An Effective Heuristic Algorithm for Sum Coloring of Graphs

Qinghua Wu and Jin-Kao Hao *

LERIA, Université d'Angers
2 Boulevard Lavoisier, 49045 Angers Cedex 01, France

Computers and Operations Research
(DOI10.1016/j.cor.2011.09.010)

Abstract

Given an undirected graph $G = (V, E)$, the minimum sum coloring problem (MSCP) is to find a legal vertex coloring of G, using colors represented by natural numbers $(1, 2, \ldots)$ such that the total sum of the colors assigned to the vertices is minimized. In this paper, we present EXSCOL, a heuristic algorithm based on independent set extraction for this NP-hard problem. EXSCOL identifies iteratively collections of disjoint independent sets of equal size and assign to each independent set the smallest available color. For the purpose of computing large independent sets, EXSCOL employs a tabu search based heuristic. Experimental evaluations on a collection of 52 DIMACS and COLOR2 benchmark graphs show that the proposed approach achieves highly competitive results. For more than half of the graphs used in the literature, our approach improves the current best known upper bounds.

Keywords: sum coloring, vertex coloring, independent set, heuristics.

1 Introduction

Given an undirected graph $G = (V, E)$ with vertex set V and edge set E, an independent set I of G is a subset of V such that no two vertices in I are joined by an edge in E. A legal k-coloring of G is a partition of V into k disjoint independent sets $I_1, I_2 \ldots, I_k$. The smallest integer k such that a legal k-coloring exists for G is the *chromatic number* of G (denoted by $\chi(G)$). The

* Corresponding author.
Email addresses: wu@info.univ-angers.fr (Qinghua Wu), hao@info.univ-angers.fr (Jin-Kao Hao).

well-known NP-hard *graph vertex coloring* problem is to find the chromatic number of a graph [7]. In this paper, we are interested in a related problem known as the *minimum sum coloring problem* (MSCP for short).

The MSCP is to find a vertex coloring $c = \{I_1, \ldots, I_k\}$ of G such that the following total sum of the colors is minimized:

$$Sum(c) = \sum_{i=1}^{k} \sum_{v \in I_i} i \tag{1}$$

The optimal (smallest) value of this sum is called the *chromatic sum* of G and denoted by $\sum(G)$. The number k of the k-coloring leading to the *chromatic sum* is called the strength of the graph and denoted by $s(G)$. It is clear that $s(G)$ is lower bounded by $\chi(G)$, i.e. $s(G) \geq \chi(G)$.

The minimum sum coloring problem is known to be NP-hard in the general case [14]. In addition to its theoretical significance as a difficult combinatorial problem, the MSCP is notable for its ability to formulate a number of important problems, including those from VLSI design, scheduling and resource allocation [1,19].

During the past two decades, the MSCP has been studied essentially from a theoretical point of view and special cases (e.g. tree, interval graphs, line graphs etc) have been identified which admit efficient approximation algorithms or polynomial algorithms [1–3,8,9,12,19,23]. For the purpose of practical solving of the general MSCP, several heuristic algorithms have recently been proposed to find suboptimal solutions. Notice that this heuristic based approach is expected to find good approximate solutions within reasonable computing time, but without provable solution quality.

For instance, Kokosiński and Kawarciany proposed a parallel genetic algorithm [13]. In [17], Y. Li et al. presented MRLF, an effective greedy algorithm based on the well-known RLF graph coloring heuristic [16]. Moukrim et al. showed a technique for computing the lower bound for the MSCP based on extraction of specific partial graphs [21]. Bouziri and Jouini adapted a tabu coloring algorithm to sum coloring [4]. Douiri and Elbernoussi illustrated a hybrid algorithm which combines a genetic algorithm with a local search heuristic [5]. Finally, in [1] Bar-Noy et al. presented a theoretical study of a heuristic algorithm based on finding iteratively maximum independent sets (MaxIS) and showed that the MaxIS is a 4-approximation to the MCSP, which is a tight bound to within a factor of 2. Nevertheless, the practical performance of this heuristic was not verified with computational experiments.

In this paper, we present a heuristic algorithm called EXSCOL for the MSCP based on the idea of independent set extraction. The similar approach was

initially applied to the vertex coloring problem [25]. Basically, EXSCOL iteratively extracts from the graph as many large disjoint independent sets of equal size as possible. For each extracted independent set, we assign to it the smallest available color (colors are represented by natural numbers 1, 2...). This process is repeated until the graph becomes empty. The rationale behind this approach is that by extracting many large disjoint independent sets, we naturally favor the construction of large color classes and reduce the number of needed color classes, leading to a reduced total sum of colors. Since computing a maximum independent set of a graph is NP-hard [7], we employ the tabu search based heuristic introduced in [26] to find large independent sets.

We present experimental results on a set of 52 benchmark graphs in the literature, showing that the proposed algorithm achieves very competitive results with respect to the existing sum coloring heuristics. Indeed, for more than half of the instances used in the literature, the proposed approach improves the current best known results. Furthermore, we assess the relative performance of two other solution methods using respectively the conventional independent set extraction strategy and graph vertex coloring algorithms.

The rest of this paper is organized as follows. In the next section we give a formal description of the proposed EXSCOL algorithm. In section 3, computational results are presented and compared with several state-of-the-art algorithms from the literature. In section 4, we show additional studies and comparisons with respect to two other solutions methods, followed by conclusions in section 5.

2 EXSCOL: an algorithm for the MSCP

2.1 Rationale and general procedure

Let $c = \{I_1, \ldots, I_k\}$ be a legal coloring of graph $G = (V, E)$, each independent set I_i is a color class of c such that all the vertices $v \in I_i$ receive color i. Given the coloring c, its sum of colors $Sum(c)$ according to Eq. (1) counts the total sum of the colors induced by c. Suppose $|I_1| \geq |I_2| \geq \ldots \geq |I_1|$, Eq. (1) can be rewritten as follows.

$$Sum(c) = 1 \cdot |I_1| + 2 \cdot |I_2| + \ldots + k \cdot |I_k| = \sum_{i=1}^{k} i \cdot |I_i| \tag{2}$$

It is clear that the sum depends on both the number k of the used colors and the size of the color classes. To minimize this sum, one can try to construct large color classes and assign to them small colors. For this purpose, one

can remove iteratively the maximum number of disjoint independent sets of the maximum size from the graph until the graph becomes empty. However, both computing a maximum independent set of a graph and a maximum set of disjoint sets (which is the maximum set packing problem) are NP-hard problems [7]. Consequently, heuristics are needed in the general case to find approximate solutions, in particular for large graphs.

The proposed EXSCOL algorithms follows this basic idea and can be summarized by the following procedure.

(1) Identify an independent set of the largest size possible from the graph;
(2) Identify as many pairwise disjoint independent sets of that size as possible and extract them from the graph;
(3) Assign to each extracted independent set the smallest available color (the first color used is 1);
(4) Stop if the graph becomes empty, goto Step 1 otherwise.

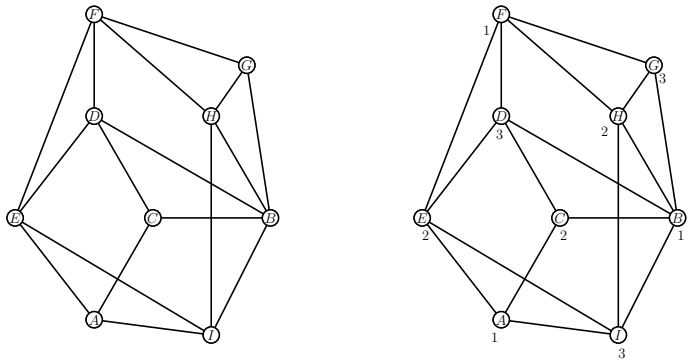

Fig. 1. An illustration of the proposed EXSCOL algorithm

Fig. 1 illustrates how this approach works on a graph with 9 vertices. At the first step, we find a maximum independent set of size 3 (e.g. $\{A, D, H\}$). Then we try to identify as many disjoint independent sets of size 3 as possible from the graph, leading to 3 disjoint independent sets $\{A, B, F\}, \{C, E, H\}, \{D, G, I\}$. We assign colors $1, 2, 3$ to these independent sets. Since the graph becomes empty after removing these independent sets, the procedure stops. We obtain a coloring $c = \{\{A, B, F\}, \{C, E, H\}, \{D, G, I\}\}$ with $Sum(c) = 18$ for the graph.

2.2 The EXSCOL algorithm

The proposed EXSCOL algorithm (Alg. 1) implements the general procedure given in section 2.1. EXSCOL starts by identifying a first largest possible independent set I_M (line 4) whose size $|I_M|$ is used later to build a pool M of independent sets of that size (lines 5-15). The search for a new independent set of size $|I_M|$ 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 is found after p_{max} consecutive tries.

From M, EXSCOL tries to determine a maximum number of disjoint independent sets (line 16). This task corresponds in fact to the maximum set packing problem, which is equivalent to the maximum clique (thus the maximum independent set) problem [7]. Given $M = \{I_1, ..., I_n\}$, we construct an instance of the independent set problem as follows.

Algorithm 1 Pseudo-code of the EXSCOL algorithm

Require: Graph $G = (V, E)$
Ensure: A coloring of G
1: **Begin**
2: $k \leftarrow 1$
3: **while** ($|V| > 0$) **do**
4: $I_M \leftarrow ATS(G)$ {Apply ATS to find an independent set as large as possible, Sect. 2.3}
5: $M \leftarrow \{I_M\}$
6: $r \leftarrow 0$
7: **while** ($r \leq p_{max}$ & $|M| \leq M_{max}$) **do**
8: $I \leftarrow ATS(G, |I_M|)$ {Apply ATS to find an independent set of size $|I|$, Sect. 2.3}
9: **if** $I \in M$ **then**
10: $r \leftarrow r + 1$
11: **else**
12: $M \leftarrow M \bigcup \{I\}$
13: $r \leftarrow 0$
14: **end if**
15: **end while**
16: Find in M as many pairwise disjoint independent sets as possible: $(I_1, ..., I_l) \leftarrow arg\ max\{|A| : A \subseteq M, \forall I^a, I^b \in A, I^a \bigcap I^b = \phi\}$ (Break ties by taking $(I_1, ..., I_l)$ such that the removal of $(I_1, ..., I_l)$ reduces the most G)
17: Remove $(I_1, ..., I_l)$ from G
18: **for** $i = 1$ to l **do**
19: Assign color k to I_i
20: $k = k + 1$
21: **end for**
22: **end while**
23: **End**

We define a new graph $G' = (V', E')$ where $V' = \{1, ..., n\}$ and $\{i, j\} \in E'$ $(i, j \in V')$ if I_i and I_j share at least one element, i.e. $I_i \cap I_j \neq \emptyset$. Now it is clear that there is strict equivalence between an independent set in G' and a set of disjoint independent sets in M. Consequently, to obtain a maximum set of disjoint independent sets in M, we can search for a maximum independent set in G'. Given that the maximum independent set problem is NP-hard in general, we approximate it with the so-called ATS heuristic [26].

To illustrate the idea, consider again Fig. 1 which contains 8 independent sets of size 3: $I_1 = \{A, B, F\}$, $I_2 = \{A, D, G\}$, $I_3 = \{A, D, H\}$, $I_4 = \{C, E, G\}$, $I_5 = \{C, E, H\}$, $I_6 = \{C, F, I\}$, $I_7 = \{C, G, I\}$, $I_8 = \{D, G, I\}$. Let $M = \{I_1, I_2, \ldots, I_8\}$, the associated graph $G' = (V', E')$ is shown in Fig. 2. It is clear that the maximum independent set $\{1, 5, 8\}$ in Fig. 2 leads to the maximum set of disjoint independent sets $\{I_1, I_5, I_8\}$ in M.

If more than one solutions exist for the maximum independent set problem, we prefer the solution IS such that the removal of the independent sets of IS will reduce the graph the most in terms of the number of edges. The rationale behind this choice is that a graph with fewer edges has more chance to contain larger independent sets.

2.3 Adaptive Tabu Search for independent set identification

The Adaptive Tabu Search (ATS) heuristic described in [26] is designed to determine an independent set of size k where k is a fixed integer. To approximate the maximum independent set problem, it suffices to search for independent sets of increasing k.

To find an independent set of size k in graph $G = (V, E)$, ATS explores a search space Ω composed of subsets of V of size k which are formally identified by: $\Omega = \{S \subset V : |S| = k\}$. To assess the quality of a candidate solution $S \in \Omega$, ATS uses an evaluation function $f(S)$ that counts the number of edges in the subgraph G_S induced by S. In other words, let $G_S = (S, E_S)$ such that $E_S = \{\{u, v\} : \{u, v\} \in E, u, v \in S\}$, then $f(S) = |E_S|$. Obviously, if $f(S) = 0$, no two vertices in S are joined by an edge, implying that S is an independent set. Otherwise, S is not an independent set of G. The goal of ATS is to find a solution S^* such that $f(S^*)$ reaches its minimum value $f(S^*) = 0$.

To this end, ATS starts with a solution $S \in \Omega$ and then replaces iteratively the current solution S by a neighboring solution S' according to a neighborhood. Basically a neighboring solution is generated by exchanging a vertex x of S against a vertex y of $V \setminus S$. For the purpose of minimizing the function f

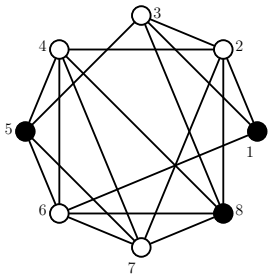

Fig. 2. A transformed maximum independent set instance $G' = (V', E')$

above, we constraint x and y to belong to some specific subsets $X \subset S$ and $Y \subset V \setminus S$. More precisely, we define X as the set of vertices of S having the maximum number of adjacent vertices in S, and Y as the set of vertices of $V \setminus S$ having the minimum number of adjacent vertices in S.

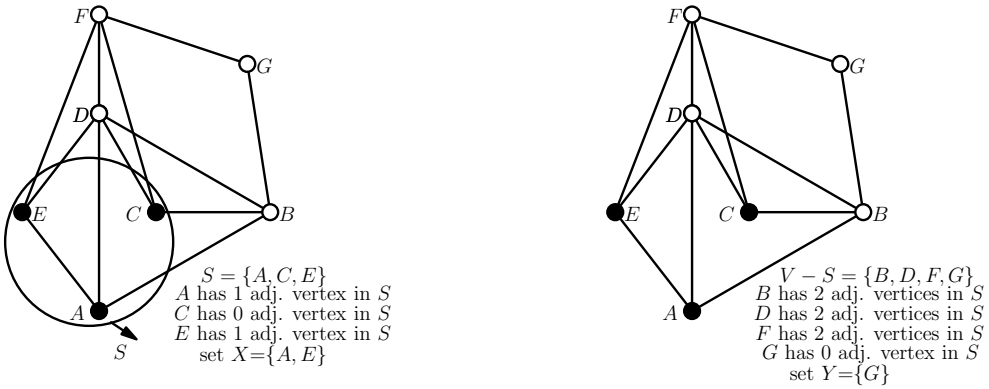

Fig. 3. An example for the neighborhood defined by swap move. The solution S has two neighboring solutions $S' = \{C, E, G\}$ or $S' = \{A, C, G\}$.

Now, to obtain a neighboring solution S' from S, we swap one vertex $x \in X$ with another vertex $y \in Y$. All possible swap moves induced by X and Y define our constrained neighborhood $N(S)$, i.e.

$$N(S) = \{S' \mid S' = S \backslash \{x\} \cup \{y\},\ x \in X, y \in Y\}.$$

Fig. 3 shows an illustrative example where S has two possible neighboring solutions.

ATS explores the search space by following this neighborhood. At each step, ATS selects the best non-tabu neighboring solution $S' \in N(S)$ with a minimum $f(S')$ value to replace S.

To prevent the search from short-term cycling, once a swap move is performed, the swapped vertices x and y are marked tabu for the next T_x and T_y iterations respectively. The tabu tenures T_x and T_y are dynamically and adaptively adjusted as follows: $T_x = f(S) + Random(4)$ and $T_y = 0.6 * T_x$ where $Random(4)$ is a random number in $\{1, \ldots, 4\}$.

Notice that with the use of tabu lists, it may happen that no move is possible (i.e., all moves are forbidden). In this case, we simply select the most favorable tabu move, i.e., the move with the largest move gain. As experimentally shown in [26], this tabu search using the swap neighborhood combined with this tabu mechanism performs quite well on a large set of benchmark graphs.

3 Experimental results

To assess the practical efficiency of our proposed EXSCOL algorithm, we carry out experiments on a total of 52 graphs in the literature and compare EXSCOL with 4 state-of-the-art existing algorithms. We also assess the interest of the conventional strategy of extracting independent set extractions one by one and the pertinence of using graph coloring algorithms to solve the sum coloring problem.

3.1 Problem instances and experimental protocol

Two sets of benchmark graphs from the literature are considered in the experiments. The first set (Table 1) is composed of 29 well-known DIMACS graphs [1] . These graphs are very popular for testing graph coloring algorithms [6,11,18,20,22]. However only the 12 DSJC random graphs have been recently used for sum coloring [4,17].

The second set of benchmarks (see Table 2) is composed of 23 graphs from the COLOR02 website [2] . Like the first set, these graphs are initially collected for the purpose of the COLOR02 competition. Among these graphs, sum coloring results have been reported in the literature for 16 graphs [5,13,21].

Our EXSCOL algorithm is programmed in C and compiled using GNU GCC on a PC with 2.8 GHz CPU and 2G RAM. To report computational statistics, we run our EXSCOL algorithm on each graph 20 times (5 times for the two huge Cxxxx.5 instances) and reports the following information: the minimum sum of colors, the number of used colors, the average sum of colors over the multiple runs, the average CPU time.

A desirable feature of EXSCOL is that it does not need any external stop condition to terminate. In fact, it stops when the graph under consideration becomes empty. Nevertheless, EXSCOL has three parameters to be fixed (see Alg. 1): p_{max}, M_{max} and $Iter$ (required by the ATS algorithm [26]). As explained in Section 2.2, large values for p_{max} and M_{max} could lead to more independent sets collected in M and thus increase the chance of finding more disjoint independent sets. On the other hand, large values for p_{max} or M_{max} also imply long computing times. Based on preliminary experiments we have fixed $p_{max} = 100$ and $M_{max} = 2000$ for all our experiments, although fine-tuning these parameters would lead to better results. Finally, $Iter$ is the maximum number of iterations before ATS stops its search if no independent set

[1] `ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/`
[2] `http://mat.gsia.cmu.edu/COLOR02/`

of the desired size is found. We have set $Iter = 10^8$ according to [26].

3.2 Computational results

Table 1 and Table 2 summarize respectively the computational statistics of our EXSCOL algorithm obtained on the two sets of benchmark instances [3]. Columns 2-4 give the features of the tested instances: the number of vertices ($|V|$), the number of edges ($|E|$) and the density of the graph (Den). Column 5 indicates the current best known sum values (Sum^*) from the literature along with the references reporting these values in brackets. Column 6 presents the smallest number of colors (k^*) for which a solution (k-coloring) has ever been found by a graph vertex coloring algorithm. In columns 7-9, the computational statistics of our EXSCOL algorithm are given, including the smallest sum of colors (Sum) with the number of required colors in parentheses (k), the average sum of colors ($Avg.$) with the standard deviation in parentheses, the average CPU time in minutes ($T[min.]$). The last column indicates the difference (gain) in sum of colors between our best results (column Sum) and the current best known results (column Sum^*).

Concerning the 29 DIMCAS graphs, one observes from Table 1 that our EXSCOL algorithm obtains excellent results (columns 7) with respect to the current best known results reported in the literature (column 5). Indeed, for the 12 random DSJC graphs used in the literature, our EXSCOL algorithm significantly improves the best known results by reducing largely the sum values (column 10). Even our averaged results are better than the current best ones. For the remaining 17 graphs, we report for the first time computational results for the sum coloring problem. EXSCOL needs from 1 to 27 minutes to achieve these results except for the two huge Cxxxx.5 graphs for which much larger computing times are required. Finally Table 1 discloses that for this set of instances, the results may vary with large standard deviation for several graphs.

Concerning the 23 COLOR02 graphs, one notices from Table 2 that EXSCOL improves the current best known results for 5 graphs while equaling the best ones for 9 graphs. Only for 2 graphs, EXSCOL obtains a worse result. For the 16 graphs used in the literature, EXSCOL achieves its results within less than 2 minutes and with small standard deviations. For the 7 additional (large) graphs, EXSCOL requires a computing time ranging from 21 to 127 minutes.

[3] The results of EXSCOL are available at `http://www.info.univ-angers.fr/pub/hao/exscol.html`

Table 1
Computational results of EXSCOL on 29 DIMACS challenge benchmarks. The symbol '-' means that the information is not available.

Instance	$\|V\|$	$\|E\|$	*Den*	Sum^*	k^*	EXSCOL			
						$Sum(k)$	$Avg.(Std.)$	$T[min.]$	Diff
DSJC125.1	125	736	0.09	344[4]	5	326(7)	326.7(0.8)	1	−18
DSJC125.5	125	3891	0.50	1103[4]	17	1017(20)	1019.7(8.7)	1	−86
DSJC125.9	125	6961	0.89	2631[4]	44	2512(44)	2512.0(0.0)	1	−119
DSJC250.1	250	3218	0.10	1046[4]	8	985(10)	985.0(0.0)	4	−61
DSJC250.5	250	15668	0.50	3658[17]	28	3246(31)	3253.9(6.7)	6	−412
DSJC250.9	250	27897	0.90	8942[17]	72	8286(75)	8288.8(6.8)	7	−656
DSJC500.1	500	12458	0.10	3205[4]	12	2850(14)	2857.4(141.0)	9	−355
DSJC500.5	500	62624	0.50	12717[17]	48	10910(51)	10918.2(54.1)	11	−1807
DSJC500.9	500	112437	0.90	32703[17]	126	29912(132)	29936.2(630.4)	15	−2791
DSJC1000.1	1000	49629	0.10	10276[17]	20	9003(22)	9017.9(91.1)	28	−1273
DSJC1000.5	1000	249826	0.50	45408[17]	83	37598(87)	37673.8(2288.0)	24	−7810
DSJC1000.9	1000	449449	0.90	119111[17]	223	103464(231)	103531(5070)	27	−15647
flat300_20_0	300	21375	0.48	-	20	3150(20)	3150.0(0.0)	3	-
flat300_26_0	300	21633	0.48	-	26	3966(26)	3966.0(0.0)	3	-
flat300_28_0	300	21695	0.48	-	28	4282(34)	4286.1(30.0)	3	-
flat1000_50_0	1000	245000	0.49	-	50	25500(50)	25500.0(0.0)	9	-
flat1000_60_0	1000	245830	0.49	-	60	30100(60)	30100.0(0.0)	11	-
flat1000_76_0	1000	246708	0.49	-	82	37167(86)	37213.2(575.0)	19	-
le450_15a	450	8168	0.08	-	15	2632(18)	2641.9(29.0)	5	-
le450_15b	450	8169	0.08	-	15	2642(19)	2643.4(7.2)	7	-
le450_15c	450	16680	0.17	-	15	3866(24)	3868.9(9.8)	6	-
le450_15d	450	16750	0.17	-	15	3921(26)	3928.5(36.5)	5	-
le450_25a	450	8260	0.08	-	25	3153(26)	3159.4(12.4)	7	-
le450_25b	450	8263	0.08	-	25	3366(26)	3371.9(18.0)	6	-
le450_25c	450	17343	0.17	-	25	4515(31)	4525.4(161.8)	8	-
le450_25d	450	17425	0.17	-	25	4544(31)	4550.0(24.0)	7	-
latin_sqr_10	900	307350	0.76	-	98	42223(109)	42392.7(4445)	4	-
C2000.5	2000	999836	0.50	-	148	132515(150)	132682(7342)	656	-
C4000.5	4000	4000268	0.50	-	271	473234(266)	473211(1027)	2588	-

3.3 Comparison with other algorithms

In this section, we compare our EXSCOL algorithm with 4 recent reference algorithms in the literature: Hybrid Local Search (HLS) [5], MRLF [17], Parallel Genetic Algorithm (PGA) [13], Tabu Search (TS) [4]. The comparisons are based on the criterion of quality, i.e., the smallest sum of colors reached by a given algorithm. Notice that information like computing time are not available for the reference algorithms.

Table 3 and Table 4 show the best results of our EXSCOL algorithm compared

Table 2
Computational results of EXSCOL on 23 COLOR02 benchmarks. The symbol '-' means that the information is not available.

Instance	$\|V\|$	$\|e\|$	*Den*	Sum^*	k^*	EXSCOL			
						$Sum(k)$	$Avg.(Std.)$	$T[min.]$	Diff
myciel3	11	20	0.40	21[5,21,13]	4	21(4)	21.0(0.0)	1	0
myciel4	23	71	0.28	45[5,21,13]	5	45(5)	45.0(0.0)	1	0
myciel5	47	236	0.22	93[5,21,13]	6	93(6)	93.0(0.0)	1	0
myciel6	95	755	0.17	189[5,21,13]	7	189(7)	189.0(0.0)	2	0
myciel7	191	2360	0.13	381[5,21]	8	381(8)	381.0(0.0)	2	0
anna	138	493	0.05	277[21]	11	283(11)	283.2(0.2)	2	6
david	87	406	0.11	241[21]	11	237(11)	238.1(1.0)	1	−4
huck	74	301	0.11	243[5,13]	11	243(11)	243.8(1.0)	1	0
jean	80	254	0.08	217[21]	10	217(10)	217.3(0.2)	1	0
queen5.5	25	160	0.53	75[21,13]	5	75(5)	75.0(0.0)	1	0
queen6.6	36	290	0.46	138[5,21,13]	7	150(10)	150.0(0.0)	1	12
queen7.7	49	476	0.40	196[21,13]	7	196(7)	196.0(0.0)	1	0
queen8.8	64	728	0.36	302[13]	9	291(9)	291.0(0.0)	1	−11
games120	120	638	0.09	446[5,21]	9	443(9)	447.9(3.2)	2	−3
miles250	128	387	0.05	334[21]	8	328(9)	333.0(7.6)	2	−6
miles500	128	1170	0.14	715[21]	20	709(20)	714.5(26.8)	2	−6
wap05	905	43081	0.10	-	50	13680(51)	13718.4(1047.1)	21	-
wap06	947	43571	0.10	-	46	13778(48)	13830.9(905.4)	27	-
wap07	1809	103368	0.06	-	46	28629(51)	28663.8(721.8)	112	-
wap08	1870	104176	0.06	-	45	28896(51)	28946.0(1361.2)	127	-
qg.order30	900	26100	0.06	-	30	13950(30)	13950.0(0.0)	28	-
qg.order40	1600	62400	0.05	-	40	32800(40)	32800.0(0.0)	35	-
qg.order60	3600	212400	0.03	-	60	110925(74)	110993.0(7054.9)	87	-

with these reference algorithms. Column 2 recall the current best known Sum^* with the references reporting these values (indicated in brackets). Columns 3-6 present the best results obtained by these reference algorithms. For indicative purposes, the number of colors k required by the best sum coloring is given in parentheses.

From Table 3, one observes that, for the 12 DIMACS random graphs, EXSCOL dominates the TS and MRLF algorithms. For each of these graphs, our EXSCOL algorithm obtains a better sum coloring compared with the reference algorithms. One notices that in most cases, the colorings of EXSCOL require a smaller number of colors.

Table 4 discloses that EXSCOL competes favorably with HLS, MRLF and PGA on the set of 16 COLOR02 instances. Indeed, our EXSCOL algorithm finds a better sum coloring for 5 graphs while equaling the best known results for 9 graphs. Only for 2 cases (anna and queen6.6), it fails to attain the best known results.

Table 3
Comparison on 12 graphs of the DIMACS challenge benchmarks. The symbol '-' means that the related statistics are not available. The best *sum* values are highlighted in bold.

Instance	*Sum**	sum coloring algorithms		
		EXSCOL	MRLF[17] (2009)	TS[4] (2010)
DSJC125.1	344 [4]	**326**(7)	352(6)	344(6)
DSJC125.5	1103 [4]	**1017**(20)	1141(21)	1103(18)
DSJC125.9	2631 [4]	**2512**(44)	2653(50)	2631(49)
DSJC250.1	1046 [4]	**985**(10)	1068(10)	1046(10)
DSJC250.5	3658 [17]	**3246**(31)	3658 (34)	3779(33)
DSJC250.9	8942 [17]	**8286**(75)	8942(83)	9198(82)
DSJC500.1	3205 [4]	**2850**(14)	3229(15)	3205(15)
DSJC500.5	12717 [17]	**10910**(51)	12717(60)	-
DSJC500.9	32703 [17]	**29912**(132)	32703(148)	-
DSJC1000.1	10276 [17]	**9003**(22)	10276(25)	-
DSJC1000.5	45408 [17]	**37598**(87)	45408(104)	-
DSJC1000.9	119111 [17]	**103464**(231)	119111(265)	-

Table 4
Comparison on 16 graphs of the COLOR02 benchmarks. The symbol '-' means that the related statistics are not available. The best *sum* values are highlighted in bold.

Instance	*Sum**	sum coloring algorithms			
		EXSCOL	HLS[5] (2011)	MRLF[21] (2010)	PGA[13] (2007)
myciel3	21 [5,21,13]	21(4)	21(4)	21(-)	21(4)
myciel4	45 [5,21,13]	45(5)	45(5)	45(-)	45(5)
myciel5	93 [5,21,13]	93(6)	93(6)	93(-)	93(6)
myciel6	189 [5,21,13]	189(7)	189(7)	189(-)	189(7)
myciel7	381 [5,21]	381(8)	381(8)	381(-)	382(8)
anna	277 [21]	283(11)	-(-)	**277**(-)	281(11)
david	241 [21]	**237**(11)	-(-)	241(-)	243(11)
huck	243 [5,13]	243(11)	243(11)	244(-)	243(11)
jean	217 [21]	217(10)	-(-)	217(-)	218(10)
queen5.5	75 [21,13]	75(5)	-(-)	75(-)	75(5)
queen6.6	138 [5,21,13]	150(10)	**138**(8)	**138**(-)	**138**(8)
queen7.7	196 [21,13]	196(7)	-(-)	196(-)	196(7)
queen8.8	302 [13]	**291**(9)	-(-)	303(-)	302(10)
games120	446 [5,21]	**443**(9)	446(9)	446(-)	460(9)
miles250	334 [21]	**328**(9)	343(10)	334(-)	347(8)
miles500	715 [21]	**709**(20)	755(22)	715(-)	762(20)

4 Discussion and analysis

4.1 Influence of the method to extract independent sets

Our EXSCOL algorithm uses a heuristic method to extract at each iteration as many disjoint independent sets as possible. A conventional method like MaxIS [1] extracts at each iteration exactly one independent set. The MaxIS method can be considered as a simplified version of our EXSCOL algorithm where lines 6–15 of Algorithm 1 (see Section 2.2) are disabled. Thus for MaxIS, each time a (large) independent set is found in the graph, it is assigned the smallest available color and its vertices are removed from the graph. This process is repeated until the graph becomes empty.

In order to highlight the difference between these two methods, we carry out additional experiments on the 12 DIMACS random DSJC graphs and show a comparison between EXSCOL and MaxIS. We run both methods 20 times and report in Table 5 the best sum values together with the number of used colors in brackets. The results show a clear dominance of EXSCOL over MaxIS. Indeed, EXSCOL finds a better sum coloring than MaxIS for each graph. Now it is interesting to observe that even if MaxIS cannot compete with EXSCOL, MaxIS does compete favorably with any of the existing sum coloring algorithms in the literature (compare column 2 of Table 3 and column 4 of Table 5). This observation highlights the interest of the general independent set extraction approach for the sum coloring problem.

Finally, if one checks the number k of colors used in the solutions, one notices that the solutions of EXSCOL need in general fewer colors compared to the solutions obtained by MaxIS. This can be explained by the fact that by extracting at each iteration as many disjoint independent sets as possible, EXSCOL is able to pack more vertices in the extracted independent sets, reducing thus the needed color classes to pack the vertices of the graph.

4.2 Sum coloring v.s. graph coloring

Given the relation between sum coloring and vertex coloring, one would wonder whether an effective vertex coloring algorithm could remain effective to approximate the sum coloring problem. Indeed, one can use a graph vertex coloring algorithm to find k-colorings with k as small as possible. Since the sum value depends partially on k, this coloring approach could help to solve the sum coloring problem. Yet, this approach does not necessarily lead to a good sum coloring as shown in the example of Fig. 4. Indeed, while the *chromatic number* $\chi(G)$ for this graph is 3, a 4-coloring is needed to obtain the

Table 5
Comparisons between EXSCOL and MaxIS. EXSCOL dominates the conventional independent set approach (extraction of one independent set each time).

Instance	Sum^*	sum coloring algorithms		
		EXSCOL	MaxIS	EXSCOL−MaxIS
DSJC125.1	344	326(7)	335(7)	-9(0)
DSJC125.5	1103	1017(20)	1047(21)	-30(-1)
DSJC125.9	2631	2512(44)	2599(50)	-87(-6)
DSJC250.1	1046	985(10)	1001(11)	-16(-1)
DSJC250.5	3658	3246(31)	3377 (34)	-131(-3)
DSJC250.9	8942	8286(75)	8548(83)	-262(-8)
DSJC500.1	3205	2850(14)	2932(16)	-82(-2)
DSJC500.5	12717	10910(51)	11163(56)	-253(-5)
DSJC500.9	32703	29912(132)	30957(145)	-1045(-13)
DSJC1000.1	10276	9003(22)	9161(24)	-158(-2)
DSJC1000.5	45408	37598(87)	38452(93)	-854(-6)
DSJC1000.9	119111	103464(231)	108487(254)	-5023(-23)

chromatic sum $\sum(G) = 15$, i.e., the strength $s(G) = 4$.

In this section we provide computational evidence to show the limit of this graph coloring based method with respect to the proposed EXSCOL method. For this purpose, we adopt the Memetic Coloring Algorithm (MACOL) [18] which is a recent and competitive graph coloring algorithm. For this experiment, we consider again the set of 12 DSJC random instances. For each graph, we apply MACOL to obtain a legal coloring with k^* colors where k^* is the smallest number of colors for which a k^*-coloring has ever been found by a graph coloring algorithm. For such a k^*-coloring, we assign color 1 to the largest color class, color 2 to the next largest color class and so on. Table 6 shows the comparative results between EXSCOL and MACOL for the chosen graphs. The differences of sum values obtained by the two methods are given in the last column.

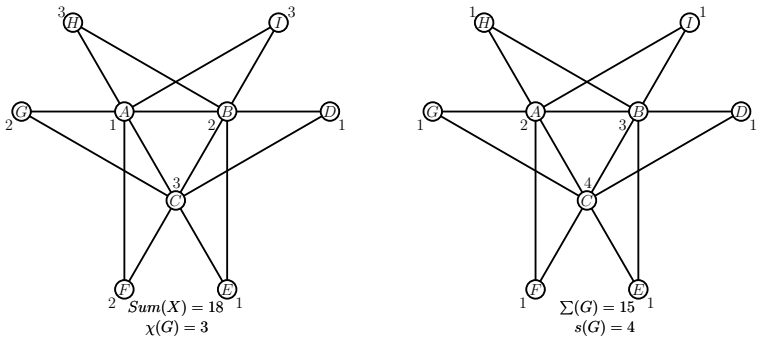

Fig. 4. A coloring with a sum of colors equal to 18 using 3 colors (Left), and a coloring with a sum of colors equal to 15 using 4 colors (Right).

Table 6 shows clearly that EXSCOL reaches smaller sum values than MACOL

Table 6
Comparisons between EXSCOL and MACOL. EXSCOL performs always better than the graph coloring approach.

Instance	Sum^*	sum coloring algorithms		
		EXSCOL	MACOL[18]	EXSCOL-MACOL
DSJC125.1	344	326(7)	355(5)	-29(2)
DSJC125.5	1103	1017(20)	1058(17)	-41(3)
DSJC125.9	2631	2512(44)	2581(44)	-69(0)
DSJC250.1	1046	985(10)	1071(8)	-86(2)
DSJC250.5	3658	3246(31)	3344(28)	-97(3)
DSJC250.9	8942	8286(75)	8372(72)	-86(3)
DSJC500.1	3205	2850(14)	3127(12)	-277(2)
DSJC500.5	12717	10910(51)	11176(48)	-266(3)
DSJC500.9	32703	29912(132)	30073(126)	-161(6)
DSJC1000.1	10276	9003(22)	9767(20)	-764(2)
DSJC1000.5	45408	37598(87)	38636(83)	-2038(4)
DSJC1000.9	119111	103464(231)	104203(223)	-739(8)

Table 7
Detailed computational results on DSJC1000.5, including the size of the independent set (size $|I|$) and the number of independent sets of size $|I|$ (No. of IS of $|I|$) contained in the best solutions found by EXSCOL and MACOL.

EXSCOL		MACOL	
size $\|I\|$	No. of IS of $\|I\|$	size $\|I\|$	No. of IS of $\|I\|$
15	6	15	1
14	28	14	16
13	9	13	27
12	13	12	14
11	7	11	9
10	5	10	6
9	4	9	6
8	3	8	1
7	4	7	3
6	0	6	0
5	4	5	0
4	1	4	0
3	1	3	0
2	1	2	0
1	1	1	0

for the tested graphs though EXSCOL may require more colors. In order to get some insights about this difference, we show in Table 7 additional information about the computational results for DSJC1000.5 reached by EXSCOL and MACOL.

Table 7 indicates the size of each independent set (column 'size $|I|$') and the number of extracted independent sets of size $|I|$ (column 'No. of IS of $|I|$')

contained in the best solutions found by EXSCOL and MACOL. From Table 7, one notices that compared to the solution obtained by MACOL (a 83-coloring), the solution obtained by EXSCOL (a 87-coloring) contains much more independent sets of sizes 15 and 14 (which are the largest and second largest sizes). These independent sets cover a larger number of vertices and receive smaller colors, reducing thus the sum of colors.

Finally, even if MACOL performs worse than EXSCOL, its results remain competitive with other existing algorithms. This is the case because MACOL is able to find k-colorings with k equaling or being close to $\chi(G)$.

5 Conclusion

In this paper, we have presented EXSCOL, a heuristic algorithm based on independent set extraction for the minimum sum coloring problem. Instead of extracting independent sets one by one, the proposed algorithm tries to extract as many disjoint independent sets as possible at each iteration. This strategy helps create more and large independent sets in the solution such that a larger number of vertices can be colored with small colors, thus leading to a smaller sum of colors. Another important element contributing to the effectiveness of EXSCOL is the ATS algorithm which is able to identify efficiently large independent sets.

We have shown that with respect to the existing sum coloring heuristics, the proposed EXSCOL algorithm obtains highly competitive results on a set of 52 DIMACS and COLOR02 benchmark graphs. Indeed, among the 28 graphs used in the literature, EXSCOL improves the best known upper bounds for 17 graphs and equals the best known results for 9 graphs. Only in two cases, EXSCOL fails to attain the best known sum values. The bounds obtained by EXSCOL on the set of 52 graphs are made available online and are useful to assess the performance of both exact and heuristic algorithms for the sum coloring problem.

Finally, this study has verified the (good) performance of the basic extraction method which extracts maximum independent sets one by one and the interest of the method based on applying directly powerful graph vertex coloring algorithms, even though these two methods cannot compete with the proposed EXSCOL algorithm.

Acknowledgment

We are grateful to the referees for their comments and questions which helped us to improve the paper. This work was partially supported by the Region of "Pays de la Loire" (France) within the Radapop and LigeRO Projects.

References

[1] A. Bar-Noy, M. Bellareb, M. M. Halldórsson, H. Shachnai, T. Tamir. On Chromatic sums and distributed resource allocation. Information and Computation 140(2): 183–202, 1998.

[2] A. Bar-Noy and G. Kortsarz. Minimum color sum of bipartite graphs. Journal of Algorithms 28(2): 339?-365, 1998.

[3] F. Bonomo, G. Durán, J. Marenco, M. Valencia-Pabon. Minimum sum set coloring of trees and line graphs of trees. Discrete Applied Mathematics 159(5): 288–294, 2011.

[4] H. Bouziri and M. Jouini. A tabu search approach for the sum coloring problem. Electronic Notes in Discrete Mathematics 36(1): 915–922, 2010.

[5] S.M. Douiri and S. Elbernoussi. New algorithm for the sum coloring problem. International Journal of Contemporary Mathematical Sciences 6(10): 453–463, 2011.

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

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

[8] H. Hajiabolhassan, M.L. Mehrabadi, R. Tusserkani. Minimal coloring and strength of graphs. Discrete Mathematics 215(1–3): 265–270, 2000.

[9] K. Jansen. Approximation results for the optimum cost chromatic partition problem. Journal of Algorithms 34(1): 54–89, 2000.

[10] T. Jiang and D. West. Coloring of trees with minimum sum of colors. Journal of Graph Theory 32(4): 354–358, 1999.

[11] D.S. Johnson and M.A.Trick (Eds). Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. vol 26, The American Mathematical Society, Providence, RI, 1996.

[12] L. G. Kroon, A. Sen, H. Deng, A. Roy. The optimal cost chromatic partition problem for trees and interval graphs. Lecture Notes in Computer Science 1197: 279–292, 1996.

[13] Z. Kokosiński and K. Kawarciany. On sum coloring of graphs with parallel genetic algorithms. Lecture Notes In Computer Science 4431: 211–219, 2007.

[14] E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. Proceedings of the 17th Annual ACM Computer Science Conference, pp 39–45, 1989.

[15] E. Kubicka, G. Kubicki, D. Kountanis. Approximation algorithms for the chromatic sum. Proceedings of the First Great Lakes Computer Science Conference. Lecture Notes in Computer Science 507: 15–21, 1991.

[16] F. T. Leighton. A Graph Coloring Algorithm for Large Scheduling Problems. Journal of Research of the National Bureau of Standards. 84(6): 489–506, 1979.

[17] Y. Li, C. Lucet, A. Moukrim, K. Sghiouer. Greedy algorithms for minimum sum coloring algorithm. Proceedings LT2009 Conference, Tunisia, March 2009.

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

[19] M. Malafiejski. Sum coloring of graphs. Graph Colorings, Contemporary Mathematics 352, AMS, 55–65, 2004.

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

[21] A. Moukrim, K. Sghiouer, C. Lucet, Y. Li. Lower bounds for the minimal sum coloring problem. Electronic Notes in Discrete Mathematics 36: 663–670, 2010.

[22] 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.

[23] M. R. Salavatipour. On sum coloring of graphs. Discrete Applied Mathematics 127(3): 477–488, 2003.

[24] C. Thomassen, P. Erdös, Y. Alavi, P.J. Malde, A.J. Schwenk. Tight bounds on the chromatic sum of a connected graph. Journal of Graph Theory 13: 353–357, 1989.

[25] Q. Wu and J.K. Hao. Coloring large graphs based on independent set extraction. Computers and Operations Research 39(2): 283–290, 2012.

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