Adaptive feasible and infeasible tabu search for weighted vertex coloring

Wen Sun [a], Jin-Kao Hao [a,b,*], Xiangjing Lai [c], Qinghua Wu [d]

[a] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[b] *Institut Universitaire de France, 1 Rue Descartes, 75231 Paris, France*

[c] *Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

[d] *School of Management, Huazhong University of Science and Technology, Wuhan 430074, China*

Information Sciences, https://doi.org/10.1016/j.ins.2018.07.037

Abstract

The Weighted Vertex Coloring Problem of a vertex weighted graph is to partition the vertex set into k disjoint independent sets such that the sum of the costs of these sets is minimized, where the cost of each set is given by the maximum weight of a vertex (representative) in that set. To solve this NP-hard problem, we present the adaptive feasible and infeasible search algorithm (AFISA) that relies on a mixed search strategy exploring both feasible and infeasible solutions. From an initial feasible solution, AFISA seeks improved solutions by oscillating between feasible and infeasible regions. To prevent the search from going too far from feasibility boundaries, we introduce a control mechanism that adaptively makes the algorithm to go back and forth between feasible and infeasible solutions. To explore the search space, we use a tabu search optimization procedure to ensure an intensified exploitation of candidate solutions and an adaptive perturbation strategy to escape local optimum traps. We show extensive experimental results on 161 benchmark instances and present new upper bounds that are useful for future studies. We assess the benefit of the key features of the proposed approach. This work demonstrates that examining both feasible and infeasible solutions during the search is a highly effective search strategy for the considered coloring problem and could beneficially be applied to other constrained problems as well.

Keywords: weighted vertex coloring; tabu search; heuristics; feasible and infeasible search; adaptive penalty-based evaluation function.

1 Introduction

Graph vertex coloring problems are very general and useful models to formulate numerous practical problems [23]. Given an undirected graph $G = (V, E)$ with the vertex set $V = \{1, 2, \ldots, n\}$, the edge set $E \subset V \times V$. Graph coloring typically involves assigning a color to each vertex of V such that two vertices linked by an edge must receive different colors while optimizing a given optimization objective. A coloring can also be considered as a partition of the vertex set V into disjointed color classes (also called independent sets or stables), where the vertices of each color class are not linked by an edge of E. For instance, the very popular NP-hard Vertex Coloring Problem (VCP) [8,26] is to find a legal coloring while minimizing the number of colors used (the smallest number of colors needed is the chromatic number of the graph), while the Equitable Vertex Coloring Problem [21], which is also NP-hard, is a special case of the VCP such that the sizes of the stables of the coloring differ by at most one. Graph coloring problems can also concern weighted graphs where a weight (typically a positive value) is associated to each vertex. The Weighted Vertex Coloring Problem (WVCP) considered in this work is a typical example of this class of coloring problems. Informally, the WVCP aims to find a legal coloring such that the sum of the costs of its stables is minimized, where the cost of each stable is given by the maximum weight of a vertex (representative) in that stable. A formal definition of the problem is given in Section 2.1 together with an illustrative example.

One notices that an instance of the NP-hard vertex coloring problem can be conveniently reduced to an instance of the WVCP by defining a weight of 1 for each vertex. As a result, the WVCP is NP-hard [10,24], and thus computationally challenging in the general case. From a practical perspective, the WVCP has a number of practical applications in different fields and arises naturally in the context of buffer management in operating systems [28,30], batch scheduling [11] and manufacturing [18]. As a result, effective solution methods for the WVCP can help to solve these practical problems.

Our literature review given in Section 2.2 indicates that unlike the popular vertex coloring problem for which numerous solution methods are available (see the reviews [8,9,26]), research on algorithms for the WVCP is still in its infancy with very few advanced methods. In this work, we aim to fill the gap by investigating effective heuristics that can be used to find high-quality approximate solutions for problem instances that cannot be solved exactly. Our interest on heuristics for the WVCP is fully motivated by the hardness of the considered problem. Indeed, unless P=NP, exact algorithms for the

* Corresponding author.
Email address: jin-kao.hao@univ-angers.fr (Jin-Kao Hao).

WVCP will have an exponential time complexity and can only be applied to solve problem instances of limited sizes or with particular features.

Our work on investigating a feasible and infeasible search procedure is driven by the following consideration. The WVCP is a constrained combinatorial optimization problem where a feasible solution must satisfy the coloring constraint (i.e., two adjacent vertices must receive different colors). Due to the presence of the coloring constraint, the feasible region can be broken into several zones that are separated from each other by infeasible regions in the search space. In this case, an algorithm searching only feasible solutions could be blocked in a particular feasible zone, thus miss the global optima or high quality solutions located in other feasible zones. On the other hand, as illustrated in numerous studies on constrained optimization, e.g., [4,15,19,20,22,27,29,31,32], methods that are allowed to oscillate between feasible and infeasible regions constitute an appropriate means to cope with such a situation. Indeed, allowing a controlled exploration of infeasible solutions may facilitate transitions between structurally different solutions and help discover high-quality solutions that are difficult to locate if the search is limited to the feasible region. Based on previous studies of examining feasible and infeasible solutions for solving other constrained optimization problems, we present in the work the first study of mixing both feasible and infeasible searches with the context of the WVCP. We summarize the main contributions of this work as follows.

First, the adaptive feasible and infeasible search algorithm (AFISA) presented in this work is the first heuristic method that explores both feasible and infeasible solutions for the WVCP. To prevent the search from going too far away from the feasible boundary, we design an adaptive penalty-based evaluation function that is used to guide the search for a fruitful examination of candidate solutions, by enabling the search to oscillate between feasible and infeasible regions. To ensure an effective exploration of both feasible and infeasible regions, we adopt the popular tabu search meta-heuristic [14] and design specific search components to cope with the particular features of the considered coloring problem.

Second, we assess the proposed algorithm on 111 conventional benchmark instances from the literature (one set of 46 instances from the DIMACS and COLOR competitions and two sets of 65 instances from matrix-decomposition problems). We report especially 5 improved best solutions (new upper bounds). We also present results on an additional set of 50 (larger) DIMACS instances. These results and the proposed algorithm can serve as new references to assess future WVCP algorithms and can be useful for the design of effective exact algorithms as well.

The remainder of the paper is organized as follows. Section 3 presents the proposed algorithm. Section 4 describes computational results and comparisons

with state-of-the-art algorithms. Section 5 analyzes three essential components of the proposed algorithm to shed light on how they affect the performance of the algorithm. Conclusions and future work are discussed in the last section.

2 Problem definition and literature review

This section formally introduces the weighted vertex coloring problem considered in this work, followed by a review of the existing solution methods available in the literature.

2.1 Weighted vertex coloring problem

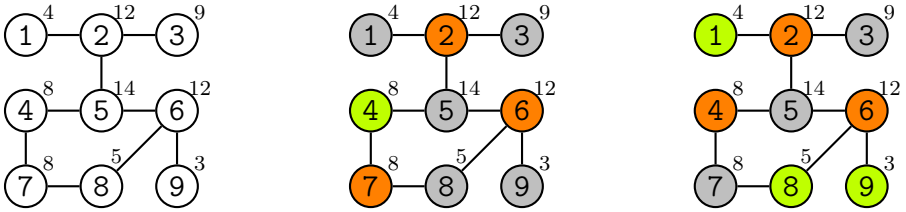
Given an undirected graph $G = (V, E)$ with the vertex set $V = \{1, 2, \ldots, n\}$, the edge set $E \subset V \times V$. Let $W = \{w_1, w_2, \ldots, w_n\}$ be the set of positive weights associated to the vertices of V. Recall that an independent set (a stable or a color class) of G is a subset of V such that any pair of its vertices is not linked by an edge of E. A legal or feasible k-coloring of G is a partition of the vertex set V into k disjoint independent sets $\{V_1, V_2, \ldots, V_k\}$. Let $s = \{V_1, V_2, \ldots, V_k\}$ be a partition of the vertex set V, the Weighted Vertex Coloring Problem can be stated as follows.

$$(WVCP) \quad \text{minimize} \quad f(s) = \sum_{i=1}^{k} max_{j \in V_i} w_j \tag{1}$$

$$\text{subject to} \quad \forall u, v \in V_i, \ \{u, v\} \notin E, i = 1, 2, \ldots, k \tag{2}$$

where the constraints (2) ensure that partition $\{V_1, V_2, \ldots, V_k\}$ is a legal k-coloring (i.e., each V_i $(i = 1, 2, \ldots, k)$ is a stable) and the objective (1) is to minimize the maximum weight of a vertex (representative) in each of the k stables V_i $(i = 1, 2, \ldots, k)$. Notice that for a given graph, the number of colors k is unknown before the optimal solution is discovered.

Figure 1(a) shows a graph $G = (V, E)$ with 9 vertices whose weights are indicated next to the vertices. Figure 1(b) shows a coloring with three stables, leading to an objective value of $14 + 12 + 8 = 34$, since the three stables have respectively a maximum weight of 14 (gray stable), 12 (orange stable) and 8 (green stable). Figure 1(c) illustrates an optimal solution with a minimum objective value of $14 + 12 + 5 = 31$.



(a) A graph $G = (V, E)$ (b) A feasible coloring (c) An optimal solution

Fig. 1. A graph, a feasible solution, an optimal solution

2.2 Literature review on existing solution methods

From a perspective of solution methods for the WVCP in the general case, several exact algorithms have been proposed. Specifically, a column generation approach combined with the general branch-and-bound method was investigated in [30]. A branch-and-price approach was presented in [6] and computational results were shown on a subset of benchmark instances from the DIMACS and COLOR competitions and two sets of instances from matrix-decomposition problems. In [5], the WVCP was solved as Maximum Weight Stable Set Problems on an associated graph, and this approach showed excellent results on the tested benchmark graphs.

Given that the WVCP is a NP-hard problem, a number of heuristic algorithms have also been investigated, which aim to provide high-quality solutions in acceptable computation time, but without provable optimal guarantee of the attained solutions. Heuristic algorithms constitute an indispensable complementary approach with respect to exact algorithms to solve instances whose optimal solutions cannot be attained. For example, a Greedy Randomized Adaptive Search Procedure (GRASP) was introduced in [28] in the context of a practical problem called TDMA traffic assignment (an application of the WVCP). This algorithm iterates a mixed search strategy combining a randomized greedy construction procedure followed by a local optimization procedure. In [25], an effective 2-phase algorithm was proposed, where in the first phase a large number of independent sets are heuristically produced, and in the second phase the set covering problem associated with these sets is solved by the Lagrangian heuristic algorithm introduced previously in [3]. These heuristics have reported interesting results on a number of benchmark instances and will serve as our main references for our computational studies. One notices that these methods only examine feasible solutions.

The above review indicates that despite the theoretical and practical significance of the WVCP, solution methods for the problem are quite limited and the WVCP benchmark instances are of small sizes in comparison with those used for other graph coloring problems. To enrich the solution arsenal for the WVCP and solve large problem instances, we present in this work a new

heuristic algorithm that relies on a mixed strategy of exploring feasible and infeasible solutions and show extensive computational results on both conventional benchmark instances and new (larger) instances.

3 Adaptive feasible and infeasible search for the WVCP

In this section, we present the adaptive feasible and infeasible search algorithm for solving the WVCP. We first show the general approach and then explain in detail the components of the proposed algorithm including the search space, the evaluation function, the neighborhood, the tabu search procedure, the adaptive mechanism to control feasible and infeasible searches and the perturbation strategy.

3.1 General approach

Unlike existing methods for the WVCP [25,28] that only consider feasible colorings, the adaptive feasible and infeasible search algorithm (AFISA) proposed in this work enlarges the search to include both feasible and infeasible colorings. Indeed, as explained in the introduction, methods that are allowed to oscillate between feasible and infeasible regions can help to attain solutions of high-quality that would not be discovered otherwise. Specifically, we first generate an initial feasible solution with a greedy procedure (Section 3.2). Then, we improve the solution by enlarging the search to include infeasible solutions. To enable the search to oscillate between feasible and infeasible regions, we devise an extended evaluation function F that combines the objective function of Eq. (1) with a penalty function (Section 3.3). To control the importance given to the penalty function, we introduce an adaptive parameter that is dynamically adjusted according to the search context (Section 3.4.3). To explore the search space, we use a tabu search procedure that relies on a neighborhood induced by the one-move operator and that is guided by the penalty-based evaluation function (Section 3.4.2). Finally, to escape local optimum traps, we introduce an adaptive perturbation strategy to generate a new starting solution for the next round of tabu search (Section 3.5). The pseudo-code of the proposed algorithm is presented in Algorithm 1.

3.2 Initial solution

The purpose of the initialization step is to generate an initial feasible solution of acceptable quality. This is achieved by adopting the greedy procedure of [28]

Algorithm 1 Main scheme of the AFISA algorithm for the WVCP

1: **Input**: Graph G
2: **Output**: The best solution found
3: $s_0 \leftarrow greedy\_initial(G)$ /* Generate an initial feasible k-coloring, Section 3.2 */
4: $s_{best} \leftarrow s_0$ /* Record the best legal solution */
5: $\gamma \leftarrow 0$ /* γ is the counter for consecutive non-improving local optima */
6: $s_1 \leftarrow s_0$
7: $L \leftarrow L_0$/* Initialize the depth of perturbation, Section 3.5 */
8: $\varphi \leftarrow 1$ /* Initialize penalty coefficient of the extended evaluation function, Sections 3.3 and 3.4.3 */
9: **while** stopping condition is not met **do**
10: $s_2 \leftarrow tabu\_search(s_1)$ /* Section 3.4 */
11: **if** $F(s_2) < F(s_{best})$ *and* $F(s_2) = f(s_2)$ **then**
12: $s_{best} \leftarrow s_2$ /* Record the best legal solution */
13: $L \leftarrow L_0$ /* Reinitialize the depth of perturbation */
14: $\gamma \leftarrow 0$ /* Reset the counter for consecutive non-improving local optima */
15: **else**
16: $\gamma \leftarrow \gamma + 1$
17: **end if**
18: **if** $\gamma = T$ **then**
19: $L \leftarrow L_{max}$ /* Increase the depth of perturbation if the best solution is not updated during T consecutive tabu search runs */
20: **end if**
21: $\varphi \leftarrow adaptive\_parameter(s_2, \varphi)$ /* Adjust penalty coefficient, Section 3.4.3 */
22: $s_1 \leftarrow perturbation(s_2, L)$ /* Section 3.5 */
23: **end while**
24: **return** s_{best}

that iteratively assigns the vertices to a suitable color class (Algorithm 2). Let $\{V_1, V_2, \ldots, V_k\}$ be the current partial solution with k color classes (initially k is set to 1, $V_1 = \emptyset$), we choose an unassigned vertex v (Algorithm 2, line 6) and assign it to the color class $V_i, 1 \leq i \leq k$ such that the objective function f is minimized while the coloring constraint is met. If no suitable color class $V_i, 1 \leq i \leq k$ exists for vertex v, we create a new color class V_k by setting $k \leftarrow k + 1$ and assign the vertex v to this new color class (Algorithm 2, line 15-16). This process is repeated until all vertices are assigned to a color class.

This initialization procedure provides thus the AFISA algorithm with a legal k-coloring of certain quality, which will be further improved during the tabu search phase of the algorithm.

3.3 Search space and penalty-based evaluation function

To further improve the initial solution provided by the above initialization procedure, the search phase of the AFISA algorithm explores an enlarged

Algorithm 2 Greedy initialization for the WVCP

Input: Graph $G = (V, E)$
Output: A legal k-coloring s
$U \leftarrow V$ /* U is the set of unassigned vertices */
$i = 1, k = 1, V_i = \emptyset$
while $U \neq \emptyset$ **do**
  choose a vertex v from U with maximum weight
  $assign = 0$ /* $assign$ is a flag indicating if vertex v is assigned a color */
  for $i = 1, 2, \ldots, k$ **do**
    if *there is no edge between v and any vertex of V_i* **then**
      $V_i \leftarrow V_i \cup \{v\}$ /* v is assigned to color class V_i */
      $assign = 1$ /* v is now assigned a color */
    end if
  end for
  if $assign = 0$ **then**
    $k \leftarrow k + 1$
    $V_k \leftarrow V_k \cup \{v\}$ /* Create a new color class V_k to hold v */
  end if
  $U \leftarrow U \setminus \{v\}$
end while
return $s = \{V_1, V_2, \ldots, V_k\}$ /* s is a legal or feasible k-coloring */

space Ω including both feasible and infeasible solutions. In other words, the space Ω is composed of the partitions of the vertex set V into k disjoint subsets.

$$\Omega = \{\{V_1, V_2, \ldots, V_k\} : \cup_{i=1}^{k} V_i = V, V_i \cap V_j = \emptyset, 1 \leq k \leq |V|\} \tag{3}$$

where $i \neq j, 1 \leq i, j \leq k$.

Following the general idea of penalty function for constrained optimization, we introduce an extended evaluation function F to asses both feasible and infeasible solutions of Ω, which enriches the objective function f (Equation (1)) with a penalty function P. Let $s = \{V_1, V_2, \ldots, V_k\}$ be a candidate solution in Ω, we define its penalty $P(s)$ as $P(s) = \sum_{i=1}^{k} |C(V_i)|$ where $C(V_i)$ counts the number of pair of conflicting vertices in color class V_i that are linked by an edge. In other words, $P(s)$ indicates the number of conflicts in the candidate solution s. Therefore, for the candidate solution s, if the penalty $P(s)$ equals 0, then s corresponds to a feasible k-coloring satisfying the coloring constraint. Otherwise (i.e., $P(s) > 0$), the solution includes at least two adjacent vertices violating the coloring constraint, i.e., belonging to a same color class.

Then the quality of the solution $s = \{V_1, V_2, \ldots, V_k\}$ is given by the following extended evaluation function:

$$F(s) = f(s) + \varphi P(s) \tag{4}$$

where $f(s)$ is the objective function value and $\varphi \geq 1$ is a parameter that is used to control the relative importance given to the penalty function.

Since F is to be minimized, increasing φ augments the value of F, making thus the infeasible solution under consideration less attractive. Inversely, decreasing φ lowers the evaluation value, making the solution more attractive. By varying φ, we can control the transition between feasible and infeasible regions. We explain in Section 3.4.3 the adaptive technique to dynamically tune φ according to the search situation. We investigate in Section 5.2 the impact of the φ parameter.

Using the extended evaluation function F, we assess the relative quality of two candidate solutions x and y as follows: x is better than y if $F(x) < F(y)$.

3.4 Searching feasible and infeasible solutions with tabu search

3.4.1 Neighborhood and its evaluation

The AFISA algorithm examines the search space Ω by making transitions from the current solution to one neighboring solution. A neighboring solution is generated by using the popular "one-move" operator. Give a solution $s = \{V_1, V_2, \ldots, V_k\}$, the one-move operator displaces a vertex v from its current color class V_i to a different color class V_j ($i \neq j$, $j \in \{1, 2, \ldots, |V|\}$), leading to a neighboring solution designated by $s \oplus < v, V_i, V_j >$. The one-move neighborhood is then given by:

$$N(s) = \{s \oplus < v, V_i, V_j > : v \in V_i, 1 \leq i \leq k, 1 \leq j \leq |V|, i \neq j\} \tag{5}$$

This neighborhood allows a vertex to be moved to a currently empty color class V_j with $j > k$ (thus the number of color classes can increase). Inversely, a color class can also become empty and thus be removed when its last vertex is transferred to another existing class (thus k is decreased). As a result, search algorithms using this neighborhood like the tabu search procedure presented in Section 3.4.2 typically visit solutions whose number of color classes varies during the search.

Notice that this neighborhood is different from the one-move neighborhood

used for other coloring problems (e.g., conventional vertex coloring [8,26] and equitable coloring problem [21,31,32], because 1) in these studies the vertex $v \in V_i$ to be displaced must be a conflicting vertex (i.e., there exists in V_i at least another vertex u such that v and u are adjacent in the graph), and 2) the number of color classes k remains fixed.

Finally, one important issue concerns the evaluation of the neighborhood that impacts significantly the computational efficiency of any local search algorithm. In our case, we employ an incremental evaluation technique that is similar to the evaluation technique designed for another graph coloring problem (i.e., equitable coloring) [21,31]. With this technique, the objective variation of each neighbor solution can be conveniently obtained in constant time. We refer the readers to [21,31] for more details of this evaluation technique.

3.4.2 Tabu search

To explore the above neighborhood, the proposed AFISA algorithm uses the well-known tabu search (TS) meta-heuristic [12,13] that has been applied to many difficult combinatorial optimization problems [14]. In particular, TS is known to be quite successful in solving several different graph coloring problems such as general graph coloring [7,17], minimum sum coloring [19] and equitable coloring [21]. As a general meta-heuristic, TS has some attractive features. First, it can be adapted to graph coloring problems rather easily. Moreover, TS offers simple strategies to promote a suitable and necessary search balance between intensification (with the best-improvement principle to explore a given neighbourhood) and diversification (with a tabu list).

In our case, the tabu search procedure make transitions between various k-colorings guided by the extended evaluation function F of Section 3.3. Our TS procedure is based on the popular TabuCol algorithm for the conventional Vertex Coloring Problem [17] and adopts the improvements presented in [7]. For the sake of completeness, we show the pseudo-code of the tabu search procedure in Algorithm 3 and provide the following description.

The tabu search procedure iteratively replaces the current solution s by a neighboring solution s' taken from the one-move neighborhood $N(s)$ defined in Section 3.4.1 until a stopping condition is met. At each iteration, TS examines the neighborhood and selects a best admissible neighboring solution s' (see below) to substitute s. After each iteration, the associate one-move is recorded on the tabu list to prevent the search from revisiting s for the next tt iterations (tt is called the tabu tenure). To tune the tabu tenure, we use $tt = Random(A) + \alpha F$ where F stands for the extended evaluation function, the function $Random(A)$ returns a random number in $\{0, \ldots, \text{A-1}\}$ (A is set to 10 in this work) and α is a parameter set to 0.6. Meanwhile, the best so-

lution found is updated if the new solution is better than all previous visited solutions. A neighboring solution s' is considered to be admissible if it is not forbidden by the tabu list or if it is better (according to the extended evaluation function F) than the best solution found. If the best solution is not updated during β consecutive iterations (β is called the depth of tabu search), then the current tabu search process is considered to be stagnating and stops.

Algorithm 3 Tabu search algorithm

1: **Input**: initial solution s, depth of tabu search β
2: **Output**: The best solution s_b found during the tabu search process
3: $s_c \leftarrow s$ /* s_c is the current solution */
4: $s_b \leftarrow s_c$ /* s_b is the best solution found */
5: $d \leftarrow 0$ /* d counts the consecutive iterations during which s_b is not updated */
6: **repeat**
7: Choose a best admissible neighboring solution $s' \in N(s_c)$
/* s' is admissible if it is not forbidden by the tabu list or better than s_b */
8: $s_c \leftarrow s'$
9: /* Update the best solution */
10: **if** $F(s_c) < F(s_b)$ **then**
11: $s_b \leftarrow s_c$
12: $d \leftarrow 0$
13: **else**
14: $d \leftarrow d + 1$
15: **end if**
16: **until** $d = \beta$
17: **return** s_b

3.4.3 Adaptive mechanism to control feasible and infeasible searches

The AFISA algorithm uses the extended evaluation function F (see Section 3.3) combining the objective function f and the penalty function P to assess the quality of candidate solutions and guide the search process. By varying the penalty coefficient φ of F, we can change the search trajectory according to the search situation. Basically, a large (small) φ value strongly (weakly) penalizes infeasible solutions and incites the search process to give more importance to feasible (infeasible) solutions. To allow the search process to go back and forth between feasible and infeasible zones, we devise an adaptive mechanism to dynamically adjust φ such that a suitable diversification-intensification balance can be reached. This adaptive mechanism relies on known ideas proposed for continuous optimization like those reviewed in [16].

The adaptive adjustment mechanism is shown in Algorithm 4. According to whether the solution s (obtained from the last round of tabu search) is a feasible k-coloring, we adjust φ to influence the search trajectory of the next round of tabu search. Specifically, if $P(s) \neq 0$ (i.e., $F(s) \neq f(s)$) (Algorithm 4, line 3), which means solution s is an infeasible k-coloring, we increase φ

to penalize more strongly the infeasible solutions. As such, if the tabu search procedure always ends up with an infeasible solution during several consecutive runs, the search is considered to perform enough exploration in infeasible zones and thus encouraged to move towards feasible regions to intensify its search during the next round of tabu search. Inversely, if $P(s) = 0$ (i.e., $F(s) = f(s)$), the solution returned by the last round of tabu search is a feasible k-coloring, indicating that the search just examined a feasible region. In this case, we decrease φ to raise the chance of visiting infeasible regions during the next tabu search run and thus diversify the search. The computational results of Section 4 show that the AFISA algorithm equipped with this adaptive mechanism of exploring feasible and infeasible solutions reaches a high performance.

Algorithm 4 Adaptive adjustment mechanism for penalty coefficient φ

1: **Input**: Penalty coefficient φ, the best solution from the last round of tabu search s
2: **Output**: Adjusted penalty coefficient φ
3: **if** $F(s) \neq f(s)$ **then**
4: $\varphi \leftarrow \varphi + 1$ /* Increase the penalty term to guide the search toward feasible regions */
5: **else**
6: $\varphi \leftarrow \varphi - 1$ /* Decrease the penalty term to increase the chance of visiting infeasible solutions */
7: **end if**
8: **if** $\varphi \leq 0$ **then**
9: $\varphi \leftarrow 1$
10: **end if**

3.5 Perturbation strategy

As illustrated in Section 3.4.2, the tabu search procedure ends up with a local optimal solution. To enable the search to move to new search zones, we apply a perturbation strategy to modify the last local optimum that is then used as the new starting solution of the next round of tabu search (line 22, Algorithm 1). To make the perturbation strategy as effective as possible, we borrow ideas from breakout local search [2] whose perturbation strategy relies on two factors: the jump magnitude L (also called depth of perturbation) and the perturbation type.

The jump magnitude L indicates the number of perturbation moves to be applied and in our case, is set to a small value (L_0) in the beginning (line 7, Algorithm 1). If the search is observed to be stagnating (i.e., no better feasible solution can be found during T consecutive tabu search runs), L is increased to a large value (L_{max}) to enable a stronger diversification (line 19, Algorithm 1). Then each time the search moves to a new promising search zone by discovering a better feasible solution, L is switched to L_0 again (line 13, Algorithm 1).

To perform a perturbation, two types of moves, both being based on the one-move operator (see Section 3.4.1), are applied according to a probability P_0. The first type of perturbation, called tabu-based perturbation, relies on the tabu principle and selects a one-move $< v, V_i, V_j >$ that minimizes the objective degradation while considering the tabu list, as in Section 3.4.2. Starting from an empty tabu list, each time a perturbation move is performed, the move is recorded in the tabu list and will not be considered during the current perturbation procedure (this is achieved by setting the tabu tenure to ∞). The second type of perturbation, called direct perturbation, also chooses a one-move that leads to the least objective degradation, but without considering any tabu restriction (this is achieved by setting the tabu tenure to 0). Finally, during the perturbation procedure, if a feasible solution that is better than the best solution found from the start of the search is reached, the best recorded solution is updated accordingly.

3.6 Connections with existing studies

For the WVCP considered in this work, one notices that existing heuristic algorithms [25,28] visit only feasible solutions while ignoring infeasible solutions. As such, these algorithms could encounter difficulties when the feasible solutions are scattered in different zones that are separated by infeasible zones. In this work, we explore for the first time the idea of searching both feasible and infeasible solutions for solving the WVCP. Indeed, as illustrated in other settings, e.g., [4,27,31,32], such a mixed search strategy is highly effective for solving several difficult problems (e.g., capacitated arc routing, capacitated clustering and equitable coloring). Among these studies, two of them [31,32] are worthy of a special mention, because they consider the related equitable coloring problem (ECP). Given that the WVCP studied in this work and the ECP considered in [31,32] are two different coloring problems, our work posses several particular features that distinguish itself from these studies.

First, the proposed AFISA algorithm incorporates search components (dedicated neighborhood, specific penalty-based evaluation function, perturbation technique...) that are customized to the WVCP. Second, unlike [31,32] where different algorithms are designed to search separately feasible and infeasible solutions, AFISA uses the same tabu search procedure to exploit both types of solutions, making the algorithm simpler in design and implementation. Third, AFISA integrates an adaptive mechanism to dynamically control the feasible and infeasible searches by the self-tuned parameter φ. Such a mechanism is missing in the studies mentioned above. Fourth, unlike [31,32], AFISA does not solve a series of k-coloring problems where each coloring problem is defined for a fixed number of colors k. Instead, the number of colors k varies during the search of our algorithm as explained in Section 3.4.1. Finally, as we show

in Section 4, the proposed algorithm integrating these features competes very favorably with the state of the art methods for the WVCP in the literature.

4 Experimental results and comparisons

This section is dedicated to a large experimental assessment of the proposed AFISA algorithm for solving the WCVP and comparisons with other state of the art methods. The study is based on 111 conventional benchmark instances in the literature as well as 50 new (large) instances from the DIMACS and COLOR competitions.

4.1 Benchmark instances

Test instances. We consider 111 instances from the literature on the WCVP [5,25,28] and 50 additional instances from the DIMACS and COLOR competitions initially proposed for the conventional graph coloring problems [1,2]. We classify these instances into four sets.

(1) The first set contains 46 (small) instances from the DIMCAS/COLOR competitions. Graphs of this set have $DSJC^*$, $GEOM^*$ or R^* in their name with up to 125 vertices. The exact algorithm ($MWSS$) [5] is able to find the optimal solutions for 40 instances in this set. These instances are tested in [5,25], whose results will be used as our references.
(2) The second set contains 35 instances from matrix-decomposition problems. These graphs named as pxx have up to 138 vertices and 1186 edges. The exact algorithm ($MWSS$) [5] is able to find the optimal solutions for all graphs in this set. These instances are tested in [5,25,28], whose results will be used as our references.
(3) The third set contains 30 rxx instances proposed in [28] from matrix-decomposition problems. These instances have the same structure as the pxx instances, but are larger, having up to 301 vertices and 4122 edges. These instances are tested in [5,28] and the exact algorithm ($MWSS$) [5] is able to find the optimal solutions for all graphs in this set.
(4) The fourth set contains 50 additional larger instances (with at least 120 vertices) [3]. These instances are created by adding random vertex weights between 1 to 20 to DIMACS/COLOR graphs.

[1] `http://www.dimacs.rutgers.edu/`
[2] `http://www.cs.hbg.psu.edu/txn131/graphcoloring.html/`
[3] These new instances are available at:
`http://www.info.univ-angers.fr/~hao/wvcp.html`

4.2 Experimental settings

The proposed algorithm was coded in C++ and compiled by GNU g++ 4.1.2 with -O3 flag (option). The experiments were conducted on a computer with an Intel Xeon E5-2670 processor (2.5 GHz and 2 GB RAM) running Ubuntu 12.04. When solving the DIMACS machine benchmark procedure 'dfmax.c' [4] without compilation optimization flag, the run time on the computer is 0.46, 2.68 and 10.70 seconds for graphs r300.5, r400.5 and r500.5, respectively.

Parameters. The setting of the parameters is given in Table 1, which was determined by a preliminary experiment. For this, we first identify a rough range of values for each parameter. To identify the default value of a particular parameter, we test different values from the range while fixing the other parameters to their default values (typically those of Table 1). As to the penalty coefficient φ of the extended evaluation function F, it is tuned adaptively as explained in Section 3.4.3. We use the default setting of Table 1 to report the experimental results shown in the rest of the paper, though fine-tuning some parameters could lead to improved results.

Table 1
Settings of important parameters

Parameters	Description	Value
L_0, L_{max}	Small and large jump magnitude	0.05*N, 0.5*N
T	Max number of non-improving local optima visited before strong perturb	50
P_0	Probability for applying tabu-based or direct perturbation	0.7
β	Depth of tabu search	100,$N < 50$
		10000, $N \geqslant 50$

Reference algorithms. For our comparative study, we use the most recent heuristic algorithms [25,28] as our references. The *GRASP* algorithm [28] was run on an IBM 9672 model R34 mainframe computer under a limit of 1000 iterations. The 2_*Phase* algorithm [25] was run on a PIV 2.4 MHz with 512 MB RAM under Windows XP and tested by stopping phase 1 after 500 iterations and phase 2 after 75 seconds. When solving the DIMACS machine benchmark procedure 'dfmax.c', the run time on the instance r500.5 reported in [25] for this machine is 7 seconds (against 10.7 seconds for our computer). We also include the lower and upper bounds reported by the exact algorithm *MWSS* in [5]. The results of *MWSS* were obtained on a computer equipped with an Intel Xeon E3-1220 at 3.10 GHz with 8 GB RAM, which spent 4.5 seconds to solve the benchmark instance r500.5 (this computer is roughly 2 times faster than our computer). These bounds provide useful information when they are contrasted with the results (upper bounds) obtained by the compared heuristic algorithms (*GRASP*, 2_*Phase* and AFISA).

[4] `dfmax:ftp://dimacs.rutgers.edu/pub/dsj/clique/`

Following [1], we show computational results in terms of solution quality and computation time. Since computation time can be greatly biased by many factors like computing platform, programming language, data structures and so on, solution quality is the primary criterion while timing information is provided only for indicative purposes.

Stopping condition. Following [25,28], we ran our AFISA algorithm 20 times for the instances from the first, second and fourth sets with a cutoff time of 1 hour per run. For the rxx instances of the third set, a cutoff time of 4 hours is used in [5] used on their computer (Intel Xeon E3-1220 processor, 3.10 GHz and 8 GB RAM), which roughly corresponds to 8 hours on our computer. We set a cutoff time of 2 hours for the instances with up to 200 vertices and 4 hours for larger instances for our AFISA algorithm (longer times do not lead to significantly improved results). Finally, note that for 8 instances of the first set that cannot be solved by $MWSS$ [5] in 1 hour, a large time limit of 10 hours was allowed (marked with $^{\circledast}$ in Table 2), leading to optimality proof of two instances (R100_1g and R100_1gb).

4.3 Computational results and comparisons with state-of-the-art algorithms

Table 2 reports the results of our AFISA algorithm on the first set of 46 DIMACS/COLOR instances commonly used in the literature, together with the results of the reference algorithms $MWSS$ [5] and 2_$Phase$ [25]. The first 3 columns indicate for each instance its name, the number of vertices and the number of edges. The fourth column shows the best-known value (BKV) reported in the literature [5,25,28]. The next four columns show the results of the AFISA algorithm for each instance: the best result (i.e., the smallest objective function value) over 20 independent runs ($Best$), the success rate (SR) to achieve the best result over 20 runs, the average result (Avg) and the average computation time (in seconds) of the successful runs to obtain the best result ($t(s)$) (0 is given if the time is less than 0.009 second). The following four columns report the results and the computation time obtained by the reference algorithms (MWSS and 2_$Phase$). For MWSS, the indicated time corresponds to the time of 1000 iterations. For 2_$Phase$, the time is the sum of the time of 500 iterations of phase 1 and the time of phase 2. The last two columns (Δ_1,Δ_2) indicate the difference between our result ($Best$) and the result of $MWSS$ and 2_$Phase$. To verify the statistical significance of the comparisons between AFISA and each reference algorithm, we show in the row "p-values" the results from the non-parametric Friedman test applied to the best values of AFISA and each compared algorithm, and a p-value smaller than 0.05 implies a significant difference between two sets of compared results.

Additionally, the rows #Better, #Equal and #Worse indicate respectively the

Table 2
Comparative results of AFISA with state-of-the-art algorithms on the 46 DIMACS benchmark instances. Improved upper bounds are indicated in bold.

				AFISA				$MWSS$[5]		2_$Phase$[25]			
Instance	$\|V\|$	$\|E\|$	BKV	*Best*	SR	*Avg*	$t(s)$	*Best*	$t(s)$	*Best*	$t(s)$	Δ_1	Δ_2
DSJC125_1g.col$^{\circledast}$	125	736	24	**23**	2	24.0	3016.32	25	36000.0	24	152	-2	-1
DSJC125_1gb.col$^{\circledast}$	125	736	95	**90**	1	92.5	402.57	95	36000.0	95	170	-5	-5
DSJC125_5g.col$^{\circledast}$	125	3891	76	**71**	2	72.3	216.04	78	36000.0	76	180	-7	-5
DSJC125_5gb.col$^{\circledast}$	125	3891	251	**243**	1	250.2	369.34	263	36000.0	251	182	-20	-8
DSJC125_9g.col	125	6961	169*	169*	3	169.9	16.00	169*	0.3	169*	162	0	0
DSJC125_9gb.col	125	6961	604*	604*	3	605.5	443.96	604*	0.4	605	237	0	-1
GEOM100.col	100	547	65*	65*	20	65.0	0.81	65*	3.8	65*	131	0	0
GEOM100a.col	100	992	89*	89*	10	89.5	110.42	89*	2.4	89*	112	0	0
GEOM100b.col	100	1050	32*	32*	1	33.1	59.11	32*	3.8	32*	15	0	0
GEOM110.col	110	638	68*	68*	20	68.0	33.81	68*	59.5	69	172	0	-1
GEOM110a.col	110	1207	97*	97*	6	97.8	176.76	97*	12.9	97*	111	0	0
GEOM110b.col	110	1256	37*	37*	14	37.9	130.85	37*	5.0	37*	5	0	0
GEOM120.col	120	773	72*	72*	20	72.0	33.14	72*	157	72*	157	0	0
GEOM120a.col	120	1434	105*	105*	1	106.3	156 .00	105*	7.0	105*	136	0	0
GEOM120b.col	120	1491	35*	35*	7	37.3	67.72	35*	16.5	35*	14	0	0
GEOM30b.col	30	81	12*	12*	20	12.0	0.02	12*	0.0	12*	0	0	0
GEOM40b.col	40	157	16*	16*	20	16.0	0.03	16*	0.1	16*	1	0	0
GEOM50b.col	50	249	18*	18*	20	18.0	0.02	18*	0.1	18*	0	0	0
GEOM60b.col	60	366	23*	23*	20	23.0	0.22	23*	0.5	23*	0	0	0
GEOM70.col	70	267	47*	47*	20	47.0	4.99	47*	0.3	47*	96	0	0
GEOM70a.col	70	459	73*	73*	20	73.0	4.42	73*	0.4	73*	3	0	0
GEOM70b.col	70	488	24*	24*	20	24.0	12.03	24*	0.9	24*	6	0	0
GEOM80.col	80	349	66*	66*	20	66.0	2.11	66*	1.1	66*	0	0	0
GEOM80a.col	80	612	76*	76*	19	76.1	137.1	76*	1.1	76*	102	0	0
GEOM80b.col	80	663	27*	27*	5	27.8	66.8	27*	2.5	27*	90	0	0
GEOM90.col	90	441	61*	61*	16	61.2	88.92	61*	2.0	61*	166	0	0
GEOM90a.col	90	789	73*	73*	3	74.0	512.41	73*	4.8	73*	157	0	0
GEOM90b.col	90	860	30*	30*	19	30.1	67.38	30*	1.7	30*	11	0	0
R100_1g.col$^{\circledast}$	100	509	21*	21*	1	22.0	113.77	21*	28788.5	22	155	0	-1
R100_1gb.col$^{\circledast}$	100	509	81*	81*	1	83.8	3.04	81*	9362.2	-	171	0	-
R100_5g.col$^{\circledast}$	100	2456	59	59	5	60.1	6.97	59	36000.0	-	179	0	-
R100_5gb.col$^{\circledast}$	100	2456	225	**221**	1	224.1	186.81	225	36000.0	-	179	-4	-
R100_9g.col	100	4438	141*	141*	15	141.3	21.36	141*	0.1	-	123	0	-
R100_9gb.col	100	4438	518*	518*	1	549.3	1152.83	518*	0.3	-	127	0	-
R50_1g.col	50	108	14*	14*	20	14.0	0.14	14*	0.8	14*	0	0	0
R50_1gb.col	50	108	53*	53*	20	53.0	0.24	53*	1.6	53*	95	0	0
R50_5g.col	50	612	37*	37*	20	37.0	0.95	37*	1.4	37*	167	0	0
R50_5gb.col	50	612	135*	135*	14	135.3	3.72	135*	1.5	137	145	0	-2
R50_9g.col	50	1092	74*	74*	20	74.0	0.74	74*	0.0	74*	36	0	0
R50_9gb.col	50	1092	262*	262*	20	262.0	12.61	262*	0.0	262*	33	0	0
R75_1g.col	70	251	18*	18*	12	18.4	10.96	18*	132.8	19	154	0	-1
R75_1gb.col	70	251	70*	70*	19	70.1	2.46	70*	192.2	72	166	0	-2
R75_5g.col	75	1407	51*	51*	12	51.4	0.08	51*	1056.0	53	172	0	-2
R75_5gb.col	75	1407	186*	186*	2	189.0	19.42	186*	989.3	190	173	0	-4
R75_9g.col	75	2513	110*	110*	20	110.0	2.65	110*	0.0	110*	79	0	0
R75_9gb.col	75	2513	396*	396*	12	396.4	145.89	396*	0.0	399	50	0	-3
#Better				5/46				0/46		0/41			
#Equal				41/46				43/46		32/41			
#Worse				0/46				3/46		9/41			
p_value				-				2.5e-2		3.1e-4			

number of instances for which an algorithm performs better, equally well or worse compared to the best-known values (BKV).

Finally, an entry with * indicates the optimal objective value. A bold entry highlights an improved upper bound, i.e., an improved result over the current best-known value. Entries with "-" mean that the corresponding results are not available in the literature.

From Table 2, we observe that AFISA reaches a remarkable performance on the first set of 46 DIMACS/COLOR instances. Compared to the most recent exact algorithm $MWSS$, AFISA attains all known optimal results (40 cases). For 5 out of the 6 remaining instances whose optimal values are still unknown, AFISA improves the current best upper bounds (see negative entries in column

Table 3
Comparative results of AFISA with state-of-the-art algorithms on the 35 pxx benchmark instances.

				AFISA				$MWSS$[5]		2_$Phase$[25]		$GRASP$[28]				
Instance	$\|V\|$	$\|E\|$	*BKV*	*Best*	*SR*	*Avg*	$t(s)$	*Best*	$t(s)$	*Best*	$t(s)$	*Best*	$t(s)$	Δ_1	Δ_2	Δ_3
p06.col	16	38	565*	565*	20	565.0	0	565*	0.0	565*	0.4	565*	1.1	0	0	0
p07.col	24	92	3771*	3771*	20	3771.0	0.02	3771*	0.0	3771*	0.1	3771*	4.0	0	0	0
p08.col	24	92	4049*	4049*	20	4049.0	0.17	4049*	0.0	4049*	1.3	4049*	1.3	0	0	0
p09.col	25	100	3388*	3388*	16	3388.2	0.95	3388*	0.0	3388*	0.1	3388*	3.0	0	0	0
p10.col	16	32	3983*	3983*	20	3983.0	0.68	3983*	0.0	3983*	0.1	3983*	4.5	0	0	0
p11.col	18	48	3380*	3380*	20	3380.0	0.01	3380*	0.0	3380*	0.1	3380*	4.7	0	0	0
p12.col	26	90	657*	657*	20	657.0	0	657*	0.0	657*	0.1	657*	3.8	0	0	0
p13.col	34	160	3220*	3220*	17	3221.1	0.68	3220*	0.0	3225	2.3	3230	7.8	0	-5	-10
p14.col	31	110	3157*	3157*	20	3157.0	0	3157*	0.0	3157*	0.1	3157*	10.1	0	0	0
p15.col	34	136	341*	341*	20	341.0	1.81	341*	0.0	341*	0.1	341*	4.7	0	0	0
p16.col	34	134	2343*	2343*	20	2343.0	0.76	2343*	0.0	2343*	0.5	2343*	14.5	0	0	0
p17.col	37	161	3281*	3281*	7	3322.2	2.72	3281*	0.0	3281*	1.4	3281*	5.5	0	0	0
p18.col	35	143	3228*	3228*	20	3228.0	0.05	3228*	0.0	3228*	0.2	3228*	10.4	0	0	0
p19.col	36	156	3710*	3710*	20	3710.0	0.36	3710*	0.0	3710*	0.1	3710*	14.6	0	0	0
p20.col	37	142	1830*	1830*	13	1841.0	4.86	1830*	0.0	1830*	1.3	1860	20.0	0	0	-30
p21.col	38	155	3660*	3660*	19	3660.5	0.75	3660*	0.0	3660*	0.2	3660*	18.4	0	0	0
p22.col	38	154	1912*	1912*	18	1912.2	0.29	1912*	0.0	1912*	0.2	1912*	20.0	0	0	0
p23.col	44	204	3770*	3770*	3	3793.0	0.28	3770*	0.1	3770*	1.4	3810	21.4	0	0	-40
p24.col	34	104	661*	661*	20	661.0	0	661*	0.0	661*	0.1	661*	27.9	0	0	0
p25.col	36	120	504*	504*	20	504.0	0.28	504*	0.0	504*	0.1	504*	23.9	0	0	0
p26.col	37	131	520*	520*	20	520.0	0.11	520*	0.0	520*	0.1	520*	28.3	0	0	0
p27.col	44	174	216*	216*	20	216.0	0.08	216*	0.1	216*	0.2	216*	7.8	0	0	0
p28.col	44	174	1729*	1729*	14	1735.1	2.56	1729*	0.1	1729*	0.1	1729*	44.5	0	0	0
p29.col	53	254	3470*	3470*	20	3470.0	0.10	3470*	0.1	3470*	65.7	3470*	65.7	0	0	0
p30.col	60	317	4891*	4891*	20	4891.0	53.79	4891*	0.2	4891*	2.1	4891*	56.6	0	0	0
p31.col	47	179	620*	620*	20	620.0	3.69	620*	0.1	620*	0.1	620*	70.9	0	0	0
p32.col	51	221	2480*	2480*	20	2480.0	0.35	2480*	0.1	2480*	0.0	2480*	70.9	0	0	0
p33.col	56	258	3018*	3018*	7	3029.7	0.43	3018*	0.3	3018*	0.1	3018*	62.3	0	0	0
p34.col	74	421	1980*	1980*	19	1980.5	3.05	1980*	0.6	1980*	0.1	1980*	131.9	0	0	0
p35.col	86	566	2140*	2140*	15	2145.0	4.48	2140*	0.6	2140*	0.1	2140*	135.0	0	0	0
p36.col	101	798	7210*	7210*	12	7385.0	0.13	7210*	1.4	7210*	0.1	7210*	163.1	0	0	0
p38.col	87	537	2130*	2130*	1	2139.5	9.54	2130*	1.2	2130*	0.4	2130*	231.8	0	0	0
p40.col	86	497	4984*	4984*	1	5016.6	5.05	4984*	1.0	4984*	0.2	4984*	224.2	0	0	0
p41.col	116	900	2688*	2688*	2	2688.1	0.10	2688*	3.2	2688*	0.1	2688*	313.7	0	0	0
p42.col	138	1186	2466*	2466*	4	2671.2	930.96	2466*	3.2	2509	2.8	2480	405.8	0	-43	-14
#Better				0/35				0/35		0/35		0/35				
#Equal				35/35				35/35		33/35		31/35				
#Worse				0/35				0/35		2/35		4/35				
p-value				-				-		1.6e-1		4.6e-2				

Δ_1). AFISA also dominates the 2_$Phase$ algorithm on the 41 instances tested by both algorithms, by obtaining better results for 13 instances (see negative entries in column Δ_2) and the same results for the remaining 28 instances. The small p-values (< 0.05) indicates that there is a significant difference between our best results and those of the two reference algorithms $MWSS$ (p-value=2.5e-2) and 2_$Phase$ (p-value=3.1e-4).

Table 3 reports the comparative results on the first set of 35 pxx instances from matrix-decomposition problems with one more reference algorithm ($GRASP$) [28]. One observes that AFISA performs very well on these pxx instances. AFISA attains always the known optimal values of $MWSS$, while 2_$Phase$ and $GRASP$ miss 2 and 4 instances respectively. The difference between AFISA and 2_$Phase$ is however not statistically significant (p-value of 1.6e-1), while the difference between AFISA and $GRASP$ is significant with a p-value of 4.6e-2.

Table 4 shows the results of AFISA on the second set of 30 rxx instances from matrix-decomposition problems, along with those of $MWSS$ and $GRASP$. Table 4 indicates that AFISA finds the optimal solutions for 26 out of the 30 rxx instances, which were previously obtained by the exact algorithm $MWSS$.

Table 4
Comparative results of AFISA with state-of-the-art algorithms on the 30 rxx benchmark instances.

				AFISA				$MWSS$[5]		$GRASP$[28]			
Instance	$\|V\|$	$\|E\|$	BKV	*Best*	*SR*	*Avg*	$t(s)$	*Best*	$t(s)$	*Best*	$t(s)$	Δ_1	Δ_2
r01.col	144	1280	6724*	6724*	8	6727.8	49.57	6724*	17.6	6724*	887	0	0
r02.col	142	1246	6771*	6771*	3	6780.6	85.33	6771*	11.3	6771*	1041	0	0
r03.col	139	1188	6473*	6473*	10	6490.8	190.19	6473*	10.4	6475	966	0	-2
r04.col	151	1406	6342*	6342*	1	6403.2	467.37	6342*	11.4	6342*	989	0	0
r05.col	142	1266	6408*	6408*	1	6466.3	71.68	6408*	12.3	6409	904	0	-1
r06.col	148	1381	7550*	7550*	4	7555.9	29.24	7550*	10.3	7550*	899	0	0
r07.col	141	1253	6889*	6889*	3	7555.9	34.76	6889*	13.7	6889*	6889	0	0
r08.col	138	1191	6057*	6057*	1	6080.3	311.66	6057*	4.1	6076	810	0	-19
r09.col	129	1027	6358*	6358*	1	6393.8	395.24	6358*	9.5	6424	868	0	-66
r10.col	150	1409	6508*	6508*	1	6519.3	461.98	6508*	13.1	6525	1048	0	-17
r11.col	208	2247	7654*	7654*	1	7710.6	259.25	7654*	57.2	7669	2423	0	-15
r12.col	199	2055	7690*	7691	1	7710.4	9542.18	7690*	53.7	7691	2267	1	0
r13.col	217	2449	7500*	7521	1	7558.3	619.53	7500*	105.2	7524	2365	21	-3
r14.col	214	2387	8254*	8254*	1	8283.9	8044.07	8254*	65.3	8254*	2342	0	0
r15.col	198	2055	8021*	8021*	1	8126.8	2559.06	8021*	25.1	8021*	2395	0	0
r16.col	188	1861	7755*	7755*	2	7789.2	195.53	7755*	26.7	7755*	2696	0	0
r17.col	213	2392	7979*	7979*	2	8030.3	855.38	7979*	79.3	8025	3175	0	-46
r18.col	200	2079	7232*	7232*	1	7278.9	868.19	7232*	58.2	7232*	1902	0	0
r19.col	185	1803	6826*	6840	1	6868.1	395.5	6826*	32.7	6858	2082	14	-18
r20.col	217	2447	8023*	8023*	1	8102.0	1028.5	8023*	104.1	8027	3452	0	-4
r21.col	281	3554	9284*	9284*	1	9384.5	4588.72	9284*	390.0	9287	4948	0	-3
r22.col	285	3684	8887*	8887*	1	8959.3	12911	8887*	302.6	8887*	5603	0	0
r23.col	288	3732	9136*	9136*	1	9267.9	3251.96	9136*	375.3	9145	5887	0	-9
r24.col	269	3284	8464*	8464*	1	8572.9	13142.6	8464*	201.7	8464*	4997	0	0
r25.col	266	3177	8426*	8468	1	8560.8	874.75	8426*	225.6	8504	5139	42	-36
r26.col	284	3629	8819*	8819*	1	8927.9	14225.1	8819*	439.6	8819*	5462	0	0
r27.col	259	3019	7975*	7975*	1	8019.7	14074.9	7975*	248.1	7975*	5064	0	0
r28.col	288	3765	9407*	9407*	1	9599.4	8691.00	9407*	222.7	9407*	5874	0	0
r29.col	281	3553	8693*	8693*	1	8743.7	7613.14	8693*	388.0	8693*	4923	0	0
r30.col	301	4122	9816*	9816*	1	10003.2	8838.59	9816*	346.9	9816*	6145	0	0
#Better				0/30				0/30		0/30			
#Equal				26/30				30/30		16/30			
#Worse				4/30				0/30		14/30			
p-value				-				4.6e-2		3.2e-4			

Among the four cases where AFISA misses the optimal solution, the gap to the optimal value is no more than 0.498% (instance $r25$). Compared to the reference heuristic algorithm $GRASP$, AFISA (Column 4) dominates $GRASP$ by attaining 13 better and 17 equal results. The p-value of 4.6e-2 between AFISA and $MWSS$ indicates that there is a slight difference and the p-value of 3.2e-4 between AFISA and $GRASP$ indicates that there is significant difference between the results of AFISA and $GRASP$.

Notice that according to [25], 2_$Phase$ is able to find (in comparable computing times) 3 solutions better than those of $GRASP$ reported in [28], 9 solutions of equal quality and 18 worse solutions. However the detailed results of 2_$Phase$ on the rxx instances are not available.

Finally, Table 5 summarizes our results on the set of 50 additional (larger) DIMACS/COLOR instances. Since these instances are not tested previously by any WVCP method, we use the general MIP solver CPLEX (version 12.6) as our reference method. We run CPLEX, with a cutoff limit of one hour, to solve the 0/1 ILP model shown in Appendix A. Entries with $-$ in Table 5 indicate that no feasible solution is found by CPLEX within the time limit and this happens for 43 out of the 50 instances. The last column (Δ) indicates the difference between our best result and the upper bound of CPLEX (a negative value implies thus a better result). We observe that only one instance can be

Table 5
Comparative results of FISA with CPLEX on the additional set of 50 larger DIMACS/COLOR instances.

			AFISA				$CPLEX$[5]			
Instance	$\|V\|$	$\|E\|$	*Best*	*SR*	*Avg*	$t(s)$	UB	LB	*status*	Δ
miles250.col	128	387	102*	8	102.7	56.61	102*	102*	Optimal	0
miles500.col	128	1170	260	1	261.3	48.46	-	-	-	-
miles1000.col	128	3216	432	1	444.7	480.02	437	31.692	Feasible	-5
miles1500.col	128	5198	587	1	644.3	32.31	-	-	-	-
mulsol.i.5.col	186	3973	367.0	20	367.0	416.91	360	109.034	Feasible	7
queen10_10.col	100	2940	166	3	169.2	68.43	-	-	-	-
queen11_11.col	121	3960	178	1	182.3	55.24	-	-	-	-
queen12_12.col	144	5192	194	1	198.6	92.7	208	47.000	Feasible	-14
queen13_13.col	169	6656	204	2	207.5	199.85	-	-	-	-
queen14_14.col	196	8372	224	2	227.4	360.05	316	23.000	Feasible	-92
queen15_15.col	225	10360	237	1	241.2	183.44	-	-	-	-
queen16_16.col	256	12640	253	2	256.3	300.85	365	22.033	Feasible	-112
zeroin.i.1.col	211	4100	518	20	518.0	0	-	-	-	-
zeroin.i.2.col	211	3541	336	3	337.6	440.84	300	26.103	Feasible	36
zeroin.i.3.col	206	3540	299	2	301.7	139.64	-	-	-	-
DSJC250.1.col	250	3218	140	1	141.9	48.94	-	-	-	-
DSJC250.5.col	250	15668	415	1	428.1	269.23	-	-	-	-
DSJC250.9.col	250	55794	925	1	942.7	856.25	-	-	-	-
DSJC500.1.col	500	12458	210	1	215.6	426.64	-	-	-	-
DSJC500.5.col	500	125248	778	1	845.1	159.27	-	-	-	-
DSJC500.9.col	500	224874	1790	1	1854.5	831.07	-	-	-	-
DSJR500.1.col	500	3555	169	1	175.4	458.86	-	-	-	-
DSJC1000.1.col	1000	99258	359	4	362.9	430.54	-	-	-	-
DSJC1000.5.col	1000	499652	1357	1	1430.9	371.65	-	-	-	-
DSJC1000.9.col	1000	898898	3166	1	3231.0	490.2	-	-	-	-
inithx.i.1.col	864	18707	587	5	587.9	527.46	-	-	-	-
inithx.i.2.col	645	13979	341	8	341.6	0.03	-	-	-	-
inithx.i.3.col	621	13969	352	11	355.6	0.01	-	-	-	-
le450_15a.col	450	8168	241	1	247.1	288.37	-	-	-	-
le450_15b.col	450	8169	239	2	245.1	368.35	-	-	-	-
le450_15c.col	450	16680	313	1	320.8	432.95	-	-	-	-
le450_15d.col	450	16750	306	2	314.1	113.7	-	-	-	-
le450_25a.col	450	8260	317	1	329.9	362.26	-	-	-	-
le450_25b.col	450	8263	318	1	325.8	285.88	-	-	-	-
le450_25c.col	450	17343	378	1	387.9	359.37	-	-	-	-
le450_25d.col	450	17345	375	1	385.3	254.76	-	-	-	-
flat1000_50_0.col	1000	245000	1289	1	1315.7	981.77	-	-	-	-
flat1000_60_0.col	1000	245830	1338	1	1354	201.98	-	-	-	-
flat1000_76_0.col	1000	246708	1314	1	1337.6	2396.63	-	-	-	-
C2000.5.col	2000	999836	2400	1	2425.1	3133.97	-	-	-	-
C2000.9.col	2000	1799532	6228	1	6284.0	2798.3	-	-	-	-
latin_square_10.col	900	307350	1690	1	1900.0	780.26	-	-	-	-
wap01a.col	2368	110871	638	1	653.1	1133.51	-	-	-	-
wap02a.col	2464	111742	637	1	638.1	3270.46	-	-	-	-
wap03a.col	4730	286722	687	1	707.5	2901.51	-	-	-	-
wap04a.col	5231	294902	698	1	709.0	4.79	-	-	-	-
wap05a.col	905	43081	598	1	610.9	1574.52	-	-	-	-
wap06a.col	947	43571	599	1	607.6	65.32	-	-	-	-
wap07a.col	1809	103368	680	1	692.5	384.82	-	-	-	-
wap08a.col	1870	104176	663	1	673.4	2627.2	-	-	-	-
#Better			47/50				2/7			
#Equal			1/50				1/7			
#Worse			2/50				4/7			

solved to optimality by CPLEX within the one hour time limit. For the six instances for which CPLEX reaches a feasible solution, but fails to find the optimal solution, its upper bounds are worse than the bounds of AFISA in four cases. Due to the small number of instances solved by CPLEX, we omit the statistical test.

To sum up, this computational assessment indicates that AFISA performs very well on the four sets of benchmark instances. The new upper bounds (5 for the first set and the results for the fourth set) established by AFISA can serve as valuable reference values to evaluate future algorithms for the WVCP. Meanwhile, AFISA failed to attain the optimal values for four instances of

the third set, indicating there is room for further improvement. Finally, this experimental study confirms that like for many NP-hard problems, both exact and heuristic algorithms are complementary and can be used to solve problem instances of different sizes and different characteristics. These approaches can even be combined to create powerful hybrid algorithms.

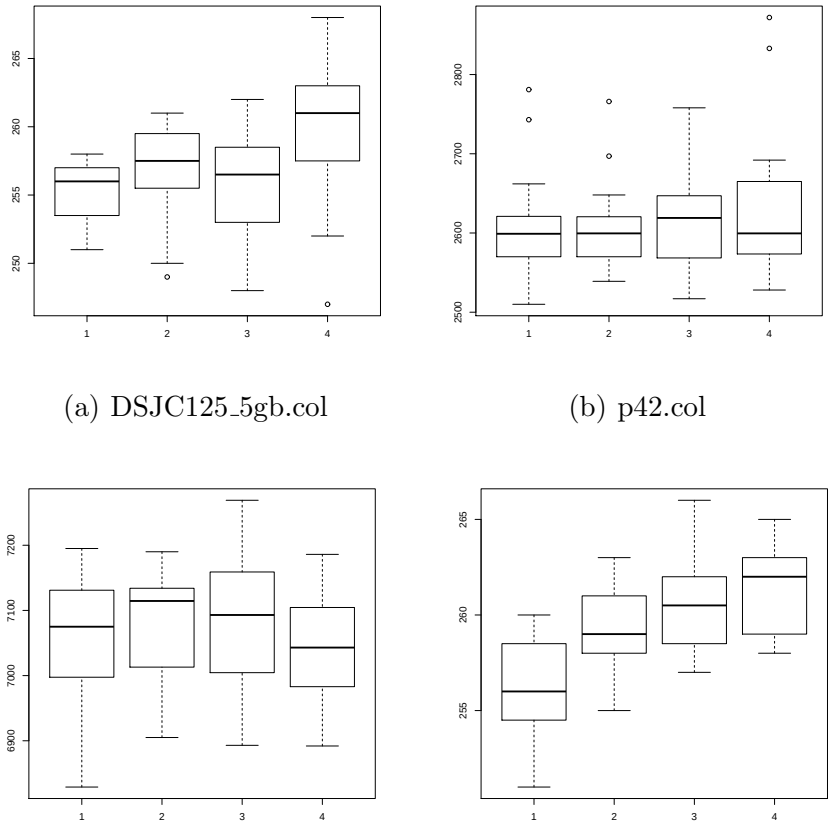
5 Analysis

This section performs additional experiments to analyze the benefits of three important ingredients of the proposed AFISA algorithm: the penalty coefficient of the extended evaluation function, the strategy of visiting both feasible and infeasible solutions and the perturbation strategy.

5.1 Impact of the penalty coefficient

AFISA uses the extended evaluation function F defined in Section 3.3 to explore both feasible and infeasible solutions. The oscillation between feasible and infeasible zones is adaptively controlled by increasing or decreasing the penalty coefficient φ (≥ 1). In this study, we analyze the impact of the increment/decrement value used to adjust φ and for this purpose, we test the following increment/decrement values: 1,2,3,4 (larger values make the search oscillate too much between feasible and infeasible zones).

Box and whisker plots of the results on 4 representative instances from the four benchmark sets (which are relatively difficult according to Tables 2-5) are shown in Figure 2, where the X-axis indicates the tested increment/decrement values and the Y-axis indicates the objective values. As a supplement, we also compute the p-value for each tested instance. The results are based on 20 independent runs for each instance with a cutoff time of 300 seconds per run. One observes that the performance of the AFISA algorithm is significantly influenced by the increment/decrement value on the instances DSJC125_5gb (p-value = 1.4e-2) and queen16_16 (p-value = 1.2e-4). This is less the case for the instances p42 (p-value = 2.0e-1) and r05 (p-value = 7.6e-1). Furthermore, the AFISA algorithm with the increment/decrement value of 1 gives the best performance compared to other values. This explains why we adopt 1 as the default increment/decrement value in this study. Finally, one notices that the outcome of this experiment remains coherent with the intuitive understanding that the search will go back and forth more frequently between feasible and infeasible regions with a large increment/decrement value than with a small value. This implies that a large increment/decrement value may make the search leave each newly discovered (feasible or infeasible) zone too early before



(a) DSJC125_5gb.col (b) p42.col

(c) r05.col (d) queen16_16.col

Fig. 2. Influence of the increment/decrement value of the penalty efficient

the search zone is sufficiently exploited.

5.2 Benefit of searching both feasible and infeasible solutions

To assess the strategy of oscillating between feasible and infeasible regions of the proposed algorithm, we create a algorithmic variant (called Tabu_Feasible) in which the search visits only feasible solutions. For this, we set the penalty coefficient φ of the extended evaluation function (Eq. (4), Section 3.3) to a large value in order to penalize strongly any infeasible solution. In our case, φ is set to the largest weight of vertices of the given graph. For this experiment, we select 38 instances that are relatively difficult according to the results reported in Tables 2-5, i.e., their best-known results cannot consistently be attained by all algorithms. We ran both algorithms 20 times to solve each selected instance with a cutoff time of 1 hour.

The comparative results of this experiment are presented in Table 6 with the

Table 6
Assessment of searching both feasible and infeasible solutions.

Instance	$\|V\|$	$\|E\|$	AFISA				*Tabu_Feasible*				Δ
			Best	*SR*	*Avg*	*t(s)*	*Best*	*SR*	*Avg*	*t(s)*	
DSJC1000.1.col	1000	99258	359	4	362.9	430.54	354	1	358.9	450.15	5
DSJC1000.5.col	1000	499652	1357	1	1430.9	371.65	1354	1	1371.3	84.75	3
DSJC1000.9.col	1000	898898	3166	1	3231.0	490.2	3166	1	3231.1	490.2	0
DSJC125_1g.col	125	736	23	2	24.0	3016.32	23	4	23.8	1467.63	0
DSJC125_1gb.col	125	736	90	1	92.5	402.57	91	1	94.6	30.09	-1
DSJC125_5g.col	125	3891	71	2	72.3	216.04	71	1	72.5	199.14	0
DSJC125_5gb.col	125	3891	243	1	250.2	369.34	246	3	250.9	292.08	-3
DSJC125_9g.col	125	6961	169	3	170.0	16	169	4	170.2	42.1	0
DSJC125_9gb.col	125	6961	604	3	605.0	443.96	606	2	608.7	18.24	-2
DSJC250.1.col	250	3218	140	1	141.9	48.94	140	4	142.1	57.37	0
DSJC250.5.col	250	15668	415	1	428.1	269.23	421	1	431.2	318.86	-6
DSJC250.9.col	250	55794	925	1	942.7	856.25	948	1	965.8	365.4	-23
DSJC500.1.col	500	12458	210	1	215.6	426.64	212	1	215.3	84.83	-2
DSJC500.5.col	500	125248	778	1	845.1	159.27	780	1	796.1	450.89	-2
DSJC500.9.col	500	224874	1790	1	1854.5	831.07	1791	1	1869.2	345.13	-1
DSJR500.1.col	500	3555	169	1	175.4	458.86	170	2	171.9	272.89	-1
GEOM100.col	100	547	65	20	65.0	0.81	65	20	65.0	9.86	0
miles1500.col	128	5198	587	1	644.3	32.31	797	4	798.6	0.13	-210
p13.col	34	160	3220	17	3221.1	0.68	3220	16	3221.5	0.83	0
p20.col	37	142	1830	13	1841	4.86	1830	16	1835.5	0.76	0
p42.col	138	1186	2466	4	2671.2	930.96	2646	1	2659.7	12.51	-180
queen12_12.col	144	5192	194	1	198.6	92.7	196	2	199.0	96.04	-2
queen14_14.col	196	8372	224	2	227.4	360.05	225	1	227.7	129.47	-1
queen16_16.col	256	12640	253	2	256.3	300.85	250	1	254.8	14.15	3
r03.col	139	1188	6473	10	6490.8	190.19	6487	1	6536	119.3	-14
r05.col	142	1266	6408	1	6466.3	71.68	6495	1	6525.6	143.13	-87
R100_1g.col	100	509	21	1	22.0	113.77	21	3	21.9	533.65	0
R100_5g.col	100	2456	59	5	60.1	6.97	59	2	60.3	2.56	0
R100_5gb.col	100	2456	221	1	224.1	186.81	222	1	224.7	1362.52	-1
R100_9g.col	100	4438	141	15	141.3	21.36	141	11	141.5	2.43	0
R100_9gb.col	100	4438	518	1	549.3	1152.83	518	13	518.5	21.25	0
R50_5gb.col	50	612	135	14	135.3	3.72	135	18	135.3	0.11	0
R75_1g.col	70	251	18	12	18.4	10.96	18	10	18.5	32.89	0
R75_1gb.col	70	251	70	19	70.1	2.46	70	17	70.3	6.58	0
R75_5g.col	75	1407	51	12	51.4	0.08	51	7	51.7	5.11	0
R75_5gb.col	75	1407	186	2	189.0	19.42	186	2	189.2	31.93	0
R75_9gb.col	75	2513	396	12	396.4	145.89	396	9	396.7	4.35	0
zeroin.i.2.col	211	3541	336	3	337.6	440.84	336	13	336.4	0.91	0
#Better			16/38		23/38		3/38		13/38		
#Equal			19/38		2/38		19/38		2/38		
#Worse			3/38		13/38		16/38		23/38		
p_value			-		-		2.9e-3		4.6e-2		

same information as before. The rows #Better/#Equal/#Worse indicate the number of instances for which each algorithm attains a better, equal and worse result compared to the other algorithm in terms of the best objective value found. The last column indicates the difference between the best results of AFISA and Tabu_Feasible. We observe that even if both algorithms obtain 19 equal results, AFISA achieves 16 better results (against 3 for Tabu_Feasible). The small p-values (< 0.05) in terms of *Best* and *Avg* confirm the statistical significance of the reported differences between AFISA and Tabu_Feasible. This experiment shows that searching both feasible and infeasible solutions enables the algorithm to reach a better performance.

5.3 Impact of the perturbation operation

As shown in Section 3.5, the proposed algorithm uses a perturbation strategy as an additional means of diversification. To assess this strategy, we compare AFISA with a AFISA variant (denoted as AFISA$^-$) where the perturbation

Table 7
Assessment of the perturbation strategy.

Instance	$\|V\|$	$\|E\|$	AFISA				AFISA$^-$				Δ
			Best	*SR*	*Avg*	$t(s)$	*Best*	*SR*	*Avg*	$t(s)$	
DSJC1000.1.col	1000	99258	359	4	362.9	430.54	384	10	385.5	0.08	-25
DSJC1000.5.col	1000	499652	1357	1	1430.9	371.65	1427	10	1434.5	0.07	-70
DSJC1000.9.col	1000	898898	3166	1	3231.0	490.2	3291	10	3302.5	0.04	-125
DSJC125_1g.col	125	736	23	2	24.0	3016.32	33	6	33.7	0	-10
DSJC125_1gb.col	125	736	90	1	92.5	402.57	120	13	122.8	0	-30
DSJC125_5g.col	125	3891	71	2	72.3	216.04	87	14	88.5	0	-16
DSJC125_5gb.col	125	3891	243	1	250.2	369.34	293	20	293.0	0.01	-50
DSJC125_9g.col	125	6961	169	3	170.0	16	186	12	188.8	0	-17
DSJC125_9gb.col	125	6961	604	3	605.0	443.96	652	10	668.0	0	-48
DSJC250.1.col	250	3218	140	1	141.9	48.94	168	10	170.5	0.01	-28
DSJC250.5.col	250	15668	415	1	428.1	269.23	472	8	476.5	0	-57
DSJC250.9.col	250	55794	925	1	942.7	856.25	1039	11	1060.0	0	-114
DSJC500.1.col	500	12458	210	1	215.6	426.64	242	10	243.0	0.02	-32
DSJC500.5.col	500	125248	778	1	845.1	159.27	853	8	856.0	0.03	-75
DSJC500.9.col	500	224874	1790	1	1854.5	831.07	1937	20	1937.0	0.01	-147
DSJR500.1.col	500	3555	169	1	175.4	458.86	184	10	191.5	0.01	-15
GEOM100.col	100	547	65	20	65.0	0.81	74	9	74.6	0.01	-9
miles1500.col	128	5198	587	1	644.3	32.31	799	8	809.8	0	-212
p13.col	34	160	3220	17	3221.1	0.68	3568	7	3764.3	0	-348
p20.col	37	142	1830	13	1841.0	4.86	2270	20	2270	0.01	-440
p42.col	138	1186	2466	4	2671.2	930.96	2880	12	2887.2	0	-414
queen12_12.col	144	5192	194	1	198.6	92.7	229	9	230.0	0	-35
queen14_14.col	196	8372	224	2	227.4	360.05	254	14	255.5	0	-30
queen16_16.col	256	12640	253	2	256.3	300.85	280	9	282.2	0	-27
r03.col	139	1188	6473	10	6490.8	190.19	6603	1	6687.3	376.19	-130
r05.col	142	1266	6408	1	6466.3	71.68	6495	2	6637.7	0	-87
R100_1g.col	100	509	21	1	22.0	113.77	30	8	30.6	0	-9
R100_5g.col	100	2456	59	5	60.1	6.97	72	10	73.5	0	-13
R100_5gb.col	100	2456	221	1	224.1	186.81	260	1	264	0	-39
R100_9g.col	100	4438	141	15	141.3	21.36	153	9	157.4	0	-12
R100_9gb.col	100	4438	518	1	549.3	1152.83	548	9	551.5	0	-30
R50_5gb.col	50	612	135	14	135.3	3.72	152	8	156.8	0	-17
R75_1g.col	70	251	18	12	18.4	10.96	24	7	25.2	0	-6
R75_1gb.col	70	251	70	19	70.1	2.46	88	11	89.4	0	-18
R75_5g.col	75	1407	51	12	51.4	0.08	63	20	63.0	0	-12
R75_5gb.col	75	1407	186	2	189.0	19.42	231	11	233.7	0	-45
R75_9gb.col	75	2513	396	12	396.4	145.89	425	12	428.2	0.01	-29
zeroin.i.2.col	211	3541	336	3	337.6	440.84	337	13	337.7	0	-1
#Better			38/38		38/38		0/38		0/38		
#Equal			0/38		0/38		0/38		0/38		
#Worse			0/38		0/38		38/38		38/38		
p-value			-		-		7.1e-10		7.1e-10		

strategy is disabled (i.e., by removing line 22 in Algorithm 1). This experiment is based on the 38 instances used in Section 5.2. We ran 20 times both algorithms to solve each selected instance with a cutoff time of 1 hour.

The results of this experiment are shown in Table 7 with the same statistics as before. We observe that AFISA dominates, in terms of *Best* and *Avg*, the AFISA$^-$ variant by obtaining a better result for each instance. The small p-values confirm the dominance of AFISA over AFISA$^-$. This experiment demonstrates the interest of the adopted perturbation strategy as a meaningful means of diversification that enables the algorithm to better explore the search space.

6 Conclusions

The weighted vertex coloring problem (WVCP) considered in this work is a generalization of the conventional vertex coloring problem with a number of

practical applications. Motivated by the observation that existing methods limit their search to feasible solutions, we investigated for the first time the benefit of examining both feasible and infeasible solutions for solving the problem. The resulting AFISA algorithm oscillates between feasible and infeasible search zones guided by an extended evaluation function that combines the initial objective function and an adaptive penalty function. To explore feasible and infeasible spaces, we introduced a tabu search procedure enhanced by a dedicated perturbation strategy to escape local optima traps.

We assessed the performance of the AFISA algorithm on three sets of 111 instances commonly tested in the literature and an additional set of 50 (large) DIMACS and COLOR instances initially proposed for graph coloring problems. We presented 5 improved best results (new upper bounds) among the 111 instances of the literature and the first upper bounds for the new set of 50 instances. These new bounds can serve as valuable references to assess future WVCP algorithms and might be used by a branch-and-bound algorithm as high-quality initial bounds. This study demonstrates the benefit of the search strategy examining both feasible and infeasible solutions for solving the WVCP. The computational results on different types of benchmark instances also confirm that exact and heuristic algorithms are complementary solution approaches that can be advantageously employed to handle instances of different sizes with particular features.

For future work, several directions could be followed. First, other penalty-based evaluation functions could be devised to enable a better strategic oscillation between feasible and infeasible spaces. Second, other neighborhoods (rather than the one-move based neighborhood used in this work) can be sought to further improve the performance of the search algorithm. Third, the proposed algorithm could be advantageously integrated into a hybrid population-based method (e.g., memetic search, path-linking) as a key intensification component. Fourth, this work uses tabu search to explore candidate solutions. Other meta-heuristics can be investigated to ensure this task while reusing most algorithmic components of AFISA. Finally, few exact algorithms are available for the WVCP, there is thus much room for research in this direction. In this context, AFISA could be used to generate high-quality initial bounds or to obtain upper bound estimations of subproblems during the search process.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive comments and suggestions, Prof. David Lesaint for his useful comments on the work, and Prof. André Rossi for suggesting the symmetry breaking

constraint used in the ILP model presented in Appendix A. Support (PhD scholarship, 2015-2019) for Wen Sun from the China Scholarship Council is also acknowledged.

References

[1] Richard S. Barr, Bruce L. Golden, James P. Kelly, Mauricio G.C. Resende, William R. StewartJr. Designing and reporting on computational experiments with heuristic methods. *Journal of Heuristics*, 1(1):9-32, 1995.

[2] Una Benlic and Jin-Kao Hao. Breakout local search for the quadratic assignment problem. *Applied Mathematics and Computation*, 219(9):4800-4815, 2013.

[3] Alberto Caprara, Matteo Fischetti and Paolo Toth. A heuristic method for the set covering problem. *Operations Research*, 47(5):730-743, 1999.

[4] Yuning Chen, Jin-Kao Hao and Fred Glover. A hybrid metaheuristic approach for the capacitated arc routing problem. *European Journal of Operational Research*, 253(1):25-39, 2016

[5] Denis Cornaz, Fabio Furini and Enrico Malaguti. Solving vertex coloring problems as maximum weight stable set problems. *Discrete Applied Mathematics*, 217:151-162, 2017.

[6] Fabio Furini and Enrico Malaguti. Exact weighted vertex coloring via branch-and-price. *Discrete Optimization*, 9(2):130-136, 2012.

[7] Philippe Galinier and Jin-Kao Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379-397, 1999.

[8] Philippe Galinier and Alain Hertz. A survey of local search methods for graph coloring. *Computers & Operations Research*, 33(9):2547–2562, 2006.

[9] Philippe Galinier, Jean-Philippe Hamiez, Jin-Kao Hao and Daniel Porumbel. Recent advances in graph vertex coloring. In Zelinka I., Snášel V., Abraham A. (eds) Handbook of Optimization. Intelligent Systems Reference Library, vol 38. Springer, Berlin, Heidelberg, 2013.

[10] Michael R. Garey and David S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. San Francisco, Calif.: W. H. Freeman and Co, 1979.

[11] Haris Gavranovic and Gerd Finke. Graph partitioning and set covering for the optimal design of a production system in the metal industry. *IFAC Proceedings Volumes*, 33(17):603-608, 2000.

[12] Fred Glover. Tabu search – part I. *ORSA Journal on Computing*, 1(3):190-206, 1989.

[13] Fred Glover. Tabu search – part II. *ORSA Journal on Computing*, 2(1):4-32, 1990.

[14] Fred Glover and Manuel Laguna. Tabu search. Kluwer Publisher, 1997.

[15] Fred Glover and Jin-Kao Hao. The case for strategic oscillation. *Annals of Operations Research*, 183(1): 163-173, 2011.

[16] Sana Ben Hamida and Marc Schoenauer. An Adaptive Algorithm for constrained optimization problems. Proceedings of PPSN VI, Paris, France, September 18–20, Lecture Notes in Computer Science 1917, pages 529-538, 2000.

[17] Alain Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345-351, 1987.

[18] Dorit S. Hochbaum and Dan Landy. Scheduling semiconductor burn-in operations to minimize total flowtime. *Operations Research*, 45(6):874–885, 1997.

[19] Yan Jin and Jin-Kao Hao. Hybrid evolutionary search for the minimum sum coloring problem of graphs. *Information Sciences*, 352-353:15-34, 2016.

[20] Xiangjing Lai, Jin-Kao Hao, Fred Glover, and Zhipeng Lü. A two-phase tabu-evolutionary algorithm for the 0–1 multidimensional knapsack problem. *Information Sciences*, 436-437:282–301, 2018.

[21] Xiangjing Lai, Jin-Kao Hao and Fred Glover. Backtracking based iterated tabu search for equitable coloring. *Engineering Applications of Artificial Intelligence*, 46:269-278, 2015.

[22] Chih-Hao Lin. A rough penalty genetic algorithm for constrained optimization. *Information Sciences*, 241:119-137, 2013.

[23] hyd M.R. Lewis. A guide to graph colouring: algorithms and applications. Springer International Publishing, 2016.

[24] Enrico Malaguti. The vertex coloring problem and its generalizations. *4OR: A Quarterly Journal of Operations Research*, 7(1):101-104, 2009.

[25] Enrico Malaguti, Michele Monaci and Paolo Toth. Models and heuristic algorithms for a weighted vertex coloring problem. *Journal of Heuristics*, 15(5):503-526, 2009.

[26] Enrico Malaguti and Paolo Toth. A survey on vertex coloring problems. *International Transactions in Operational Research*, 17(1):1-34, 2010.

[27] Anna Martínez-Gavara, Dario Landa-Silva, Vicente Campos, and Rafael Martí. Randomized heuristics for the capacitated clustering problem. *Information Sciences*, 417:154-168, 2017.

[28] Marcelo Prais and Celso C Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, 12(3):164-176, 2000.

[29] Chao Qian, Yang Yu, and Zhi-Hua Zhou. Subset Selection by Pareto Optimization. In: Advances in Neural Information Processing Systems 28 (NIPS'15), Montreal, Canada, 2015, pages 1765-1773.

[30] Celso Carneiro Ribeiro, Michel Minoux and Manoel Camillo Penna. An optimal column-generation-with-ranking algorithm for very large scale set partitioning problems in traffic assignment. *European Journal of Operational Research*, 41(2):232-239, 1989.

[31] Wen Sun, Jin-Kao Hao, Xiangjing Lai and Qinghua Wu. On feasible and infeasible search for equitable graph coloring. Proceedings of GECCO-2017, Berlin, Germany, July 15-19, pages 369-376, 2017.

[32] Wenyu Wang, Jin-Kao Hao and Qinghua Wu. Tabu search with feasible and infeasible searches for equitable coloring. *Engineering Applications of Artificial Intelligence*, 71:1-14, 2018.

A ILP formulation of the WVCP

In this section, we describe the 0-1 integer linear programming (ILP) model for the WVCP proposed in [25], extended with a symmetry breaking constraint. This model is used for the computational experiment presented in Section 4.3 (Table 5). First, we define the following decision variables:

- Let $s = \{V_1, V_2, \ldots, V_k\}$ be a candidate solution.
- $x_{vk} \in \{0, 1\}$ is a binary decision variable that is equal to 1 if and only if vertex v is part of set V_k, 0 otherwise for all $v \in V$ and for all $k \in \{1, \ldots, N\}$.
- W_k is a positive decision variable that is equal to the weight of stable V_k, for all $k \in \{1, \ldots, N\}$.

We obtain the following ILP model for the WVCP:

$$f(s) = min \sum_{k=1}^{N} W_k \tag{A.1}$$

$$\begin{cases} \sum_{k=1}^{N} x_{vk} = 1 & \forall v \in V \\ x_{vk} + x_{uk} \leq 1 & \forall (u, v) \in E, \forall k \in \{1, \ldots, N\} \\ x_{vk} w_v \leq W_k & \forall (v, k) \in V \times \{1, \ldots, N\} \\ W_k \geq W_{k+1} & \forall k \in \{1, \ldots, N-1\} \\ W_k \geq 0 & \forall k \in \{1, \ldots, N\} \\ x_{vk} \in \{0, 1\} & \forall (v, k) \in V \times \{1, \ldots, N\} \end{cases}$$

The objective function (A.1) is to minimize the sum of the weights of the stables used. The first constraint ensures that each vertex belongs to exactly one set. The second constraint enforces that the same color cannot be assigned to adjacent vertices. The third constraint sets W_k to the weight of the maximum weight vertex in the k-th set. The fourth constraint partially breaks symmetry by enforcing that the stables are ordered by decreasing order of their sizes.