A solution-driven multilevel approach for graph coloring

Wen Sun [a], Jin-Kao Hao [b,*], Yuhao Zang [a], Xiangjing Lai [c]

[a] *School of Cyber Science and Engineering, Southeast University, 2 Road Southeast University, 211189 Nanjing, China*

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

[c] *Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, 66 Road Xinmofan, 210023 Nanjing, China*

Applied Soft Computing, February 2021
https://doi.org/10.1016/j.asoc.2021.107174

Abstract

Graph coloring is one of the most studied NP-hard problems with a wide range of applications. In this work, the first solution-driven multilevel algorithm for this computationally challenging problem is investigated. Following the general idea of multilevel optimization, the proposed algorithm combines an original solution-driven coarsening procedure with an uncoarsening procedure as well as an effective refinement procedure. The algorithm is assessed on 47 popular DIMACS and COLOR benchmark graphs, and compared with 13 state-of-the-art coloring methods in the literature. We close one large graph (wap01a.col) by providing its chromatic number for the first time. Impacts of the key ingredients of the algorithm are also investigated.

Keywords: Multilevel optimization; heuristics; *k*-graph coloring; tabu search.

1 Introduction

Given an undirected graph $G = (V, E)$ with a vertex set $V = \{1, 2, \ldots, n\}$ and an edge set $E \subset V \times V$, a k-coloring of G is a mapping $S : V \to \{1, \ldots, k\}$. If the k-coloring verifies the condition $S(u) \neq S(v)$ for all edges

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

(u, v) in E (this is called the coloring constraint), then it is a legal k-coloring. Otherwise, the k-coloring is said to be illegal or conflicting. Equivalently, a k-coloring can be defined by a partition of the vertex V into k subsets (also called color classes) $V_1, V_2, \cdots, V_k$, such that $\cup_{i=1}^{k} V_i = V$ and $V_i \cap V_j = \emptyset$. A legal k-coloring implies that if $u, v \in V_i$ $(i = 1, \ldots, k)$, then $(u, v) \notin E$.

The graph k-coloring problem (k-GCP) is to determine if a legal k-coloring of G exists for a given k. The classical graph coloring problem (GCP) is to find the minimum integer k (chromatic number $\chi(G)$) for which a legal k-coloring of G exists. k-GCP (decision problem) is known to be NP-complete while GCP (optimization problem) is NP-hard [1]. In addition to its theoretical importance in graph theory, graph coloring has numerous applications, such as register allocation [2], timetabling [3], frequency assignment [4] and scheduling [5].

To enable more applications to be formulated, several graph coloring variants and generalizations have been introduced, including for instance equitable coloring (ECP), weighted vertex coloring (WVCP), bandwidth coloring (BCP), and minimum sum coloring (MSCP) [6,7]. The ECP imposes an equity constraint on the sizes of color classes of a coloring and requires that the color classes differ in size by one unit at most. The ECP is applied in garbage collection [8], memory allocation in parallel systems [9], scheduling computer and manufacturing processes with load balancing [10], and timetabling [11]. The WVCP of a vertex weighted graph is to partition the vertex set into k disjoint color classes such that the sum of the costs of these color classes is minimized, where the cost of a color class is given by the maximum weight of a vertex (representative) in that class. The WVCP degenerates to the GCP when the vertices have all the unit weight. From a practical perspective, the WVCP arises naturally in the context of buffer management in operating systems [12], batch scheduling [13] and manufacturing [14]. As another generalization of the GCP, the BCP imposes a stronger coloring constraint and requires that the difference of the colors assigned to any two adjacent vertices (u, v) in E must not be smaller than a given value $d_{uv} \geq 1$ (the GCP corresponds to the case when $d_{uv} = 1$). The BCP is notable for its applicability especially in the area of frequency assignment in mobile networks [15]. Finally, the MSCP involves finding a legal k-coloring that minimizes the sum of the colors (i.e., an integer in $\{1, \ldots, k\}$) assigned to the vertices. Typical applications of the MSCP include scheduling, resource allocation, and VLSI design [16].

Given the intrinsic difficulty of graph coloring, many efforts have been devoted to developing approximate or heuristic algorithms. Comprehensive reviews on graph coloring algorithms can be found in [17,18,19]. Representative examples include, but are not limited to, the following algorithms: constructive algorithms (DSATUR [20], RLF [5]), local search algorithms

(TabuCol [21], IGrAl [22], VSS [23], PLSCOL [24]) and population-based hybrid algorithms (HEA [25], AMA [26], MMT [27], MACOL [28], Evo-Div [29], QACOL [30], HEAD [31]). It is worth mentioning that the population-based hybrid algorithms represent the state-of-the-art methods, which often typically employ local search (e.g., TabuCol [21]) as their key local optimization component.

Meanwhile, we notice that even if various approaches have been studied for graph coloring, the well-known multilevel (also called multiscale) optimization approach [32,33] has attracted very little attention in the literature for this important problem. The basic idea of the multilevel approach for graph optimization problems is to reduce the complexity of the initial (large) graph by creating a sequence of intermediate smaller graphs (coarsening phase), solving the smallest graph (initial solution phase) and projecting the solution back to the initial graph through the intermediate graphs (uncoarsening phase). This approach has been applied with great success to many difficult problems including graph partitioning [32,34,35], graph drawing [36], manifold learning [37], graph clustering [38], segmentation [39], VLSI design [40], and vertex separator problem [41].

To our knowledge, there is only one early attempt [42] on using the multilevel approach to solve the graph coloring problem. However, the reported results are disappointing given that it is just a simple and straightforward application of the general multilevel method with no dedicated strategies. In this work, we revisit the multilevel approach for graph coloring by adopting more advanced strategies and techniques in its key components. We summarize the main contributions as follows.

First, we propose the solution-driven multilevel algorithm (SDMA) (Section 2) that for the first time integrates a special solution-guided coarsening strategy with an effective local refinement based on a weight tabu search procedure within the multilevel paradigm. To reinforce its search capacity, SDMA also uses a perturbation-based strategy to iterate the underlying multilevel process.

Second, we provide extensive computational assessments on well-known DIMACS benchmark graphs and perform comparisons with state-of-the-art coloring algorithms (Section 3). We also investigate the impacts of key components of the proposed algorithm on its performance (Section 4).

In the next section, we present the proposed approach as well as its components.

2 Solution-driven multilevel algorithm for GCP

As explained in [17], GCP can be approximated by solving a series of k-GCP with decreasing k values where k is the number of available colors. Specifically, we start with a large k (e.g., the number of vertices of the input graph G), and try to find a legal k-coloring. Once such a solution is found (k gives an upper bound of the chromatic number of G), we decrease k by one and then try to solve the new k-GCP instance. This process is repeated until no legal k-coloring can be found. Therefore, the proposed SDMA algorithm is designed to solve the k-coloring problem.

For a given graph $G = (V, E)$, the algorithm explores the search space Ω_k that is composed of all legal (non-conflicting) and illegal (conflicting) candidate k-colorings, i.e., $\Omega_k = \{\{V_1, V_2, \cdots, V_k\} : \cup_{i=1}^{k} V_i = V, V_i \cap V_j = \emptyset\}$. For a candidate solution $S \in \Omega_k$, its quality is assessed by the objective function $f(S) = \sum_{i=1}^{k} |C(V_i)|$, where $C(V_i)$ is the set of conflicting edges in color class V_i (a conflicting edge means that its endpoints are in the same color class, or equivalently, receive the same color). In other words, this function counts the conflicting edges in the color classes of S. Thus, a solution for G is a non-conflicting coloring S with $f(S) = 0$. The objective of SDMA is then to minimize f to find a non-conflicting k-coloring.

2.1 General approach

The proposed SDMA approach (Algorithm 1) starts with an initial solution S_0 (line 3) for the input graph G_0. Then, SDMA iterates the multilevel optimization process (lines 4-15, m represents the current level) and a perturbation phase (lines 16-18) to find improved solutions.

(1) Initialization phase: SDMA constructs its initial solution (i.e., a k-coloring solution with as few conflicts as possible) in two steps: 1) create a k-coloring solution with as few conflicts as possible by the greedy DANGER algorithm of [43]; 2) improve the solution by the tabu search procedure of [25]. Basically, the tabu search procedure progressively reduces the number of coloring constraint violations by changing the color of a conflicting vertex once a time (see Section 2.3).

(2) Coarsening phase: The coarsening phase aims to transform the input graph G_0 into a hierarchy of monotonically smaller coarsened graphs by merging vertices. Specifically, let m be the current level, then the graph G_m is transformed into a smaller graph G_{m+1} based on the current solution (i.e., a candidate k-coloring for G_m). Given that the algorithm used to color each coarsened graph is a heuristic procedure (see Section 2.3), some ver-

Algorithm 1 The SDMA algorithm for solving k-GCP

```
1: Input: Graph $G_0 = (V_0, E_0)$, number of colors $k$, level limit $L$.
2: Output: the best $k$-coloring $S$ found
3: $S_0 \leftarrow Initialization(G_0)$
4: while Stopping condition is not meet do
5:    $m = 0$                                                   /*The lowest level*/
      /*Coarsening phase*/
6:    while $m < L$ do
7:       $(G_{m+1}, w_{m+1}) \leftarrow Coarsen(G_m, w_m, S_m)$                   /*Section 2.2*/
8:       $S_{m+1} \leftarrow Weight\_tabu\_coloring(G_{m+1}, w_{m+1}, S_m)$         /*Section 2.3*/
9:       $m = m + 1$
10:   end while
      /*Uncoarsening phase*/
11:   while $m > 0$ do
12:      $(G_{m-1}, w_{m-1}) \leftarrow Uncoarsen(G_m, w_m, S_m)$                  /*Section 2.4*/
13:      $S_{m-1} \leftarrow Weight\_tabu\_coloring(G_m, w_m, S_m)$                /*Section 2.3*/
14:      $m = m - 1$
15:   end while
      /*Perturbation phase*/
16:   if search stagnation is observed then
17:      $S_0 \leftarrow Perturbation(S_0)$                                  /*Section 2.5*/
18:   end if
19: end while
20: return The best $k$-coloring found $S$
```

tices may wrongly receive the same color and therefore can be incorrectly merged. To remedy this problem, in the coarsening phase we adopt a coarsening strategy that merges non-conflicting vertices in a probabilistic way (see Section 2.2). The coarsening operation repeats L times (a parameter, set to 5 by default), leading to L coarsened graphs.

(3) Weight tabu coloring phase: For each new coarsened graph G_{m+1}, we first generate a k-coloring S_{m+1} by inheriting from the solution S_m. Then we apply the weight tabu coloring algorithm (see Section 2.3) to ameliorate S_{m+1}. The improved solution is then used to create the next coarsened graph as explained above.

(4) Uncoarsening phase: When the coarsening phase attains its last level L, the uncoarsening phase is triggered, which unfolds the current coarsened graph G_m to the less coarsened graph G_{m-1}. Then, the solution S_m for G_m is correspondingly projected back to S_{m-1} for G_{m-1}, which is further improved by the weight tabu coloring algorithm of Section 2.3. As a result, the uncoarsening phase recovers L intermediate coarsened graphs (up to G_0) in the reverse order of the coarsening phase, while the weight tabu coloring algorithm is applied to each uncoarsened graph to improve the solution.

(5) Perturbation phase: When the uncoarsening phase terminates, SDMA moves to the next round of the multilevel process if the best solution for G_0 has been improved during the last rounds of the multilevel process. Otherwise, if the best solution cannot be improved for λ (λ is set to 10 in this work) consecutive rounds of the multilevel process, the search is judged to be in a deep local optimum trap. To escape the trap, the perturbation phase is triggered to change the colors of some specifically identified vertices (see Section 2.5). The perturbed solution is then used to seeding the next round of the multilevel process.

In the next subsections, we explain the coarsening phase, the weight tabu coloring algorithm, the uncoarsening phase and the perturbation phase.

2.2 *Coarsening phase*

Algorithm 2 The coarsening phase

```
 1: Input: Current graph $G_m$ at level $m$, solution $S_m$ for $G_m$, edge weights $w_m$.
 2: Output: Coarsened graph $G_{m+1}$, edge weights $w_m$.
    /*Step 1: Initialize the to-be-coarsened graph $G_{m+1}$*/
 3: $G_{m+1} \leftarrow G_m$
 4: $S_{m+1} \leftarrow S_m$
 5: $w_{m+1} \leftarrow w_m$
    /*Step 2: Increase the weight of edges of conflicting vertices $v_m$ and $u_m$*/
 6: for each pair $S(v_{m+1}) = S(u_{m+1})$ and $w(v_{m+1}, u_{m+1}) \neq 0$ do
 7:     $w(v_{m+1}, u_{m+1}) = w(v_{m+1}, u_{m+1}) + |E|$
 8: end for
    /*Step 3: Collapse the non-conflicting vertices $v_{m+1}$ and $u_{m+1}$*/
 9: for each pair of non-conflicting vertices $S(v_{m+1}) = S(u_{m+1})$ and $v_{m+1}$ and $u_{m+1}$
    do
10:     Calculate the $score[v_{m+1}][u_{m+1}]$
11: end for
12: if $the\ rank\ of\ score[v_{m+1}][u_{m+1}]\ is\ in\ the\ top\ p$ then
13:     $v_{m+1} \leftarrow v_{m+1} \cup u_{m+1}$
14:     Update the weights of the edges in $G_{m+1}$
15: end if
```

Let $G_m = (V_m, E_m)$ be the graph at level m. The coarsening phase coarsens G_m to G_{m+1}. Contrary to the classical coarsening strategy that only merges vertices based on structural information of the input graph, we adopt an original coarsening strategy that relies on additional information provided by the current solution.

Specifically, the proposed coarsening phase, as illustrated in Algorithm 2, performs the following steps.

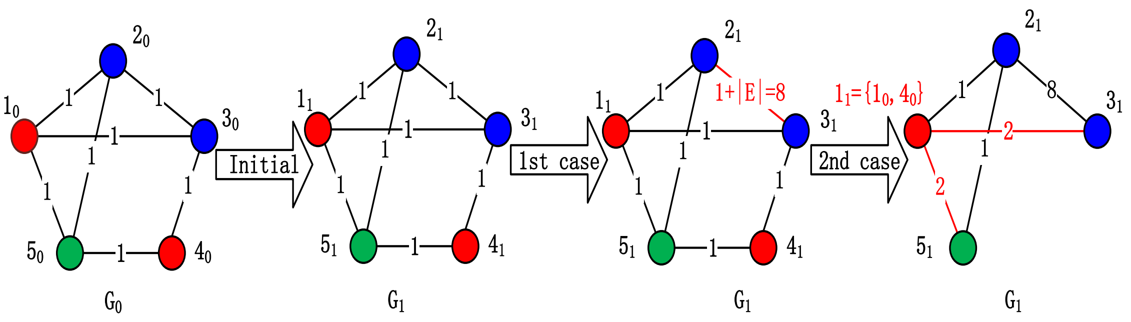


Fig. 1. Illustrative example for the coarsening phase.

Step 1: Initialize the to-be-coarsened graph G_{m+1} by setting the G_{m+1} as G_m, the solution S_{m+1} as S_m and the weight of edge w_{m+1} as w_m (lines 3-5, Algorithm 2). In the example of Figure 1, G_1 is the coarsened graph of G_0 with 5 vertices and is initialized to G_0 at the first step in Figure 1.

Step 2: Increase the weight of edge between conflicting vertices. In the weight tabu search algorithm, the weight of edge affects the coloring solution generated at the mth level. Thus, in the coarsening phase, we adjust the weight dynamically. In order to avoid the same conflict of the vertices, the algorithm adopts a heuristic strategy, which dynamically adds a large value (set to be $|E|$) to the weight of a conflicting edge. In Figure 1 (second step), the weight of the aggregated edge between 2_1 and 3_1 increases by a large number, $w(2_1, 3_1) = w(2_0, 3_0) + |E| = 8$.

Step 3: Merge non-conflicting vertices. If vertices v_{m+1} and u_{m+1} are not conflicting, they will be taken as a pair whose score will be calculated then. Repeat this process until all non-conflicting vertices are traversed. Then, we collapse probabilistically these non-conflicting vertices according to the scores and update the weights of the edges that are adjacent to vertices v_{m+1} and u_{m+1} (line 9-15, Algorithm 2). The merge procedure is decomposed into three operations:

(a) Score each pair of vertices by using historical information. We define a function $score[u_{m+1}][v_{m+1}]$ to score each pair of vertices u_{m+1} and vertex v_{m+1} on the $m+1$th level. Let $Record[u_{m+1}][v_{m+1}]$ be the number of times that vertices u_{m+1} and v_{m+1} have been assigned to the same color class in the past solutions. Besides, we define the sum of times that vertices have been coarsened as $sumCoarsen$. The scoring function is then given by $score[u_{m+1}][v_{m+1}] = Record[u_{m+1}][v_{m+1}]/(sumCoarsen + 1)$.

(b) Choose coarsened vertices. According to the score of each pair of vertices, we merge the vertices if the rank of their score is in the top p (line 12, Algorithm 2), where p is an integer from 0 to N (N being the number of vertices) randomly generated at each coarsening phase.

(c) Update the weights of the coarsened vertices. The weight of the edge (v_{m+1}, u_{m+1}) of the coarsened graph G_{m+1} equals the weight sum of the edges of the last level graph G_m. Suppose that the coarsened vertex v_{m+1} is formed by merging v_m and u_m, p_{m+1} is the result of merging p_m and q_m, the weight of the edge (v_{m+1}, p_{m+1}) of the coarsened graph G_{m+1} equals the weight sum of the edges (v_m, p_m), (v_m, q_m), (u_m, p_m) and (u_m, q_m) of the last level graph G_m.

The last step of Figure 1 shows a possible coarsening situation of non-conflicting vertices in the coarsening procedure. Suppose that vertices 1_0 and 4_0 are to be merged according to the coarsening rule, leading to the coarsened vertex 1_1 of G_1. Then, the corresponding edge weights are updated, i.e., $w(1_1, 5_1) = w(1_0, 5_0) + w(4_0, 5_0) = 2$, $w(1_1, 3_1) = w(1_0, 3_0) + w(4_0, 3_0) = 2$.

After coarsening the graph G_m to G_{m+1}, we apply the weight tabu coloring algorithm to improve the solution of the graph G_{m+1} (see the Section 2.3). Normally, the solution S_{m+1} that is inherited from S_m has less degree of freedom.

2.3 *Solution refinement by weight tabu coloring algorithm*

Recall that the coarsening phase is to obtain a coarsened graph G_{m+1} at the $m+1$th level from the graph G_m at the mth level while the uncoarsening phase is to recover G_m from G_{m+1}. After the coarsening phase and uncoarsening phase, we adopt the weight tabu coloring algorithm (WTS) to refine the quality of the solution S_m for each new graph. For the reason of clarity, we use below $G^{'} = (V^{'}, E^{'})$ to represent the new graph and $S^{'}$ to represent the solution inherited from its original graph.

Given $G^{'} = (V^{'}, E^{'})$, WTS explores all possible k-colorings (candidate solutions), similar to what is presented at the beginning of Section 2. However, given that $G^{'}$ is a weight graph, WTS adopts an adjusted weight objective function $f^{'}(S^{'}) = \sum_{i=1}^{k} |C_w(V_i^{'})|$ where $|C_w(V_i^{'})|$ is the sum of the weights of conflicting edges in color class $V_i^{'}$. Accordingly, a k-coloring $S^{'}$ with $f^{'}(S^{'}) = 0$ corresponds to a legal k-coloring. For the purpose of optimization, WTS extends the popular TabuCol algorithm [21,44,25] to the case of weight graphs.

Given a conflicting k-coloring solution $S^{'} = \{V_1^{'}, V_2^{'}, \ldots, V_k^{'}\}$, the basic idea of WTS is to move a conflicting vertex $v^{'}$ from its color class $V_i^{'}$ to another color class $V_j^{'}$. To ensure a high computational efficiency of such moves, we maintain a matrix B of size $n * k$ with elements $B[v^{'}][i]$ recording the number of vertices adjacent to $v^{'}$ in color class $V_i^{'} (1 \leqslant i \leqslant k)$. The move

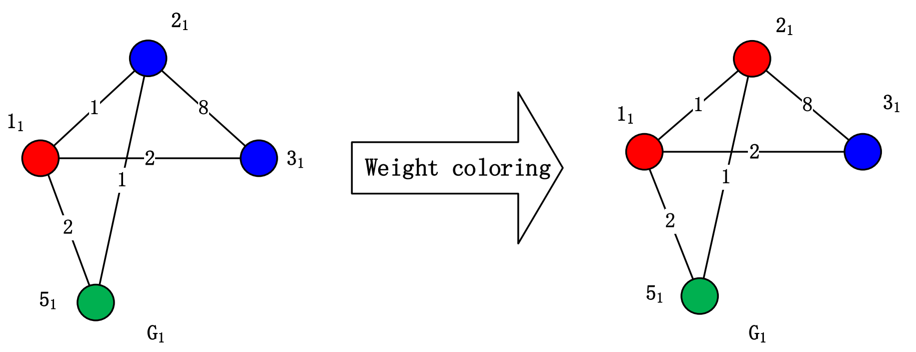


Fig. 2. Illustrative example for the weight coloring of the coarsened graph G_1.

gain that represents the change in the objective function is expressed as $\Delta f' = B[v'][j] - B[v'][i]$. According to an incremental technique, each time a vertex v' is moved to another color class, we just need to update a subset of values affected by this move as follows. For each vertex u' adjacent to vertex v', $B[u'][i] \leftarrow B[u'][i] - w[v'][u']$, and $B[u'][j] \leftarrow B[u'][j] + w[v'][u']$.

At each iteration, WTS selects a best (according to f') eligible vertex to perform the move. An eligible vertex is a conflicting vertex, which is not forbidden by the tabu list or its move leads to a solution better than any seen solution. WTS stops when $f'(S') = 0$ or when the objective function cannot be improved within a given number of iterations.

Figure 2 shows a possible iteration of the WTS procedure. Given the conflicting 3-coloring for G_1 in the last graph of Figure 1 and suppose that the vertex 3_1 is forbidden by the tabu list, the vertex 2_1 will be colored red, leading to a new solution with $f' = 1$.

2.4 *Uncoarsening phase*

The uncoarsening phase is the reverse of the coarsening phase that recovers the graph G_{m-1} from the collapsed graph G_m level by level until the initial graph G_0 is reached from the graph G_1. Specifically, the underlying steps from the mth level to the $m-1$th level are described as follows.

(1) Initialize the graph G_{m-1} with G_m, the solution S_{m-1} with S_m and the edge weights w_{m-1} with w_m.

(2) Unfold the coarsened vertices and update the corresponding solution and edge weights. As each pair of coarsened vertices is never adjacent, when the k-coloring S_m for G_m is extended to G_{m-1}, we simply assign the same color to them when the vertices are unfold. Finally, update the corresponding weights of the edges of the unfolded vertices.

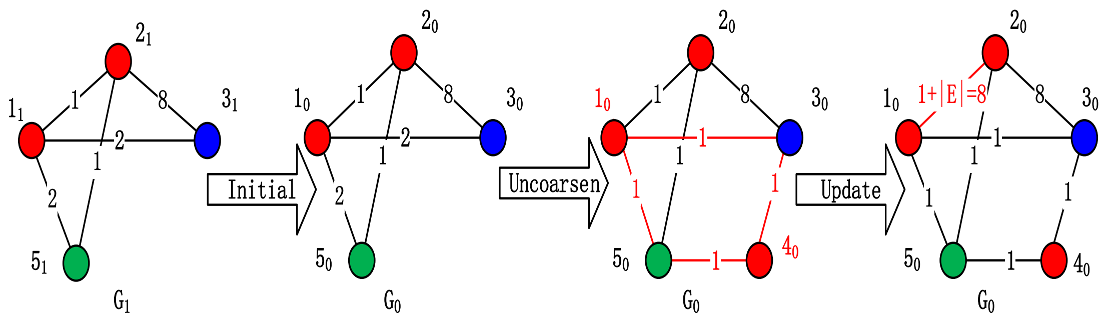


Fig. 3. Illustrative example for the uncoarsening phase.

(3) Increase the weights of the edges between conflicting vertices. If the vertices v_{m-1} and u_{m-1} are conflicting vertices, we add a big number ($|E|$ in this work) to the weight of the underlying edge.

As shown in Figures 1 and 2, after the coarsening procedure and the weight tabu search procedure, the graph G_0 is coarsened to the graph G_1, which is colored with the solution S_1. An initialization of the uncoarsening phase is shown in the first step of Figure 3, $G_0 = G_1, S_0 = S_1, w_0 = w_1$. Then, vertex 1_1 of G_1 is recovered and its color is passed to the vertices 1_0 and 4_0. Meanwhile, those edges between the vertices adjacent to vertices 1_0 and 4_0 are restored and the corresponding weights are updated, i.e., $w(1_0, 3_0) = w(1_0, 5_0) = 1$, $w(4_0, 3_0) = w(4_0, 5_0) = 1$. Finally, the weight of the edge between conflicting vertices 1_0 and 2_0 is increased, $w(1_0, 2_0) = w(1_1, 2_1) + |E| = 8$.

2.5 Perturbation process

The above multilevel process combining coarsening, uncoarsening and weight tabu search is able to progressively improve the initial solution S_0. To further improve the quality of the solution, we iteratively repeat the multilevel process until no solution improvement can be reached during 10 consecutive rounds of the multilevel process. At this point, the search is judged to be trapped deeply. To get rid of the trap, we apply a simple perturbation before entering the next round of the multilevel process.

The perturbation displaces a fixed number of vertices (set to $0.1 * N$, N being the number of vertices) in the following way. To avoid a too strong deterioration of the perturbed solution, the perturbation takes into consideration the objective value and makes sure that the reverse move is forbidden during $tt = f' + rand(1000)$ steps where $rand(1000)$ is a random number between 0 to 999. The resulting solution from the perturbation procedure is then used as the new starting solution of the next round of the coarsening phase.

3 Experimental results and comparisons

This section is dedicated to an experimental assessment of the proposed SDMA algorithm. The study was based on 47 conventional benchmark instances that are commonly used in the literature and initially proposed for the DIMACS and COLOR competitions for graph coloring problems [1] [2].

3.1 *Benchmark instances*

The 47 benchmark instances are classified into three categories in this paper: easy graphs, small difficult graphs and large difficult graphs [17,26]. Let BKV (the best-known value) represents $\chi(G)$ (if known) or the best-known upper bound in the literature.

(1) The first category, easy graphs, contains 19 instances that can be colored with BKV colors by a basic coloring algorithm like DSATUR [20] (thus by numerous algorithms). This category of graphs includes some graphs of DSJC*, DSJR*, flat*, le450*, R* and two school* graphs. For these graphs, modern coloring algorithms can consistently reach the best-known or the optimal results.
(2) The second category contains 20 difficult instances. These graphs are much more difficult for which a legal k-coloring with k =BKV or a slightly larger value than BKV is difficult to detect. This category of graphs includes the remaining graphs of DSJC*, DSJR*, flat*, le450*, R* graph and latin* graphs. For several graphs of this category, only few algorithms can reach the best-known results.
(3) The third category contains 8 large difficult instances whose best-known results can only be achieved by a few advanced coloring algorithms. Graphs having wap* in their name fall into this category.

Among these graphs, DSJC* are standard (n, p) random graphs. DSJR* and R* are geometric graphs, with DSJR* being complements of geometric graphs. le450* are random graphs generated with a special procedure ensuring known chromatic numbers. flat* are quasi-random graphs generated by partitioning the vertex set into k random classes of almost equal size and then by adding random edges only between vertices of different classes. latin_square_10 and school* are structured graphs from the Latin square problem and school class scheduling problems. wap* are large structured sparse graphs from the wavelength assignment problem in real-life optical networks [45]. These

[1] http://archive.dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/
[2] http://www.cs.hbg.psu.edu/txn131/graphcoloring.html/

graphs are highly diverse with very different characteristics (size, topology, edge density...).

Following the common practice to report comparative results in the coloring literature, the performance assessment focuses on the best solution found by an algorithm corresponding to the smallest number k of colors needed to reach a legal k-coloring for a graph. It is worth noting that for the 28 difficult graphs, no single algorithm can reach all the best-known results. As one can observe from the results presented in Section 3.3 (e.g., Tables 3 and 5), even the best performing algorithms miss at least two best-known results. In fact, these instances have been studied for a long time (over 30 years) and the current best-known k values (BKV) have been attained jointly by a small number of complex hybrid algorithms under specific and relaxed conditions (e.g., large run time from several days to one month). Moreover, for some of these benchmark graphs, even finding a legal k-coloring for a k value slightly above the chromatic number or the current best-known result remains difficult. Thus, an algorithm able to attain (or draw near to) most BKV values can be considered to be interesting. Given these observations, one understands that it is not suitable to apply statistical tests when discussing computational results.

3.2 *Experiment settings*

The proposed algorithm was written in C++ and compiled by GNU g++ 4.1.2 with the -O3 flag. 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.

3.2.1 *Parameters*

The main parameters of the SDMA algorithm are related to the weight tabu coloring algorithm. Since the weight tabu coloring algorithm is built up on the TabuCol implementation of [25], we adopted the parameter setting in [25] as SDMA's default setting. Specially, the depth of the weight tabu coloring algorithm (β) was set to 10^6. The tabu tenure (tt_w) was defined by $rand(10) + f'$ in the weight tabu coloring algorithm and by $rand(10000) + f'$ in the perturbation phase. In addition, we fixed the level limit (L) to 5 according to the experiment reported in Section 4.1. The unimproved consecutive rounds for best solution is set 10. For each instance, the initial value of k was set to a value lightly larger (typically, 1 to 5 more colors) than the best-known value in the literature. Table 1 gives the descriptions of the parameters and their settings adopted in our SDMA algorithm..

Table 1
Settings of important parameters

Parameters	Description	Section	Value
β	search depth of weight tabu coloring	2.3	10^6
tt_w	tabu tenue of weight tabu coloring	2.3	$rand(10) + f'$
tt	tabu tenue of perturbation	2.5	$rand(1000) + f'$
L	level limit of coarsening phase	2.1	5
λ	unimproved consecutive rounds for best solution	2.1	10

3.2.2 *Reference algorithms*

To evaluate the performance of the proposed algorithm, 13 state-of-the-art heuristic algorithms in the literature are used as the main reference algorithms.

(1) Iterated local search algorithm (IGrAl) [22] (a 2.8 GHz Pentium 4 processor and a cut off time of 1 hour);
(2) Variable space search algorithm (VSS) [23] (a 2.0 GHz Pentium 4 processor and a cut off time of 10 hours);
(3) Local search algorithm using partial solutions (Partial) [46] (a 2.0 GHz Pentium 4 and a time limit of 10 hours together with a limit of $2 * 10^9$ iterations without improvement);
(4) Hybrid evolutionary algorithm (HEA) [25] (the processor used is not available for this oldest algorithm and the results were obtained with different parameter settings);
(5) TabuCol search refinement (TabuCol) [25];
(6) The basic multilevel algorithm (MLTS) [42] (a DEC Alpha machine with a 466 MHz CPU and 1 Gbyte of memory with 2048 iteration);
(7) Adaptive memory algorithm (AMA) [26] (the processor applied is not available and the results were obtained with different parameter settings);
(8) Two-phase evolutionary algorithm (MMT) [27] (a 2.4 GHz Pentium processor and a cut off time of 6000 or 40000 seconds);
(9) Evolutionary algorithm with diversity guarantee (Evo-Div) [29] (a 2.8 GHz Xeon processor and a cut off time of 12 hours);
(10) Distributed quantum annealing algorithm (QACOL) [30,47] (a 3.0 GHz Intel processor with 12 cores and a cut off time of 5 hours);
(11) Memetic algorithm (MACOL, renamed simply MA) [28] (a 3.4 GHz processor and a cutoff time of 5 hours);
(12) The newest parallel memetic algorithm (HEAD) [31] (a 3.1 GHz Intel Xeon processor with 4 cores used to run in parallel the search processes with a cut off time of at least 3 hours);
(13) Probability learning based local search (PLSCOL) [24] (A 2.8 GHz Intel Xeon E5-2760 processor and 2 GB RAM with a cutoff of 5 hours).

We only compare the results of the second category graphs with 12 algorithms for an overall discussion since most of the reference algorithms did not report their results on the third category graphs. Then, we compare our results of the second category and third category graphs with three typical algorithms: TabuCol (the popular tabu search coloring algorithm), PLSCOL (the latest learning-based local search algorithm) and MLTS (the basic multilevel coloring algorithm).

For our comparative studies, we follow the common practice in the literature on graph coloring and cite the best results reported by the reference algorithms. Indeed, as indicated in Section 3.1, these results were obtained on very different computing platforms and under various relaxed conditions (e.g., specific parameter settings, large run time from several days to one month for the most difficult instances). As such, the cited results can be considered as the very best results that can be achieved by the reference algorithms.

3.2.3 *Stopping condition*

Given the high difficulty of the above challenging benchmarks, the majority of the reference algorithms allowed long run time limits of at least 5 hours. We thus adopt the same cut off limit for our experiments. It is worth noticing that in the literature, the main assessment criterion is the quality, i.e., the smallest number of colors used to find a legal coloring. Indeed, it is impossible to compare computation times in a controllable manner, given that the reference algorithms were implemented with different programming languages, and executed under various computing platforms and different stopping conditions (maximum allowed generations, maximum allowed objective evaluations, maximum allowed iterations or maximum allowed time limit). Thus, the timing information, when it is shown, was only provided for indicative purposes. Given its stochastic nature, SDMA was run 10 times to solve each problem instance.

3.3 *Comparison with state-of-the-art algorithms*

In this section, we compare the results of the 19 instances in the first category, the 28 difficult instances in the second and the third categories.

3.3.1 *Comparative results on easy instances*

Table 2 reports the results of our SDMA algorithm on the set of 19 DIMACS/COLOR easy instances. As our basic reference algorithms, we adopt TabuCol (the popular tabu search coloring algorithm, which is also the underlying refinement algorithm for SDMA), PLSCOL (the latest learning-based local search algorithm) and MLTS (the basic multilevel coloring algorithm). We focus on the criterion of solution quality in terms of the smallest number of colors used to find a legal coloring.

The first column of Table 2 indicates the name of each instance and the second column indicates the best results (BKV) obtained in the literature. The following 9 columns report respectively the best results of TabuCol, PLSCOL and MLTS (i.e., the smallest number of colors used) over 10 independent runs (k), the success rate (SR) (when available) to achieve the best result over 10 runs and the needed computation time. The last 4 columns show the results of SDMA for each instance: the best result over 10 runs (k), the success rate (SR), the average computation time in seconds ($t(s)_{Avg}$) of the successful runs to obtain the best result and the minimal computation time ($t(s)_{Min}$). Additionally, the rows #Better_SDMA, #Equal_SDMA and #Worse_SDMA indicate respectively the number of instances for which an algorithm performs better, equally well or worse as compared with SDMA. Finally, an entry with * indicates the known chromatic number.

Table 2
Comparison between SDMA and state-of-the-art algorithms (the first category of instances).

		TabuCol				PLSCOL			MLTS		SDMA			
Instance	BKV	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	k	SR	$t(s)_{Avg}$	k	$t(s)_{Avg}$	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$
DSJC125.1.col	5*	5	10/10	0.00	0.00	5	10/10	< 60	5	8.09	5	10/10	0.01	0.00
DSJC125.5.col	17*	17	10/10	0.28	0.01	17	10/10	< 60	18	7.27	17	10/10	0.26	0.03
DSJC125.9.col	44*	44	10/10	0.02	0.01	44	10/10	< 60	44	5.84	44	10/10	0.03	0.01
DSJC250.1.col	8	8	10/10	0.04	0.02	8	10/10	< 60	9	11.05	8	10/10	0.06	0.01
DSJC250.9.col	72*	72	10/10	1.89	0.61	72	10/10	< 60	75	5.09	72	10/10	2.91	0.28
DSJR500.1.col	12*	12	10/10	125.63	22.15	12	10/10	< 60	12	9.37	12	10/10	0.14	0.04
flat300_20_0.col	20*	20	10/10	0.6	0.4	20	10/10	< 60	20	13.26	20	10/10	0.09	0.05
le450_15a.col	15*	15	10/10	0.35	0.17	15	10/10	< 60	16	14.96	15	10/10	0.36	0.12
le450_15b.col	15*	15	10/10	0.18	0.09	15	10/10	< 60	16	14.96	15	10/10	0.22	0.09
le450_25a.col	25*	25	10/10	0.01	0.01	25	10/10	< 60	25	11.92	25	10/10	0.04	0.04
le450_25b.col	25*	25	10/10	0.01	0.01	25	10/10	< 60	25	8.50	25	10/10	0.04	0.04
R125.1.col	5*	5	10/10	0.72	0.07	5	10/10	< 60	5	5.89	5	10/10	0.00	0.00
R125.1c.col	46*	46	4/10	0.00	0.00	46	10/10	< 60	47	5.58	46	10/10	6.15	3.11
R125.5.col	36*	36	6/10	5.12	0.03	36	10/10	< 60	38	7.10	36	10/10	1.76	0.05
R250.1.col	8*	8	10/10	0.76	0.00	8	10/10	< 60	8	7.40	8	10/10	0.01	0.00
R250.1c.col	64*	67	1/10	159.50	159.50	64	10/10	60	67	11.12	64	10/10	19.95	0.06
R1000.1.col	20*	20	10/10	0.72	0.07	20	10/10	< 60	20	12.18	20	10/10	0.27	0.25
school1.col	14*	14	10/10	0.02	0.01	14	10/10	< 60	14	11.93	14	10/10	0.04	0.03
school1_nsh.col	14	14	10/10	0.05	0.02	14	10/10	< 60	14	12.35	14	10/10	0.04	0.03
#Better_SDMA		0/19				0/19			0/19					
#Equal_SDMA		18/19				19/19			11/19					
#Worse_SDMA		1/19				0/19			8/19					

Table 2 indicates that PLSCOL and SDMA can easily find the best-known results with a 100% success rate, while TabuCol is less stable on 2 instances (R125.1c and R125.5), and fails to attain one best-known result (R250.1c). Besides, MLTS fails to attain 8 best-known results. We notice that SDMA is computationally efficient by attaining the best-known solutions in less than 20s.

3.3.2 *Comparative results on difficult instances*

Table 3
Comparison between SDMA and state-of-the-art algorithms (the 2nd and 3rd categories of instances).

		TabuCol				PLSCOL			MLTS		SDMA			
Instance	BKV	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	k	SR	$t(s)_{Avg}$	k	$t(s)_{Avg}$	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$
DSJC250.5.col	28	28	9/10	41.88	3.53	28	10/10	4	30	9.21	28	10/10	20.27	3.12
DSJC500.1.col	12	12	10/10	784.94	134.24	12	7/10	43	13	18.25	12	10/10	656.47	8.16
DSJC500.5.col	47	49	10/10	1393.98	147.98	48	3/10	1786	54	15.68	48	1/10	2127.11	2127.11
DSJC500.9.col	126	126	7/10	7150.97	972.45	126	10/10	747	136	12.39	126	9/10	4116.60	584.23
DSJC1000.1.col	20	21	10/10	2.65	1.30	20	1/10	3694	23	27.49	20	7/10	13397.85	6901.68
DSJC1000.5.col	82	89	9/10	3909.42	1941.63	87	10/10	1419	97	26.02	87	4/10	22923.98	15107.20
DSJC1000.9.col	222	225	10/10	4013.43	1377.62	223	5/10	12094	253	24.74	223	5/10	22093.84	13301.7
DSJR500.1c.col	85*	85	10/10	1105.22	113.19	85	10/10	386	90	24.05	85	8/10	788.67	27.40
DSJR500.5.col	122*	126	2/10	3929.72	14.45	126	8/10	1860	134	49.29	**124**	1/10	1539.74	1539.74
flat300_26_0.col	26*	26	10/10	11.24	51.91	26	10/10	195	34	10.72	26	10/10	23.11	5.31
flat300_28_0_30.col	28*	30	6/10	198.68	6202.80	30	10/10	233	34	10.44	**29**	3/10	32601.40	10334.20
flat1000_76_0.col	81	88	9/10	5052.25	1175.60	86	1/10	5301	95	25.87	86	3/10	23971.83	17304.90
latin_square_10.col	97	100	1/10	11134.60	11134.60	99	8/10	2005	113	20.92	99	3/10	3911.05	1109.18
le450_15c.col	15*	15	1/10	40.12	40.12	15	7/10	1718	22	16.81	15	6/10	150.75	1410.97
le450_15d.col	15*	16	10/10	0.44	4.45	15	3/10	2499	22	13.30	15	1/10	6.19	3423.09
le450_25c.col	25*	25	4/10	1886.15	9178.21	25	10/10	1296	28	17.08	25	10/10	4007.70	316.64
le450_25d.col	25*	26	10/10	0.43	0.32	25	10/10	1704	28	13.01	25	9/10	7820.19	172.18
R250.5	65*	67	6/10	227.93	0.50	66	10/10	705	70	17.63	**65**	3/10	5610.90	963.19
R1000.1c.col	98	98	9/10	412.10	28.79	98	10/10	256	107	41.16	98	10/10	713.29	116.73
R1000.5.col	234*	250	1/10	17231.80	17231.80	254	4/10	7818	259	156.39	**246**	3/10	21984.50	14435.40
wap01a.col	42	43	10/10	276.70	11.04	-	-	-	-	-	**41**	1/10	12368.80	12368.80
wap02a.col	41	42	10/10	2366.80	512.13	-	-	-	-	-	**41**	7/10	10786.50	2820.18
wap03a.col	44	45	10/10	4280.36	324.08	-	-	-	-	-	**44**	4/10	16572.03	5275.43
wap04a.col	42	44	10/10	821.55	120.30	-	-	-	-	-	**42**	3/10	16359.17	10538.10
wap05a.col	50*	50	10/10	1.89	0.32	-	-	-	-	-	50	10/10	0.40	0.37
wap06a.col	40*	44	1/10	5741.87	5741.87	-	-	-	-	-	**40**	8/10	3490.96	1220.78
wap07a.col	41	42	1/10	2455.08	2455.08	-	-	-	-	-	**41**	1/10	21236.90	21236.90
wap08a.col	41	42	10/10	7743.11	241.71	-	-	-	-	-	**41**	9/10	5607.51	2230.79
#Better_SDMA		0/28				0/20			0/20					
#Equal_SDMA		9/28				16/20			0/20					
#Worse_SDMA		19/28				4/20			20/20					

We first show in Table 3 a comparison of SDMA with the three basic reference algorithms (TabuCol, PLSCOL and MLTS) in terms of quality on all the 28 instances of the second and the third categories. In this table, we show the same information as in Table 2, while entries with "-" mean that the corresponding results are not available.

From Table 3, we can see that SDMA competes very favorably with the three basic reference algorithms. Indeed, SDMA reports 11 strictly better results (indicated in boldface) including 4 better results as compared with the recent PLSCOL algorithm. In particular, SDMA performs far better than the previous multilevel algorithm MLTS, which reports the worst results among the compared algorithms. This provides strong evidences in favor of our solution-driven multilevel approach.

SDMA performs remarkably well on the family of large structured wap* graphs by reaching all the best-known results. Importantly, SDMA finds a

41-coloring for wap01a.col, which improves the current best upper bound of 42 reported in [48]. Given that the chromatic number of wap01a.col is lower-bounded by 41 [49] (in fact wap01a.col has a clique of size 41), our 41-coloring [3] closes definitively this graph and proves the chromatic number of 41. To shed light on SDMA's good performance on these graphs that come from the wavelength assignment problem, we show their structural characteristics in Table 4. The first column indicates the name of each graph. The following 4 columns report the number of vertices and edges, density, and maximum/minimum degree. We observe that unlike other DIMACS instances, the wap* graphs are both large and sparse with 905 to 5231 vertices, an edge density between 0.02 and 0.11 and a large degree variation (from 9 to 351). As shown in the convergence analysis of Section 4.3, the search trajectory of SDMA for solving the wap* graphs is eased due to the structural features of this type of graphs.

Table 4
Main structural properties of the wap* instances from wavelength assignment in optical networks.

Instances	$\|V\|$	$\|E\|$	Density	Max_degree	Min_degree
wap01a.col	2368	110871	0.0395611	288	14
wap02a.col	2464	111742	0.0368249	294	14
wap03a.col	4730	286722	0.0256366	344	26
wap04a.col	5231	294902	0.0215586	351	26
wap05a.col	905	43081	0.1053170	228	9
wap06a.col	947	43571	0.0972717	230	9
wap07a.col	1809	103368	0.0632090	298	13
wap08a.col	1870	104176	0.0596138	308	13

To further assess the performance of the proposed algorithm, we now present in Table 5 a comparison of SDMA with 10 other reference algorithms that are based on various approaches and represent the state-of-the-art methods on graph coloring. These algorithms altogether cover the current best-known results ever reported in the literature (shown in the column BKV) that have been obtained under various conditions. Notice that no single algorithm can reach all the best-known results even under much relaxed conditions (e.g., a running time more than several days). Since the results for the 8 instances of the third category are not available for these reference algorithms, Table 5 only concerns the 20 hard instances of the second category and shows the best results of each algorithm in terms of the smallest number of colors k used to find a legal k-coloring. Once again, the rows #Better_SDMA, #Equal_SDMA and #Worse_SDMA indicate the number of instances for which an algorithm has a better, equal or worse result compared to SDMA.

[3] The newly found 41-coloring for wap01a.col is available at http://www.info.univ-angers.fr/pub/hao/SDMA.html

Table 5
Comparison between SDMA and state-of-the-art algorithms (the 2nd and 3rd categories of instances).

		Local search algorithms				Population-based algorithms						
Instance	BKV	IGrAl	VSS	Partial	SDMA	HEA	AMA	MMT	Evo-Div	MA	QACOL	HEAD
		2008	2008	2008	Best	1999	2008	2008	2010	2010	2011	2015
DSJC250.5.col	28	29	-	-	28	-	28	28	-	28	28	28
DSJC500.1.col	12	12	12	12	12	12	12	12	12	12	12	12
DSJC500.5.col	47	50	48	48	48	48	48	48	48	48	48	47
DSJC500.9.col	126	129	126	127	126	126	126	127	126	126	126	126
DSJC1000.1.col	20	22	20	20	20	20	20	20	20	20	20	20
DSJC1000.5.col	82	94	86	89	87	83	84	84	83	83	83	82
DSJC1000.9.col	222	239	224	226	223	224	224	225	223	223	222	222
DSJR500.1c.col	85*	85	85	85	85	-	86	85	85	85	85	85
DSJR500.5.col	122*	126	125	125	124	-	127	122	122	122	122	-
flat300_26_0.col	26*	-	-	-	26	-	26	26	-	26	-	-
flat300 28 0.col	28*	-	28	28	29	31	31	31	31	29	31	31
flat1000_76_0.col	81	-	85	87	86	83	84	83	82	82	82	81
latin_square_10.col	97	100	-	-	99	-	104	101	100	99	98	-
le450_15c.col	15*	16	15	15	15	15	15	15	-	15	15	-
le450_15d.col	15*	16	15	15	15	15	15	15	-	15	15	-
le450_25c.col	25*	27	25	25	25	26	26	25	25	25	25	25
le450_25d.col	25*	27	25	25	25	26	26	25	25	25	25	25
R250.5.col	65*	-	-	66	65	-	-	65	65	65	65	65
R1000.1c.col	98	-	-	-	98	-	-	98	98	98	98	98
R1000.5.col	234*	238	245	238	246	255	-	234	238	245	238	245
#Better_SDMA		1/15	4/15	2/16	-	2/13	2/17	4/20	4/16	4/20	6/19	5/15
#Equal_SDMA		2/15	9/15	8/16	-	6/13	8/17	12/20	10/16	16/20	12/19	9/15
#Worse_SDMA		12/15	2/15	6/16	-	5/13	7/17	4/20	2/16	0/20	1/19	1/15

Table 5 shows that SDMA remains globally competitive as compared with the reference algorithms. In order to facilitate comparisons, we divide these reference algorithms into three groups according to whether the results are better than SDMA.

The first group contains 2 local search algorithms (IGrAI, Partial) and 2 hybrid algorithms (HEA, AMA), which report respectively their results for 15, 16, 13 and 17 out of the 20 instances. Compared to these four algorithms, SDMA algorithm obtains respectively (12, 6, 5 and 7) better results, (2, 8, 6 and 8) equal results, and (1, 2, 2 and 2) worse results. This indicates that SDMA competes very favorably with these reference algorithms.

The second group only includes the highly sophisticated MMT algorithm (column 9), which reports its results on all 20 instances of the second category. Table 5 shows that SDMA is worse than MMT for 4 instances, better for 4 instances and matches the remaining results, which indicates that SDMA performs similarly to MMT.

The third group contains the 5 remaining algorithms (VSS, Evo-Div, MA, QACOL and HEAD). SDMA is less competitive when it is compared with these algorithms though it shares the same results for many instances. Specifically, SDMA has fewer winning cases than losing cases: 2 vs 4 for 15 instances for VSS, 2 vs 4 for 16 instances for Evo-Div, MACOL 0 vs 4 for the

20 instances for MA, 1 vs 6 for 19 instances for QACOL and 1 vs 5 for 15 instances for HEAD.

To sum up, the results of Tables 3 and 5 indicate that the proposed algorithm remains competitive with respect to 8 of the 13 leading reference coloring algorithms, and is less competitive than 5 of them. Moreover, SDMA tends to be more suitable for coloring large structured (sparse) graphs such as the wap* instances while other leading algorithms are more appropriate for coloring random or quasi-random instances. As such, SDMA can be considered to be a complementary alternative that enriches the toolkit for graph coloring.

4 Analysis

This section performs additional experiments to deepen the understanding of some important features of the proposed SDMA approach: the number of level for multilevel and the coarsening strategy. The experiments are based on 15 graphs with different characteristics. Each algorithm was run 10 times respectively on each instance with a time limit of 9000s (2.5 hour) per run unless otherwise stated.

4.1 Influences of the number of the coarsening levels

The proposed SDMA algorithm uses the parameter L to control the number of coarsened graphs, which was set to 5 by default. We now present a study to assess the influence of the number of the levels on the algorithm. For this purpose, we test two alternative values $L = 3, 7$.

In this experiment, we ran the algorithm 10 times to solve each selected instance with a cutoff time of 9000 seconds. The results are presented in Table 6 with the same information as before. Columns δ_1 and δ_2 indicate the color difference between SDMA (with its default value $L = 5$) and the two other values ($L = 3$ and $L = 7$) (a negative value indicates that SDMA reports a better result). We note that the best results are obtained with $L = 5$, which justifies the default setting for this parameter.

4.2 Effectiveness of the coarsening strategy

The coarsening phase uses a historical information based probability procedure (Section 2.2) during the coarsening process. In this section, we discuss

Table 6
Influences of the number of the levels L on the algorithm.

	SDMA ($L = 3$)				SDMA ($L = 5$, defaut value)				SDMA ($L = 7$)					
Instance	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	δ_1	δ_2
DSJC1000.5.col	88	4/10	5169.41	1397.90	88	5/10	3775.84	655.07	88	5/10	5015.70	552.59	0	0
DSJC1000.9.col	224	1/10	4072.58	4072.58	224	2/10	4633.51	4230.82	224	2/10	5400.58	3407.85	0	0
DSJC500.5.col	49	10/10	758.20	151.13	49	10/10	1599.93	75.90	49	10/10	1127.31	7.11	0	0
DSJC500.9.col	126	6/10	3030.90	389.67	126	4/10	3538.56	2911.30	126	5/10	2218.92	679.11	0	0
DSJR500.1c.col	85	1/10	1159.28	1159.28	85	1/10	1029.30	1029.30	86	2/10	659.46	28.74	0	-1
flat1000_76_0.col	87	5/10	4563.49	2103.53	87	5/10	4696.42	2643.76	87	4/10	3302.04	237.23	0	0
flat300_28_0.col	30	3/10	5829.91	3523.22	29	1/10	6136.43	6136.43	30	4/10	1540.71	78.37	-1	-1
le450_25c.col	25	3/10	2820.42	85.99	25	2/10	231.31	129.59	25	1/10	2594.13	2594.13	0	0
le450_25d.col	25	2/10	5924.27	5847.12	25	4/10	3435.04	303.92	25	1/10	241.46	241.46	0	0
R1000.5.col	249	3/10	5109.91	4400.25	248	2/10	5648.30	5513.54	249	1/10	3463.03	3463.03	-1	-1
wap01a.col	42	9/10	2142.89	197.63	42	7/10	907.225	411.52	42	7/10	1017.67	432.13	0	0
wap05a.col	50	9/10	7.66	0.37	50	10/10	5.65	0.36	50	10/10	95.97	0.44	0	0
wap06a.col	40	6/10	1826.09	447.60	40	7/10	2335.37	220.32	40	8/10	828.25	280.31	0	0
wap07a.col	41	2/10	3431.62	1146.24	41	2/10	3016.67	2503.04	41	5/10	3703.32	1228.83	0	0
wap08a.col	41	4/10	2785.52	1742.16	41	4/10	3411.05	1280.35	41	5/10	4375.18	726.57	0	0

how this strategy contributes to the overall performance of our SDMA algorithm.

For this study, we create $SDMA\_variant$ that is a SDMA variant where we coarsen all the vertices without the probability test (i.e., all the non-conflicting vertices with the same color are merged). We run $SDMA\_variant$ under the same stopping condition as in Section 4.1 (i.e., if a legal k-coloring is found or the maximum allowed run time of 2.5 CPU hours is reached). Each tested instance is solved 10 times.

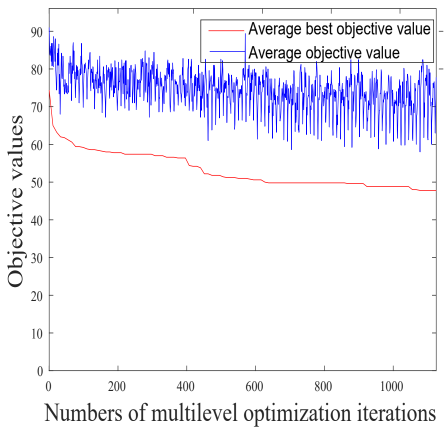
Table 7 displays the comparative results between SDMA and $SDMA\_variant$, based on the same indicators adopted in Table 6. From this table, we observe that SDMA outperforms $SDMA\_variant$, achieving better values for 5 out of 15 instances and equal results for the remaining 9 instances. Moreover, for the 9 instances with equal results, SDMA always obtains a higher successful rate than $SDMA\_variant$. These observations confirm the usefulness of the probability strategy of the coarsening procedure based on history information for the SDMA algorithm.

4.3 Convergence analysis

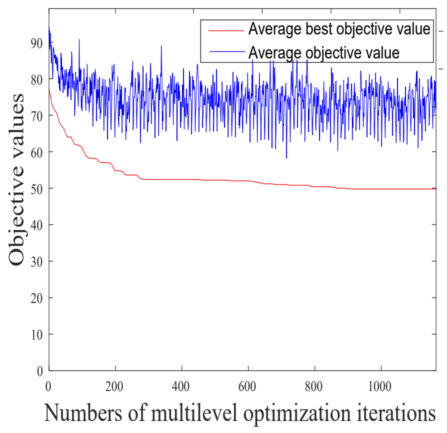
This section shows a convergence analysis of the SDMA algorithm. For this purpose, we carried out an experiment to obtain the running profiles of the algorithm on 4 representative instances (DSJC1000.5.col, flat1000_76_0.col, R1000.5.col, wap01a.col) with the k values reported in the literature. To eliminate the possible influence of randomness, we ran SDMA 10 times on each instance with a cutoff time of 3600 seconds per run. Given this reduced cutoff time, we limited accordingly the search depth of weight tabu coloring β to 10^4.

Table 7
Influences of the coarsening strategy on the algorithm.

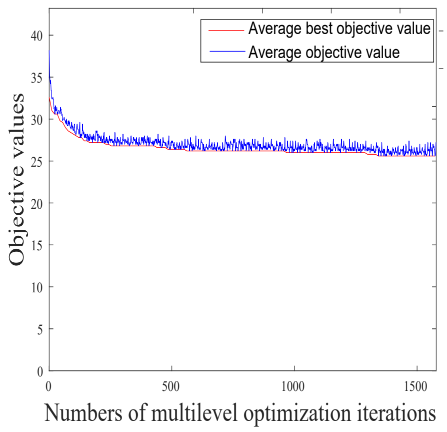
	SDMA_variant				SDMA				
Instance	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	k	SR	$t(s)_{Avg}$	$t(s)_{Min}$	δ
DSJC1000.5.col	88	2/10	4653.98	1738.86	88	5/10	3775.84	655.07	0
DSJC1000.9.col	226	1/10	1769.46	1769.46	224	2/10	4633.51	4230.82	-2
DSJC500.5.col	49	5/10	2782.48	225.36	49	10/10	1599.93	75.90	0
DSJC500.9.col	126	1/10	4605.12	4605.12	126	4/10	3538.56	2911.30	0
DSJR500.1c.col	88	2/10	4.29	0.32	85	1/10	1029.30	1029.30	-3
flat1000_76_0.col	86	1/10	7799.25	7799.25	87	5/10	4696.42	2643.76	1
flat300_28_0.col	30	2/10	4110.15	3172.38	29	1/10	6136.43	6136.43	-1
le450_25c.col	25	1/10	1447.73	1447.73	25	2/10	231.31	129.59	0
le450_25d.col	25	1/10	1813.77	1813.77	25	4/10	3435.04	303.92	0
R1000.5.col	254	1/10	8010.04	8010.04	248	2/10	5648.30	5513.54	-6
wap01a.col	42	2/10	4167.23	3366.09	42	7/10	907.225	411.527	0
wap05a.col	50	4/10	53.36	0.39	50	10/10	5.65	0.36	0
wap06a.col	40	1/10	6033.32	6033.32	40	7/10	2335.37	220.32	0
wap07a.col	42	5/10	1843.35	496.25	41	2/10	3016.67	2503.04	-1
wap08a.col	41	1/10	6345.52	6345.52	41	4/10	3411.05	1280.35	0



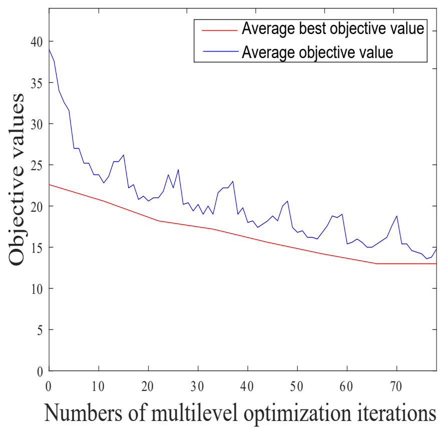
(a) DSJC1000.5.col



(b) flat1000_76_0.col



(c) R1000.5.col



(d) wap01a.col

Fig. 4. Convergence charts (running profiles) of SDMA on four representative instances (DSJC1000.5.col, flat1000_76_0.col, R1000.5.col, wap01a.col).

For each run, we collect the following statistics. For each multilevel optimization iteration (lines 4-15, Algorithm 1), we record the best objective value and $2 \times L$ current objective values corresponding to the objectives of L coarsening iterations (lines 6-10, Algorithm 1) and L uncoarsening iterations (lines 11-15, Algorithm 1), where L is the level limit (see Algorithm 1 and Table 1). Figure 4 shows the running behaviors of SDMA where the X-axis indicates the numbers of multilevel optimization iterations for each graph and the Y-axis shows the average best objective values and average current objective values along the multilevel iterations.

From Figure 4, we observe the following general trends. The algorithm improves its solution quality significantly and quickly in the beginning of the search. Then the improvement slows down and the algorithm progressively converges to its best solution. For the three random (dense) graphs (DSJC1000.5.col, flat1000_76_0.col, R1000.5.col), the quality of the current solution fluctuates greatly during the multilevel iterations, allowing the algorithm to escape many local optimal solutions to reach solutions of increasing quality. For the sparse wap01a.col graph from real wavelength assignment, due to its specific structures, the improvement of both the average best objective value and the average objective value is more regular and the search fluctuates much less. This provides an interesting indicator for the high performance of the SDMA algorithm on this graph (and the other graphs of this family). Note that due to its large size and structure, the number of multilevel iterations performed by the algorithm within 3600 seconds on this graph is much smaller than on the other graphs.

These convergence profiles are typical and representative for the tested graphs, even if the exact curves and convergence points vary according to the studied instances, as the four curves show.

5 Conclusions and perspectives

We investigated the first solution-driven multilevel approach for solving the well-known graph coloring problem. Contrary to the conventional multilevel approach where the coarsening phase is performed based on structural information of the given graph, we adopted an innovative coarsening strategy that merges vertices based on the solution provided by the refinement procedure. Another new feature of the proposed algorithm is to apply the refinement procedure to improve each coarsened or uncoarsened graph. This ensures that the solution is continually ameliorated all along both the coarsening phase and the uncoarsening phase. Finally, for solution refinement, we devised an effective weight tabu coloring algorithm that generalizes the popular TabuCol procedure to the case of weight graphs.

We presented extensive computational assessments on 47 well-known benchmark graphs and extensive comparative studies with 13 reference algorithms in the literature. The computational results showed that the proposed SDMA algorithm is particularly effective for solving structured (sparse) graphs while other leading algorithms are more suitable for (large) random instances. As such, SDMA provides a promising alternative that enriches the toolkit for graph coloring and could contribute to a portfolio approach with an ensemble of solvers and an intelligent decision-maker on the basis of instance structures.

Finally, given that the underlying optimization procedure (i.e., weight tabu coloring procedure) is rather simple compared with the leading state-of-the-art algorithms, this work suggests that the proposed approach could be further improved by adopting more powerful refinement procedures. Although the multilevel approach has been applied to many problems, this is the first work demonstrating the potential of using solutions to guide the coarsening process. As such, one promising direction for future work is to investigate this solution-driven multilevel approach to other applications including those that have been solved with the conventional multilevel approach.

Declaration of competing interest

The authors declare that they have no known competing interests that could have appeared to influence the work reported in this paper.

Acknowledgment

We are grateful to the reviewers for their useful comments and suggestions that helped us to significantly improve the paper. Support from the Natural Science Foundation Program of Jiangsu Province [Grant No. SBK2020040023] for the first author is also acknowledged.

References

[1] M. R. Garey, D. S. Johnson, Computers and intractability, Vol. 174, freeman San Francisco, 1979.

[2] G. J. Chaitin, Register allocation & spilling via graph coloring, in: ACM Sigplan Notices, Vol. 17, ACM, 1982, pp. 98–105.

[3] D. de Werra, An introduction to timetabling, European Journal of Operational Research 19 (2) (1985) 151–162.

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

[5] F. T. Leighton, A graph coloring algorithm for large scheduling problems, Journal of research of the national bureau of standards 84 (6) (1979) 489–506.

[6] M. Kubale (Ed.), Graph colorings, Vol. 352 of Contemporary Mathematics, American Mathematical Society, 2004.

[7] R. Lewis, A Guide to graph colouring: algorithms and applications, Springer, 2015.

[8] W. Meyer, Equitable coloring, The American Mathematical Monthly 80 (8) (1973) 920–922.

[9] S. K. Das, I. Finocchi, R. Petreschi, Conflict-free star-access in parallel memory systems, Journal of Parallel and Distributed Computing 66 (11) (2006) 1431–1441.

[10] J. Blazewicz, K. H. Ecker, E. Pesch, G. Schmidt, J. Weglarz, Scheduling computer and manufacturing processes, Journal of the Operational Research Society 48 (6) (1997) 659–659.

[11] H. Furmańczyk, Equitable coloring of graph products, Opuscula Mathematica 26 (1) (2006) 31–44.

[12] C. C. Ribeiro, M. Minoux, M. C. 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) (1989) 232–239.

[13] H. Gavranovic, G. Finke, Graph partitioning and set covering for the optimal design of a production system in the metal industry, IFAC Proceedings Volumes 33 (17) (2000) 603–608.

[14] D. S. Hochbaum, D. Landy, Scheduling semiconductor burn-in operations to minimize total flowtime, Operations Research 45 (6) (1997) 874–885.

[15] K. I. Aardal, S. P. Van Hoesel, A. M. Koster, C. Mannino, A. Sassano, Models and solution techniques for frequency assignment problems, Annals of Operations Research 153 (1) (2007) 79–129.

[16] M. Malafiejski, Sum coloring of graphs, in: M. Kubale (Ed.), Graph Colorings, Chapter 4, Vol. 352 of Contemporary Mathematics, American Mathematical Society, 2004, pp. 55–66.

[17] P. Galinier, J.-P. Hamiez, J.-K. Hao, D. Porumbel, Recent advances in graph vertex coloring, in: Handbook of Optimization, Springer, 2013, pp. 505–528.

[18] P. Galinier, A. Hertz, A survey of local search methods for graph coloring, Computers & Operations Research 33 (9) (2006) 2547–2562.

[19] E. Malaguti, P. Toth, A survey on vertex coloring problems, International Transactions in Operational Research 17 (1) (2010) 1–34.

[20] D. Brélaz, New methods to color the vertices of a graph, Communications of the ACM 22 (4) (1979) 251–256.

[21] A. Hertz, D. de Werra, Using tabu search techniques for graph coloring, Computing 39 (4) (1987) 345–351.

[22] M. Caramia, P. Dell Olmo, Coloring graphs by iterated local search traversing feasible and infeasible solutions, Discrete Applied Mathematics 156 (2) (2008) 201–217.

[23] A. Hertz, M. Plumettaz, N. Zufferey, Variable space search for graph coloring, Discrete Applied Mathematics 156 (13) (2008) 2551–2560.

[24] Y. Zhou, B. Duval, J.-K. Hao, Improving probability learning based local search for graph coloring, Applied Soft Computing 65 (2018) 542–553.

[25] P. Galinier, J.-K. Hao, Hybrid evolutionary algorithms for graph coloring, Journal of Combinatorial Optimization 3 (4) (1999) 379–397.

[26] P. Galinier, A. Hertz, N. Zufferey, An adaptive memory algorithm for the k-coloring problem, Discrete Applied Mathematics 156 (2) (2008) 267–279.

[27] E. Malaguti, M. Monaci, P. Toth, A metaheuristic approach for the vertex coloring problem, INFORMS Journal on Computing 20 (2) (2008) 302–316.

[28] Z. Lü, J.-K. Hao, A memetic algorithm for graph coloring, European Journal of Operational Research 203 (1) (2010) 241–250.

[29] D. C. Porumbel, J.-K. Hao, P. Kuntz, An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring, Computers & Operations Research 37 (10) (2010) 1822–1832.

[30] O. Titiloye, A. Crispin, Quantum annealing of the graph coloring problem, Discrete Optimization 8 (2) (2011) 376–384.

[31] L. Moalic, A. Gondran, Variations on memetic algorithms for graph coloring problems, Journal of Heuristics 24 (1) (2018) 1–24.

[32] B. Hendrickson, R. W. Leland, A multilevel algorithm for partitioning graphs, in: S. Karin (Ed.), Proceedings 1995 ACM/IEEE Conference on Supercomputing, San Diego, USA, ACM, 1995, pp. 28–es.

[33] C. Walshaw, Multilevel refinement for combinatorial optimisation problems, Annals of Operations Research 131 (1-4) (2004) 325–372.

[34] U. Benlic, J.-K. Hao, A multilevel memetic approach for improving graph k-partitions, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 624–642.

[35] R. Shaydulin, J. Chen, I. Safro, Relaxation-based coarsening for multilevel hypergraph partitioning, Multiscale Modeling & Simulation 17 (1) (2019) 482–506.

[36] C. Walshaw, A multilevel algorithm for force-directed graph-drawing, Journal of Graph Algorithms and Applications 7 (3) (2006) 253–285.

[37] H.-R. Fang, S. Sakellaridi, Y. Saad, Multilevel manifold learning with application to spectral clustering, in: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, 2010, pp. 419–428.

[38] I. Dhillon, Y. Guan, B. Kulis, A fast kernel-based multilevel algorithm for graph clustering, in: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM, 2005, pp. 629–634.

[39] E. Sharon, M. Galun, D. Sharon, R. Basri, A. Brandt, Hierarchy and adaptivity in segmenting visual scenes, Nature 442 (7104) (2006) 810.

[40] G.-J. Nam, J. J. Cong, Modern circuit placement: Best practices and results, Springer Science & Business Media, 2007.

[41] W. W. Hager, J. T. Hungerford, I. Safro, A multilevel bilinear programming algorithm for the vertex separator problem, Computational Optimization and Applications 69 (1) (2018) 189–223.

[42] C. Walshaw, A multilevel approach to the graph colouring problem, Citeseer, 2001.

[43] F. Glover, M. Parker, J. Ryan, Coloring by tabu branch and bound, DIMACS Series in Discrete Mathematics and Theoretical Computer Science 26 (1996) 285–307.

[44] R. Dorne, J.-K. Hao, A new genetic local search algorithm for graph coloring, in: International Conference on Parallel Problem Solving from Nature, Springer, 1998, pp. 745–754.

[45] A. Koster, Wavelength assignment in multifiber wdm networks, Proceedings of INOC 2005 (2005) 60–66.

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

[47] O. Titiloye, A. Crispin, Parameter tuning patterns for random graph coloring with quantum annealing, PloS one 7 (11) (2012) e50060.

[48] J.-K. Hao, Q. Wu, Improving the extraction and expansion method for large graph coloring, Discrete Applied Mathematics 160 (16-17) (2012) 2397–2407.

[49] S. Held, W. J. Cook, E. C. Sewell, Safe lower bounds for graph coloring, in: O. Günlük, G. J. Woeginger (Eds.), Integer Programming and Combinatoral Optimization - 15th International Conference, New York, USA, Proceedings, Vol. 6655 of Lecture Notes in Computer Science, Springer, 2011, pp. 261–273.