A constructive matheuristic approach for the vertex colouring problem

Reshma Chirayil Chandrasekharan ·
Tony Wauters

Received: date / Accepted: date

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. In addition, the impact of algorithm design parameters on the particular decompositions used and their influence on final solution quality is also explored.

Keywords Vertex colouring · matheuristic · decomposition

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

Reshma Chirayil Chandrasekharan
KU Leuven
Department of Computer Science
CODeS
Gebroeders De Smetstraat 1
9000 Gent
Belgium
E-mail: ReshmaCC@kuleuven.be

Tony Wauters
KU Leuven
Department of Computer Science
CODeS
Gebroeders De Smetstraat 1
9000 Gent
Belgium
E-mail: tony.wauters@kuleuven.be

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)), 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.

Malaguti and Toth (2010) provide a useful survey on the various exact and heuristic algorithms developed for the VCP. Several high performing algorithms for the VCP such as Malaguti, Monaci, and Toth (2008), Funabiki and Higashino (2000), Galinier and Hao (1999) suggest that hybrid methods are efficient in colouring some of the very large random graphs. This paper presents some preliminary experiments conducted in order to 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.

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 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). 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. 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 Vanden Berghe (2014). This constructive heuristic sequentially solves subproblems and utilizes these optimal solutions to construct a solution for the entire original problem. The subproblems of a CMH strategy are called blocks.

The following CMH design parameters introduced in Chandrasekharan, Toffolo, and Wauters (2019) are also tested.

1. Block size (η): This parameter defines the size of subproblems and often significantly influences algorithmic runtime.
2. Overlap (θ): This parameter allows blocks to share some constraints instead of being completely disjoint. θ denotes the extent of overlap between consecutive blocks.
3. Relaxed future (ρ): This feature allows the CMH to have larger subproblems by solving a part of the block in a relaxed fashion, which tends to have less of an impact on algorithmic runtime than increasing block size. The size of the relaxed part is expressed as percentage (ρ) of the size of the original block. The relaxed part must be later solved again with the original formulation to ensure feasibility.

A CMH configuration is therefore represented by the three tuple (η, θ, ρ). Figure 1 illustrates the overall CMH strategy utilized in this paper and the design parameters. The CMH approach utilized relaxes one or more constraints of an IP formulation for the VCP in order to define the decomposition. The subproblems generated are then sequentially solved by a MIP solver. Depending on the solutions of previously solved subproblems, additional constraints may need to be added to the blocks or objective functions utilized in the blocks modified to ensure feasibility of the final solution. Given a block b, its definition and the precise optimization problem it solves is realized by means of its *block objective function* $Z_b(\eta, \theta, \rho)$.

The CMH strategy 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

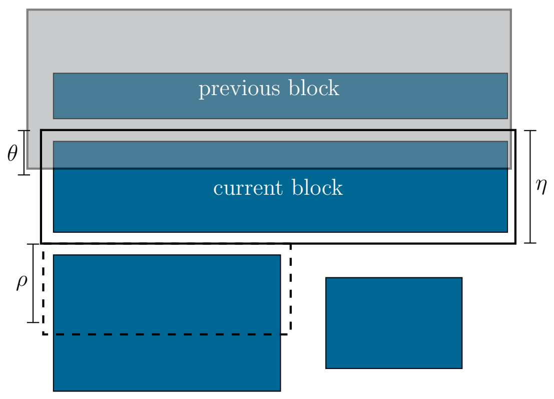


Fig. 1: An overview of the general CMH strategy. Blue rectangles represent subproblems (blocks) in the CMH strategy and the solid window represents current block. The gray rectangle represents the block previously solved and the dotted rectangle represents the part of a future unsolved block solved in a relaxed fashion.

or not.

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

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

The model can then be formulated as follows:

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

$$\text{subject to: } \quad \sum_{h=1}^{n} x_{ih} = 1 \quad \forall i \in V \tag{4}$$

$$x_{ih} + x_{jh} \leq y_h \quad \forall (i,j) \in E, h = 1,\dots,n \tag{5}$$

$$y_{h+1} \leq y_h \quad \forall i = 0,\dots,n-1 \tag{6}$$

$$x_{ih} \in \{0,1\} \quad \forall i \in V, h = 1,\dots,n \tag{7}$$

$$y_h \in \{0,1\} \quad \forall h = 1,\dots,n \tag{8}$$

Constraints 4 ensure that vertices are assigned exactly one colour, while Constraints 5 prevent adjacent vertices from being assigned the same colour. Constraints 6 are introduced to break the symmetry incurred by the interchangeability of colours.

The aim is to develop a CMH algorithm for the VCP inspired by the recent success of CMH techniques for task scheduling problems, as discussed in both Smet et al. (2014) and Chandrasekharan, Smet, and Wauters (2020). The CMH strategy aims to optimally colour subgraphs of G in a sequential fashion by utilizing their MIP formulations, and then eventually combining them together into a solution for the entire graph. The two obvious candidates for how to decompose a graph led to the development of two decomposition strategies: vertex-based and colour-based CMH strategies.

3.1 Vertex-based CMH

The vertex-based CMH (VBC) defines blocks by way of groups of vertices. The induced subgraph defined by a block is coloured to optimality by utilizing its MIP formulation. Once a block is solved, the colour assignments are fixed and the following blocks are solved such that they do not contradict previously fixed colour assignments. Let b_k denote the current block and E_{b_k} denote the induced subgraph corresponding to the vertices belonging to blocks $b_0, \ldots, b_k$. The MIP formulation solved in block b_k can now be formulated as follows.

$$x_{ih} = \begin{cases} 1 & \text{if vertex } i \in b_k \text{ is assigned to colour } h \in H \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$y_h = \begin{cases} 1 & \text{if colour } h \in H \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$\text{minimize:} \quad \sum_{h=1}^{n} y_h \tag{11}$$

$$\text{subject to:} \quad \sum_{h=1}^{n} x_{ih} = 1 \quad \forall i \in b_k \tag{12}$$

$$x_{ih} + x_{jh} \leq y_h \quad \forall (i,j) \in E_{b_k}, h = 1, \ldots, n \tag{13}$$

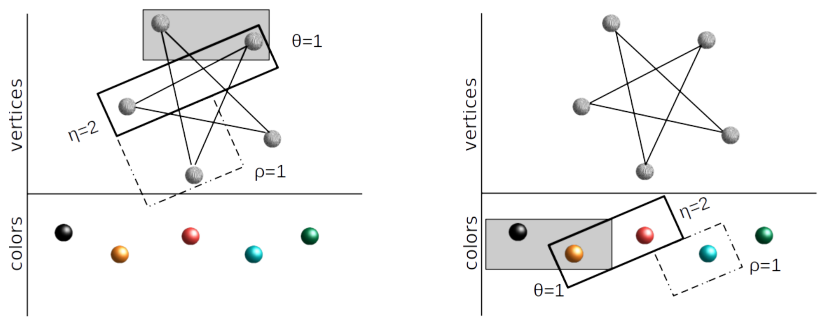
$$y_{h+1} \leq y_h \quad \forall h = 0, \ldots, n-1 \tag{14}$$

$$x_{ih} \in \{0,1\} \quad \forall i \in b_k, h = 1, \ldots, n \tag{15}$$

$$y_h \in \{0,1\} \quad \forall h = 1, \ldots, n \tag{16}$$

3.2 Colour-based CMH

The colour-based CMH (CBC) defines blocks by means of grouping colours. In each block of colours, the CMH solves for the maximal subgraph that can be coloured by the colours in the block and fixes those assignments. The CMH stops when all vertices have been coloured. In contrast with the vertex-based CMH, the block objective function of the CBC maximizes the number of vertices that can be coloured by the colours in a given block. The following is the



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

Fig. 2: The CBC and VBC strategies. Solid rectangles represent blocks. Gray windows correspond to blocks previously solved, while the dotted rectangles represent the relaxed future part of the current block.

MIP formulation solved in each block b_k:

$$x_{ih} = \begin{cases} 1 & \text{if vertex } i \in V \text{ is assigned to colour } h \in b_k \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$y_h = \begin{cases} 1 & \text{if colour } h \in b_k \text{ is used} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

The model can then be formulated as:

$$\text{maximize: } \sum_{h \in b_k} \sum_{i \in V} x_{ih} \tag{19}$$

$$\text{subject to: } x_{ih} + x_{jh} \le y_h \quad \forall (i,j) \in E, h \in b_k \tag{20}$$

$$y_{h+1} \le y_h \quad \forall h \in \{b_{k_0}, \dots, b_{k_{|b_k|-1}}\} \tag{21}$$

$$x_{ih} \in \{0,1\} \quad \forall i \in V, h \in b_k \tag{22}$$

$$y_h \in \{0,1\} \quad \forall h \in b_k \tag{23}$$

The final block is re-optimized with the original objective of minimizing the number of colours. Figure 2 illustrates CBC and VBC strategies.

The proposed 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, Marble, and Isaacson (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 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 composed of uncoloured vertices with the first η largest saturation degrees in the partially coloured graph. Ties are broken utilizing the SL order.

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 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

4.1 Colour-based CMH

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 2 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 to isolate independent sets in the graph, which can be an NP hard 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 2: Summary of performance details of colour-based CMH strategies

Configuration	Average runtime	Average %gap	#feasible solutions	#optimal solutions	Average %gap(only feasible solutions)
"CBC(1,0,0)"	569.17	521.17	116	45	52.41
"CBC(4,0,0)"	1,694.07	2,806.29	72	41	63.77
"CBC(8,0,0)"	1,839.26	1,247.49	86	51	57.85
"CBC(4,50,0)"	2,129.31	3,490.89	52	14	48.79
"CBC(8,50,0)"	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 3. 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 3: 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(1,0,0)"	917.08	1,103.59	104	35	34.45	35.41
"RVC(10,0,0)"	797.95	1,029.42	109	31	34.68	35.65
"RVC(20,0,0)"	767.18	1,142.73	107	38	34.02	34.35
"RVC(10,50,0)"	954.28	1,105.3	104	38	36.6	36.95
"RVC(20,50,0)"	913.79	1,103.09	104	37	33.82	34.8
"SLC(1,0,0)"	973.73	595.88	108	49	27.18	26.47
"SLC(10,0,0)"	703.43	879.43	112	60	23.04	23.29
"SLC(20,0,0)"	669.67	878.55	112	55	22.02	22.26
"SLC(10,50,0)"	899.41	1,250.6	106	58	21.41	22
"SLC(20,50,0)"	838.19	1,054.18	108	61	19.66	20.8
"DSC(1,0,0)"	1,204.19	2,148.55	95	48	26.14	26.14
"DSC(10,0,0)"	831.33	1,208.23	109	55	22.24	22.8
"DSC(20,0,0)"	762.19	1,207.75	109	60	21.67	21.98
"DSC(10,50,0)"	992.31	1,539.85	102	56	16.33	16.68
"DSC(20,50,0)"	928.37	1,489.96	105	56	17.07	17.64

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.

Figure 3 compares the vertex-based CMH performance with respect to design parameters η and θ. These experiments are based on the performance 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 figure 3 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 clear that while overlap can improve the CMH solution quality, its influence on algorithm runtime makes the overall CMH strategy rather inefficient. However, it is worth noting that, for overlap to be able to handle constraints linking blocks more effectively, it requires vertices which share Constraints 5 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.

Similar to all other CMH strategies discussed, the solution quality of DSC also improves upon increasing block size, as is evident from Figure 3 and Table 3. Note that concerning experiments without overlap, the SLC exhibits the best overall performance. Table 3 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 coloring heuristic that colors 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 and their performance on the difficult VCP instances is presented in Table 4 along with that of the other best performing heuristics for comparison.

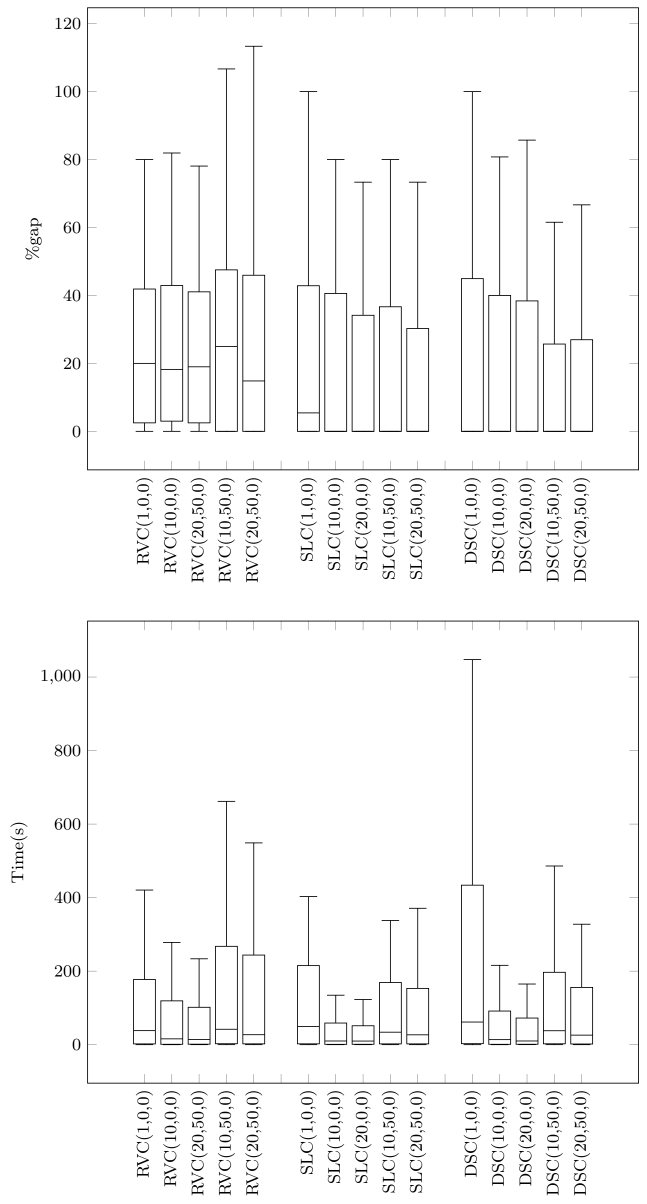


Fig. 3: Performance comparison of various vertex-based CMH approaches

Table 4: Performance of SLC and DSC on some of the difficult VCP instances. Performance details of some best performing algorithms are also provided for comparison such as Impasse: (Morgenstern, 1996), VSS-Col: (Hertz et al., 2008), MIPS_CLR: (Funabiki and Higashino, 2000) and MMT: (Malaguti et al., 2008). k denotes the number of colors used and T(s) denotes the algorithm runtime in seconds.

					Impasse		**VSS-Col**		**MIPS_CLR**					**MMT**	**SLC'(10,50,0)**		**SLC'(20,50,0)**		**DSC'(10,50,0)**		**DSC'(20,50,0)**	
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)	k	T(s)	k	T(s)
DSJC125.1	125	736		5					5	5	0	5	5	21	7	1.15	7	0.42	6	0.95	6	0.58
DSJC125.5	125	3,891		17	17	1			17	17	1	17	17	122	24	8.97	24	6.11	21	9.8	23	5.99
DSJC125.9	125	6,961		44					44	44	0	44	44	121	55	30.02	51	19.17	51	31.29	50	19.49
DSJC250.1	250	3,218		8					8	8	5	8	8	21	12	7.46	11	3.84	11	11	11	5.75
DSJC250.5	250	15,668		28	28	22			28	28.4	14	28	28	117	41	110.45	42	66.57	37	124.01	37	71.62
DSJC250.9	250	27,897		72					72	72.4	31	72	72	89	95	381.77	92	217.38	92	414.78	92	234.5
DSJC500.1	500	12,458		12			12	97	12	12.4	84	12	12.25	210	18	84.57	18	45.93	16	192.83	17	97.7
DSJC500.5	500	62,624		48	49	660	48	1,331	49	49.4	349	48	48.25	388	72	1,368.51	69	783.7	65	1,721.85	66	958.1
DSJC500.9	500	$1.12 \cdot 10^6$		126			126	1,686	127	127.8	480	127	127.75	433	176	3,600	176	3,600	176	3,600	176	3,600
DSJC1000.1	1,000	49,629		20			20	2,396	21	21	90	20	20.25	260	30	1,032.55	30	3,600	26	2,682.38	26	2,013.54
DSJC1000.5	1,000	$2.5 \cdot 10^5$		83	89	1,148	88	2,028	88	89	4,658	84	84.25	8,407	124	3,600	124	3,600	124	3,600	124	3,600
DSJC1000.9	1,000	$4.49 \cdot 10^5$		224			224	3,326	228	229.6	1,565	225	226	3,234	318	3,600	318	3,600	318	3,600	318	3,600
DSJR500.1	500	3,555	12	12	12	0			12	12	0	12	12	25	12	16.25	12	8.34	12	66.43	12	33.99
DSJR500.1c	500	$1.21 \cdot 10^5$		85	85	5	85	736	85	85	6	85	85	88	107	3,600	103	2,366.91	107	3,600	95	2,703.06
DSJR500.5	500	58,862	122	122	123	14	126	1,409	122	123.4	276	122	122	163	125	2,729.63	128	1,440.24	127	3,545.39	126	1,855.03
le450_15a	450	8,168	15	15	15	0			15	15	1	15	15	0	18	47.86	18	24.18	18	119.75	18	76.34
le450_15b	450	8,169	15	15	15	0			15	15	1	15	15	0	17	3,600	17	3,600	17	120.08	16	58.96
le450_15c	450	16,680	15	15	15	5	15	6	15	15.2	11	15	15	3	26	74.15	25	40.58	24	251.45	25	130.83
le450_15d	450	16,750	15	15	15	3	15	44	15	15	5	15	15	4	27	78.45	27	3,600	25	254.95	26	131.65
le450_25c	450	17,343	25	25			26	1	26	26	7	25	25	1,321	30	94.31	30	50.18	29	319.64	30	162.16
le450_25d	450	17,425	25	25			26	1	26	26.4	1	25	25	436	30	3,600	30	48.09	28	310.25	29	154.7
r250.1	250	867	8	8	8	0			8	8	0	8	8	26	8	0.91	8	0.44	8	3.17	8	1.7
r250.1c	250	30,227	64	64	64	0			64	64	2	64	64	21	67	191.97	67	103.77	67	401.03	65	212.73
r250.5	250	14,849	65	65	65	7			65	65.8	16	65	65	64	67	91.83	68	3,600	68	204.42	67	106.4
r1000.1	1,000	14,378	20	20	20	1			20	20	0	20	20	37	20	113.8	20	58.51	20	2,021.99	20	1,004.68
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	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	251	3,600	251	3,600
latin_square_10	900	$3.07 \cdot 10^5$		98	98	415			99	100.2	938	101	102	5,156	213	3,600	213	3,600	213	3,600	213	3,600
flat300_20_0	300	21,375	20	20	20	0			20	20	2	20	20	21	46	96.89	44	56.49	43	205.43	42	118
flat300_26_0	300	21,633	26	26	26	1			26	26	1	26	26	36	46	96.67	44	57.59	42	211.27	41	121.14
flat300_28_0	300	21,695	28	28	31	156	29	867	31	31	133	31	31	212	45	97.08	46	57.64	43	212.38	44	122.38
flatl000_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	125	3,600	125	3,600
flatl000_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	123	3,600	123	3,600
flatl000_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	125	3,600	125	3,600
Average gap											1.78			0.71		44.13		42.66		38.19		38.48
Average run-time											386.79			1,020.41		1,469.27		1,468.22		1,454.01		1,258.85
Maximum runtime											4,658			8,407		3,600		3,600		3,600		3,600

5 Results and Discussion

The present paper presents preliminary research conducted in order to design an efficient constructive matheuristic (CMH) strategy for the vertex colouring problem. 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 high quality solutions. 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. The major challenge, however, is the very high impact of such features on algorithmic runtime. Previous research has shown that identifying smarter decomposition strategies and introducing components that better navigate the CMH may overcome this drawback. Therefore, future research will explore decomposition strategies for CMH and study their suitability when implemented alongside CMH design parameters.

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

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 AM 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*, 46:64–73, 2014.