INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

I F O R S International Federation of Operational Research Societies

INTERNATIONAL TRANSACTIONS IN OPERATIONAL RESEARCH

Intl. Trans. in Op. Res. 17 (2010) 1–34
DOI: 10.1111/j.1475-3995.2009.00696.x

A survey on vertex coloring problems

Enrico Malaguti and Paolo Toth

Dipartimento di Elettronica, Informatica e Sistemistica, University of Bologna Viale Risorgimento, Bologna, Italy
E-mails: enrico.malaguti@unibo.it [Malaguti], paolo.toth@unibo.it [Toth]

Received 23 October 2008; received in revised form 8 January 2009; accepted 9 January 2009

Abstract

This paper surveys the most important algorithmic and computational results on the *Vertex Coloring Problem* (VCP) and its generalizations. The first part of the paper introduces the classical models for the VCP, and discusses how these models can be used and possibly strengthened to derive exact and heuristic algorithms for the problem. Computational results on the best performing algorithms proposed in the literature are reported. The second part of the paper is devoted to some generalizations of the problem, which are obtained by considering additional constraints [*Bandwidth (Multi) Coloring Problem, Bounded Vertex Coloring Problem*] or an objective function with a special structure (*Weighted Vertex Coloring Problem*). The extension of the models for the classical VCP to the considered problems and the best performing algorithms from the literature, as well as the corresponding computational results, are reported.

Keywords: vertex coloring; bandwidth coloring; bounded vertex coloring; weighted vertex coloring; mathematical formulation; algorithms; computational results

1. Introduction

Given an undirected graph $G = (V, E)$, the *Vertex Coloring Problem* (VCP) requires to assign a color to each vertex in such a way that colors on adjacent vertices are different and the number of colors used is minimized. This paper surveys recent algorithmic and computational contributions to the exact and heuristic solution of the VCP and its generalizations.

The VCP is known to be NP-hard (see Garey and Johnson, 1979), and has received much attention in the literature, not only for its real world applications in many engineering fields, including, among many others, scheduling (see Leighton, 1979), timetabling (see de Werra, 1985), register allocation (see Chow and Hennessy, 1990), train platforming (see Caprara et al., 2007), frequency assignment (see Gamst, 1986) and communication networks (see Woo et al., 2002), but also for its theoretical aspects and for its difficulty from the computational point of view. In fact, exact algorithms proposed for VCP are able to solve consistently only small randomly generated instances, with up to 80 vertices. On the other hand, real world applications commonly deal with graphs of hundreds or thousands of vertices, for which the use of heuristic and metaheuristic techniques is necessary.

Some definitions will be useful in the following. Let n and m be the cardinalities of vertex set V and edge set E, respectively. A subset of V is called an *independent set* if no two adjacent vertices belong to it (note that, in the VCP, a subset of vertices having the same color corresponds to an independent set, and vice versa). A *clique* of a graph G is a complete subgraph of G (note that the size of a clique represents a valid lower bound for the VCP). An independent set (resp. clique) is *maximal* if no vertex can be added still having an independent set (resp. clique). A *k coloring* of G is a partition of V into k independent sets. Each independent set of a coloring is called a *color class*. An optimal coloring of G is a *k coloring* with the smallest possible value of k (the *chromatic number* $\chi(G)$ of G). The *chromatic degree* of a vertex in a given coloring is the number of different colors of its adjacent vertices. For each vertex $v \in V$, let $N(v)$ be the neighborhood of v, i.e., the set of vertices adjacent to v, and $\delta(v)$ the degree of v (i.e., $\delta(v) = |N(v)|$). We say that v *dominates* w if $N(w) \subset N(v)$.

The paper is organized as follows: in the next section we revise the best known Integer Linear Programming models for the VCP, and present exact approaches to the problem in Section 2. Section 3 is devoted to the heuristic and metaheuristic algorithms for the VCP. The following sections consider some generalizations of the VCP, which are obtained by introducing additional constraints, like in the Bandwidth (Multi)Coloring Problem (BMCP) (Section 4.1) and in the Bounded Vertex Coloring Problem (Section 4.2), or an objective function with a special structure, like in the Weighted Vertex Coloring Problem (WVCP) (Section 4.3).

1.1. Classical integer programming models for the VCP

Since n colors will always suffice for coloring graph G, a straightforward Integer Linear Programming (ILP) model for VCP can be obtained by defining the following two sets of binary variables: variables x_{ih} ($i \in V$, $h = 1, \ldots, n$), with $x_{ih} = 1$ iff vertex i is assigned to color h, and variables y_h ($h = 1, \ldots, n$), with $y_h = 1$ iff color h is used in the solution. A possible model for VCP, that we call VCP-ASS since vertices are assigned to colors, reads:

$$\text{(VCP-ASS)} \quad \min \sum_{h=1}^{n} y_h, \tag{1}$$

$$\sum_{h=1}^{n} x_{ih} = 1 \quad i \in V, \tag{2}$$

$$x_{ih} + x_{jh} \leqslant y_h \quad (i,j) \in E, h = 1, \ldots, n, \tag{3}$$

$$x_{ih} \in \{0, 1\} \quad i \in V, h = 1, \ldots, n, \tag{4}$$

$$y_h \in \{0, 1\} \quad h = 1, \ldots, n. \tag{5}$$

Objective function (1) minimizes the number of colors used. Constraints (2) require that each vertex is colored, while (3) impose that at most one of a pair of adjacent vertices receives a given color, when the color is used. Finally, (4) and (5) impose the variables to be binary. Albeit more sophisticated models can lead to better computational results when solved by means of exact or heuristic techniques, the model using binary variables x_{ih} and y_h has the advantage of clarity, and can be easily extended to VCP generalizations, as discussed in the following sections.

Model VCP-ASS has a polynomial number of variables, which can be explicitly considered when solving it. However, the model has two major drawbacks, which make it practically useless, unless it is strengthened by using sophisticated techniques (see Section 2):

- Colors are indistinguishable in the model, which thus suffers from symmetry. Suppose that a solution uses k of the n available colors. There are $\binom{n}{k}$ possible combinations of k colors. Once the k colors have been chosen, they can be permutated in $k!$ ways. Thus, given a solution of value k, there exist $\binom{n}{k}k!$ equivalent solutions.
- The continuous relaxation of the model, obtained by (1)–(5) by replacing (4) and (5), respectively, with

$$x_{ih} \geqslant 0 \quad i \in V, h = 1, \ldots, n, \tag{6}$$

$$y_h \geqslant 0 \quad h = 1, \ldots, n \tag{7}$$

is extremely weak: a feasible solution of value 2, valid for any graph, is simply $x_{i1} = x_{i2} = 1/2$, $i \in V$; $y_1 = y_2 = 1$; $x_{ih} = 0$, $i \in V$, $h > 2$; $y_h = 0$, $h > 2$.

An alternative model for the VCP, based on the so-called *Set Covering* formulation (VCP-SC), was proposed by Mehrotra and Trick (1996). Let S be the family of all the independent sets of G. Each independent set (column) $s \in S$ has associated a binary variable x_s having value 1 iff the vertices of s receive the same color. VCP can be formulated through the following model:

$$(\text{VCP-SC}) \quad \min \sum_{s \in S} x_s, \tag{8}$$

$$\sum_{s \in S : i \in s} x_s \geqslant 1 \quad i \in V, \tag{9}$$

$$x_s \in \{0, 1\} \quad s \in S. \tag{10}$$

The objective function (8) minimizes the total number of independent sets (and hence of colors) used. Constraints (9) state that every vertex i in the graph must belong to at least one independent set (i.e., must receive at least one color). Indeed, if a vertex i is assigned to more than one independent set in a feasible solution, it can be removed from all the independent sets but one (in other words if a vertex is assigned more than one color, a feasible solution of the same value can be obtained by using any one of these colors for the vertex). Thus, we can define S as the family of all the *maximal* independent sets of G. Finally, constraints (10) impose variables x_s to be binary. The VCP-SC model is not affected by the symmetry of the VCP-ASS model, since it does not assign a specific color to the vertices, but simply defines which vertices receive the same color. The lower bound provided by the continuous relaxation of VCP-SC, obtained by (8)–(10) by replacing (10) with

$$x_s \geqslant 0 \quad s \in S \tag{11}$$

is at least as good as the one provided by the continuous relaxation of VCP-ASS (see Mehrotra, 1992)

1.2. Benchmark instances

The DIMACS benchmark collects a large set of instances, which represent the standard set for experimenting algorithms for the VCP (see Johnson and Trick, 1996, all instances are available at ftp://dimacs.rutgers.edu/pub/challenge/graph/). The benchmark set includes: random graphs (DSJC), where for each pair of vertices $i, j \in V$, edge $(i,j) \in E$ is created with uniform probability; geometric random graphs (DSJR and r), where vertices are randomly distributed in a unit square, and an edge $(i,j) \in E$ is created if the distance between i and j is less than a given threshold; "quasi-random" graphs (flat); artificial graphs (le, Queen, myciel, Insertion, FullIns, latin_square_10); register allocation graphs (musol, zeroin, inithx, fpsol); graphs from real world applications (see the web page http://mat.gsia.cmu.edu/COLOR03/ for a complete description of the instances).

To allow a meaningful – although approximate – comparison on results obtained with different machines, a benchmark program (dfmax), together with a benchmark instance (r500.5), are available at http://mat.gsia.cmu.edu/COLOR03/. Computing times obtained on different machines can be scaled with respect to the performance obtained on this program.

2. Exact approaches to the VCP

This section reviews the most important exact approaches proposed for the VCP. Despite the relevance of the problem, the number of exact approaches is small with respect to the variety of heuristic algorithms proposed.

The first implicit enumeration algorithm for the VCP was proposed by Brown (1972), and is based on the idea of coloring the graph one vertex at the time, by using colors already assigned to vertices of the graph or a new color. The idea was further developed by Brélaz (1979) in the DSATUR algorithm. This algorithm works by subdividing the problem into subproblems. Each subproblem corresponds to a partial coloring of the graph. The algorithm is based on the observation that, when this partial coloring uses k colors, and $k \geqslant UB$, where UB is an upper bound on the needed number of colors, the corresponding subproblem can be fathomed. When all the vertices are colored and $k < UB$, the upper bound is updated (by setting $UB = k$). From each subproblem using k colors, up to $k+1$ new subproblems are generated, by assigning, when feasible, one of the k colors to the next vertex to be colored, and by coloring it with color $k+1$ (if $k+1 < UB$). Brélaz suggests to choose, as next vertex to color, the one having the largest chromatic degree. This rule produces an initial partial coloring corresponding to a clique, whose size is used as a lower bound during the computation. Ties are broken by choosing the vertex with highest degree in the uncolored subgraph. These rules work well within a Branch and Bound framework, since they reduce the number of subproblems which are generated. Brélaz reports computational results on randomly generated graphs.

Improvements to the DSATUR algorithm are suggested by Sewell (1996), and consist in pre-coloring a maximum clique in graph G, and in computing an initial upper bound through heuristic algorithms. In addition, the author proposes an alternative tie-breaking rule, aimed at reducing the number of generated subproblems: the vertex that produces the largest decrease in the number of colors available for the remaining uncolored vertices is chosen. According to the computational experiments reported in the paper, the improvements proposed by the author are effective, and the improved DSATUR algorithm can solve a larger set of instances than the original version. Random graphs with up to 70 vertices are consistently solved. The alternative tie-breaking rule is effective in reducing the number of generated subproblems, but is computationally expensive, and does not improve the overall performance of the algorithm.

Mehrotra and Trick (1996) proposed a Branch and Price algorithm based on the VCP-SC formulation. Since the formulation has exponentially many variables, when solving its continuous relaxation only a subset of variables (columns) are generated explicitly, by means of column generation techniques. The detection of possible negative reduced cost columns is performed by solving, at each iteration, a *Weighted Independent Set Problem* (WISP) which is known to be NP-hard (see Garey and Johnson, 1979) and can be modeled as

$$\text{(WISP)} \quad \max \sum_{i \in V} \pi_i z_i, \tag{12}$$

$$z_i + z_j \leqslant 1 \quad (i,j) \in E, \tag{13}$$

$$z_i \in \{0,1\} \quad i \in V, \tag{14}$$

where, for each $i \in V$, z_i is a binary variable taking value 1 iff vertex i is included in the independent set, and π_i is the optimal value of the dual variable associated with the corresponding constraint (9). To generate a column having negative reduced cost, WISP is heuristically solved. If no negative reduced cost column is found, an exact algorithm is executed, to generate the optimal column to be added to the VCP-SC formulation or to prove the optimality of the current solution.

When the solution to the continuous relaxation of the VCP-SC formulation is fractional, the authors choose a branching rule that ensures that the subproblem to be solved at each search node is itself a VCP without additional constraints, and the only operation to perform is the modification of the input graph G. Consider a fractional solution, say x_s, to the continuous relaxation of the VCP-SC. It is always possible to find two independent sets s_1 and s_2, and two vertices i and j such that $i \in s_1 \cap s_2$ and $j \in s_1 \backslash s_2$, with at least one between x_{s_1} and x_{s_2} fractional. Then, two subproblems are generated:

1. by imposing that i and j have the same color, which is equivalent to collapsing the two vertices in graph G;
2. by imposing that i and j have different colors, which is equivalent to adding edge (i,j) to set E.

Thus, at each vertex of the search tree the problem to solve is again the continuous relaxation of a VCP, on a modified graph where two vertices are collapsed or an edge has been added. The paper compares the proposed approach with an implementation of the DSATUR algorithm including the improvements suggested by Sewell (1996). Computational results show that both algorithms can consistently solve instances with up to 70 vertices for random graphs and 250 vertices for random geometric graphs.

Hansen et al. (2005) propose a *Set Packing* formulation (VCP-SP), which is obtained from a *Set Partitioning* formulation (i.e., the VCP-SC formulation where the inequality constraints (9) are replaced with the corresponding equality constraints) after a simple transformation. The formulation reads:

$$\text{(VCP-SP)} \quad \max \sum_{s \in \Omega} (|s| - 1)x_s, \tag{15}$$

$$\sum_{s \in \Omega : i \in s} x_s \leqslant 1 \quad i \in V, \tag{16}$$

$$x_s \in \{0, 1\} \quad s \in \Omega, \tag{17}$$

where $\Omega = \{s \in S : |s| \geqslant 2\}$. If z is the value of a solution to the Set Packing model, the corresponding number of colors is $k = n - z$. The authors prove the equivalence of the continuous relaxations of the VCP-SP and VCP-SC formulations in terms of solution value, and characterize several families of facet defining inequalities: inequalities in the initial formulations and further inequalities. Necessary and sufficient conditions for additional classes of facets are also given.

An interesting result, which explains why the VCP-SC formulation yields strong continuous relaxations, is that all constraints (9) induce facets under very general conditions:

Proposition 1. Hansen et al. (2005) *Let i be a vertex of V. Then the corresponding inequality* (9) *is facet defining if and only if i is not dominated.*

In order to experimentally compare the VCP-SC and VCP-SP formulations, the authors implemented, following Mehrotra and Trick (1996), two exact algorithms for solving the corresponding continuous relaxations. Computational experiments on a set of random, geometric random and queen graphs, however, do not show the superiority of one formulation over the other. The paper is concluded with two preprocessing techniques used to reduce the size of the graphs without changing their chromatic number, and a cutting plane procedure which is embedded in the exact algorithm for the VCP-SC formulation, where the cuts are $\left(0, \frac{1}{2}\right)$-Chvátal-Gomory cuts (see Caprara and Fischetti, 1996). Computational experiments show that, albeit the cut generation reduces the number of generated search nodes, the overall computing time is not reduced.

Two recent papers by Méndez-Díaz and Zabala (2006, 2008) tackle the VCP-ASS formulation, trying to overcome its major drawbacks, namely the weakness of the continuous relaxation of the model and the existence of many symmetric solutions. Méndez-Díaz and Zabala (2006) add symmetry breaking inequalities to model (1)–(5), ensuring that color $h+1$ can be assigned to a vertex only if color h is assigned to some other vertex:

$$y_h \geqslant y_{h+1} \quad h = 1, \ldots, n-1. \tag{18}$$

They study the properties of the polytope corresponding to the new model, and introduce several families of valid inequalities, namely the *Independent Set inequalities*, *Clique inequalities*, *Hole inequalities*, *Neighborhood inequalities*, *Block Color inequalities*, *Multicolor Path inequalities* and *Multicolor Clique inequalities*. These inequalities are embedded in a Branch and Cut algorithm, and are used to strengthen the continuous relaxation of the model at the nodes of the search tree. The algorithm includes preprocessing procedures, which remove vertices having the property that, when they are eliminated from the graph, any coloring of the resulting graph can be extended to a coloring of the original graph without using new colors. An initial upper bound is obtained by the best feasible solution value found by the DSATUR algorithm, executed with a short time limit. Instead of using the classical 0–1 branching rule on the x_{ih} variables, the authors implement the branching rule proposed by Brélaz (1979), as well as its strategy for choosing the next vertex to color. By combining two tie-breaking strategies and four possible orders of exploration for the search nodes, they obtain height different configurations, which are experimentally compared. The authors report the results of their computational experiments on randomly generated graphs, which can be consistently solved up to 80 vertices, as well as on the DIMACS instances.

Méndez-Díaz and Zabala (2008) propose further improvements to the model, in order to remove the symmetry that arises from color indistinguishability. In addition to (18), they define two additional sets of constraints, thus obtaining three alternative models:

- model (1)–(5) with constraints (18),
- model (1)–(5) with constraints imposing that the cardinality of color class h must be not smaller than the cardinality of color class $h+1$:

$$\sum_{i\in V} x_{ih} \geqslant \sum_{i\in V} x_{i,h+1} \quad h = 1,\ldots,n-1; \tag{19}$$

- a model where the independent sets are sorted by the minimum label of the vertices they include, and only colorings assigning color h to the hth set are considered. This model completely removes the symmetry due to color indistinguishability, and is obtained by imposing the following additional constraints to (1)–(5):

$$x_{ih} = 0 \quad h \geqslant i+1, \tag{20}$$

$$x_{ih} \leqslant \sum_{k=h-1}^{i-1} x_{k,h-1} \quad i \in V\backslash\{1\},\ 2 \leqslant h \leqslant i-1. \tag{21}$$

Three lower bounding procedures are then obtained by combining each model with the inequalities introduced in Méndez-Díaz and Zabala (2006), and by solving the corresponding continuous relaxation. The three lower bounds are computationally compared on random and DIMACS instances.

Campêlo et al. (2004), independent of Méndez-Díaz and Zabala (2008), propose a formulation of the VCP, called the *Representatives Formulation*, which is based on the similar idea of defining a representative vertex for each color class, and imposing that a color can be used only if the color class is initialized by the corresponding vertex. In addition, the authors present a polyhedral study of the proposed formulation. In a recent paper, Campêlo et al. (2008) revisit this formulation, and remove the symmetry arising from the fact that each vertex of a color class can be representative of the class. They extend the previous polyhedral study to the new formulation, presenting new versions of the inequalities proposed in the previous study. The same idea of using a representative vertex for each color class, was independently proposed by Malaguti et al. (2008b) for the WVCP (see Section 4.3).

We conclude this section with the paper by Desrosiers et al. (2008), who propose several procedures for finding *critical subgraphs*. A graph $G=(V,E)$ is vertex critical if $\chi(H)<\chi(G)$ for every induced subgraph H obtained by removing any vertex from V. A *k vertex critical subgraph* (k-VCS) of G is a vertex critical subgraph G' such that $\chi(G')=k$. The authors focus on the computation of k-VCS, once k is given. When k is the value of a feasible coloring of G, finding a k-VCS of G proves the optimality of the coloring. The paper reports computational experiments on random and DIMACS instances, and proves that some previously known upper bounds are actually the chromatic numbers (i.e., the optimal solution values) of the corresponding graphs.

2.1. Computational results

We report in this section the computational results obtained by the best performing exact algorithms on a subset of the DIMACS benchmark instances. The reported values are from the corresponding papers. Many papers also consider randomly generated graphs. Since a direct comparison on different instances is not possible, we do not report detailed results of such instances, and refer the reader to the relevant papers.

Table 1 summarizes the computational results reported in Mehrotra and Trick (1996) and in the more recent papers by Méndez-Díaz and Zabala (2006, 2008). When comparing these results, it must be noted that 10 years have passed, and not only the computers but also the Linear Programming solvers have largely improved. The first four columns report the instance name, the number of nodes and edges, and the chromatic number of the graph (χ), when known. The following two columns report the results obtained by Mehrotra and Trick (1996). For the considered instances, solved with a DEC ALPHA 3000 model 300 workstation with a time limit of 1 h, the authors report the computing time (expressed in seconds) and the number of nodes of their algorithm, called LPCOLOR (columns 5 and 6) and of their implementation of DSATUR, which includes the improvements proposed by Sewell (1996) (columns seven and eight). The corresponding code DSATUR is available at http://mat.gsia.cmu.edu/COLOR/solvers/trick.c. When an algorithm fails, a line appears in the table. Albeit LPCOLOR requires on average larger computing times than DSATUR, the number of explored nodes is much smaller, and it solves two instances which are not solved by DSATUR, but cannot solve one instance solved by the latter.

Then the table considers the results of the Branch and Cut algorithm BC-Col proposed by Méndez-Díaz and Zabala (2006), obtained on a Sun ULTRA 1 workstation at 140 MHz and 288 MB RAM. According to Dongarra (2006), we estimate this computer to be from two to three times faster than that used in Mehrotra and Trick (1996). The authors, before starting the algorithm, preprocess the graph in order to reduce its size, and compute a maximal clique by means of a greedy algorithm, preassigning a color to each vertex of the clique. Then they run code DSATUR by Mehrotra and Trick (1996) for 5 seconds, in order to obtain an upper bound. They consider this phase an initialization of the algorithm, which is then run with a time limit of 2 h. The ninth and tenth columns of Table 1 report the best lower and upper bounds achieved within the 2 h, the 11th column reports the corresponding gap, and the 12th column reports the computing time (expressed in seconds). An *init* entry in the 12th column means that the instance was solved to optimality during the initialization phase of the algorithm. The following four columns report the same information for code DSATUR in the tests performed by the authors, in order to asses the performance of their algorithm. When one of the two algorithms can solve an instance while the other algorithm fails, the authors do not report the lower and upper bounds corresponding to the algorithm which failed, and thus we simply report that the gap is positive (>0). Note that for DSATUR the lower bound at the root node corresponds to the cardinality of the maximal clique computed during the initialization phase, while, when the instance is solved, the lower bound equals the upper bound. Comparing BC-Col and DSATUR, the former can solve eight instances not solved by the latter, and cannot solve one instance solved by the latter. The advantage of BC-Col is that, when an instance is not solved to optimality, the corresponding lower bound always dominates that of DSATUR. On the other hand, the solution values found by DSATUR, when the instance is not solved within the time limit, are always better than or

Table 1
Exact approaches

Instance	n	m	χ	LPCOLOR Mehrotra and Trick (1996)		DSATUR Mehrotra and Trick (1996)		BC-Col Méndez-Díaz and Zabala (2006)				DSATUR Méndez-Díaz and Zabala (2006)				$\mathscr{CP}$ Méndez-Díaz and Zabala (2008)	
				Time	Nodes	Time	Nodes	LB	UB	%Gap	Time	LB	UB	%Gap	Time	LB	Time
DSJC125.1	125	736	5					5	5	0	0.9	5	5	0	0.1	5	1
DSJC125.5	125	3891	?					13	20	35	7200	9	19	52.63	7200	12	77
DSJC125.9	125	6961	?					42	47	10.6	7200	29	45	35.5	7200	42	354
DSJC250.1	250	3218	?					5	9	44.4	7200	4	9	55.5	7200	5	11
DSJC250.5	250	15668	?					13	36	63.8	7200	9	35	74.2	7200	14	3339
DSJC250.9	250	27897	?					48	88	45.4	7200	34	87	60.9	7200	48	3605
DSJC500.1	500	12458	?					5	15	66.6	7200	5	15	66.6	7200		
DSJC500.5	500	62624	?					13	63	79.3	7200	9	63	85.7	7200	13	538
DSJC500.9	500	1124367	?					59	161	63.3	7200	43	160	73.1	7200	59	5870
DSJC1000.1	1000	49629	?					6	26	76.9	7200	5	25	80	7200		
DSJC1000.5	1000	249826	?					15	116	87	7200	10	114	91.22	7200		
DSJC1000.9	1000	449449	?					65	301	78.4	7200	53	300	82.3	7200	66	4546
DSJR500.1	500	3555	12					12	12	0	init	12	12	0	init		
DSJR500.1c	500	121275	?					78	88	11.3	7200	70	88	20.4	7200	80	4470
DSJR500.5	500	58862	122					119	130	8.4	7200	103	130	20.7	7200	119	1211
latin_sq._10	900	307350	?					90	129	30.2	7200	90	129	30.2	7200		
le450_5a	450	5714	5					5	9	44.4	7200	5	9	44.4	7200		
le450_5b	450	5734	5					5	9	44.4	7200	5	9	44.4	7200		
le450_5c	450	9803	5					5	5	0	init	5	5	0	init		
le450_5d	450	9757	5					5	10	50	7200	5	8	37.5	7200		
le450_15a	450	8168	15					15	17	11.7	7200	15	17	11.7	7200		
le450_15b	450	8169	15					15	17	11.7	7200	15	16	6.2	7200		
le450_15c	450	16680	15					15	24	37.5	7200	13	23	43.4	7200		
le450_15d	450	16750	15					15	23	34.7	7200	13	23	43.4	7200		
le450_25a	450	8260	25					25	25	0	init	25	25	0	init		
le450_25b	450	8263	25					25	25	0	init	25	25	0	init		
le450_25c	450	17343	25					25	28	10.7	7200	20	28	28.5	7200		
le450_25d	450	17425	25					25	28	10.7	7200	21	27	22.2	7200		
queen5_5	25	160	5	0	1	0	21										
queen6_6	36	290	7	1	4	0	865										
queen7_7	49	476	7	4	1	0	6849										
queen8_8	64	728	9	19	18	–	–	9	9	0	3	9	9	0	18		
queen8_12	96	1368	12	79	42	0	164	12	12	0	init	12	12	0	init		
queen9_9	81	1056	10	515	77	–	–	9	11	18.1	7200	9	10	10	7200		
queen10_10	100	2940	11					10	12	16.6	7200	10	12	16.6	7200		

Table 1. (Contd.)

Instance	n	m	χ	LPCOLOR Mehrotra and Trick (1996)		DSATUR Mehrotra and Trick (1996)		BC-Col Méndez-Díaz and Zabala (2006)				DSATUR Méndez-Díaz and Zabala (2006)				$\mathcal{CP}$ Méndez-Díaz and Zabala (2008)	
				Time	Nodes	Time	Nodes	LB	UB	%Gap	Time	LB	UB	%Gap	Time	LB	Time
queen11_11	121	3960	11					11	14	21.4	7200	11	13	15.38	7200		
queen12_12	144	5192	12					12	15	20	7200	12	14	14.2	7200		
queen13_13	169	6656	13					13	16	18.7	7200	13	15	13.3	7200		
queen14_14	196	8372	14					14	17	17.6	7200	14	17	17.6	7200		
queen15_15	225	10360	15					15	18	16.6	7200	15	18	16.6	7200		
queen16_16	256	12640	16					16	20	20	7200	16	19	15.7	7200		
myciel2	5	5	3	0	1	0	4										
myciel3	11	23	4	0	9	0	27										
myciel4	20	71	5	9	303	0	848										
myciel5	47	236	6	–	–	18	378311										
myciel6	95	755	7					5	7	28.5	7200	2	7	71.4	7200		
myciel7	191	2360	8					5	8	37.5	7200	2	8	75	7200		
1-Insertions_4	67	232	5					5	5	0	2			>0	7200	3	2
1-Insertions_5	202	1227	?					4	6	33.3	7200	2	6	66.6	7200	3	0
1-Insertions_6	607	6337	?					4	7	42.8	7200	2	7	71.4	7200	3	3
2-Insertions_4	149	541	?					4	5	20	7200	2	5	60	7200	3	0
2-Insertions_5	597	3936	?					3	6	50	7200	2	6	66.6	7200	3	3
3-Insertions_3	56	110	4					4	4	0	1	4	4	0	5	3	0
3-Insertions_4	281	1046	?					3	5	40	7200	2	5	60	7200	3	0
3-Insertions_5	1406	9695	?					3	6	50	7200	2	6	66.6	7200	3	61
4-Insertions_3	79	156	4					3	4	25	7200	2	4	50	7200	3	0
4-Insertions_4	475	1795	?					3	5	40	7200	2	5	60	7200	3	2
1-FullIns_4	93	593	5					5	5	0	0.1			>0	7200	4	0
1-FullIns_5	282	3247	6					4	6	33.3	7200	3	6	50	7200	4	0
2-FullIns_3	52	201	5					5	5	0	0.1	5	5	0	1014	5	0
2-FullIns_4	212	1621	6					5	6	16.6	7200	4	6	50	7200	6	4
2-FullIns_5	852	12201	7					5	7	28.5	7200	4	7	42.8	7200		
3-FullIns_3	80	346	6					6	6	0	0.1			>0	7200	6	0
3-FullIns_4	405	3524	7					6	7	14.2	7200	5	7	28.5	7200	6	4
3-FullIns_5	2030	33751	8					6	8	25	7200	5	8	37.5	7200	6	292
4-FullIns_3	114	541	7					7	7	0	3			>0	7200	7	0
4-FullIns_4	690	6650	8					7	8	12.5	7200	6	8	25	7200	7	16
4-FullIns_5	4146	77305	?					6	9	33.3	7200	6	9	33.3	7200		
5-FullIns_3	154	792	8					8	8	0	20			>0	7200		
5-FullIns_4	1085	11395	?					8	9	11.1	7200	7	9	22.2	7200	8	55
wap01	2368	110871	?					41	46	10.8	7200	39	48	18.7	7200		
wap02	2464	111742	?					40	45	11.1	7200	39	46	15.21	7200		

wap03	4730	286722	?					40	56	28.5	7200	40	55	27.27	7200		
wap04	5231	294902	?					40	50	20	7200	20	48	58.3	7200		
wap05	905	43081	?					50	51	1.9	7200	27	51	47	7200		
wap06	947	43571	?					40	44	9	7200	33	45	26.6	7200		
wap07	1809	103368	?					40	46	13	7200	23	46	50	7200		
wap08	1870	104176	?					40	47	14.8	7200	23	45	48.8	7200		
qg_order30	900	26100	30					30	30	0	init	30	30	0	init		
qg_order40	1600	62400	40					40	42	4.7	7200	40	42	4.7	7200		
qg_order60	3600	212400	60					60	63	4.7	7200	60	63	4.7	7200		
fpsol2_i_1	496	11654	65	10	1	0	811	65	65	0	0.6	65	65	0	0.1	65	8
fpsol2_i_2	451	8691	30	9	1	2	615	30	30	0	1.2	30	30	0	0.1	30	1
fpsol2_i_3	425	8688	30	16	1	2	591	30	30	0	1.1	30	30	0	0.1	30	1
mug88_1	88	146	4					4	4	0	11			>0	7200		
mug88_25	88	146	4					4	4	0	184	4	4	0	4756		
mug100_1	100	166	4					4	4	0	60			>0	7200		
mug100_25	100	166	4					4	4	0	60			>0	7200		
ash331GPIA	662	4185	4					4	4	0	51	4	4	0	0.7	4	46
ash608GPIA	1216	7844	4					4	4	0	692	4	4	0	3	4	692
ash958GPIA	1916	12506	?					4	5	20	7200	3	5	40	7200	4	4236
abb313GPIA	1557	46546	?					8	10	20	7200	6	10	40	7200		
will199GPIA	701	6772	7							>0	7200	7	7	0	1.2		
inithx.i.1	864	18707	54	31	1	1	432	54	54	0	init	54	54	0	init		
inithx.i.2	645	13979	31	19	1	1	422	31	31	0	init	31	31	0	init		
inithx.i.3	621	13969	31	14	1	1	396	31	31	0	init	31	31	0	init		
mulsol.i.1	197	3925	49	2	1	0	149	49	49	0	init	49	49	0	init		
mulsol.i.2	188	3885	31	1	1	0	158	31	31	0	init	31	31	0	init		
mulsol.i.3	184	3916	31	3	1	0	154	31	31	0	init	31	31	0	init		
mulsol.i.4	185	3946	31	1	1	0	155	31	31	0	init	31	31	0	init		
mulsol.i.5	185	3973	31	1	1	0	156	31	31	0	init	31	31	0	init		
school1	385	19095	14					14	14	0	init	14	14	0	init		
school1_nsh	352	14612	14					14	14	0	init	14	14	0	init		
zeroin.i.1	211	4100	49	2	1	0	163	49	49	0	init	49	49	0	init		
zeroin.i.2	211	3541	30	2	1	0	182	30	30	0	init	30	30	0	init		
zeroin.i.3	206	3540	30	2	1	0	177	30	30	0	init	30	30	0	init		
anna	138	493	11	1	0	0	128	11	11	0	init	11	11	0	init		
david	87	406	11	1	0	0	77	11	11	0	init	11	11	0	init		
homer	561	1629	13	1	24	1	549	13	13	0	init	13	13	0	init		
huck	74	301	11	1	0	0	64	11	11	0	init	11	11	0	init		
jean	80	254	10	1	0	0	71	10	10	0	init	10	10	0	init		
games120	120	638	9	1	1	0	112	9	9	0	init	9	9	0	init		
miles250	128	387	8	2	1	0	121	8	8	0	init	8	8	0	init		
miles500	128	1170	20	1	1	0	109	20	20	0	init	20	20	0	init		
miles750	128	2113	31	1	3	0	98	31	31	0	init	31	31	0	init		
miles1000	128	3216	42	0	1	0	87	42	42	0	0.02	42	42	0	0.1	42	0
miles1500	128	5198	73	1	1	0	56	73	73	0	0.1	73	73	0	0.1	73	0

equal to those found by BC-Col, with the exception of three wap instances. Comparing LPCOLOR by Mehrotra and Trick (1996) and BC-Col by Méndez-Díaz and Zabala (2006) on the common instances, the former can solve one instance (queen9.9) not solved by the latter, although it has larger computing time on the instances solved by both algorithms.

Finally, the last two columns report the results obtained by Méndez-Díaz and Zabala (2008), concerning the three lower bounding procedures described in the previous section. We report the results obtained by the best procedure, namely that corresponding to the VCP-ASS formulation with inequalities (18), called *CP*. We consider only the subset of instances which were not solved during the initialization phase (greedy computation of a maximal clique and execution of DSATUR for 5 s), and for which procedure *CP* can improve the initial lower bound (represented by the cardinality of the maximal clique). We report the final lower bound achieved by the procedure and the corresponding computing time (expressed in seconds on the previously mentioned machine). For four instances (DSJC250.5, DSJC1000.9, DSJR500.1c and 2-FullIns_4), this lower bound improves that found by BC-Col at the end of the computation.

3. Heuristic approaches to the VCP

In this section we focus on the most effective and original contributions among the hundreds of algorithms that have been proposed in the literature for the heuristic solution of the VCP.

3.1. Greedy algorithms

The first heuristic approaches to VCP were based on greedy constructive algorithms. These algorithms sequentially color the vertices of the graph following some rule for choosing the next vertex to color and the color to use. They are generally very fast, but the results they produce can be very sensitive to some input parameter, like the input ordering of the vertices.

The simplest approach is the greedy sequential algorithm (generally called SEQ): assume that the vertices are labelled $v_1, \ldots, v_n$. Vertex v_1 is assigned to the first color class, and thereafter vertex v_i $(i = 2, \ldots, n)$ is assigned to the lowest indexed color class that contains no vertices adjacent to v_i.

The greedy version of the DSATUR algorithm by Brélaz (1979) is obtained by branching on the first admissible color, until the whole graph is colored: initially the graph is uncolored, at each iteration the first vertex having the current maximum chromatic number is chosen, and it is colored with the first available color.

Leighton (1979) proposed the *Recursive Largest First* (RLF) algorithm, which colors the vertices, one class at a time, in the following greedy way. Let C be the next color class (initially empty) to be constructed, V' the set of uncolored vertices (initially containing all the currently uncolored vertices) that can be placed in C, and U the set (initially empty) of uncolored vertices that cannot be placed in C.

- Choose the first vertex $v_0 \in V'$ that has the *maximum* number of adjacent vertices in V'. Place v_0 in C and move all the vertices $u \in V'$ that are adjacent to v_0 from V' to U.

- While V' remains nonempty, do the following: choose the first vertex $v \in V'$ that has the maximum number of adjacent vertices in U; add v to C and move all the vertices $u \in V'$ that are adjacent to v from V' to U.

An experimental study by Johnson et al. (1991) analyzed the dependence of the value of the solutions found by these algorithms on the input ordering of the vertices, showing that the values tend to have a normal distribution when the input ordering of the vertices is random. Concerning the average solution values, RLF is the best among the three algorithms, and SEQ the worst.

Bollobàs and Thomason (1985) proposed an algorithm that recursively selects the maximum independent set from the set of uncolored vertices, by means of an exponential backtrack algorithm. The vertices of the selected independent set receive a new color, and the process is iterated until the whole graph is colored. The algorithm performs well on dense graphs, while for sparse graphs, where the independent sets are large, the exponential algorithm can be very time consuming.

Culberson and Luo (1996) proposed the *Iterated Greedy Algorithm*, which iteratively colors the graph by means of the SEQ algorithm, in such a way that at each iteration the number of used colors is not increased. The algorithm starts with an initial feasible coloring. At each successive iteration, the ordering of the vertices is such that the vertices of each independent set identified in the previous coloring are consecutive. This ensures that the new coloring will use no additional color. Given a coloring, the vertices in the same color class have to be consecutive in the input of the next iteration, but the order of the color classes is changed according to different criteria (e.g., vertices which received the color 2 can precede those that received color 1, and so on), and this may lead to a reduction of the number of colors used.

3.1.1. Metaheuristic methods

This section reviews the best performing algorithms for the VCP based on local search. On the same topic, we also refer the reader to the specific surveys by Galinier and Hertz (2006) and by Chiarandini et al. (2007). Starting from an initial solution, one may improve it by means of a local search method, which iteratively moves from the current solution to an improved one. Local search requires the definition of three elements: (i) a set of candidate solutions (search space), which can correspond to partial colorings or complete colorings; (ii) a neighborhood relation; (iii) a solution evaluating function. Concerning the number of colors to be used (generally denoted as k), this can be either fixed or variable. In the first case the algorithm solves the decision version of the problem, while in the second case it solves the optimization version. Following Chiarandini et al. (2007), local search algorithms for the VCP can be partitioned in three families: (i) fixed k, complete colorings; (ii) fixed k, partial colorings; (iii) variable k, complete colorings.

One of the first metaheuristic algorithms proposed for the VCP was the Tabu Search procedure by Hertz and de Werra (1987). The algorithm, called TABUCOL, considers a fixed number k of available colors, and moves among complete colorings. A solution is thus represented by a k coloring of the graph, where some edges are conflicting, i.e., both endpoints share the same color. A move consists in changing the color of one vertex, and the evaluation function measures the number of conflicts in the current coloring. The algorithm considers only moves involving critical vertices, i.e., vertices whose color is currently in conflict with some adjacent vertex. A tabu list stores the assignment of colors to vertices, and forbids this assignment to be replicated for a specified number of iterations, the so-called *tabu tenure*. TABUCOL is a well-known algorithm which was successfully improved and embedded as a procedure in more complex algorithms (see, e.g., Dorne and Hao, 1998; Galinier and Hao, 1999).

Simulated annealing was tested by Johnson et al. (1991), who compared three different neighborhoods, with k variable and complete colorings. One neighborhood uses only feasible colorings, and the others allow infeasible colorings. In addition, they propose an algorithm, called XRLF, integrating the independent set extraction with an improved version of RLF. When the number of vertices to color is reduced under a given threshold, the residual graph is colored with an exact algorithm. The method is competitive with the simulated annealing approach.

A pure genetic algorithm was proposed by Davis (1998), encoding a solution as a permutation of the vertices, which are then colored through the SEQ algorithm. However, pure genetic methods do not produce satisfactory results for the VCP, since they are not able to capture the specificity of the problem.

Morgenstern (1996) defined the *Impasse Class Neighborhood*, a structure used to improve a partial *k coloring* to a complete coloring of the same value, thus, according to the adopted classification, a method based on fixed k, partial coloring. The method requires a target value k for the number of colors to be used. A solution S is a partition of V in $k+1$ color classes $\{V_1, \ldots, V_k, V_{k+1}\}$ in which all classes, but possibly the last one, are independent sets. This means that the first k classes constitute a partial feasible *k coloring*, while all vertices that do not fit in the first k classes are in the last one. Making this last class empty gives a complete feasible *k coloring*. To move from a solution S to a new solution S' in the neighborhood, one can choose an uncolored vertex $v \in V_{k+1}$, assign v to a different color class, say h, and move to class $k+1$ all vertices v' in class h that are adjacent to v. This ensures that color class h remains feasible. The uncolored vertex v is randomly chosen, while the class h is chosen by comparing different target classes by means of an evaluating function $f(S)$. Rather than simply minimizing $|V_{k+1}|$, Morgenstern proposed to minimize the global degree of the uncolored vertices:

$$f(S) = \sum_{v \in V_{k+1}} \delta(v). \tag{22}$$

This choice forces vertices having small degree, which are easier to color, to enter class $k+1$. Morgenstern used this idea, together with a procedure for the recombination of the solutions, to define a simulated annealing algorithm, which obtains very good results, compared with the algorithms previously proposed. The definition of the *Impasse Class Neighborhood* was a fundamental contribution to the local search methods for the VCP: some of the best performing methods later proposed are based on this neighborhood.

The development of genetic algorithms led to the integration of local search procedures with population based algorithms, in the so-called *Evolutionary Algorithms*. Fleurent and Ferland (1996) proposed an algorithm based on the evolution of a population of solutions (like in genetic techniques), with the mutation operator substituted by a local search operator, which in this case is TABUCOL. The crossover operator they use is a standard uniform crossover, which does not exploit the specificity of the VCP.

Galinier and Hao (1999) proposed an evolutionary algorithm (called HCA: Hybrid Coloring Algorithm), working with fixed k, and combining an improved version of TABUCOL with a crossover operator which is specialized for the VCP, thus obtaining one of the best performing algorithms for the problem. The crossover, called *Greedy Partitioning Crossover*, works as follows: given two solutions, which represent the parents (partitions of the vertices in k sets, not necessarily independent), the operator alternatively considers each parent to generate the next color class of the offspring. The color class of maximum cardinality of the considered parent becomes the next color class of the offspring; all the vertices in this color class are deleted from the parents. When k steps are performed, some vertices may remain unassigned. Each of these vertices is then assigned to a randomly chosen class. The power of this operator resides in its capacity to transmit to the offspring the structure of the parents, namely the partition into sets.

Funabiki and Higashino (2000) proposed an effective algorithm for the problem, called MIPS-CLR (MInimal-state Processing Search algorithm for the graph CoLoRing problem), which combines a Tabu Search technique, based on the *Impasse Class Neighborhood*, with different heuristic procedures, color fixing and solution recombination in an attempt to expand a feasible partial coloring to a complete coloring. The algorithm works with variable k.

Variable Neighborhood Search (VNS) is a technique that combines local search with different neighborhoods: when the algorithm is trapped in a local optimum, the neighborhood is changed (which possibly makes the current solution not locally optimal, according to the new neighborhood), and the search can continue. An algorithm for the VCP based on VNS was proposed by Avanthay et al. (2003).

In population-based algorithms, the pool of solutions evolving during the computation can be interpreted as a central memory, where the information collected during the search is stored. In *adaptive memory algorithms*, the central memory stores pieces of solutions instead of complete ones. Galinier et al. (2008) proposed an algorithm, called AMACOL, working with fixed k, where independent sets corresponding to the solutions generated during the computation are stored. They use a recombination operator to generate new solutions from these independent sets: a sample of independent sets is selected from the pool and, iteratively, the color classes are built by choosing the independent set from the sample which includes the largest number of uncolored vertices. Then, the new solutions evolve through an improved version of TABUCOL.

Two Tabu Search algorithms, working with fixed k and based on the *Impasse Class Neighborhood*, called DYN-PARTIALCOL and FOO-PARTIALCOL, were recently proposed by Blöchliger and Zufferey (2008): in this work the next vertex to color is not chosen randomly in the set of the uncolored vertices, but selected so that it, entering the best color class, produces the best solution in the neighborhood. This approach increases the size of the neighborhood reducing at the same time the randomness introduced in the search. Thus, to avoid premature convergence, the authors use an evaluating function that simply minimizes $|V_{k+1}|$. In addition, the authors compare the use of a dynamic tabu tenure (DYN-PARTIALCOL), which depends on the last solution visited by the algorithm, and a *reactive tabu tenure* (FOO-PARTIALCOL), based on the fluctuation of the objective function. If the value of the objective function is almost constant during a long period, the tabu tenure is increased in an attempt to escape the region where the search process has been trapped. In order to balance this increase, the tenure is constantly reduced during the search process. In this way the search process alternates between intensification (when the tenure is low) and diversification (when the tenure is high).

Recently, Malaguti et al. (2008a) proposed an evolutionary algorithm combining a Tabu Search procedure, based on the *Impasse Class Neighborhood*, and a crossover operator, which is an adaptation of the *Greedy Partitioning Crossover* to the neighborhood they use. The evolutionary algorithm, which solves the problem for a fixed value of k, is then embedded in an overall algorithm, called MMT, which solves the optimization version of the problem. The algorithm is initialized by generating a pool of solutions through the greedy heuristics SEQ, DSATUR and RLF. The best value found by the greedy heuristics defines a valid upper bound (initial value of k). Then, the evolutionary algorithm tries to improve on this upper bound, by solving the decision version of the VCP for decreasing values of k. The independent sets corresponding to the feasible colorings obtained during the computation are stored in a pool of columns. The evolutionary algorithm is stopped when the problem is solved to optimality (certified by means of a clique-based lower bound) or when a time limit is reached. The final phase of the algorithm is a post optimization procedure based on the VCP-SC formulation. The Set Covering instance corresponding to the columns stored in the pool is heuristically solved by means of a heuristic algorithm from the literature (the CFT algorithm by Caprara et al., 1999), thus possibly improving the best incumbent solution. MMT is one of the best heuristic algorithms for the VCP, and, in contrast to many of its competitors, it does not require the set-up of a large set of parameters. In fact, only the time limit of the algorithm needs to be chosen, since the other parameters are dynamically adjusted during the computation.

We conclude this review with a recent paper by Hertz et al. (2008), who propose an extension of the VNS called *Variable Search Space* (VSP). In VNS, the neighborhood is changed when the search is trapped in a local optimum. The idea of VSP is to completely change the search space (i.e., the space of the solutions which are explored during the computation) and consider different objective functions for each space. They propose an algorithm working with fixed k, called VSS-Col, which moves between three different search spaces, obtaining excellent results on a limited set of benchmark instances.

3.2. Computational results

In this section we report the computational results obtained by the best performing metaheuristic algorithms proposed for the VCP. It must be pointed out that it is very difficult to fairly compare the performance of different algorithms with a single table. On one hand, some algorithms work in decision version (i.e., for fixed k) and require the fine tuning of a large set of parameters, which change for each instance: Morgenstern (1996) observed that reporting computing times for algorithms that require the value of k and other parameters as input does not reflect the real effort required to find the corresponding values, and this hidden cost can increase substantially the actual computing time. On the other hand, some algorithms solve the problem in optimization version (which is clearly much more difficult than finding a coloring when k is given), and the authors use a single set-up of the parameters for all the instances they consider. In the following, we report solution values and computing times obtained by the considered algorithms. While the solution value is absolute, when considering the computing times the reader should carefully take into account the above.

All the computing times reported in this section are expressed in seconds of a Pentium IV 2.4 GHz with 512 MB RAM, which spent 7 s user time to solve the benchmark instance mentioned in Section 1.2.

In Table 2 we report the results obtained by the best performing algorithms for the subset of the DIMACS instances commonly used for testing heuristic approaches, which includes all the most difficult instances. For each instance, the first part of the table reports the instance name, the number of nodes, the number of edges and the value of the best known solution found in the literature (in bold when it is the proven optimal value).

In the following four parts we report the performance of the algorithms solving the decision version of the VCP. All these algorithms need the target value k for the coloring as an input parameter.

The second part of the table reports the results obtained by the *Impasse* algorithm by Morgenstern (1996), which requires some input parameters that are tuned for every instance. For each considered instance we report the smallest value of k for which no failure occurred over 5 runs and the average computing time to solve the instance, scaled with respect to the benchmark instance time. In addition to the values for which all 5 runs were successful, we report in brackets the best solution values for which at least one solution was found during the computational study performed to tune the parameters of the algorithm. The algorithm does not use an explicit time limit, instead, it is stopped when the solution is not improved for a specified number of iterations.

The third part of the table considers the HCA by Galinier and Hao (1999). HCA requires as input some parameters that are tuned for every instance. For each considered instance we report the smallest value of k for which there was at least one successful run over 10 (or 5) runs (with a limit on the number of iterations of the algorithm which varies with the considered instance), the corresponding ratio between the number of successful runs and the number of runs (succ.), and an approximate average computing time for the successful runs (we divided the reported time, obtained on an UltraSPARC-IIi 333 MHz with 128 MB RAM, for which the authors do not report the performance on the benchmark instance, by 6, following the performance ratio reported by Dongarra (2006), in order to approximately obtain the time on a PIV 2.4 GHz computer). We report as well the best solution values found on a set of instances for which the computing times and the number of successful runs are not given (*NA* in the table).

The fourth part of the table reports the results obtained by Blöchliger and Zufferey (2008) with DYN-PARTIALCOL and FOO-PARTIALCOL, with a time limit of 1 h for each run (the computer used is equivalent to the reference one). For each considered instance we report the smallest value of k for which there was at least one successful run over 50 runs of each of the two algorithms, the corresponding ratio between the number of successful runs and the number of runs (succ.), and the average computing time for the successful runs of the best algorithm (considering as "best" the algorithm which finds the best solution value, breaking ties by considering first the number of successful runs and then the computing time). In brackets, we report some improved results obtained with a time limit of 10 h.

The fifth part of the table reports the results obtained by Hertz et al. (2008) with the VSS-Col algorithm, with a time limit of 1 h (the computer used is equivalent to the reference one). The algorithm uses a common set-up of the parameters for all the instances. For each considered instance we report the smallest value of k for which there was at least one successful run over 10 runs, the corresponding ratio between the number of successful runs and the number of runs (succ.), and the average computing time for the successful runs. In brackets, we report some improved results obtained with a time limit of 10 h.

Table 2
Performance of the most effective heuristics for the VCP

Instance	n	m	Best	Impesse Morgenstern (1996)		HCA Galinier and Hao (1999)			DYN/FOO-P.COL Blöchliger and Zufferey (2008)			VSS-Col Hertz et al. (2008)			MIPS_CLR Funabiki and Higashino (2000)			MMT Malaguti et al. (2008a)		
				k	Time	Succ.	k	Time	succ.	k	Time	Succ	k	Time	Best$k$	Avg$k$	Time	Best$k$	Avg$k$	Time
DSJC125.1	125	736	5												5	5.0	0	5	5.00	21
DSJC125.5	125	3891	17	17	1										17	17.0	1	17	17.00	122
DSJC125.9	125	6961	44												44	44.0	0	44	44.00	121
DSJC250.1	250	3218	8												8	8.0	5	8	8.00	21
DSJC250.5	250	15668	28	28	22	9/10	28	13							28	28.4	14	28	28.00	117
DSJC250.9	250	27897	72												72	72.4	31	72	72.00	89
DSJC500.1	500	12458	12	(12)					50/50	12	193	10/10	12	97	12	12.4	84	12	12.25	210
DSJC500.5	500	62624	48	(48)49	660	5/10	48	268	1/50	(48) 49	811	3/10	48	1331	49	49.4	349	48	48.25	388
DSJC500.9	500	1124367	126	(127)					1/50	127	1680	8/10	126	1686	127	127.8	480	127	127.75	433
DSJC1000.1	1000	49629	20			NA	20	NA	3/50	20	2301	3/10	20	2396	21	21.0	90	20	20.25	260
DSJC1000.5	1000	249826	83	(83)89	1148	NA	83	2258	6/50	89	2786	8/10	(86) 88	2028	88	89.0	4658	(83) 84	84.25	8407
DSJC1000.9	1000	449449	224	(226)		NA	224	NA	30/50	(226) 228	2275	1/10	224	3326	228	229.6	1565	225	226.00	3234
DSJR500.1	500	3555	**12**	**12**	0										**12**	**12.0**	0	**12**	**12.00**	25
DSJR500.1C	500	121275	85	85	5				50/50	85	438	9/10	85	736	85	85.0	6	85	85.00	88
DSJR500.5	500	58862	**122**	123	14				28/50	(125) 126	1544	9/10	(125) 126	1409	**122**	123.4	276	**122**	**122.00**	163
le450_15a	450	8168	**15**	**15**	0										**15**	**15.0**	1	**15**	**15.00**	0
le450_15b	450	8169	**15**	**15**	0										**15**	**15.0**	1	**15**	**15.00**	0
le450_15c	450	16680	**15**	**15**	5	6/10	**15**	8	50/50	**15**	1	10/10	**15**	6	**15**	15.2	11	**15**	**15.00**	3
le450_15d	450	16750	**15**	**15**	3				50/50	**15**	3	10/10	**15**	44	**15**	**15.0**	5	**15**	**15.00**	4
le450_25c	450	17343	**25**	**(25)**		10/10	26	55	50/50	**(25)** 27	4	10/10	**(25)** 26	1	26	26.0	7	**25**	**25.00**	1321
le450_25d	450	17425	**25**	**(25)**					50/50	**(25)** 27	3	10/10	**(25)** 26	1	26	26.4	1	**25**	**25.00**	436
r250.1	250	867	**8**	**8**	0										**8**	**8.0**	0	**8**	**8.00**	26
r250.1c	250	30227	**64**	**64**	0				50/50	**64**	4				**64**	**64.0**	2	**64**	**64.00**	21
r250.5	250	14849	**65**	**65**	7				6/50	66	2400				**65**	65.8	16	**65**	**65.00**	64
r1000.1	1000	14378	**20**	**20**	1										**20**	**20.0**	0	**20**	**20.00**	37
r1000.1c	1000	485090	98	98	46				20/50	99	1895				98	98.8	557	98	**98.00**	518
r1000.5	1000	238267	**234**	241	77				10/50	(248) 249	2098				237	238.6	1345	**234**	**234.00**	753
latin_square_10	900	307350	98	98	415										99	100.2	938	101	**102.00**	5156
flat300_20_0	300	21375	**20**	**20**	0				10/10	**20**	0				**20**	**20.0**	2	**20**	**20.00**	21
flat300_26_0	300	21633	**26**	**26**	1				10/10	**26**	0				**26**	**26.0**	1	**26**	**26.00**	36
flat300_28_0	300	21695	**28**	31	156	6/10	31	20	35/50	**28**	1905	1/10	**(28)** 29	867	31	31.0	133	31	**31.00**	212
flat1000_50_0	1000	245000	**50**	**50**	0				50/50	**50**	26	10/10	**50**	318	**50**	**50.0**	14	**50**	**50.00**	1417
flat1000_60_0	1000	245830	**60**	**60**	0				50/50	**60**	91	10/10	**60**	694	**60**	**60.0**	59	**60**	**60.00**	3645
flat1000_76_0	1000	246708	82	89	897	4/5	83	1471	10/50	(87) 88	2332	6/10	(85) 87	1689	87	87.8	2499	(82) 83	83.50	7325

The last two parts of the table consider the algorithms solving the optimization version of the problem, which do not require the number k of colors as input parameter.

The sixth part of the table considers the MIPS-CLR algorithm by Funabiki and Higashino (2000). This algorithm works in optimization version but requires as input an initial target value k_{init} for the coloring, together with some other parameters whose values are tuned for the most difficult instances. The algorithm dynamically increases this target value when the number of uncolored vertices of the solution is not reduced for a given number of iterations. We report the best value (best$k$) and the average value (avg$k$) of k found over 5 runs, and the average scaled computing time.

The last part of the table reports the results obtained by Malaguti et al. (2008a) with the MMT algorithm, whose performance is summarized reporting the best value (best$k$) and the average value (avg$k$) of k over 4 runs, and the average computing time up to the time limit (set to a value depending on the instance size, and varying from 20 to 3000 s) or to optimality, which can be proved for a subset of instances through a lower bound based on the computation of a maximal clique. The MMT algorithm does not take advantage of any external information (such as the expected value of the coloring), and the set-up of the parameters is dynamically performed by the algorithm during the computation. The time limit is the only parameter to be specified as input.

From the table, it clearly appears that, as far as the solution value is concerned, the best algorithms are MMT and VSS-Col. On the common set of instances, MMT finds four better solutions and one worse than Impasse, two better solutions and one worse than HCA (on a very limited set of instances), seven better solutions and one worse than DYN/FOO-PARTIALCOL, eight better solutions and one worse than MIPS-CLR (15 better solutions and one worse by considering the average values). In addition, MMT is the only algorithm which can solve to optimality instance r1000.5, and the only one which can color instance flat1000_76_0 with 82 colors.

On the common set of instances, VSS-Col finds four better solutions and two worse than Impasse, two better solutions and two worse than HCA (on a very limited set of instances), four better solutions than DYN/FOO-PARTIALCOL, nine better solutions and one worse than MIPS-CLR. In addition, VSS-Col is the only algorithm which can color instance DSJC500.9 with 126 colors.

When comparing directly MMT and VSS-Col, there is no clear winner, since, on the common set of instances, MMT finds three (five if the time limit of 1 h is considered for VSS-Col) better solutions than VSS-Col, and vice versa. However, observing that the computing times of the algorithms are comparable, it must be noticed that MMT is solving the problem in optimization version, without any external input on the number k of colors, while VSS-Col is solving the problem for a specified value of k (namely the smallest value of k for which the algorithm does not fail), and thus the reported computing time does not reflect the global effort needed to determine this value.

4. Generalizations of the VCP

In this section we survey recent results on some generalizations on the VCP which have received attention in the literature. All the problems mentioned in this section are NP-hard, since they generalize the VCP.

4.1. BMCP

In the *Bandwidth Coloring Problem* (BCP) "distance" constraints are imposed between adjacent vertices, replacing the "diversity" constraints. A distance $d(i,j)$ is defined for each edge $(i,j) \in E$, and the absolute value of the difference between the colors assigned to i and j must be at least equal to this distance: $|c(i) - c(j)| \geqslant d(i,j)$, where, for each vertex $i \in V$, $c(i)$ is the color assigned to i. The problem asks for minimizing the largest color assigned to the vertices. The problem generalizes the VCP, arising when $d(i,j) = 1$ for each edge $(i,j) \in E$.

In the *Multicoloring Problem* (MCP) a positive weight w_i is defined for each vertex $i \in V$, representing the number of colors that must be assigned to vertex i, so that for each edge $(i,j) \in E$ the intersection of the color sets assigned to vertices i and j is empty. The problem generalizes the VCP, arising when $w_i = 1$ for each vertex $i \in V$.

The BMCP is the combination of the two problems above. Each vertex i must be assigned w_i colors, and each of these colors must respect the distance $d(i,j)$ with all the colors assigned to any adjacent vertex j. In this case, loop $d(i,i)$ represents the minimum distance between different colors assigned to the same vertex i. The problem generalizes both the BCP and the MCP, arising, respectively, when $w(i) = 1$ for each vertex $i \in V$, and $d(i,j) = 1$ for each edge $(i,j) \in E$.

Let $\bar{h}$ be an upper bound on the number of colors needed to color graph $G = (V, E)$, and define $H = \{1, \ldots, \bar{h}\}$ as the set of usable colors. A possible model for BMCP, using the same binary variables x_{ih} and y_h used in model VCP-ASS (see Section 1.1), and a continuous variable k, reads:

$$\text{(BMCP-ASS)} \quad \min k, \tag{23}$$

$$k \geqslant y_h h \quad h \in H, \tag{24}$$

$$\sum_{h \in H} x_{ih} = w_i \quad i \in V, \tag{25}$$

$$x_{ih} + x_{jl} \leqslant 1 \quad (i,j) \in E,\ h \in H,\ l \in \{h - d(i,j) + 1, \ldots, h + d(i,j) - 1\}, \tag{26}$$

$$x_{ih} \leqslant y_h \quad i \in V,\ h \in H, \tag{27}$$

$$x_{ih} \in \{0,1\} \quad i \in V,\ h \in H, \tag{28}$$

$$y_h \in \{0,1\} \quad h \in H. \tag{29}$$

The objective function (23) [in conjunction with constraints (24)] minimizes the maximum color used. Note that in BCP and BMCP the number of colors assigned to the vertices can be smaller than the maximum color used. Constraints (25) impose that each vertex $i \in V$ receives the specified number of colors. Constraints (26) state that, for each edge $(i,j) \in E$, the absolute value of the difference between the colors assigned to vertices i and j must be at least equal to $d(i,j)$. Constraints (27) ensure that if a vertex i uses a color h, then color h results as used.

BCP, MCP and BMCP allow more complex situations than VCP to be modeled, in particular in the domain of the *Frequency Assignment Problem* (FAP), concerning the allocation of frequencies to transmitters (see the survey by Aardal et al., 2003) for an overview on FAPs). The BMCP models the minimization of the radio spectrum span for the frequencies assigned to a set of transmitters, with constraints on the largest acceptable mutual interference. Vertices correspond to transmitters which have to receive a given number of frequencies, and distances correspond to the minimum distance between frequencies that can be re-used by adjacent (i.e., possibly interfering) transmitters. For models, exact and heuristic approaches to other specific FAPs, we refer the reader to Aardal et al. (2003). Most of the papers considered in that survey describe algorithms designed to solve instances with a special structure, representing real cases arising in telecommunications.

The Computational Symposium on Graph Coloring and its Generalizations (see Trick, 2002) renewed the interest in computational approaches for BMCP, and a set of random instances specifically designed for the problem was proposed. At the symposium, Phan and Skiena (2002) proposed to solve VCP and BCP by means of a general heuristic, called *Discropt* (designed for "black box optimization"), adapted to the specific coloring problems. During the same symposium, Prestwich (2002) proposed a combination of local search and constraint propagation in a method called FCNS to solve generalized graph coloring problems. In a subsequent work, Prestwich (2008) proposed a hybrid heuristic algorithm for BCP and BMCP combining a local search operator, based on a variation of the *impasse* neighborhood, with the constraint programming technique of forward checking, in order to prune the coloring neighborhood during the search. Lim et al. (2003) proposed a method for solving VCP, BCP, MCP and BMCP combining iterated greedy techniques and Squeaky Wheel Optimization, a general heuristic approach for optimization, originally proposed by Joslin and Clements (1999). The algorithm starts from a greedy coloring, computed as a sequential coloring (each vertex, considered in the given input ordering, receives the first available color). A penalty is assigned to each vertex whose color exceeds a given target, and the order of the vertices is modified according to the penalties associated with the vertices. The greedy algorithm is re-executed, thus obtaining a new coloring. Lim et al. (2005) studied the performance of different heuristic methods, including Squeaky Wheel and Tabu Search and their hybridization, for the solution of BCP, MCP and BMCP. In a recent work, Malaguti and Toth (2008) proposed an evolutionary approach for the BCP, based on the integration of a Tabu Search algorithm with population management procedures. The Tabu Search, which moves among partial colorings, is based on an adaptation of the *impasse* neighborhood to BCP, and the paper experimentally investigates the behavior of different crossover operators, used to combine existing solutions into new ones. Note that in the BCP it is not possible to design a crossover that simply copies the independent sets of the parent solutions into the offspring, since the distance constraints impose a distance among the independent sets. The best performing crossover proposed in Malaguti and Toth (2008), called *distance_crossover*, is based on the idea that the important structure to be transferred from the parents to the offspring is the relative color distance between the vertices, in particular when the distance constraints are satisfied tightly. The crossover copies all the "tight distance" pairs (i.e., $\{(v,w) \in E: \ v < w, \ |c_1(v) - c_1(w)| = d(v,w)\}$, where, for each vertex $i \in V$, $c_1(i)$ denotes the color assigned to i in the solution corresponding to the first parent) from the first parent to the offspring, keeping the same color assignment. Then it tries to do the same with "tight distance" pairs from the second parent, keeping the same color assignment, when this does not violate the distance constraints with respect to the already colored vertices of the offspring, or changing the color assignment (but keeping the relative distance between colors) so as to produce a feasible coloring.

Although the method proposed in Malaguti and Toth (2008) is explicitly designed for BCP, it can be applied to BMCP instances as well. Indeed, in BMCP each vertex $i \in V$ must be assigned a number of colors corresponding to its weight $w(i)$. To deal with these constraints, the authors transform each instance of BMCP into the corresponding instance of BCP, as suggested for example in Lim et al. (2005). In particular, each vertex i is split into a clique of cardinality $w(i)$, with each edge of the clique having distance $d(i, i)$, corresponding to the distance of the loop edge of vertex i in the original graph. The new graph will then have $\sum_{i \in V} w_i$ vertices. This transformation introduces extra symmetry, and hence the approach is effective only when the weights of the vertices are "small" (which is the case for the instances proposed in Trick, 2002).

To the best of our knowledge, exact algorithms were proposed only for FAPs, and not for BCP and BMCP.

4.1.1. Computational results for the BCP and BMCP

In Tables 3 and 4, respectively, we report the best known results for the BCP and the BMCP instances proposed at the Computational Symposium on Graph Coloring and its Generalizations (see Trick, 2002). Computing times, reported from the respective papers, are scaled according to the time needed to solve the benchmark instance (see Section 1.2), and represent seconds of a Pentium IV 2.4 GHz with 512 MB RAM, which spent 7 s user time to solve the benchmark instance. For each instance, the first three columns of the tables report the instance name (which includes the number of vertices), the number of edges (m) and the corresponding best published solution value (*best*). All the reported computing times are expressed in seconds.

The solution values (k) and the corresponding computing times obtained by Lim et al. (2003) are reported in the fourth and fifth columns of the tables. The algorithm works in optimization version (i.e., it does not require as input the maximum color k to be used). We report the best solution value found over a single run and the scaled computing time needed to get this result.

The sixth column of Table 3 reports the solution values (k) obtained by Phan and Skiena (2002) by means of the *Discropt* general heuristic (no results are reported for BMCP). The algorithm works in optimization version. The reported results represent the best value obtained over 3 runs, using 3 different versions of the algorithm. For each instance, the total computing time was 300 s (100 s for each run), on an Athlon K7 AMD processor with 768 MB of RAM under RedHat 7.2. According to Dongarra (2006), we estimate that this computer is between 3 and 4 times slower than the Pentium IV used as reference. Thus, the 300 s approximately correspond to 75–100 s of the latter computer.

The solution values (k) and the corresponding computing times obtained by Lim et al. (2005), with an algorithm combining Squeaky Wheel Optimization with Tabu Search, are reported in the sixth and seventh columns of Table 3 in the case of BMCP (results for BCP are not competitive and were not reported in detail in Lim et al., 2005). The algorithm works in optimization version. We report, for each instance, the best solution value found over a single run with 10 restarts and the scaled computing time.

The solution values (k) and the corresponding scaled computing times obtained by Prestwich (2008) are reported in the seventh and eighth columns of Table 3 in the case of BCP, and in the eighth and ninth columns of Table 4 in the case of BMCP. The algorithm, working in optimization version, requires some input parameters, whose values are tuned, among a limited set of possibilities, for each instance.

Table 3
Performance of the algorithms for the Bandwidth Coloring Problem

Instance	m	Best	Lim et al. (2003)		Phan and Skiena (2002)	Prestwich (2008)		Malaguti and Toth (2008)	
			k	Time	k	k	Time	k	Time
GEOM20	40	20	21	0	20	21	0	21	0
GEOM20a	57	20	22	0	20	20	0	20	0
GEOM20b	52	13	14	0	13	13	0	13	0
GEOM30	80	27	29	0	27	28	0	28	0
GEOM30a	111	27	32	0	27	27	0	27	0
GEOM30b	111	26	26	0	26	26	0	26	0
GEOM40	118	27	28	1	27	28	0	28	0
GEOM40a	186	37	38	1	38	37	0	37	0
GEOM40b	197	33	34	1	36	33	0	33	0
GEOM50	177	28	28	1	29	28	0	28	0
GEOM50a	288	50	52	1	54	50	2	50	0
GEOM50b	299	35	38	2	40	35	0	35	0
GEOM60	245	33	34	0	34	33	0	33	0
GEOM60a	339	50	53	2	54	50	1	50	0
GEOM60b	426	41	46	1	47	43	0	41	29
GEOM70	337	38	38	0	40	38	0	38	0
GEOM70a	529	61	63	0	64	62	2	61	12
GEOM70b	558	48	54	0	54	48	1	48	52
GEOM80	429	41	42	2	44	41	0	41	0
GEOM80a	692	63	66	0	69	63	12	63	150
GEOM80b	743	60	65	8	70	61	0	60	145
GEOM90	531	46	46	0	48	46	3	46	0
GEOM90a	879	63	69	2	74	64	2	63	150
GEOM90b	950	70	77	6	83	72	2	70	1031
GEOM100	647	50	51	9	55	50	0	50	2
GEOM100a	1092	68	76	6	84	68	9	68	273
GEOM100b	1150	73	83	2	87	73	15	73	597
GEOM110	748	50	53	1	59	50	4	50	3
GEOM110a	1317	72	82	11	88	73	7	72	171
GEOM110b	1366	78	88	5	87	79	2	78	676
GEOM120	893	59	62	0	67	60	4	59	0
GEOM120a	1554	84	92	1	101	84	4	84	614
GEOM120b	1611	84	98	1	103	86	9	84	857
avg. ratio (k/best)			1.0710		1.0972	1.0096		1.0038	
avg. time				2	~ 75–100		2.5		144

The last two columns of Tables 3 and 4 report the results obtained by Malaguti and Toth (2008) for BCP and BMCP, respectively. Their algorithm works in decision version. The authors start by computing an initial upper bound UB, obtained by adapting the sequential algorithm SEQ (Section 3) to the BCP, and set $k = UB - 1$. Then, they solve the problem for decreasing values of k with a time limit, for each k, of 500 s for BCP and 3000 s for BMCP. The last columns of the tables report the solution value ("k") and the total computing time (including the initial UB computation) corresponding to the last improvement.

Table 4
Performance of the algorithms for the Bandwidth Multicoloring Problem

Instance	m	Best	Lim et al. (2003)		Lim et al. (2005)		Prestwich (2008)		Malaguti and Toth (2008)	
			k	Time	k	Time	k	Time	k	Time
GEOM20	40	149	149	0	149	17	149	4	149	18
GEOM20a	57	169	169	4	169	16	170	2	169	9
GEOM20b	52	44	44	0	44	2	44	0	44	5
GEOM30	80	160	160	0	160	23	160	0	160	1
GEOM30a	111	209	211	3	209	40	214	11	210	954
GEOM30b	111	77	77	0	77	7	77	0	77	0
GEOM40	118	167	167	1	167	47	167	1	167	20
GEOM40a	186	213	214	99	213	54	217	299	214	393
GEOM40b	197	74	76	2	74	10	74	4	74	1
GEOM50	177	224	224	11	224	77	224	1	224	1197
GEOM50a	288	316	326	27	318	12	323	51	316	4675
GEOM50b	299	83	87	15	87	15	86	1	83	197
GEOM60	245	258	258	13	258	96	258	77	258	139
GEOM60a	339	357	368	1037	358	162	373	10	357	8706
GEOM60b	426	115	119	83	116	23	116	12	115	460
GEOM70	337	272	279	7	273	138	277	641	272	1413
GEOM70a	529	469	478	115	469	188	482	315	473	988
GEOM70b	558	117	124	38	121	30	119	55	117	897
GEOM80	429	383	394	1118	383	204	398	361	388	132
GEOM80a	692	363	379	187	379	190	380	109	363	8583
GEOM80b	743	141	145	894	141	39	141	37	141	1856
GEOM90	531	332	335	1133	332	248	339	44	332	4160
GEOM90a	879	377	382	2879	377	245	382	13	382	5334
GEOM90b	950	144	157	179	157	46	147	303	144	1750
GEOM100	647	404	413	175	404	311	424	7	410	3283
GEOM100a	1092	444	462	48	459	334	461	26	444	12526
GEOM100b	1150	156	172	1354	170	55	159	367	156	3699
GEOM110	748	383	389	160	383	368	392	43	383	2344
GEOM110a	1317	490	501	1292	494	441	500	29	490	2318
GEOM110b	1366	206	210	3	206	68	208	5	206	480
GEOM120	893	396	409	505	402	408	417	9	396	2867
GEOM120a	1554	556	564	1476	556	633	565	41	559	3873
GEOM120b	1611	191	201	240	199	97	196	3	191	3292
avg. ratio (k/best)			1.0251		1.0129		1.0188		1.0020	
avg. time				397		141		87		2320

The last two rows of Tables 3 and 4 report, for each considered algorithm, the average ratio between the solution value and the best known solution value in the literature (*k*/*best*), and the average computing time.

Over the 33 considered instances of BCP (see Table 3), the algorithm by Malaguti and Toth (2008) clearly outperforms its competitors from the viewpoint of the solution value, but with a much larger computing time (144 s on average). This algorithm is able to find the best solution in

30 cases, followed by the algorithm proposed by Prestwich (2008), which finds the best solution in 21 cases, with a computing time of only 2.5 s on average. Also by considering the average ratio between the solution value and the best known solution value, the two algorithms outperform the others by an order of magnitude. In addition, it can be noted that the algorithm proposed in Malaguti and Toth (2008) never obtains solution values worse than those found in Lim et al. (2003) and Prestwich (2008). As for the algorithm proposed in Phan and Skiena (2002), it has a very good behavior for the smallest instances ($n<40$), and is able to determine in three cases a solution value better than that found in Malaguti and Toth (2008), although with a much larger computing time.

As for the BMCP instances (see Table 4), the algorithms by Lim et al. (2005) and Malaguti and Toth (2008) dominate the other algorithms from the viewpoint of the solution value. Over the 33 considered instances, the algorithm by Lim et al. (2005) finds the best solution in 20 cases, while the algorithm by Malaguti and Toth (2008) finds the best solution in 26 cases. Also the average ratio between the solution value and the best known solution value ($k/best$) is better for the latter algorithm.

Finally, Malaguti and Toth (2008), whose algorithms are more time consuming than the competitors, observe that when shorter computing times are imposed, their algorithms are still able to find feasible solutions of good quality. They report that the hard part of the computation very often corresponds to the last improvements in the number k of colors. In other words, when their algorithms spend a long time to improve the best known solution value of only few colors, a large part of the computation is devoted to solve the problem for the last values of k.

4.2. Bounded VCP

In many practical situations, the number of users that can access to a resource is bounded, or it may happen that each user consumes a (possibly different) fraction of the resource, and the total capacity of the resource is limited. This situation can be modeled by assigning a positive weight w_i to each vertex $i \in V$, and by imposing a *capacity constraint* on the total weight of the vertices that receive the same color. The corresponding problem is known as *Bounded Vertex Coloring Problem* (BVCP) or *Bin Packing Problem with Conflicts*. The Bin Packing Problem (BPP) requires to assign a set of items, each with a positive weight, to the smallest number of bins, each having the same capacity (see, e.g., Martello and Toth, 1990). The Bin Packing Problem with Conflicts requires the additional constraint that items corresponding to vertices which are adjacent in G (called conflict graph) cannot be assigned to the same bin.

If C represents the capacity of each color, we can impose the capacity constraints as follows:

$$\sum_{i=1}^{n} w_i x_{ih} \leqslant C, \quad h = 1, \ldots, n. \tag{30}$$

Model VCP-ASS with the capacity constraints, i.e., (1)–(5) and (30), define the BVCP. When all the weights of the vertices are equal to 1, constraints (30) determine the maximum cardinality of each color class.

Some BVCP real-world applications include examination scheduling (see Laporte and Desroches, 1984), assignment of processes to processors and load balancing of tasks in parallel computing (see Jansen, 1999). Other applications concern particular delivery problems, such as food distribution, where some items cannot be placed in the same vehicle (see Christofides et al., 1979).

The BVCP was addressed by Jansen and Oehring (1997) and Jansen (1999), who derived polynomial time approximation schemes for special classes of conflict graphs. Hansen et al. (1993) considered a special case of the BPPC, where all the weights are equal to 1, and proposed upper and lower bounds, as well as complexity results. Bodlaender and Jansen (1993) considered the complexity of the previous problem for different conflict graph classes.

An extensive computational work on the BVCP was presented by Gendreau et al. (2004), who provided lower bounds and greedy algorithms, and introduced benchmark instances. The authors started by adapting to the BVCP some fast greedy heuristics for the VCP (including those presented in Section 3.1), heuristic algorithms for the BPP and heuristic algorithms based on clique computations. Then they present more sophisticated and time consuming greedy algorithms, obtained by combining the previous approaches. The authors propose the so-called *constrained packing* lower bound L_{CP}. The bounding procedure starts by computing a maximal clique K of graph G through a greedy algorithm. Then a color is initialized for each vertex in the clique, and the items in $V\backslash K$ are assigned to these colors (possibly in a fractional way), by solving a transportation problem that takes into consideration both the weights and the conflicts implied by the edges of E. All the remaining vertices (or fractions) than cannot fit in the $|K|$ initialized colors are stored in a new set V_1, on which a lower bound based on the continuous relaxation of the corresponding Bin Packing instance is computed. The authors propose as well a set of instances derived from Bin Packing by adding random conflict graphs.

Fernandes Muritiba et al. (2008) describe improvements of the greedy algorithms and of the lower bounds proposed in Gendreau et al. (2004), and new lower bounds as well. In addition, they propose a metaheuristic algorithm for the problem, which integrates a Tabu Search algorithm with a crossover operator. The Tabu Search algorithm is based on an adaptation of the *Impasse* neighborhood, described in Section 3.1.1, to the BVCP. In particular, when a *move* makes a color class feasible with respect to the conflict constraints, it may remain infeasible with respect to the capacity constraints (30): the authors propose alternative moves to keep the feasibility of the new solution. The crossover operator used is an adaptation of the one proposed by Galinier and Hao (1999). Finally, the authors propose a Branch and Price algorithm for the solution of the Set Covering formulation of the BVCP (which is equivalent to the VCP-SCP formulation (8)–(10), once the set of maximal independent sets satisfying the capacity constraints is considered). The Branch and Price algorithm is integrated with the metaheuristic algorithms and with the previously mentioned greedy algorithms and lower bounds. In particular, the solutions generated by the greedy and metaheuristic algorithms are stored in a pool during the computation, and used to initialize the pool of columns of the Branch and Price algorithm. The paper reports computational experiments, performed on a set of instances generated according to the description given in Gendreau et al. (2004) (the original instance set being unavailable), which show the effectiveness of the integrated metaheuristic-exact approach. The instances are available at http://www.or.deis.unibo.it/research_pages/ORinstances/BPPC.html

4.3. WVCP

The classical VCP asks for the minimization of the number of colors used. In other words, the problem asks for the minimization of the cost of the coloring, where the cost of each color is equal to one. In this section we consider a WVCP, where each vertex $i \in V$ has a positive weight w_i, and the cost of each color is the maximum weight of the vertices assigned to the color. This generalization of the objective function reduces to the classical VCP when the weight of each vertex is one.

WVCP has real world applications in different industrial fields. In the scheduling literature it is also known as *Scheduling on a Batch Machine with Job Compatibilities*, and it models applications where the simultaneous access of the jobs to a resource is limited by a conflict graph. The weights associated with the vertices can be interpreted as the processing times associated with the jobs, and the processing time of each batch corresponds to the maximum processing time of the associated jobs. A typical case, concerning thermic treatment or mechanical processing, arises when some jobs cannot be processed within the same time slot because of geometric constraints represented by the edges of the conflict graph G (see Gavranovich and Finke, 2000), and the duration of each time slot corresponds to the maximum processing time of the associated jobs.

WVCP can also model the *Matrix Decomposition Problem in Time Division Multiple Access Traffic Assignment* (MDP), a satellite transmission scheduling problem where a traffic matrix has to be transmitted through a satellite connection, and needs to be decomposed into mode matrices (i.e., matrices where no more than one non-zero element is present in each row and column), in such a way that the sum of the costs of all the mode matrices is minimized. The cost of each mode matrix equals the maximum of its non-zero elements. We can transform each instance of the MDP into a WVCP instance, by constructing the corresponding graph G as follows: each entry of the matrix whose value, say w_i, is different from zero is associated with a vertex i of G having weight w_i. Moreover, each pair of vertices associated with elements belonging to the same row or to the same column are linked through an edge in G.

On the theoretical side, de Werra et al. (2007) analyzed some properties of the optimal solutions and discussed complexity and approximability results for the WVCP; Escoffier et al. (2006) continued the investigation of the complexity and the approximability of the problem; while Boudhar and Finke (2000) studied its complexity for several classes of graphs. These results were extended to different classes of graphs by Finke et al. (2008).

4.3.1. Integer programming models for the WVCP

Malaguti et al. (2008b) propose three alternative models for the WVCP. The first one is a straightforward generalization of model VCP-ASS, defined by (1)–(5), to the weighted problem, where, for each color h ($h = 1, \ldots, n$), the binary variable y_h is substituted by the integer variable z_h, representing the cost of color h, as imposed by the following constraints:

$$z_h \geqslant w_i\, x_{ih}, \quad i \in V,\ h = 1, \ldots, n. \tag{31}$$

The second model, called WVCP-M2, tries to remove the symmetry, due to color indistinguishability, affecting the previous model (see Section 1.1). Without loss of generality, let the vertices be numbered in such a way that

$$w_1 \geqslant w_2 \geqslant \cdots \geqslant w_n. \tag{32}$$

For any solution to WVCP, an equivalent solution exists in which each independent set $s = \{v_{i_1}, v_{i_2}, \ldots, v_{i_p}\}$ having p vertices is associated with color i_1, i.e., with the color corresponding to the first vertex included in s. Hence, the model considers only feasible solutions in which each color h, if used, is *initialized* by vertex h, has cost equal to w_h and can be assigned to vertices $i \geqslant h$. This idea is analogous to the notion of representative vertex introduced by Campêlo et al. (2004), and to the symmetry breaking inequalities introduced by Méndez-Díaz and Zabala (2008), for the VCP.

Model WVCP-M2 uses binary variables x_{ih}, taking value 1 iff vertex i is assigned to color h ($i \in V$, $h \leqslant i$), with $x_{hh} = 1$ iff color h is used, and reads:

$$\text{(WVCP-M2)} \quad \min \sum_{h=1}^{n} w_h x_{hh}, \tag{33}$$

$$\sum_{h=1}^{i} x_{ih} = 1, \; i = 1, \ldots, n, \tag{34}$$

$$x_{ih} + x_{jh} \leqslant x_{hh}, \quad (i,j) \in E, \; h = 1, \ldots, \min\{i,j\}, \tag{35}$$

$$x_{ih} \leqslant x_{hh}, \quad h = 1, \ldots, n, \; i > h, \tag{36}$$

$$x_{ih} \in \{0, 1\}, \quad h = 1, \ldots, n, \; i \geqslant h. \tag{37}$$

Objective function (33) minimizes the sum of the costs of the colors used. Constraints (34) and (35) impose that each vertex receives a color, and colors of adjacent vertices are different, respectively. Constraints (36) impose that a vertex i can receive a color $h \neq i$, among those available for the vertex, only if color h is used. Constraints (37) impose the variables to be binary.

Clearly, constraints (35) and (36) can be expressed in a stronger way by using the corresponding clique constraints, as in the classical VCP (see, e.g., Méndez-Díaz and Zabala, 2006).

Finally, the last model discussed in Malaguti et al. (2008b) generalizes the Set Covering formulation VCP-SC (8)–(10) (see Section 1.1), by replacing the objective function (8) with

$$\text{(WVCP-SC)} \min \sum_{s \in S} c_s x_s, \tag{38}$$

where $c_s = \max\{w_i\colon i \in s\}$ defines the cost of each independent set $s \in S$.

4.3.2. Exact and heuristic approaches

An exact approach to the MDP, based on column generation, is proposed by Ribeiro et al. (1989). The problem is formulated as a set partitioning problem and solved using a branch and bound algorithm combined with a column generation procedure and a column ranking technique.

Prais and Ribeiro (2000) propose a heuristic approach to the MDP, based on a *Greedy Randomized Adaptive Search Procedure* (GRASP). GRASP is a metaheuristic technique based on two phases: in a construction phase, a feasible solution is built though a greedy algorithm, which constructs the solution one element at a time (in the VCP, e.g., it colors one vertex at a time). The next element to be added to the solution is determined by computing for each candidate element a score according to a greedy function that estimates the benefit of selecting the element. One of the elements in the candidate list, among those overcoming a given threshold of the score, is randomly chosen. A local search phase is then applied to improve the solution.

Malaguti et al. (2008b) propose a two-phase heuristic approach based on the Set Covering formulation of the WVCP. The algorithm is initialized by computing a lower bound on the optimal solution value through the solution of the continuous relaxation of model WVCP-M2. Then, the first phase of the algorithm is executed, aimed at producing a very large number of independent sets. In this phase the authors apply in sequence some fast greedy heuristics designed for the problem, possibly considering different parameter sets. Every time a new greedy solution is generated, it is possibly improved by means of fast post-optimization procedures. The independent sets defining the feasible solutions found by the greedy heuristics correspond to variables (columns) of the WVCP-SC formulation of the problem, and are stored in a family S', which represents a subfamily of family S that includes all the independent sets of the graph. When the optimality of the incumbent solution is not proved during the first phase, the second phase considers the SCP instance associated with the columns in S' and heuristically solves it through a Lagrangian heuristic algorithm from the literature (CFT by Caprara et al., 1999), improving many times the best incumbent solution.

4.3.3. Computational results

Two different sets of instances of the WVCP were considered in the computational papers (Ribeiro et al., 1989; Prais and Ribeiro, 2000; Malaguti et al., 2008b). The first set is composed of 35 instances from MDP, which were proposed by Ribeiro et al. (1989), and are publicly available at http://www-di.inf.puc-rio.br/~celso/grupo_de_pesquisa.htm. The second set was proposed during the Computational Symposiumon on Graph Coloring and its generalizations (see Trick, 2002), when weights were added to vertices of the original DIMACS instances. Malaguti et al. (2008b) report experiments on all the random graphs (DSJC*n*.*x*), a subset of the geometric random graphs (Geom*n*) (the smallest ones were disregarded) and all the "quasi-random" graphs (R*n*_*x*), where n denotes the number of vertices and $10x$ is the percentage density of the graph.

The results concerning the MDP instances, which were all exactly solved by Ribeiro et al. (1989) (we do not report the corresponding large computing time, since a comparison with the computer and the Linear Programming solver they used is meaningless), are given in Table 5 (note that instances p01–p05, p37 and p39 are missing from the instance set). The first three columns report the instance name, the number of vertices (n) and the number of edges (m). The following two columns of the table give the solution value (z_{PR}), in bold when it corresponds to the optimal value, found with the GRASP algorithm proposed by Prais and Ribeiro (2000) and the corresponding computing time (T_{PR}), expressed in seconds of an IBM 9672 model R34 mainframe computer. The last three columns of the table refer to the two-phase heuristic proposed by Malaguti et al. (2008b), and report the lower bound (LB), computed by solving the LP relaxation of model WVCP-M2 with CPLEX9, the solution value (z_{MMT}) and the overall computing time (T_{MMT}), expressed in seconds of a Pentium IV 2.4 GHz with 512 MB RAM and including the lower bound computation. These results were obtained with a single set-up of the parameters. By properly tuning the parameters for the two instances not solved to optimality (i.e., p13 and p42), the whole set can be optimally solved (see the last two lines of the table).

Table 5
Weighted VCP: results on the MDP instances

Instance	n	m	Z_{PR}	T_{PR}	LB	Z_{MMT}	T_{MMT}
p06	16	38	**565**	1.1	565	**565**	0.4
p07	24	92	**3771**	4.0	3771	**3771**	0.1
p08	24	92	**4049**	1.3	4049	**4049**	1.3
p09	25	100	**3388**	3.0	3388	**3388**	0.1
p10	16	32	**3983**	4.5	3983	**3983**	0.1
p11	18	48	**3380**	4.7	3380	**3380**	0.1
p12	26	90	**657**	3.8	657	**657**	0.1
p13	34	160	3230	7.8	3220	3225	2.4
p14	31	110	**3157**	10.1	3157	**3157**	0.1
p15	34	136	**341**	4.7	341	**341**	0.2
p16	34	134	**2343**	14.5	2343	**2343**	0.6
p17	37	161	**3281**	5.5	3281	**3281**	1.5
p18	35	143	**3228**	10.4	3228	**3228**	0.3
p19	36	156	**3710**	14.6	3710	**3710**	0.2
p20	37	142	1860	20.0	1830	**1830**	1.4
p21	38	155	**3660**	18.4	3660	**3660**	0.3
p22	38	154	**1912**	20.0	1912	**1912**	0.3
p23	44	204	3810	21.4	3770	**3770**	1.5
p24	34	104	**661**	27.9	661	**661**	0.2
p25	36	120	**504**	23.9	504	**504**	0.2
p26	37	131	**520**	28.3	520	**520**	0.2
p27	44	174	**216**	7.8	216	**216**	0.3
p28	44	164	**1729**	44.5	1729	**1729**	0.2
p29	53	254	**3470**	65.7	3470	**3470**	0.2
p30	60	317	**4891**	56.6	4891	**4891**	2.3
p31	47	179	**620**	70.9	620	**620**	0.2
p32	51	211	**2480**	70.9	2480	**2480**	0.1
p33	56	258	**3018**	62.3	3018	**3018**	0.3
p34	74	421	**1980**	131.9	1980	**1980**	0.4
p35	86	566	**2140**	135.0	2140	**2140**	0.6
p36	101	798	**7210**	163.1	7210	**7210**	1.0
p38	94	537	**2130**	70.5	2130	**2130**	1.0
p40	86	497	**4984**	224.2	4879	**4984**	0.7
p41	116	900	**2688**	313.7	2688	**2688**	1.4
p42	138	1186	2480	405.8	2466	2509	4.9
p13	34	160	3230	7.8	3220	**3220**	8.9
p42	138	1186	2480	405.8	2466	**2466**	101.9

The algorithm by Malaguti et al. (2008b) can find the optimal solution on 33 of 35 instances (and on the whole set by tuning the parameter set on each instance), outperforming on this set of instances the approach by Prais and Ribeiro (2000), that finds the optimal solution on 31 instances out of 35. The computing times are roughly comparable, considering that the machine used in Malaguti et al. (2008b) is from 40 to 60 times faster than that used in Prais and Ribeiro (2000) (see Dongarra, 2006). In addition, the approach proposed in Malaguti et al. (2008b) is more general, since it was not designed by considering the special structure of these instances.

Table 6
Weighted VCP: results on the weighted DIMACS instances

Instance	n	m	LB	T_{LB}	Z_{MMT}	T_{MMT}
DSJC125_1g	125	736	19	27	24	152
DSJC125_1gb	125	736	74	26	95	170
DSJC125_5g	125	3891		34	76	180
DSJC125_5gb	125	3891		35	260	182
DSJC125_9g	125	6961	152	55	173	162
DSJC125_9gb	125	6961	568	42	628	237
GEOM30b	30	111	12	6	**12**	0
GEOM40b	40	197	16	0	**16**	1
GEOM50b	50	299	18	0	**18**	0
GEOM60b	60	426	23	0	**23**	0
GEOM70	70	337	47	1	**47**	96
GEOM70a	70	529	73	0	**73**	3
GEOM70b	70	558	24	1	**24**	6
GEOM80	80	429	66	2	**66**	0
GEOM80a	80	692	76	3	**76**	102
GEOM80b	80	743	27	3	**27**	90
GEOM90	90	531	61	3	**61**	166
GEOM90a	90	879	73	4	**73**	157
GEOM90b	90	950	30	5	**30**	11
GEOM100	100	647	65	4	**65**	131
GEOM100a	100	1092	89	7	**59**	112
GEOM100b	100	1150	32	7	**32**	15
GEOM110	110	748	66	7	69	172
GEOM110a	110	1317	97	12	**97**	111
GEOM110b	110	1366	37	19	**37**	5
GEOM120	120	893	72	8	**72**	157
GEOM120a	120	1554	105	10	**105**	136
GEOM120b	120	1611	35	28	**34**	14
R50_1g	50	108	14	0	**14**	0
R50_1gb	50	108	52	0	*53	95
R50_5g	50	612	35	1	*37	167
R50_5gb	50	612	126	1	137	145
R50_9g	50	1092	73	1	*74	36
R50_9gb	50	1092	257	1	*262	33
R75_1g	70	251	17	4	19	154
R75_1gb	75	251	63	4	72	166
R75_5g	75	1407	43	6	53	172
R75_5gb	75	1407	160	6	190	173
R75_9g	75	2513	108	3	*110	79
R75_9gb	75	2513	393	3	399	50
R100_1g	100	509	18	13	22	155
R100_1gb	100	509	70	16	84	171
R100_5g	100	2456	48	28	62	179
R100_5gb	100	2456	182	32	234	179
R100_9g	100	4438	138	11	142	123
R100_9gb	100	4438	499	11	520	127

In Table 6 we report the results obtained by the algorithm proposed by Malaguti et al. (2008b) on the second set of instances, derived from the DIMACS set. In addition to the names and dimensions of the instances, the table reports the lower bound LB, computed by solving the LP relaxation of model WVCP-M2 with CPLEX9, the corresponding computing time (T_{LB}), the solution value (z_{MMT}), in bold when it corresponds to the proven optimal value, and the overall computing time of the algorithm (T_{MMT}). For the solution of the LP relaxation of model WVCP-M2, a time limit of 30 s was imposed for each instance. For two instances (DSJC125_5 g and DSJC125_5gb) this time limit was reached. By solving model WVCP-M2 with a very large time limit, the authors were able to prove the optimality of an additional set of instances, marked with an asterisk in the sixth column of the table.

Table 6 confirms that, as for the classical VCP, coloring geometric graphs (GEOM) is easy, while pure random graphs (DSJC) are very difficult for the weighted version of the problem as well.

References

Aardal, K.I., van Hoesel, S.P.M., Koster, A.M.C.A., Mannino, C., Sassano, A., 2003. Models and solution techniques for the frequency assignment problem. *4OR: A Quarterly Journal of Operations Research* 1, 4, 261–317.

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

Blöchliger, I., Zufferey, N., 2008. A reactive tabu search using partial solutions for the graph coloring problem. *Computers and Operations Research* 35, 3, 960–975.

Bodlaender, H.L., Jansen, K., 1993. On the complexity of scheduling incompatible jobs with unit-times. In *MFCS '93: Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*. Springer-Verlag, London, pp. 291–300.

Bollobàs, B., Thomason, A., 1985. Random graphs of small order. *Annals of Discrete Mathematics* 28, 47–97.

Boudhar, M., Finke, G., 2000. Scheduling on a batch machine with job compatibilities. *Belgian Journal of Operations Research, Statistics and Computer Science* 40, 1–2, 874–885.

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

Brown, R., 1972. Chromatic scheduling and the chromatic number problem. *Management Science* 19, 4, 456–463.

Campêlo, M., Corrêa, R., Frota, Y., 2004. Cliques, holes and the vertex coloring polytope. *Information Processing Letters* 89, 4, 159–164.

Campêlo, M., Campos, V.A., Corrêa, R., 2008. On the asymmetric representatives formulation for the vertex coloring problem. *Discrete Applied Mathematics* 156, 7, 1097–1111.

Caprara, A., Fischetti, M., 1996. $(0,\frac{1}{2})$-Chvátal-Gomory cuts. *Mathematical Programming* 74, 221–235.

Caprara, A., Fischetti, M., Toth, P., 1999. A heuristic method for the set covering problem. *Operations Research* 47, 730–743.

Caprara, A., Kroon, L., Monaci, M., Peeters, M., Toth, P., 2007. Passenger railway optimization. In Barnhart, C., Laporte, G. (eds) *Transportation, Handbook in Operations Research and Management Science 14*. Elsevier, Amsterdam, pp. 129–187.

Chiarandini, M., Dumitrescu, I., Stützle, T., 2007. Stochastic local search algorithms for the graph colouring problem. In Gonzalez, T.F. (ed.) *Handbook of Approximation Algorithms and Metaheuristics. Computer & Information Science Series*. Chapman & Hall/CRC, Boca Raton, FL, pp. 63-1–63-17.

Chow, F.C., Hennessy, J.L., 1990. The priority-based coloring approach to register allocation. *ACM Transactions on Programming Languages and Systems* 12, 4, 501–536.

Christofides, N., Mingozzi, A., Toth, P., 1979. The vehicle routing problem. In Christofides, N., Mingozzi, A., Toth, P., Sandi, C. (eds) *Combinatorial Optimization*. Wiley, Chichester, pp. 315–338.

Culberson, J.C., Luo, F., 1996. Exploring the k-colorable landscape with iterated greedy. In Johnson, D.S., Trick, M.A. (eds) *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, pp. 245–284.

Davis, L., 1998. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

de Werra, D., 1985. An introduction to timetabling. *European Journal of Operational Research* 19, 151–162.

de Werra, D., Demange, M., Monnot, J., Paschos, V.T., 2007. Time slot scheduling of compatible jobs. *Journal of Scheduling* 10, 111–127.

Desrosiers, C., Galinier, G., Hertz, A., 2008. Efficient algorithms for finding critical subgraphs. *Discrete Applied Mathematics* 156, 244–266.

Dongarra, J.J., 2006. Performance of various computers using standard linear equations software, (linpack benchmark report). Technical Report CS-89-85, University of Tennessee, Computer Science Department.

Dorne, R., Hao, J.K., 1998. Tabu search for graph coloring, t-coloring and set t-colorings. In Voss, S., Martello, S., Osman, I.H., Roucairol, C. (eds) *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston, MA, pp. 33–47.

Escoffier, B., Monnot, J., Paschos, V.T., 2006. Weighted coloring: Further complexity and approximability results. *Information Processing Letters* 94, 3, 98–103.

Fernandes Muritiba, A.E., Iori, M., Malaguti, E., Toth, P., 2008. Algorithms for the bin packing problem with conflicts. Technical report, DEIS, University of Bologna.

Finke, G., Jost, V., Queyranne, M., Sebö, A., 2008. Batch processing with interval graph compatibilities between tasks. *Discrete Applied Mathematics* 156, 5, 556–568.

Fleurent, C., Ferland, J.A., 1996. Object-oriented implementation of heuristic search methods for graph coloring. In Johnson, D.S., Trick, M.A. (eds) *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, pp. 619–652.

Funabiki, N., Higashino, T., 2000. A minimal-state processing search algorithm for graph coloring problems. *IEICE Transactions Fundamentals* E83-A, 7, 1420–1430.

Galinier, P., Hao, J.K., 1999. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization* 3, 4, 379–397.

Galinier, P., Hertz, A., 2006. A survey on local search methods for graph coloring. *Computers and Operations Research* 33, 2547–2562.

Galinier, P., Hertz, A., Zufferey, N., 2008. An adaptive memory algorithm for the *k*-coloring problem. *Discrete Applied Mathematics* 156, 267–279.

Gamst, A., 1986. Some lower bounds for a class of frequency assignment problems. *IEEE Transactions on Vehicular Technology* 35, 1, 8–14.

Garey, M.R., Johnson, D.S., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co, New York.

Gavranovich, H., Finke, G., 2000. Graph partitioning and set covering for the optimal design of a production system in the metal industry. In *Proceedings of the Second Conference on Management and Control of Production and Logistics (MCPL'2000)*, Vol. 2. Grenoble, France, pp. 603–608.

Gendreau, M., Laporte, G., Semet, F., 2004. Heuristics and lower bounds for the bin packing problem with conflicts. *Computers and Operations Research* 31, 347–358.

Hansen, P., Hertz, A., Kuplinsky, J., 1993. Bounded vertex colorings of graphs. *Discrete Mathematics* 111, 305–312.

Hansen, P., Labbé, M., Schindl, D., 2005. Set covering and packing formulations of graph coloring: algorithms and first polyhedral results. Technical Report G-2005-76, GERAD, Université de Montréal.

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

Hertz, A., Plumettaz, M., Zufferey, N., 2008. Variable space search for graph coloring. *Discrete Applied Mathematics* 156, 2551–2560.

Jansen, K., 1999. An approximation scheme for bin packing with conflicts. *Journal of Combinatorial Optimization* 3, 363–377.

Jansen, K., Oehring, S., 1997. Approximation algorithms for time constrained scheduling. *Information and Computation* 132, 85–108.

Johnson, D.S., Trick, M.A. (eds) 1996. *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C., 1991. Optimization by simulated annealing: An experimental evaluation; Part II, graph coloring and number partitioning. *Operations Research* 39, 3, 378–406.

Joslin, D.E., Clements, D.P., 1999. Squeaky wheel optimization. *Journal of Artificial Intelligence Research* 10, 353–373.

Laporte, G., Desroches, S., 1984. Examination timetabling by computer. *Computers and Operations Research* 11, 351–360.

Leighton, F.T., 1979. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards* 84, 6, 489–503.

Lim, A., Zhang, X., Zhu, Y., 2003. A hybrid method for the graph coloring and its related problems. In *Proceedings of MIC2003: The Fifth Metaheuristic International Conference*. Kyoto, Japan.

Lim, A., Lou, Q., Rodrigues, B., Zhu, Y., 2005. Heuristic methods for graph coloring problems. In *Proceedings of the 2005 ACM Symposium on Applied Computing*. Santa Fe, New Mexico, pp. 933–939.

Malaguti, E., Toth, P., 2008. An evolutionary approach for bandwidth multicoloring problems. *European Journal of Operational Research* 189, 638–651.

Malaguti, E., Monaci, M., Toth, P., 2008a. A metaheuristic approach for the vertex coloring problem. *INFORMS Journal on Computing* 20, 2, 302–316.

Malaguti, E., Monaci, M., Toth, P., 2008b. Models and heuristic algorithms for a weighted vertex coloring problem. *Journal of Heuristics*, doi 10.1007/s10732-008-9075-1.

Martello, S., Toth, P., 1990. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Chichester.

Mehrotra, A., 1992. Constrained graph partitioning: Decomposition, polyhedral structure and algorithms. Ph.D. Thesis, Georgia Institute of Technology.

Mehrotra, A., Trick, M.A., 1996. A column generation approach for graph coloring. *INFORMS Journal on Computing* 8, 344–354.

Méndez-Díaz, I., Zabala, P., 2006. A branch-and-cut algorithm for graph coloring. *Discrete Applied Mathematics* 154, 5, 826–847.

Méndez-Díaz, I., Zabala, P., 2008. A cutting plane algorithm for graph coloring. *Discrete Applied Mathematics* 156, 159–179.

Morgenstern, C.A., 1996. Distributed coloration neighborhood search. In Johnson, D.S., Trick, M.A. (eds) *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI. pp. 335–358.

Phan, V., Skiena, S., 2002. Coloring graphs with a general heuristic search engine. In Johnson, D.S., Mehrotra, A., Trick, M.A. (eds) *Computational Symposium on Graph Coloring and its Generalizations*, Ithaca, New York, pp. 92–99.

Prais, M., Ribeiro, C.C., 2000. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing* 12, 3, 164–176.

Prestwich, S., 2002. Constrained bandwidth multicoloration neighborhoods. In Johnson, D.S., Mehrotra, A., Trick, M.A. (eds) *Computational Symposium on Graph Coloring and its Generalizations*. Ithaca, New York, pp. 126–133.

Prestwich, S., 2008. Generalized graph colouring by a hybrid of local search and constraint programming. *Discrete Applied Mathematics* 156, 2, 148–158.

Ribeiro, C.C., Minoux, M., Penna, M.C., 1989. An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment. *European Journal of Operational Research* 41, 232–239.

Sewell, E.C., 1996. An improved algorithm for exact graph coloring. In Johnson, D.S., Trick, M.A. (eds) *Cliques, Coloring, and Satisfiability: 2nd DIMACS Implementation Challenge, 1993. DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI, pp. 359–373.

Trick, M.A., 2002. Computational symposium: Graph coloring and its generalizations. http://mat.gsia.cmu.edu/COLOR02/

Woo, T.K., Su, S.Y.W., Newman Wolfe, R., 2002. Resource allocation in a dynamically partitionable bus network using a graph coloring algorithm. *IEEE Transactions on Communication* 39, 12, 1794–1801.