An extraction and expansion approach for graph coloring

Qinghua Wu and Jin-Kao Hao*
LERIA, Université d'Angers
2 Bd Lavoisier, 49045 Angers, France
Email: {wu,hao}@info.univ-angers.fr
Accepted to *Asian-Pacific Journal of Operational Research*
May 1, 2011

Abstract

This paper presents an extraction and expansion approach for the graph coloring problem. The extraction phase transforms a large graph into a sequence of progressively smaller graphs by removing large independent sets from the graph. The expansion phase starts by generating an approximate coloring for the smallest graph in the sequence. Then it expands the smallest graph by progressively adding back the extracted independent sets and determine a coloring for each intermediate graph. To color each graph, a simple perturbation based tabu search algorithm is used. The proposed approach is evaluated on the DIMACS challenge benchmarks showing competitive results in comparison with the state-of-the-art methods.
Keywords: graph coloring, independent set, tabu search, progressive optimization.

1 Introduction

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. A subset I of V is an independent set if no two adjacent vertices belong to I. A legal k-coloring of G is a partition of V into k independent sets (color classes). The graph k-coloring problem is to find a legal k-coloring of G for

*Corresponding author

a given k. The graph coloring problem is to determine the smallest integer k (its chromatic number $\chi(G)$) such that there exists a legal k-coloring of G.

The graph coloring problem is a well-known NP-hard problem [17] with real-word applications in many engineering fields, including scheduling [23], register allocation [7], timetabling [4, 30], and frequency assignment [29]. Exact solution methods are useful to solve problems of relatively small size whereas heuristics and metaheuristics are usually preferred to handle larger graphs.

There are a large number of heuristic algorithms for graph coloring including the following approaches: greedy construction (DSATUR [3], RLF[23]), tabu search [2, 8, 19, 27], iterated local search [6], variable neighborhood search [1], simulated annealing [5, 21], variable space search [20], multiagent fusion search [32] and evolutionary hybrid or population based search [9, 13, 15, 24, 25, 28]. A comprehensive survey of the most significant heuristic methods can be found in [14].

Another approach for dealing with large graphs is based on a general principle of "reduce-and-solve". This approach consists in applying, prior to the phase of graph coloring, a preprocessing procedure to extract large independent sets from the original graph until the remaining graph (called residual graph) becomes sufficiently small and then coloring the residual graph. The resulting color classes of the residual graph plus the extracted independent sets form a solution for the initial graph. Typically, the preprocessing procedure reduces the initial graph by removing one independent set at a time. Such an approach was described for example in early studies [11, 19, 21, 26].

This paper presents an extraction-and-expansion approach (denoted by E2COL) to handle large and very large graphs. The proposed approach also extracts independent sets to create a sequence of progressively smaller graphs to reduce the initial graph. However, E2COL goes beyond the basic preprocessing approach by introducing an advanced extraction strategy. More importantly, E2COL reconsiders the extracted independent sets during its expansion phase in order to improve the colorings of the intermediate graphs. This is motivated by the fact that not all vertices of an extracted independent set necessarily get the same color in the final coloring.

Experiments on a set of large DIMACS challenge benchmark graphs show that the proposed approach is very competitive. Indeed, E2COL is able to match most of the best-known solutions and for two large instances (C2000.5 and C4000.5), it even finds improved solutions with 1 and 9 fewer colors. E2COL also competes favorably with the currently top-performing hybrid evolutionary methods.

The rest of this paper is organized as follows. Section 2 describes the general framework and the main components of the proposed E2COL algorithm.

In Section 3, computational results are presented and compared. Section 4 investigates the role of the expansion phase, followed by the concluding section.

2 The extraction and expansion approach for graph coloring

2.1 Problem description

Given an integer k and a graph $G = (V, E)$ with vertex set V and edge set E, a k-coloring of G is a partition of V into k distinct color classes $\{C_1, C_2, ..., C_k\}$ such that each color class C_i $(i = 1, 2, ..., k)$ contains all the vertices that are colored with color i.

If two adjacent vertices u and v are in the same color class, u and v are called conflicting vertices and the edge linking u and v is called a conflicting edge. A color class without conflicting vertices is an independent set. A k-coloring without conflicting edges is said to be a legal k-coloring and corresponds to a partition of the vertices of V into k independent sets. A k-coloring with conflicting edges is said to be an illegal k-coloring.

For a given k-coloring $s = \{C_1, C_2, ..., C_k\}$ (legal or illegal), let the evaluation function $f(s)$ denote the number of conflicting edges induced by s such that:

$$f(s) =\mid \{\{u, v\} \in E : \exists C_i \in s,\ u \in C_i, v \in C_i\} \mid \tag{1}$$

Then the k-coloring problem can be solved by minimizing the function $f(s)$; $f(s) = 0$ implies that all the color classes C_i of s are independent sets, i.e. s is a legal k-coloring. The chromatic number of G can be approximated by solving a series of k-coloring problems with decreasing values of k until no legal k-coloring can be obtained. In Section 2.4, we describe the coloring algorithm used in this work.

2.2 General procedure

As observed in [5, 19, 21], it is difficult to color a large graph by using pure local search algorithms. One possible approach to improve the situation is to first extract several large independent sets from the original graph and then color the reduced residual graph.

The proposed approach in this paper reinforces this basic idea of extracting independent sets by two improvements. First, we develop an advanced

method for the extraction phase which identifies and extracts as many pairwise disjoint independent sets as possible. Second, we introduce an expansion phase which optimizes the colorings of a series of intermediate graphs by reconsidering the extracted independent sets. The general procedure of the proposed E2COL approach can be summarized by the following steps.

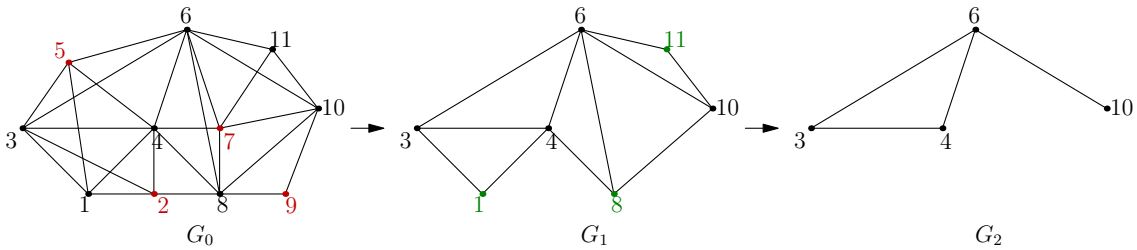

Figure 1: The extraction phase

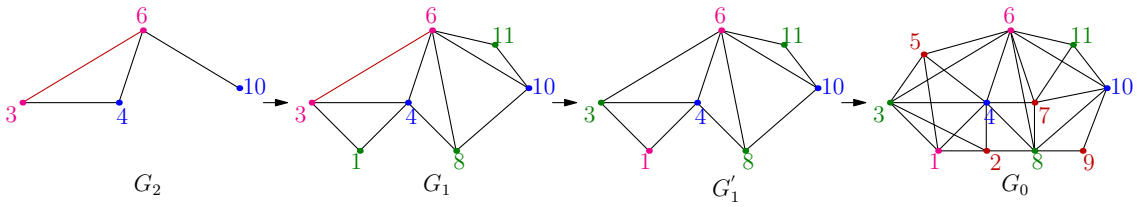

Figure 2: The expansion phase

1. Extraction phase: The initial graph $G_0 = G$ is transformed into a sequence of progressively smaller graphs $G_1, G_2, ..., G_m$ by removing large independent sets from the original graph. More precisely, each $G_i = \{V_i, E_i\}$ $(i = 1, ..., m)$ is created by removing a large independent set I_{i-1} from its upper level graph G_{i-1} (see Fig. 1). Obviously, G_i is a subgraph of G_{i-1} and $V_i = V_{i-1} - I_{i-1}$. This phase stops when the size of V_m reaches a fixed threshold q. We will present the method for constructing $I_0, ..., I_{m-1}$ in Section 2.3.

2. Initial coloring phase: A coloring algorithm is applied to the smallest graph $G_m = (V_m, E_m)$ to determine a coloring with $k - m$ colors. Let $s'_m = \{C_1, ..., C_{k-m}\}$ be this coloring. If s'_m is a legal $(k - m)$-coloring (i.e. $f(s'_m) = 0$), s'_m plus the m extracted independent sets (i.e. $\{C_1, ..., C_{k-m}, I_{m-1}, ..., I_0\}$) gives a legal k-coloring for the initial graph G_0, the initial k-coloring problem is solved. Otherwise (i.e. $f(s'_m) > 0$), s'_m is used to construct an initial $(k - m + 1)$-coloring for the immediate upper graph G_{m-1} by the expansion phase.

3. Expansion phase: This phase carries out the inverse of the extraction phase and traverses the sequence of intermediate graphs in the reverse order. At the first step, the solution $s'_m = \{C_1, ..., C_{k-m}\}$ of G_m is transformed into an initial $(k - m + 1)$-coloring s_{m-1} of G_{m-1} such that $s_{m-1} = \{C_1, ..., C_{k-m}, I_{m-1}\}$ (i.e. all vertices in I_{m-1} form a new color class in s_{m-1}, see Fig. 2 from G_2 to G_1). Obviously, $f(s_{m-1}) = f(s'_m)$. s_{m-1} is then improved by the coloring algorithm (Section 2.4) and let s'_{m-1} be the resulting coloring of G_{m-1}. If s'_{m-1} is a legal $(k-m+1)$-coloring for G_{m-1}, then a legal k-coloring of G_0 is obtained by joining s'_{m-1} and the independent sets $I_0, I_1...I_{m-2}$. Otherwise, s'_{m-1} is transformed into an initial solution s_{m-2} for coloring the immediate upper level graph G_{m-1}. The same operations are performed on the intermediate graphs $G_{m-3}, G_{m-4}, ...$ in the sequence until a legal $(k-i)$-coloring of a graph G_i is found or the uppermost graph G_0 in the sequence is colored.

Figures 1 and 2 illustrate how our proposed approach works. In this example, we want to color the initial graph G_0 with $k = 4$ colors. In Fig. 1, we obtain two smaller graphs G_1, G_2 by successively extracting two independent sets $I_1 = \{2, 5, 7, 9\}$ and $I_2 = \{1, 8, 11\}$ from G_0. The smallest graph G_2 is then colored with 2 colors and we get an initial (and illegal) 2-coloring solution s'_2 of G_2 (Fig. 2, G_2). Using s'_2, we construct an initial (and illegal) 3-coloring solution s_1 of G_1. s_1 is then improved by the coloring algorithm and a legal 3-coloring s'_1 of G_1 is obtained (Fig. 2, from G_1 to G'_1). At last, we use the legal 3-coloring s'_1 to construct a legal 4-coloring of G_0 (Fig. 2, from G'_1 to G_0).

The proposed extraction and expansion approach is described in Algorithm 1. The expansion process thus leads to a sequence of colorings s'_m, s'_{m-1}, s'_{m-2},...with progressively smaller (thus improved) f values (i.e. $f(s'_m) \geq f(s'_{m-1}) \geq f(s'_{m-2}) \geq ...$). As the expansion-coloring process proceeds, the coloring quality s'_{i-1} of graph G_{i-1} is usually better than that of G_i because there is a greater degree of freedom to improve the coloring.

The computational complexity of the proposed E2COL algorithm depends essentially on the tasks of extracting independent sets (see Section 2.3) and coloring the residual and intermediate graphs (see Section 2.4). For each of these tasks, a tabu search algorithm is employed whose complexity is bounded by the allowed maximum computing time or iterations.

Algorithm 1 The Extraction-Expansion algorithm for k-coloring

Input: An undirected graph $G_0 = (V_0, E_0)$, An integer k
Output: A legal k-coloring of G_0 or report failure
{Extraction phase}
Use the extracting procedure to get a sequence of independent sets $I_0, ..., I_t$ (Section 2.3)
$i := 0$
while ($|V_i| > q$) **do**
    $G_{i+1} = Remove(G_i, I_i)$ {Remove all vertices of I_i from G_i}
    $i = i + 1$
end while
{Initial coloring phase}
$s'_i = Tabu\_Search(G_i, k - i)$ {Use tabu search to color the smallest graph G_i with $k - i$ colors (Section 2.4), s'_i is the best coloring found in terms of the function f}
if (s'_i is a legal $(k - i)$-coloring, i.e. $f(s'_i) = 0$) **then**
    Use s'_i to construct a legal k-coloring $s_0 = \{s'_i, I_{i-1}, ..., I_0\}$ of G_0
    Return s_0 and Stop
end if
{Expansion phase}
while ($i > 0$) **do**
    $i = i - 1$
    $s_i = \{s'_{i+1}, I_i\}$ {Construct a $(k - i)$-coloring of G_i from $s'_{i+1} = \{C_1, ..., C_{k-i-1}\}$ }
    $s'_i = Perturbation\_Tabu\_Search(s_i, G_i)$ {Section 2.4}
    if (s'_i is a legal $(k - i)$-coloring, i.e. $f(s'_i) = 0$) **then**
        Use s'_i to construct a legal k-coloring $s_0 = \{s'_i, I_{i-1}, ..., I_0\}$ of G_0
        Return s_0 and Stop
    end if
    Return (No legal k-coloring found)
end while

2.3 The procedure for extracting independent sets

As stated previously, in order to create a sequence of progressively smaller graphs $G_0 = G, G_1, ..., G_m$ (which are kept in a stack), we identify from the initial graph m pairwise disjoint independent sets $I_0, ..., I_{m-1}$. We explain in this section how this is achieved.

2.3.1 The extraction procedure

Conventional extraction procedures identify only one large independent set at each preprocessing step (see for example [19, 11, 21, 26]). With our independent set extraction procedure, each time a large independent set I is identified from a given graph, we try to find as many pairwise disjoint independent sets of the same size $|I|$ as possible before moving to the next step. This process is repeated until there are no more than q vertices left in the residual graph.

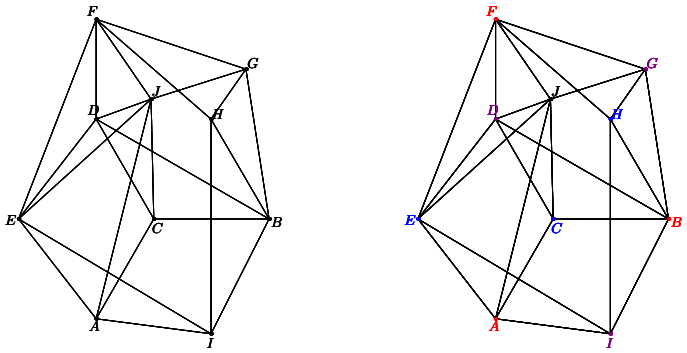

Figure 3: An illustration of the extraction procedure

Figure 3 illustrates how the extraction procedure works on a graph with 10 vertices. At the first step, we find a maximum independent set of size 3 (e.g. $\{A, D, H\}$). Then we identify as many independent sets of size 3 as possible from the graph (say 8). Finally, we determine from them 3 disjoint independent sets $\{A, B, F\}, \{C, E, H\}, \{D, G, I\}$ and remove these 3 independent sets from the graph. Since there is only one vertex left (vertex J) after removing these independent sets, the procedure stops.

Our extracting procedure iterates basically the following 4 steps.

1. Identify a maximal independent set I in the graph $G = (V, E)$ and initialize set M with I. Identification of I can be achieved by applying any maximum independent set algorithm (or equivalently any maximum clique algorithm). In our case, we employ the Adaptive Tabu Search (ATS) algorithm described in [33].

2. Apply repeatedly the ATS algorithm to generate as many independent sets of size $|I|$ as possible and add them in set M. The search for new independent sets of size $|I|$ stops when the number of independent sets contained in M reaches a desired threshold (M_{max}) or when no new independent set of that size can be found. According to our experiments we set M_{max} equal to $|V| * \rho$, where ρ is the density of the graph.

3. Identify as many *pairwise disjoint* independent sets $I_0^{'}, ..., I_{l-1}^{'}$ as possible from M (i.e. $\forall I_x', I_y' \in \{I_0^{'}, ..., I_{l-1}^{'}\}$, $x \neq y \rightarrow I_x' \bigcap I_y' = \emptyset$ and l is as large as possible. We explain how this is achieved below) and record them in $\mathcal{H}$. Remove the disjoint independent sets of $\mathcal{H}$ from G and record each resulting intermediate residual graph in $\mathcal{R}$. ($\mathcal{H}$ and $\mathcal{R}$ will be used when the intermediate graphs are colored during the expansion phase).

4. Repeat steps 1–3 until the residual graph G contains no more than q vertices.

Step (3) aims at finding a maximum number of disjoint independent sets from M. This problem, called the maximum set packing problem, is equivalent to the maximum clique (thus the maximum independent set) problem [17]. Let $M = \{I_1, ..., I_n\}$ be the given set of independent sets, we define a new graph $G' = (V', E')$ where $V' = \{1, ..., n\}$ and the edge matrix is given by:

$$e_{ij} = \begin{cases} 0, & \text{if } I_i \cap I_j = \emptyset, i, j \in \{1, ..., n\} \\ 1, & \text{otherwise.} \end{cases}$$

One observes that if $\{i_1, ..., i_r\} \subset V'$ is an independent set in G', $\{I_{i_1}, ..., I_{i_r}\}$ is a pairwise disjoint set in M. Consequently, to obtain a maximum pairwise disjoint set in M, we can determine a maximum independent set in G'. For this purpose, we apply again the ATS algorithm from [33].

2.4 Graph coloring by perturbation based tabu search

2.4.1 The tabu search for the k-coloring problem

In this paper, we employ the tabu search coloring algorithm described in [13] which itself is based on the seminal tabucol algorithm presented in [19]. This algorithm uses a search space which is defined by the set of all possible k-colorings for the given graph and integer k. For a given k-coloring s, its quality is assessed by the evaluation function f defined by formula (1) which measures the total number of conflicting edges (see Section 2.1). A neighboring solution of a given k-coloring is obtained by moving a conflicting

vertex u from its original color class C_i to another color class C_j ($i \neq j$). When a vertex u is moved from color class C_i to color class C_j, color class C_i is declared tabu to vertex u for the next l iterations (called tabu tenure).

At each iteration, our tabu search determines the best neighbor s' of the current solution s (ties are broken randomly) such that either s' is a non-tabu solution, or s' leads to a solution better than the best solution found so far (aspiration criterion). The tabu tenure is dynamically determined by $f(s) + r(10)$, where $r(10)$ returns a random number in $\{1, 2, ..., 10\}$.

The tabu search returns the most recent best solution found during the search. Hence, if a solution is encountered with the same f value as the current best solution, then the current best solution is replaced by the new one. The stop condition of our tabu search is just the allowed number of iterations with which the best solution has not been improved. We call this number depth of tabu search, denoted by α.

Algorithm 2 The perturbation based tabu search algorithm

Require: Graph $G = (V, E)$, initial coloring s, integers *max-tries* and *max-runs*
Ensure: the best coloring s^* found so far

```
1: Begin
2: i := 0
3: s* := s {s* records the best coloring found so far}
4: while (i < max-tries) do
5:    s_p ← Perturb(s*, t)
6:    i = i + 1
7:    j = 0
8:    while (j < max-runs) do
9:       s' ← Tabu_Search(s_p, t)
10:      if (f(s') ≤ f(s*)) then
11:         s* = s'
12:      end if
13:      if (f(s*) = 0) then
14:         Return s* and Stop
15:      end if
16:      j = j + 1
17:   end while
18: end while
19: End
```

2.4.2 The perturbation based tabu search algorithm

To reinforce the search capacity of this basic tabu search algorithm, we employ a perturbation strategy. Basically, the tabu search ensures the intensification, while the perturbation mechanism brings controlled diversification

into the search. The general procedure of the perturbation based tabu search is summarized in Algorithm 2.

The perturbation to a given solution s^* (i.e. a conflicting coloring using a certain number k of colors, line 5 of Algorithm 2) operates as follows: 1) select $t \in \{1, 2, 3\}$ color classes $C'_1, ..., C'_t$ from s^*, 2) empty $C'_1, ..., C'_t$ by moving all vertices in $C'_1, ..., C'_t$ into the other $k - t$ color classes of s^* and 3) run the tabu search algorithm to obtain an improved $(k - t)$-coloring s_p. After this perturbation operation, we add back the t empty color classes to s_p from which we run again tabu search to try to find a conflict-free coloring with k colors (lines 8-17 of Algorithm 2). The value of t is determined as follows: if $k < 40$, t is set equal to 1, if $40 \leq k < 80$, t is set equal to 2, otherwise t is set to 3. Notice that after each perturbation, a long tabu search instead of the multiple restart of tabu search (inner **while** loop of Algorithm 2) can be applied to improve the perturbed solution. However, our experiments showed that the multiple restart strategy performs globally better.

3 Experimental Results

In this section, we report intensive experimental results of our E2COL algorithm on the well-known DIMACS coloring benchmarks. We also compare the results with some state-of-the-art coloring algorithms from the literature.

3.1 Problem Instances and Experimental Protocol

As our E2COL algorithm is devised for coloring large and very large graphs, we evaluate our algorithm on 17 large graphs with at least 400 vertices from the DIMACS Challenge [22]. We select those graphs because they are known to be among the most challenging ones. The considered graphs are described below.

1. Six classical random graphs ($dsjcA.B$, $A = 500$ and 1000, $B = 0.1$, 0.5 and 0.9) with unknown chromatic numbers.

2. Four Leighton graphs ($le450\_15x$, $le450\_25x$, $x = c$ and d) with known chromatic numbers.

3. Three $flat$ graphs (flatn_χ_0, $n = 1000$ and $\chi = 50$, 60 and 76) with known chromatic numbers.

4. Two random geometric graphs ($DSJR500.1c$ and $DSJR500.5$) with unknown chromatic numbers.

Table 1: Settings of important parameters for E2COL

Parameters	Section	Description	Values
q	2.2	the size threshold for G_m	300
$max-tries$	2.4.2	the maximum number of perturbations	10
$max-runs$	2.4.2	the maximum number of tabu search runs after one perturbation	10
α	2.4.1	depth of tabu search	100,000

5. Two huge random graphs (C2000.5 and C4000.5) with unknown chromatic numbers.

All experiments for this study were performed on a dedicated PC with 2.8 GHz CPU and 2G RAM. For all the tested graphs, the same parameter values (see Table 1) are used which are determined within a preliminary experiment. To obtain our computational results, for each instance, 10 independent trials were performed with different random seeds. The algorithm stops if one of the following conditions is verified:

1. A legal k-coloring is found.

2. The processing time reaches its timeout limit. The timeout limit is set to be 5 CPU hours except for four large graphs DSJC1000.5, DSJC1000.9, C2000.5 and C4000.5. For DSJC1000.5 and DSJC1000.9 a limit of 10 hours is allowed while for the two largest graphs C2000.5 and C4000.5, the limit is set equal to 5 days. Notice that these timeout limits are comparable with those reported in the latest papers on large graph coloring like [24, 25, 28, 32].

3. The uppermost level graph G_0 in the sequence of the graphs is reached, in this case, we repeat using the perturbation based TS to optimize s'_0 until the timeout limit is reached or a legal k-coloring is found.

3.2 Computational Results

Table 2 summarizes the computational statistics of our E2COL algorithm on the large DIMACS challenge instances. Columns 2-4 indicate the features of the tested instances: the number of vertices $Node$, the number of edges $Edge$ and the density of the graph $Density$. Column 5 displays the best known results k^* reported in the literature. In columns 6-9, the computational statistics of our E2COL algorithm are presented, including the number of colors obtained k, the success rate (hit) and the average computation time in minutes for reaching the given k. The last column shows the average number of iterations for the successful runs.

Table 2: Computational results of E2COL on the set of difficult DIMACS challenge benchmarks.

Instance	*Node*	*Edge*	*Density*	k^*	*E2COL*			
					k	*hit*	*time(m)*	*Iterations*
DSJC500.1	500	12458	0.1	12	**12**	10/10	15	2.2×10^7
DSJC500.5	500	62624	0.5	48	**48**	10/10	55	4.2×10^7
DSJC500.9	500	224874	0.9	126	**126**	10/10	136	1.8×10^8
DSJC1000.1	1000	49629	0.1	20	**20**	10/10	45	5.2×10^7
DSJC1000.5	1000	249826	0.5	83	**83**	4/10	316	7.2×10^8
					84	10/10	26	5.3×10^7
DSJC1000.9	1000	449449	0.9	223[a]	224(223)	6/10	356	6.7×10^8
					225	10/10	76	1.8×10^8
DSJR500.1c	500	121275	0.97	85	**85**	8/10	238	1.8×10^8
					86	10/10	56	5.5×10^7
DSJR500.5	500	58862	0.47	122	123	1/10	287	2.1×10^8
					124	10/10	197	1.5×10^8
flat1000_50_0	1000	245000	0.49	50	**50**	10/10	25	1.2×10^6
flat1000_60_0	1000	245830	0.49	60	**60**	10/10	25	1.7×10^6
flat1000_76_0	1000	246708	0.49	82	**82**	10/10	176	3.5×10^8
le450_15c	450	16680	0.16	15	**15**	10/10	15	1.5×10^7
le450_15d	450	16750	0.17	15	**15**	10/10	16	1.8×10^7
le450_25c	450	17343	0.17	25	**25**	10/10	88	8.5×10^7
le450_25d	450	17425	0.17	25	**25**	10/10	94	9.1×10^7
C2000.5	2000	999836	0.5	148[b]	**147**	10/10	720	1.1×10^9
C4000.5	4000	4000268	0.5	271[b]	**262**	10/10	1520	1.8×10^9

Note *a*: For DSJC1000.9, 222-colorings were independently reported very recently in [31, 34].
Note *b*: For C2000.5 and C4000.5, 146-colorings and 260-colorings were reported very recently in [34].

From Table 2, we observe that the results obtained by E2COL are competitive when comparing to the previous best known results reported in the literature (column 5 in Table 2). For the two huge random graphs C2000.5 and C4000.5, colorings with respectively $k = 148$ and 271 were reported recently in [28] by a hybrid evolutionary algorithm using a time limit of 120 hours. It is interesting to observe that E2COL is able to obtains colorings with $k = 147$ and 262 respectively, leading to a gain of 1 and 9 colors.

For the six standard random graphs (DSJC), our E2COL algorithm can reach the previous best known results except for DSJC1000.9, for which E2COL needs 224 colors within the allowed time limit while the best known result requires 223 colors, which was reported recently in [24, 27, 28, 32]. Yet, E2COl is able to reach a 223-coloring when the timeout limit is set to 5 days.

For the three flat graphs (flat) and the four Leighton graphs (le450), E2COL finds a k-coloring on every run, with k equal to the smallest known value. However, for the graph DSJR500.5, our coloring requires one more color than the current best coloring.

Table 3: Comparisons between E2COl and 13 state of the art coloring algorithms.

Graph	k^*	k_{best}	local search algorithms				hybrid algorithms								
			[20]	[2]	[6]	[8]	[24]	[28]	[25]	[32]	[12]	[10]	[26]	[15]	[13]
DSJC500.1	12	12	12	12	12	13	12	12	12	-	12	-	-	12	-
DSJC500.5	48	48	48	48	49	50	48	48	48	48	49	49	49	48	48
DSJC500.9	126	126	126	127	126	127	126	126	127	-	127	-	-	126	-
DSJC1000.1	20	20	20	20	-	21	20	20	20	-	21	-	-	20	20
DSJC1000.5	83	83	86	89	89	90	83	83	83	84	88	84	89	84	83
DSJC1000.9	223	224(223)	224	226	-	226	223	223	225	223	228	-	226	224	224
DSJR500.1c	85	85	85	85	-	-	85	85	85	85	85	85	85	86	-
DSJR500.5	122	123	125	125	124	-	122	122	122	-	122	130	123	127	-
flat1000_50_0	50	50	50	50	-	50	50	50	50	50	50	84	50	50	-
flat1000_60_0	60	60	60	60	-	60	60	60	60	60	60	84	60	60	-
flat1000_76_0	82	82	85	87	-	89	82	82	82	83	87	84	89	84	83
le450_15c	15	15	15	15	15	-	15	15	15	15	15	16	15	15	15
le450_15d	15	15	15	15	15	-	15	15	15	-	15	16	15	15	-
le450_25c	25	25	25	25	26	-	25	25	25	27	26	-	25	26	26
le450_25d	25	25	25	25	26	-	25	25	25	-	26	-	25	26	-
C2000.5	148	147	-	-	-	-	148	148	-	150	162	153	151	-	-
C4000.5	271	262	-	-	-	-	272	271	-	-	301	280	-	-	-

3.3 Comparison with other algorithm

In this section, we compare the results of our E2COL algorithm with 13 well-known coloring algorithms from the literature, including four local search algorithms and nine hybrid algorithms. For this experiment, we are interested in the quality criterion, i.e. the lowest value of k for which a k-coloring can be found. Table 3 presents the comparative results. Columns 2 and 3 recall the best known results (k^*) and the best results found by E2COL (k_{best}). Columns 4-16 give the best results reported by these reference algorithms.

From Table 3, we observe that the coloring algorithms based on local search can hardly compete with the hybrid evolutionary algorithms in terms of solution quality. However, it is interesting and remarkable to note that our E2COL algorithm which uses tabu search as its coloring algorithm competes very favorably with the top-performing hybrid evolutionary algorithms such as [13, 15, 24, 25, 28, 32]. Indeed, for very hard graphs such as DSJC1000.5 and flat1000_76_0, only some very recently proposed hybrid algorithms can reach the best known results. Yet our E2COL algorithm performs quite well on these instances, and it performs even better than the top-performing hybrid algorithms on the two huge instances C2000.5 and C4000.5.

Table 4: Effect of the expansion phase. CEACOL uses the same strategy as E2COL to remove disjoint independent sets, but disables the expansion phase.

Graph	*Den*	k^*	E2COL		CEACOL	
			k	*hit*	k	*hit*
DSJC500.5	0.5	48	48	10/10	48	7/10
DSJC1000.5	0.5	83	83	4/10	84	10/10
DSJC1000.9	0.9	223	224	8/10	227	2/10
DSJR500.1c	0.97	85	85	8/10	88	3/10
flat1000_76_0	0.49	82	82	10/10	82	5/10

4 Discussions

4.1 Effect of the expansion phase

As explained previously, one conventional approach for handling large graphs is to first extract several large independent sets from the original graph and then color the residual graph. This approach is used for instance in [19, 11, 21, 26]. However, for some difficult instances, even if we extract m large independent sets from the original graph, it is still difficult to find a legal $(k-m)$-coloring for the residual graph with k set close to $\chi(G)$. This is essentially due to the separation between the extraction phase and coloring phase. Indeed, if an independent set is wrongly extracted during the first phase, i.e. if the extracted independent set is not part of any legal k-coloring, the error can never be repaired.

Contrary to this conventional approach, with our proposed approach, the extracted independent sets participate in the coloring process of intermediate graphs. As such, a wrongly extracted independent set has a chance to be corrected during the expansion-coloring phase. To show the effect of the expansion phase, we carry out additional experiments on five instances (DSJC1000.5, DSJC1000.9, DSJC500.5, DSJC500.1c, flat1000_76_0) and show a comparison between E2COL and the conventional extraction based coloring approach (CEACOL).

To solve each instance, we run both E2COL and CEACOL 10 times, each run being given the same amount of computation time (10 hours). For CEACOL, the extraction procedure described in Section 2.3 is applied to extract large independent sets until there are at most q ($q = 300$) vertices left in the residual graph. Then we repeatedly apply the tabu search to the residual graph until the timeout limit is reached. More precisely, each time we

Table 5: Effect of the strategy for independent set removal. SE2COL removes at a time one (large) independent set.

Graph	*Den*	k^*	E2COL		SE2COL	
			k	*hit*	k	*hit*
DSJC500.5	0.5	48	48	10/10	48	2/10
DSJC1000.5	0.5	83	83	4/10	85	10/10
DSJC1000.9	0.9	223	224	8/10	226	1/10
DSJR500.1c	0.97	85	85	8/10	85	7/10
flat1000_76_0	0.49	82	82	10/10	83	2/10

observe that the best solution is not improved for α consecutive iterations (see Section 2.4.1), we restart the tabu search if the timeout limit is not reached. Table 4 shows the comparative results between E2COL and CEACOL for the chosen graphs. For both algorithms, we indicate the smallest k that are reached and the number of successful runs for reaching these smallest k.

From Table 4, we observe that, the conventional extraction based algorithm manages to reach the best known coloring on the two medium–density instances(DSJC500.5 and flat1000_76_0) with a success rate of 70% and 50% respectively. However, on the two high–density instances (DSJC1000.9 and DSJR500.1c), CEACOL produces bad results, which are far behind the best known results. In fact, for many other high–density instances, we observe from experiments that CEACOL obtains poor results. While considering the results obtained by our E2COL algorithm, one easily observes that E2COL dominates the conventional algorithm. This shows that the expansion phase does play an important role on the performance of our E2COL algorithm.

4.2 Effect of the disjoint independent set removal

We now turn our attention to an additional study to justify steps 2 and 3 of the process described in Section 2.3. More precisely, we create from E2COL a simplified version (called SE2COL) where the extraction phase removes *one* large independent set after the other (and not a collection of pairwise disjoint independent sets) until the residual graph reaches the threshold q; at this point, we apply the expansion phase like in E2COL. Thus SE2COL differs from E2COL only by the way it extracts the independent sets.

We run SE2COL on the five selected instances and report in Table 5 the computational statistics of E2COL and SE2COL. We observe that E2COL dominates SE2COL in terms of number of colors used or number of hits when the two algorithms use the same number of colors. In particular, the

Table 6: Influence of the size of residual graph (parameter q) on the performance of E2COL

Graph	k^*	$q = 200$			$q = 300$			$q = 400$		
		k	*hit*	*Iterations*	k	*hit*	*Iterations*	k	*hit*	*Iterations*
DSJC500.5	48	48	10/10	5.5×10^7	48	10/10	4.2×10^7	48	5/10	3.8×10^7
DSJC1000.5	83	83	5/10	6.4×10^8	83	4/10	7.2×10^8	83	1/10	6.3×10^8
DSJC1000.9	223	224	7/10	7.1×10^8	224	6/10	6.7×10^8	224	7/10	5.5×10^8
DSJR500.1c	85	85	7/10	2.3×10^8	85	8/10	1.8×10^8	85	9/10	1.6×10^8
flat1000_76_0	82	82	10/10	4.4×10^8	82	10/10	3.5×10^8	82	5/10	4.7×10^8

dominance is more pronounced on the three large and hard instances with 1000 vertices. Indeed, compared to E2COL, SE2COL requires 1 or 2 more colors to find a legal coloring. This experiment confirms clearly the advantage of the independent set extraction strategy of E2COL over the conventional strategy.

4.3 Effect of the size of residual graph

With E2COL, its extraction phase stops when no more than q vertices are left in the residual graph from which the initial coloring and expansion phases are launched. It is then interesting to know whether different values of q have a strong impact on the performance of E2COL. To this end, we run E2COL 10 times on each of the five selected instances with $q \in \{200, 400\}$ and show in Table 6 the computational results together with those of $q = 300$ used in the previous experiments. In addition to k and *hit*, we also indicate the average iterations needed to find a k-coloring.

From Table 6, we observe that these q values do not change the number of used colors. Nevertheless, E2COL with $q = 200$ and $q = 300$ reaches more stable results (higher hits), but may require more iterations than with $q = 400$. Therefore, it seems that a smaller q makes the algorithm more robust but also slower. This implies that there may not be an absolute best value for this parameter and that a compromise between robustness and speed could be possible.

4.4 Possible improvements

The E2COL algorithm could be further improved by following two directions. First, E2COL currently uses a somewhat basic coloring procedure (which is based on tabu search) to color each intermediate graph (Section 2.4). The

main advantage of this tabu based algorithm is its simplicity. However, it is less effective than most of the recent coloring algorithms. the E2COL algorithm could obtain a better performance if it uses a more powerful recent coloring algorithm.

Second, E2COL generates the sequence of graphs $G_1, ..., G_m$ by removing *one* independent set at a time (Section 2.2). By doing this, E2COL tends to generate a large number of intermediate graphs and consequently needs to apply the coloring procedure many times. To reduce the number of intermediate graphs, one could generate one intermediate graph by removing *all* the pairwise disjoint independent sets of a given size obtained at Step 3 of the extraction procedure (Section 2.3.1). Similarly, during the expansion phase, instead of adding back one color class (independent set) at a time, one can add a set of color classes to generate each upper level graph.

5 Conclusion

In this paper, we have presented E2COL, an original extraction and expansion approach for graph coloring. The extraction phase transforms a large graph into a sequence of progressively smaller graphs by extracting large independent sets from the graph, while the expansion phase colors the intermediate graphs in the reverse order by reconsidering the extracted independent sets.

We have shown that even using a basic tabu search coloring algorithm, the proposed approach obtains highly competitive results on a set of DIMACS challenge benchmark graphs. Indeed, our E2COL algorithm is able to find the best known results for most of the tested graphs. Moreover, it is even able to find improved results for the two largest graphs (C2000.5 and C4000.5) using respectively 1 and 9 fewer colors. The comparison with 13 reference algorithms allows us to better appreciate the performance of the proposed approach.

We have also presented an analysis on the role of the expansion phase which confirms clearly the interest of reconsidering the extracted independent sets for coloring the intermediate graph.

Most of the current top-performing algorithms for the graph coloring problem are hybrid algorithms that combine a local search with a population based method. The proposed extraction and expansion approach constitutes an interesting alternative approach.

Acknowledgment

We are grateful to the referees for their comments and questions which helped us to improve the paper. The work is partially supported by the "Pays de la Loire" Region (France) within the RaDaPop (2009-2013) and LigeRO (2010-2013) projects.

References

[1] C. Avanthay, A. Hertz, N. Zufferey, A variable neighborhood search for graph coloring. European Journal of Operational Research 151(2) (2003) 379–388.

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

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

[4] E.K. Burke, B. McCollum, A. Meisels, S. Petrovic, R. Qu, A graph-based hyper heuristic for timetabling problems. European Journal of Operational Research 176 (2007) 177–192.

[5] M. Chams, A. Hertz, D. de Werra, Some experiments with simulated annealing for coloring graphs. European Journal of Operational Research 32 (1987) 260–266.

[6] M. Chiarandini, T. Stützle, An application of iterated local search to graph coloring. In: D.S. Johnson, A. Mehrotra, M. Trick, editors, Proc. of the Computational Symposium on Graph Coloring and its Generalizations, Ithaca, New York, USA, 112–125, 2002.

[7] D. de Werra, C. Eisenbeis, S. Lelait, B. Marmol, On a graph-theoretical model for cyclic register allocation. Discrete Applied Mathematics 93(2-3) (1999) 191–203.

[8] R. Dorne, J.K. Hao, Tabu search for graph coloring, T-colorings and set T-colorings. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, 77–92, 1998.

[9] R. Dorne, J.K. Hao, A new genetic local search algorithm for graph coloring. Lecture Notes in Computer Science 1498 (1998) 745–754.

[10] C. Fleurent, J.A. Ferland, Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63 (1996) 437–461.

[11] C. Fleurent, J.A. Ferland, Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In: [22], (1996) 619–652.

[12] N. Funabiki, T. Higashino, A minimal-state processing search algorithm for graph coloring problems. IEICE Transaction Fundamentals E83-A (2000) 1420–1430.

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

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

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

[16] M.R. Garey, D.S. Johnson, H.C. So, An application of graph coloring to printed circuit testing. IEEE Transactions on Circuits and Systems 23 (1976) 591–599.

[17] M.R Garey, D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman and Company, San Francisco 1979.

[18] J.P. Hamiez, J.K. Hao, Scatter search for graph coloring. Lecture Notes in Computer Science 2310 (2002) 168–179, Springer-Verlag.

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

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

[21] D.S. Johnson, C.R. Aragon, L.A. McGeoch, C. Schevon, Optimization by simulated annealing: an experimental evaluation; part II, graph coloring and number partitioning. Operations Research 39(3) (1991) 378–406.

[22] D.S. Johnson, M. Trick (Eds.), Cliques, coloring, and satisfiability: Second DIMACS implementation challenge, Volume 26 of Dimacs Series in Discrete Mathematics and Theoretical Computer Science (1996).

[23] 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.

[24] Z. Lü and J.K. Hao, A memetic algorithm for graph coloring. European Journal of Operational Research 200(1) (2010) 235–244.

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

[26] C. Morgenstern, Distributed coloration neighborhood search. In [22], (1996) 335–357.

[27] D.C. Porumbel, J.K. Hao, P. Kuntz, A search space cartography for guiding graph coloring heuristics. Computers and Operations Research 37(4) (2010) 769–778.

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

[29] D.H. Smith, S. Hurley, S.U. Thiel, Improving heuristics for the frequency assignment problem. European Journal of Operational Research 107(1) (1998) 76–86.

[30] B.T. Tesfaldet, Automated lecture timetabling using a memetic algorithm. Asia - Pacific Journal of Operational Research 25(4) (2008) 451–475.

[31] O. Titiloye, A. Crispin, Quantum annealing of the graph coloring problem. Discrete Optimization (2011), doi:10.1016/j.disopt.2010.12.001.

[32] X.F. Xie, J. Liu, Graph coloring by multiagent fusion search. Journal of Combinatorial Optimization 18(2) (2009) 99–123.

[33] Q. Wu, J.K. Hao, Adaptive multistart tabu search for the maximum clique problem. Technical Report, University of Angers. June (2010).

[34] Q. Wu, J.K. Hao, Coloring large graphs based on independent set extraction. Computers and Operations Research (2011), doi:10.1016/j.cor.2011.04.002.