colour classes by solving for it mathematically. It is worth noting that in case of the colour-based CMH, subproblems are attempting to generate maximal independent sets mathematically and therefore might require very long computation times for certain graph classes, as opposed to RLF where it is done heuristically.

Johnson et al. (1991) and Matula et al. (1972) studied the influence of the order in which vertices are coloured on final solution quality. Some heuristics utilize a fixed static ordering of vertices, whereas some change this ordering dynamically as the algorithm proceeds. Among static colourings, it is proven that the smallest last (SL) ordering yields the best performance when implemented in a greedy colouring heuristic that colours one vertex at a time. The random vertex-based CMH (RVC) is produced by adapting a random order for colouring vertices in the vertex-based CMH. On adapting the SL ordering, the SL-based CMH (SLC) is produced. In addition, another approach called the DSATUR-SL based CMH (DSC) is designed to utilize a dynamic ordering as in the DSATUR heuristic. In this CMH approach, the first block is composed of η vertices from the SL ordering. Once this block is solved, elements of the next block are selected such that it is composed of uncoloured vertices with the first η largest saturation degrees in the partially coloured graph. Ties are broken utilizing the SL order.

Table 1: Performance comparison of baseline algorithms VCP-ASS, CBC and RVC

	VCP-ASS	RVC(1,0,0)	CBC(1,0,0)
Number of feasible solutions	74	104	116
Number of best known solutions	48	35	45
Average calculation time(s)	2718.36	917.08	569.17
%Gap - average	3100.82	1103.59	521.17

4 Computational Study

Experiments are conducted on four threads of an Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz computer running Ubuntu 16.04.2 LTS. The CMH algorithm was coded in Java and used Gurobi 8.1 to solve blocks. The DIMACS10 vertex-colouring benchmark instances were utilized in the computational study and are available at http://www.cc.gatech.edu/dimacs10/. The benchmark time limit is considered to be 3600s.

From Table 1 it is clear that VCP-ASS, being the IP formulation, is capable of solving only 48 instances of the 131 benchmark instances. When the number of vertices is above 500, the method fails or exhibits poor performance, something which, again, justifies the need for powerful heuristics. The performance of CBC and RVC for $\eta = 1$ are also summarized for comparison purposes. Since RVC utilizes a random order of vertices, the results are averaged over 10 runs. Here, one colour or one vertex is considered per block. If the runtime exceeds the benchmark time limit, the program outputs the number of vertices (n) as the result. %Gap is calculated with respect to the best known solution available in the literature for a particular instance. In order to develop an efficient CMH for the VCP, the impact of CMH design parameters η and θ on all its variants have been tested. Table 2 presents a list of different algorithms tested in this study.

Table 2: The different algorithms tested for the VCP.

Algorithm	Description
VCP-ASS	IP formulation of the VCP (See Equations 1)
CBC	Color-based CMH; blocks are subsets of colours
VBC	Vertex-based CMH; blocks are subsets of vertices
RVC	A VBC that employs a random ordering for vertices
SLC	A VBC that employs the SL-based ordering for vertices
DSC	A VBC that employs the DSATUR-based dynamic ordering for vertices
SLC'	An SLC that utilizes lower bounds from a greedy colouring heuristic
DSC'	A DSC that utilizes lower bounds from a greedy colouring heuristic
SLC'-400	SLC' when implemented with a runtime limit of 400s
DSC'-400	DSC' when implemented with a runtime limit of 400s
SLC'-CP	SLC' with its blocks being solved using a CP formulation
DSC'-CP	DSC' with its blocks being solved using a CP formulation
SLC'-CP-400	SLC'-CP when implemented with a runtime limit of 400s
DSC'-CP-400	DSC'-CP when implemented with a runtime limit of 400s

4.1 Color-based CMH

Figures 5 and 6 illustrate the structure of the coefficient matrix for the VCP-ASS formulation for VCP benchmark instances *3-Insertions_3.col* and *Queen5_5.col* respectively. The X-axis represents variables sorted in the order $x_{1,1}, x_{2,1}, ..., x_{1,2}, x_{2,2}, ..., x_{1,n}, x_{2,n}, ..., x_{n,n}, y_1, y_2, ..., y_n$. Y-axis represents constraints ordered such that any constraint that contains a variable that appears in position i in the order appears before any constraint that contains a variable at position $j, j > i$. Constraints 1g (symmetry breaking) and Constraints 1h and 1i (variable bounding) are omitted in this illustration. In other words, the variables and constraints are sorted according to colour. The graph plots non-zero coeffecients of the formulation. The horizontal band at the top of the figures corresponds to Constraints 2d, which contains variables involving multiple colours. The figure reveals a block-diagonal structure.

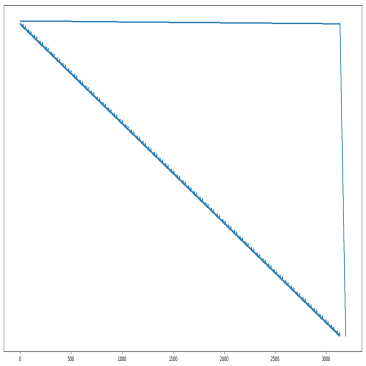


Fig. 5: The coefficient matrix of the VCP-ASS formulation for instance *3-Insertions_3.col* when rows and columns are sorted as in a CBC.

The colour-based CMH (CBC) for the VCP is an attempt to take advantage of this block diagonal structure when designing a heuristic decomposition strategy. From Table 1 it is evident that for $\eta = 1$, CBC has a better performance compared to that of the RVC. In general, larger block size is expected to result in higher solution quality, but also lead to longer runtimes. However, decreasing the block size too much may lead to too many sub problems and longer overall set up times. See Table 3 for a summary of the results obtained by CBC. From the performance of CBC, it is clear that larger block sizes lead to longer runtimes but this does not correspond to improved solution quality trends. This can be attributed to the fact that the algorithm might terminate due to the runtime limit being exceeded for larger block sizes, which contributes highly to the average %gap. This becomes more evident with the %gap calculated exclusively for the feasible solutions.

For CBC, the block objective function tries to solve for the largest subgraph that can be coloured by the elements of the block. The underlying problem therefore seeks

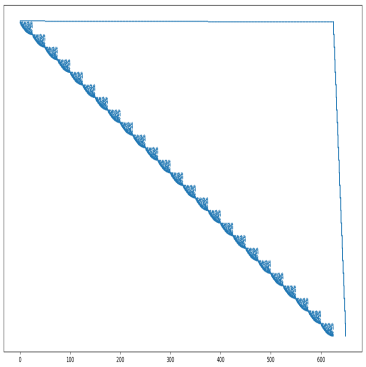


Fig. 6: The coefficient matrix of the VCP-ASS formulation for instance *Queen5_5.col* when rows and columns are sorted as in a CBC

to isolate independent sets in the graph, which can be an NP hard problem. Thus, the precise optimization problem solved in subproblems of a colour-based decomposition are not necessarily any easier than solving the original problem. Since increasing the block size beyond $\eta = 8$ may lead to very long algorithm runtimes, increasing block size further will probably not improve CMH performance. This motivates testing the impact of the overlap design parameter on the CBC. Surprisingly, overlap feature has a negative effect on CBC's performance. Both average algorithm runtime and %gap increase when the overlap feature is introduced. However, when only feasible solutions are considered there is an improvement concerning the average gap, indicating that the overlap feature need not necessarily have a negative impact on CBC's solution quality and this instead may be attributed to the premature termination of the algorithm due to the runtime limit being exceeded.

Table 3: Summary of performance details of colour-based CMH strategies

Configuration	Average runtime	Average %gap	#feasible solutions	#optimal solutions	Average %gap(only feasible solutions)
CBC(1,0)	569.17	521.17	116	45	52.41
CBC(4,0)	1,694.07	2,806.29	72	41	63.77
CBC(8,0)	1,839.26	1,247.49	86	51	57.85
CBC(4,50)	2,129.31	3,490.89	52	14	48.79
CBC(8,50)	1,842.23	2,102.66	65	20	39.46

4.2 Vertex-based CMH

The performance of various vertex-based approaches is summarized in Table 4. In case of RVC, performance trends are rather irregular, with RVC(20,0,0) exhibiting the best performance when $\theta = 0$. In contrast to that of CBC, it is evident that overlap feature can be employed in the RVC to produce more feasible solutions. The overall average gap still remains high whereas the average gap calculated only over the feasible solutions decreases. This could be due to the large RVC runtimes as a result of implementing overlap. In case of exceeding runtime limit, the CMH terminates and returns the number of vertices (n) as the result. Therefore, a subset S of 95 instances on which all vertex-based CMH strategies produce feasible solutions has been constructed in order to make a fair comparison between their performances.

Table 4: Summary of performance details of various vertex-based CMH strategies

Configuration	Average runtime	Average %gap	#feasible solutions	#optimal solutions	Average %gap(only feasible solutions)	Average %gap(over S)
RVC(1,0)	917.08	1,103.59	104	35	34.45	35.41
RVC(10,0)	797.95	1,029.42	109	31	34.68	35.65
RVC(20,0)	767.18	1,142.73	107	38	34.02	34.35
RVC(10,50)	954.28	1,105.3	104	38	36.6	36.95
RVC(20,50)	913.79	1,103.09	104	37	33.82	34.8
SLC(1,0)	973.73	595.88	108	49	27.18	26.47
SLC(10,0)	703.43	879.43	112	60	23.04	23.29
SLC(20,0)	669.67	878.55	112	55	22.02	22.26
SLC(10,50)	899.41	1,250.6	106	58	21.41	22
SLC(20,50)	838.19	1,054.18	108	61	19.66	20.8
DSC(1,0)	1,204.19	2,148.55	95	48	26.14	26.14
DSC(10,0)	831.33	1,208.23	109	55	22.24	22.8
DSC(20,0)	762.19	1,207.75	109	60	21.67	21.98
DSC(10,50)	992.31	1,539.85	102	56	16.33	16.68
DSC(20,50)	928.37	1,489.96	105	56	17.07	17.64

Figure 7 shows a similar the coefficient matrix illustration for the instance *3-Insertions_3.col*, but with the variables and contraints arranged in a random order as considered in an RVC such as $x_{1,1}, x_{1,2}, ..., x_{2,1}, x_{2,2}, ..., x_{n,1}, x_{n,2}, ..., x_{n,n}$, $y_1, y_2, ..., y_n$. By sorting constraints, any constraint that contains a variable that appears in position i in the order appears before any constraint that contains a variable at position $j, j > i$. Let vertical lines l_0 and l_1 represent how vertices are divided to define three blocks b_0, b_1, b_2 of an RVC. The horizontal band at the top of the graph corresponds to Constraints 1d, which may be evaluated independently in each block. The challenge is to evaluate edge Constraints 1e. In this figure, rectangular window C_0 represents all the edge constraints that contain variables from block b_0. It can be easily seen that many of these constraints contain variables from other blocks as well. This means several vertices of block b_0 will be coloured without evaluating all of the

constraints they are contained in. It is therefore desirable to construct blocks such that a vertex is coloured after all constraints that contain it have been evaluated.

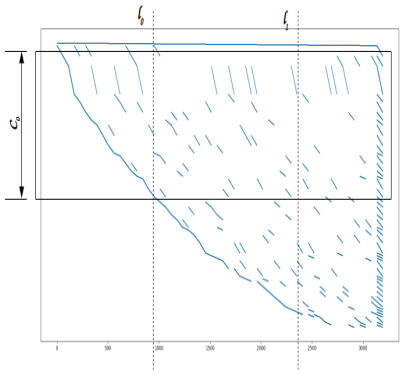


Fig. 7: The coefficient matrix of the VCP-ASS formulation when rows and columns are sorted in the order of vertices arranged in RVC.

Sequential colouring (SC) refers to greedy strategies which colours vertices of a graph one by one. It is proven that there exists an order which, when utilized by the SC, results in an optimal colouring. Matula et al. (1972) presents an in depth study of the influence of the order in which vertices are coloured in a SC heuristic. This work introduced the SL ordering and proved that when utilized by the SC, this order guarantees the max-subgraph-min-degree bound given by

$$\chi(G) \leq 1 + \max_{H:\text{subgraph of G}} \min_{v \in H} \{\deg_H(v)\}$$

Note that the configuration $(1, 0, 0)$ of vertex-based CMH corresponds to a SC heuristic and guarantees this bound when one uses SL ordering. The improvement of RVC solution quality with larger block sizes and when applying the overlap feature motivates similar experiments using SL-ordering in the vertex-based CMH, resulting in the SL-CMH or SLC.

Table 4 compares the vertex-based CMH performance with respect to design parameters η and θ. These experiments are based on the performance exhibited on the insances from set S. It is clear that utilizing SL ordering in the vertex-based CMH is an improvement over the RVC. The Table 4 also shows how increasing the block size improves solution quality. However, it must be noted that very large blocks lead to very long runtimes and hence the number of optimal solutions decreases for $\eta = 20$. Thus it is not feasible to increase block size further to improve SLC's performance. Overlap experiments show that this feature improve the overall performance of SLC. While it is clear that the algorithm generates high quality solutions, overlap also increases the algorithm runtime, thereby leading to the termination of the program before it generates a solution. This results in fewer feasible and optimal solutions and larger average %gaps compared to the results of SLC implemented without overlap.

From these experiments it is worth noting that, for overlap to be able to handle constraints linking blocks more effectively, it requires vertices which share Constraints 1e to be present in consecutive blocks. This idea motivates utilizing the DSATUR-based dynamic ordering in the CMH strategy, resulting in the DSC. While using DSATUR may not lead to solution quality improvements, it may exhibit better performance when employed with non-zero overlap.

In contrast with the CBC, the VBC offers a much more natural way of decomposing vertices. Consider a graph that can be decomposed into connected components. This leads to a natural decomposition of vertices. Each component can be independently coloured by implementing only the subset of Constraints 1d and 1e which are active in it and yet produce an optimal solution for the full problem. The objective of using a minimum number of colours can be achieved by implementing the full objective function 1c on all the components. The possibility of such an optimality-preserving decomposition is what motivates the vertex-based CMH. Although most of the graphs that we encounter in real-world examples cannot be decomposed in this fashion, whenever possible the idea is to arrange the vertices of the graph and design blocks of the CMH so that vertices that share edges are coloured in the same block. Figures 8 and 9 illustrate coefficient matrices when vertices are ordered in accordance with an SLC and DSC, respectively for the same instance *3-Insertions_3.col*.

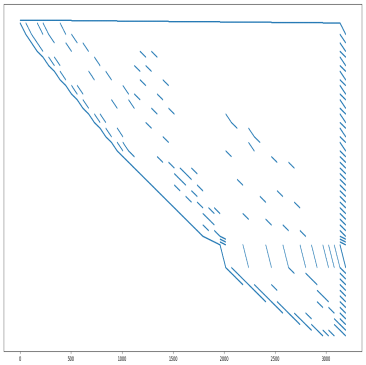


Fig. 8: The coefficient matrix of the VCP-ASS formulation for instance *3-Insertions_3.col* when rows and columns are sorted in the order of vertices arranged in SLC.

From figures 8 and 9, it is clear that SLC and DSC approaches offer the possibility of dividing vertices such that there are significantly fewer overlapping constraints across blocks. This could be a possible explanation for why they outperform the RVC approach. One must note here that the possibility of constructing such blocks depends on the structure of the graph. Moreover, the chromatic number of a graph is not a local property. For example, it is possible to construct Mycielski graphs of arbitrarily large chromatic numbers but which have a maximal clique size less than or equal to two. Moreover, SLC and DSC ordering will only result in significant

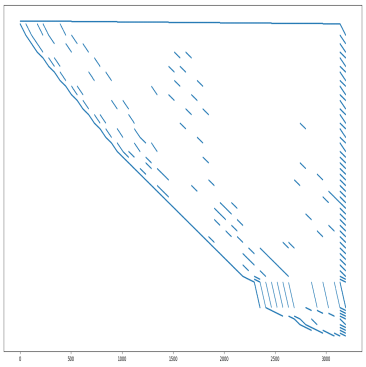


Fig. 9: The coefficient matrix of the VCP-ASS formulation for instance *3-Insertions_3.col* when rows and columns are sorted in the order of vertices arranged in DSC.

differences concerning grouping constraints when the vertices of the graph are less interconnected. This can be observed in the Figures 10, 11 and 12. They illustrate constraint matrices of instance *Queen5_5.col* when its vertices are sorted as in the RVC, SLC and DSC orders respectively. Compared to instance *3-Insertions_3.col*, instance *Queen5_5.col* is characterized by a denser graph in which vertices are highly interconnected. This means that none of the orders RVC, SLC or DSC can decompose vertices into blocks such that edge constraints only have variables from a particular block or even from consecutive blocks. Thus coefficient matrices show no significant difference concerning the distribution of constraints under any of these orders.

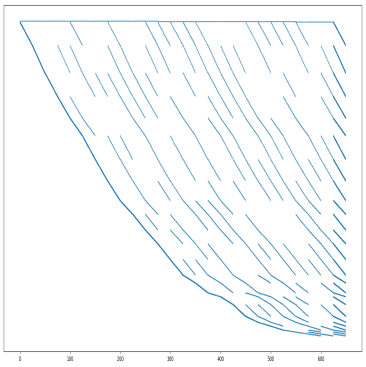


Fig. 10: The coefficient matrix of the VCP-ASS formulation for instance *Queen5_5.col* when rows and columns are sorted in the order of vertices arranged in RVC

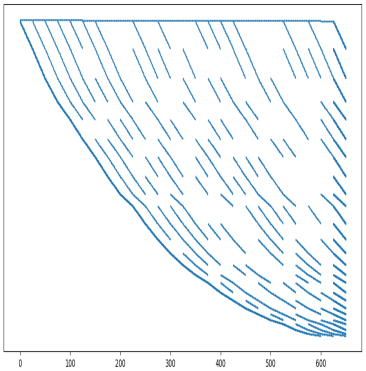


Fig. 11: The coefficient matrix of the VCP-ASS formulation for instance *Queen5_5.col* when rows and columns are sorted in the order of vertices arranged in SLC

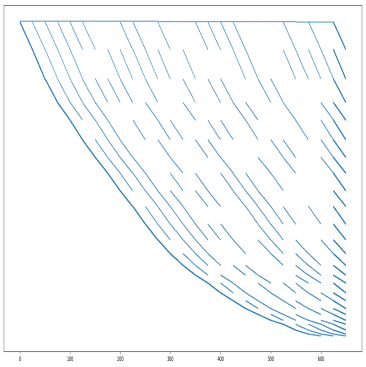


Fig. 12: The coefficient matrix of the VCP-ASS formulation for instance *Queen5_5.col* when rows and columns are sorted in the order of vertices arranged in DSC

Similar to all other CMH strategies discussed, the solution quality of DSC also improves upon increasing block size, as is evident from Table 4. Note that concerning experiments without overlap, the SLC exhibits the best overall performance. Table 4 also presents the results of DSC when implemented with overlap $\theta = 50$. Out of all the CMH strategies presented in the paper, DSC when implemented with overlap leads to the lowest %gap computed over the feasible solutions, indicating its ability to generate high quality solutions for the VCP. However, algorithm runtime also increases while adding overlap feature and affects algorithms overall efficiency. SLC exhibits the lowest impact on algorithm runtime when implementing overlap.

To further improve the method, an upper bound for the chromatic number is utilized to reduce the size of the formulation. This is done by executing a simple greedy colouring heuristic that colours vertices one after the other based on the SL ordering. Moreover, using the number of vertices(n) as the algorithm output on reaching timelimit contribute a disproportionately high value towards the average gap, making it an inefficient measure to study CMH performance with respect to other best performing algorithms. The SLC and DSC algorithms after implementing this modification is renamed as SLC' and DSC' respectively. The results of the SLC' and DSC' on the benchmark instances is summarized in Table 5. While adding this lower bounding procedure increases the average runtime, it guarantees that the algorithm outputs a feasible solution to all instances. This also explains the identical values for the average gap and average gap (feasible) for results in Table 5.

4.3 Solving subproblems using constraint programming

Even though integer programming formulations have been employed as the exact method to solve subproblems in all the previous experiments, the generality of the CMH approach allows even other exact approaches to be used for solving subproblems. Recent advances in the area of constraint programming(CP) solvers motivates using a CP formulation as the exact method within the CMH algorithm. The similarity between the integer programming and constraint programming approaches lets us use the same CMH framework except that the subproblems are solved utilizing an equivalent CP formulation. The algorithms obtained on utilizing the CP formulation in DSC' and SLC' are denoted as DSC'-CP and SLC'-CP respectively.

Let c_v denote the colour assigned to vertex v in a given block b_k and let z denote the total number of colours utilized. The simple CP formulation utilized to solve the block b_k can now be formulated as follows.

$$\text{minimize: } z \tag{4}$$

$$\text{subject to: } z \geq c_v \qquad \forall v \in b_k \tag{5}$$

$$\text{alldiff}(c_v, c_{v'}) \forall (v, v') \in E_{b_k} \tag{6}$$

$$c_v \in \{0, 1, ..., n-1\} \qquad \forall v \in b_k \tag{7}$$

$$z \in \{0, 1, ..., n-1\} \tag{8}$$

JaCoP 4.7 has been utilized to solve the CP formulations. The performance summary of DSC'-CP and SLC'-CP on the benchmark instances is presented in Table 5 for comparison.

Table 5: Summary of performance details of SLC', DSC', SLC'-CP and DSC'-CP.

Configuration	Average runtime	Average %gap	#feasible solutions	#optimal solutions	Average %gap(feasible)
SLC'(1,0)	877.32	28.14	131	53	28.14
SLC'(10,0)	561.42	24.93	131	57	24.93
SLC'(20,0)	485.33	23.68	131	58	23.68
SLC'(10,50)	652.79	23.67	131	58	23.67
SLC'(20,50)	591.18	22.33	131	61	22.33
DSC'(1,0)	1,194.75	20.35	131	55	20.35
DSC'(10,0)	773.27	24.02	131	46	24.02
DSC'(20,0)	699.66	21.84	131	53	21.84
DSC'(10,50)	854.03	19.54	131	56	19.54
DSC'(20,50)	753.77	20.48	131	52	20.48
SLC'-CP(1,0)	66.11	28.14	131	53	28.14
SLC'-CP(10,0)	115.1	24.33	131	57	24.33
SLC'-CP(20,0)	1,477.92	24.25	131	59	24.25
SLC'-CP(10,50)	143.96	23.28	131	57	23.28
SLC'-CP(20,50)	1,511.4	23.05	131	59	23.05
DSC'-CP(1,0)	379.04	27.06	131	50	27.06
DSC'-CP(10,0)	172.83	26.14	131	55	26.14
DSC'-CP(20,0)	1,453.86	26.56	131	53	26.56
DSC'-CP(10,50)	194.72	25.35	131	52	25.35
DSC'-CP(20,50)	1,528.31	24.92	131	55	24.92
SLC'-400(1,0)	174.26	27.72	131	52	27.72
SLC'-400(10,0)	100.79	24.6	131	57	24.6
SLC'-400(20,0)	91.4	23.44	131	57	23.44
SLC'-400(10,50)	118.54	23.36	131	57	23.36
SLC'-400(20,50)	107.55	22.31	131	60	22.31
DSC'-400(1,0)	219.41	21.12	131	55	21.12
DSC'-400(10,0)	133.74	23.91	131	52	23.91
DSC'-400(20,0)	120.2	21.61	131	56	21.61
DSC'-400(10,50)	159.49	20.16	131	56	20.16
DSC'-400(20,50)	139.08	20.24	131	55	20.24
SLC'-CP-400(1,0)	7.31	27.72	131	52	27.72
SLC'-CP-400(10,0)	11.5	23.9	131	56	23.9
SLC'-CP-400(20,0)	167.77	23.83	131	58	23.83
SLC'-CP-400(10,50)	15.73	22.85	131	56	22.85
SLC'-CP-400(20,50)	171.55	22.62	131	58	22.62
DSC'-CP-400(1,0)	90.08	28.03	131	47	28.03
DSC'-CP-400(10,0)	47	26.44	131	53	26.44
DSC'-CP-400(20,0)	173.52	27.28	131	51	27.28
DSC'-CP-400(10,50)	64.64	25.45	131	51	25.45
DSC'-CP-400(20,50)	183.86	24.5	131	54	24.5

For block sizes less than 20, DSC'-CP and SLC'-CP exhibit the same performance in shorter average runtimes than the SLC' and DSC' algorithms. However, for $\eta = 20$ and greater, the average runtime of DSC'-CP and SLC'-CP increases rapidly. This implies that while utilizing a CP formulation makes the CMH approach for vertex colouring more efficient for small block sizes, the same does not hold for large block sizes. The observation, however, that a large share of the CP solver runtime is dedicated towards proving the optimality of the solution rather

than at arriving the solution, experiments to test the DSC'-CP and SLC'-CP when given a runtime limit of 400 seconds to solve the subproblems were conducted. These algorithms are denoted as SLC'-CP-400 and DSC'-CP-400. Note that in these two algorithms, the subproblems are not solved using an exact method but are instead solved heuristically, deviating from the original definition of the CMH. The performance of algorithms SLC'-400 and DSC'-400 obtained by restricting runtimes of SLC' and DSC' respectively are also summarized in Table 5.

As expected and evident in Table 5, the algorithms benefit from using an upper bound. The algorithm runtimes are shorter and as a result, leads to higher number of optimal solutions and smaller %gaps. With this improvement, DSATUR-based CMHs exhibit superior performance over the SL-based heuristics with respect to %gaps. However, the algorithm runtimes are still longer than for DSATUR-based approaches due to the dynamic sorting step that must be conducted in each iteration. In general, it can be observed that algorithms utilizing CP are efficient for smaller block sizes, $\eta = 1$ and $\eta = 10$, but are inefficient for $\eta = 20$. For $\eta = 20$, the algorithm may have to be prematurely stopped due to longer runtimes, thereby generating fewer optimal solutions and larger %gap. The fact that part of the runtime is spent on proving the optimality was the motivation to implement a smaller runtime limit such as 400s. Experiments show that giving a runtime limit on subproblems significantly reduces algorithm runtimes, while still generating solutions of almost the same quality. This offers the possibility of experimenting with larger block sizes. Clearly DSC' exbits the best performance. Note here that DSC'-400 is capable of producing almost equally good solutions with much shorter runtime. Therefore, in terms of efficiency DSC'-400 outperforms all other CMHs designed for the VCP in this paper.

Tables 6 and 7 present the performance of SLC', DSC', SLC'-CP, DSC'-CP, SLC'-400, DSC'-400, SLC'-CP-400 and DSC'-CP-400 on the difficult VCP instances. The configuration used during these experiments is $\eta = 20$ and $\theta = 50$. The performance of some of the best performing algorithms available in the literature such as Morgenstern (1996), Hertz, Plumettaz, and Zufferey (2008), Funabiki and Higashino (2000) and Malaguti et al. (2008) is also presented for comparison purposes. All of them are metaheuristics and constitute the best performing algorithms for the VCP. Morgenstern (1996) introduced an interesting neighbourhood structure called the Impasse Class used within a simulated annealing scheme to design one of the first efficient metaheuristics for the VCP. Hertz et al. (2008) introduced the variable search space heuristic based on the principles of variable neighbourhood search. Funabiki and Higashino (2000) proposed a Tabu Search heuristic which incorporates the Impasse Class neighbourhoods introduced by Morgenstern (1996). Finally, Malaguti et al. (2008) presents one of the best existing heuristics for the VCP. A genetic algorithm approach is utilized which also incorporates Tabu Search principles and ideas from the Impasse Class neighbourhood structures introduced by Morgenstern (1996). The algorithm makes use of solutions of a variety of greedy heuristics such as DSATUR and RLF to generate the initial pool of solutions to be used in the genetic algorithm framework. While the CMHs introduced have been able to generate high quality solutions for several of these difficult instances, they are inefficient on others due to CMH exceeding the runtime limit of 3600s. A comparison between the performances of SLC', SLC'-CP, DSC', DSC'-CP with those of SLC'-400, SLC'-CP-400, DSC'-400, DSC'-CP-400 shows that for very large and difficult instances, it is beneficial to use

a runtime limit when solving subproblems so as to prevent early termination of the CMH.

Table 6: Performance of the different vertex-based CMHs and some of the best performing algorithms such as Impasse: (Morgenstern, 1996), VSS-Col: (Hertz et al., 2008), MIPS_CLR: (Funabiki and Higashino, 2000) and MMT: (Malaguti et al., 2008) on the difficult VCP instances. k denotes the number of colours used and T(s) denotes the algorithm runtime in seconds.

				Impasse		**VSS-Col**		**MIPS_CLR**			**MMT**			**SLC**		**DSC**	
Instance	n	m	χ_G	best	k	T(s)	k	T(s)	best	avg.	T(s)	best	avg.	T(s)	k	T(s)	k	T(s)
DSJC125.1	125	736		5					5	5	0	5	5	21	7	0.98	6	0.98
DSJC125.5	125	3,891		17	17	1			17	17	1	17	17	122	24	5.36	23	4.77
DSJC125.9	125	6,961		44					44	44	0	44	44	121	51	13.64	52	12.23
DSJC250.1	250	3,218		5					8	8	5	8	8	21	12	6.28	10	6.7
DSJC250.5	250	15,668		17	28	22			28	28.4	14	28	28	117	40	48.05	38	48.63
DSJC250.9	250	27,897		44					72	72.4	31	72	72	89	92	149.45	93	145.23
DSJC500.1	500	12,458		8			12	97	12	12.4	84	12	12.25	210	18	52.04	17	89.12
DSJC500.5	500	62,624		28	49	660	48	1,331	49	49.4	349	48	48.25	388	70	528.8	66	623.06
DSJC500.9	500	$1.12 \cdot 10^6$		72			126	1,686	127	127.8	480	127	127.75	433	176	3,600	176	3,600
DSJC1000.1	1,000	49,629		12			20	2,396	21	21	90	20	20.25	260	31	528.83	27	1,694.86
DSJC1000.5	1,000	$2.5 \cdot 10^5$		48	89	1,148	88	2,028	88	89	4,658	84	84.25	8,407	124	3,600	124	3,600
DSJC1000.9	1,000	$4.49 \cdot 10^5$		126			224	3,326	228	229.6	1,565	225	226	3,234	318	3,600	318	3,600
DSJR500.1	500	3,555	12	12	12	0			12	12	0	12	12	25	12	14.18	13	41.6
DSJR500.1c	500	$1.21 \cdot 10^5$		84	85	5	85	736	85	85	6	85	85	88	103	1,604.57	93	1,720.93
DSJR500.5	500	58,862	122	122	123	14	126	1,409	122	123.4	276	122	122	163	125	1,030.55	126	1,279.98
le450_15a	450	8,168	15	15	15	0			15	15	1	15	15	0	17	32.29	17	49.07
le450_15b	450	8,169	15	15	15	0			15	15	1	15	15	0	18	32.39	17	46.13
le450_15c	450	16,680	15	15	15	5	15	6	15	15.2	11	15	15	3	25	76.93	24	94.98
le450_15d	450	16,750	15	25	15	3	15	44	15	15	5	15	15	4	26	77.83	25	98.29
le450_25c	450	17,343	25	25			26	1	26	26	7	25	25	1,321	30	90.98	29	115.36
le450_25d	450	17,425	25	25			26	1	26	26.4	1	25	25	436	30	89.87	30	114.23
r250.1	250	867	8	8	8	0			8	8	0	8	8	26	8	1.85	8	2.04
r250.1c	250	30,227	64	64	64	0			64	64	2	64	64	21	67	149.46	66	147.02
r250.5	250	14,849	65	65	65	7			65	65.8	16	65	65	64	67	74.85	67	78.5
r1000.1	1,000	14,378	20	20	20	1			20	20	0	20	20	37	20	143.54	20	540.35
r1000.1c	1,000	$4.85 \cdot 10^5$		98	98	46			98	98.8	557	98	98	518	120	3,600	120	3,600
r1000.5	1,000	$2.38 \cdot 10^5$	234	234	241	77			237	238.6	1,345	234	234	753	251	3,600	251	3,600
latin_square_10	900	$3.07 \cdot 10^5$		100	98	415			99	100.2	938	101	102	5,156	213	3,600	213	3,600
flat300_20_0	300	21,375	20	20	20	0			20	20	2	20	20	21	43	83.37	41	83.79
flat300_26_0	300	21,633	26	26	26	1			26	26	1	26	26	36	43	85.07	43	87.44
flat300_28_0	300	21,695	28	28	31	156	29	867	31	31	133	31	31	212	46	87.16	44	87.84
flat1000_50_0	1,000	$2.45 \cdot 10^5$	50	50	50	0	50	318	50	50	14	50	50	1,417	125	3,600	125	3,600
flat1000_60_0	1,000	$2.46 \cdot 10^5$	60	60	60	0	60	694	60	60	59	60	60	3,645	123	3,600	123	3,600
flat1000_76_0	1,000	$2.47 \cdot 10^5$		82	89	897	87	1,689	87	87.8	2,499	83	83.5	7,325	125	3,600	125	3,600
Average gap											7.07			1.83		65.93		60.75
Average runtime											386.79			1,020.41		1,100.24		1,165.09
Maximum runtime											4,658			8,407		3,600		3,600

Table 7: Performance of SLC', DSC', SLC'-CP and DSC'-CP on some of the difficult VCP instances. k denotes the number of colours used and T(s) denotes the algorithm runtime in seconds.

					SLC'-CP		**DSC'-CP**		**SLC'-400**		**DSC'-400**		**SLC'-CP-400**		**DSC'-CP-400**	
Instance	n	m	χ_G	best	k	T(s)	k	T(s)	k	T(s)	k	T(s)	k	T(s)	k	T(s)
DSJC125.1	125	736		5	7	$9 \cdot 10^{-3}$	6	$1.9 \cdot 10^{-2}$	7	0.91	6	1.17	7	$1.1 \cdot 10^{-2}$	6	$2 \cdot 10^{-2}$
DSJC125.5	125	3,891		17	23	0.29	24	0.29	24	5.21	24	5.45	23	0.26	24	0.29
DSJC125.9	125	6,961		44	55	3,600	55	3,600	51	13.75	50	14.34	55	400	55	400
DSJC250.1	250	3,218		5	11	$2.2 \cdot 10^{-2}$	12	0.15	12	6.63	11	8.13	11	$5.8 \cdot 10^{-2}$	12	0.15
DSJC250.5	250	15,668		17	41	3.44	42	3.97	40	47.67	38	55.3	41	3.51	42	4.01
DSJC250.9	250	27,897		44	95	3,600	95	3,600	92	146.73	89	168.63	95	400	95	400
DSJC500.1	500	12,458		8	18	0.13	19	1.95	18	50.67	17	105.73	18	0.14	19	1.94
DSJC500.5	500	62,624		28	70	3.11	69	11.33	72	400	72	400	70	3.19	69	11.14
DSJC500.9	500	$1.12 \cdot 10^6$		72	176	3,600	174	3,600	174	400	174	400	174	400	174	400
DSJC1000.1	1,000	49,629		12	29	1.02	30	29.3	30	400	30	400	29	1.01	30	28.82
DSJC1000.5	1,000	$2.5 \cdot 10^5$		48	124	14.91	125	145.51	124	400	124	400	124	14.86	125	144.57
DSJC1000.9	1,000	$4.49 \cdot 10^5$		126	318	3,600	318	3,600	318	400	318	400	318	400	318	400
DSJR500.1	500	3,555	12	12	12	3,600	12	3,600	12	14.41	12	44.3	12	400	12	400
DSJR500.1c	500	$1.21 \cdot 10^5$		84	107	3,600	107	3,600	107	400	107	400	107	400	107	400
DSJR500.5	500	58,862	122	122	131	3,600	131	3,600	131	400	131	400	131	400	131	400
le450_15a	450	8,168	15	15	18	3,600	18	3,600	17	32.15	17	70.75	18	400	18	400
le450_15b	450	8,169	15	15	17	3,600	17	3,600	18	32.17	16	68.77	17	400	17	400
le450_15c	450	16,680	15	15	26	1.79	26	3.3	25	75.18	25	134.83	26	1.77	26	3.66
le450_15d	450	16,750	15	25	26	1.94	26	3.23	26	77.11	25	136.41	26	1.92	26	3.69
le450_25c	450	17,343	25	25	31	3,600	31	3,600	30	89.62	29	161.13	31	400	31	400
le450_25d	450	17,425	25	25	30	3,600	30	3,600	30	92.36	29	160.02	30	400	30	400
r250.1	250	867	8	8	8	4.68	8	2,237.72	8	1.74	8	2.82	8	4.49	8	400
r250.1c	250	30,227	64	64	68	3,600	68	3,600	67	149.82	66	175.08	68	400	68	400
r250.5	250	14,849	65	65	68	3,600	68	3,600	67	75.83	66	92.47	68	400	68	400
r1000.1	1,000	14,378	20	20	20	3,600	20	3,600	20	142.6	20	400	20	400	20	400
r1000.1c	1,000	$4.85 \cdot 10^5$		98	120	3,600	120	3,600	120	400	120	400	120	400	120	400
r1000.5	1,000	$2.38 \cdot 10^5$	234	234	251	3,600	251	3,600	251	400	251	400	251	400	251	400
latin_square_10	900	$3.07 \cdot 10^5$		100	213	3,600	213	3,600	213	400	213	400	213	400	213	400
flat300_20_0	300	21,375	20	20	45	11.01	46	10.74	43	80.6	41	101.51	45	9.83	46	10.78
flat300_26_0	300	21,633	26	26	46	5.05	45	5.89	43	83.62	44	106.46	46	4.72	45	5.76
flat300_28_0	300	21,695	28	28	44	0.62	46	1.65	46	85.44	44	105.5	44	0.59	46	1.64
flat1000_50_0	1,000	$2.45 \cdot 10^5$	50	50	124	15.11	123	143.7	125	400	125	400	124	14.85	123	144.14
flat1000_60_0	1,000	$2.46 \cdot 10^5$	60	60	123	16.33	123	144.76	123	400	123	400	123	15.71	123	145.42
flat1000_76_0	1,000	$2.47 \cdot 10^5$		82	119	14.65	122	145.92	125	400	125	400	119	14.25	122	142.09
Average gap						66.16		67.28		66.1		62.71		66.08		67.28
Average runtime						1,802.77		1,884.98		191.3		215.26		202.68		230.83
Maximum runtime						3,600		3,600		400		400		400		400

5 Results and Discussion

This paper presents the first extensive experimentation conducted in order to design a constructive matheuristic (CMH) strategy for the vertex colouring problem (VCP), thereby suggesting the possibility of utilizing matheuristic techniques to tackle the VCP. The primary insight gained from this work is that it is possible to extend subproblems of successive augmentation techniques and employ simple integer programming approaches to arrive at feasible solutions for the VCP. This, in turn, becomes a way of designing a CMH. Experiments show that algorithm design parameters such as overlap can be effectively utilized to improve the solution quality when used in combination with an appropriate decomposition strategy. The major challenge, however, is the very high impact of such features on algorithmic runtime.

The paper compares and contrast the efficiency of using the colour-based and vertex-based decomposition strategies in the CMH. The results of the extensive experimentation suggests that the vertex based approach is much more suited for application in CMH and offers possibilities for further improvements. These experiments also suggest the general trends one can expect when employing such decomposition strategies in other vertex colouring and vertex colouring-like problems, especially the problems that are time-based.

Specifically, there is a need to identify smarter decomposition strategies such that the subproblems involved can be solved more efficiently. The present approach designs blocks by making partitions of equal size on the vertices or colours, which need not partition the problem into equally difficult subproblems. Therefore, further research on how to design the CMH approach so as to adapt itself based on the structure of the instance concerned may be interesting. Another interesting area of research can be introducing more algorithmic parameters to better navigate the CMH to produce high quality solutions.

References

Karen I Aardal, Stan PM Van Hoesel, Arie MCA Koster, Carlo Mannino, and Antonio Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, 2007.

Hamed Babaei, Jaber Karimpour, and Amin Hadidi. A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86: 43–59, 2015.

Daniel Brélaz. New methods to color vertices of a graph. *Commun. ACM*, 22: 251–256, 1979.

Reshma Chirayil Chandrasekharan, Túlio A. M. Toffolo, and Tony Wauters. Analysis of a constructive matheuristic for the traveling umpire problem. *Journal of Quantitative Analysis in Sports*, 15(1):41–57, 2019.

Reshma Chirayil Chandrasekharan, Pieter Smet, and Tony Wauters. An automatic constructive matheuristic for the shift minimization personnel task scheduling problem. *Journal of Heuristics*, pages 1–23, 2020.

Fred C Chow and John L Hennessy. The priority-based coloring approach to register allocation. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 12(4):501–536, 1990.

Nobuo Funabiki and Teruo Higashino. A minimal-state processing search algorithm for graph coloring problems. *IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences*, 83(7):1420–1430, 2000.

Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of combinatorial optimization*, 3(4):379–397, 1999.

Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Alain Hertz, Matthieu Plumettaz, and Nicolas Zufferey. Variable space search for graph coloring. *Discrete Applied Mathematics*, 156(13):2551–2560, 2008.

David S Johnson, Cecilia R Aragon, Lyle A McGeoch, and Catherine Schevon. Optimization by simulated annealing: an experimental evaluation; part ii, graph coloring and number partitioning. *Operations research*, 39(3):378–406, 1991.

Frank Thomson Leighton. A graph coloring algorithm for large scheduling problems. *Journal of research of the national bureau of standards*, 84(6):489–506, 1979.

Enrico Malaguti and Paolo Toth. A survey on vertex coloring problems. *International transactions in operational research*, 17(1):1–34, 2010.

Enrico Malaguti, Michele Monaci, and Paolo Toth. A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing*, 20(2):302–316, 2008.

Enrico Malaguti, Michele Monaci, and Paolo Toth. An exact approach for the vertex coloring problem. *Discrete Optimization*, 8(2):174–190, 2011.

V Maniezzo, T Stützle, and S Voß. Matheuristics, volume 10 of annals of information systems, 2010.

David W. Matula, George Marble, and Joel D. Isaacson. Graph coloring algorithms. pages 109 – 122, 1972. doi: https://doi.org/10.1016/B978-1-4832-3187-7.50015-5.

Isabel Méndez-Díaz and Paula Zabala. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics*, 154(5):826–847, 2006.

Isabel Méndez-Díaz and Paula Zabala. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics*, 156(2):159–179, 2008.

Craig Morgenstern. Distributed coloration neighborhood search. *Discrete Mathematics and Theoretical Computer Science*, 26:335–358, 1996.

Pieter Smet, Tony Wauters, Mihail Mihaylov, and Greet Vanden Berghe. The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. *Omega*, 46(Supplement C):64 – 73, 2014.