A local search heuristic for chromatic sum

Anders Helmar[1], Marco Chiarandini[1]

Department of Mathematics and Computer Science,
University of Southern Denmark
Campusvej 55, 5230 Odense M, Denmark
ahelm06@student.sdu.dk, marco@imada.sdu.dk

Abstract

A coloring of an undirected graph is a labelling of the vertices in the graph such that no two adjacent vertices receive the same label. The sum coloring problem asks to find a coloring, using natural numbers as labels, such that the total sum of the colors used is minimized. We design and test a local search algorithm, based on variable neighborhood search and iterated local search, that outperforms in several instances the currently existing benchmarks on this problem.

1 Introduction

Let $G = (V, E)$ be an undirected graph with V the set of vertices and E the set of edges. A *k-coloring* of G is a mapping $\varphi : V \mapsto \mathbb{N}$ that assigns exactly one natural number, e.g., $\{1, 2, 3, \ldots, k\}$, to each vertex. A *proper k-coloring* of G is a coloring in which $\varphi(u) \neq \varphi(v)$ for all $uv \in E$. A proper k-coloring is also a partition $\mathcal{C}$ of V into mutually disjoint independent sets $C_1, ..., C_k$, $\bigcup_i C_i = V$, where $\varphi(v) = i$ for all $v \in C_i$. The *chromatic number* $\chi(G)$ is the smallest number of independent sets k for which a proper coloring for G exists. Let x_i denote the size of C_i. The *sum* of a proper coloring φ of G is

$$\mathrm{Sum}(\varphi) = \sum_{v \in V} \varphi(v) = 1 \cdot x_1 + 2 \cdot x_2 + \cdots + k \cdot x_k = \sum_{i=1}^{k} i \cdot x_i$$

The minimum sum coloring problem asks to determine

$$\Sigma(G) = \min\{\mathrm{Sum}(\varphi) \mid \varphi \text{ is a proper coloring of } G\} \qquad \text{(MSCP)}$$

The number $\Sigma(G)$ is the *chromatic sum* of G.

For a partition $\mathcal{C}$ any permutation of the independent sets is still a proper coloring. However, for any pair $i > j$ such that $x_i > x_j$, a coloring that is better in terms of sum can be obtained by swapping the color classes C_i and C_j. Hence, for a partition $\mathcal{C}$ the smallest color sum is achieved by the *sorted coloring* in which sets are sorted in decreasing order of size, i.e., $x_1 \geq x_2 \geq \ldots \geq x_k$.

The number of colors of the proper coloring associated with $\Sigma(G)$ is the *strength* of the graph G and is denoted by $s(G)$. Note that $s(G) \geq \chi(G)$ and that there exist cases where the inequality is strict, see [11] for an example.

The concept of chromatic sum was introduced in [11], where it was also proved that the MSCP is NP-hard for general graphs. Polynomial time algorithms exist for some families of graphs, like trees. A review of these cases, theoretical lower bounds and NP-hard issues is given in [10]. Heuristic algorithms have been recently under investigation. Greedy algorithms have been studied in [13]. Metaheuristics approaches include a parallel genetic algorithm [9] and a tabu search algorithm [1].

The interest in fast heuristics and computational studies is motivated by a few practical applications of MSCP. In the literature, sum coloring of graphs has been used in VLSI design, scheduling and distributed resource allocation (see [15] for a list of references). For example, in computer systems with multiprocessors that are in competition over resources, we might seek an allocation under which no two jobs with conflicting requirements are executed simultaneously while minimizing the average completion time of the jobs.

In this paper, we design and test a local search algorithm for finding proper colorings, which correspond to upper bounds of the chromatic number, as well as to find lower bounds. A fundamental choice

in the application of local search algorithms is the definition of the search space. In the literature on graph coloring several proposals appeared, combining the following characteristics: partial vs complete colorings, proper vs unproper colorings, fixed vs variable k (see [3] for a review). We opt for a solution representation consisting of complete colorings that may be unproper and k variable. We then devise an evaluation function and a search strategy that guide the search towards good solutions for MSCP. We present the algorithm, that borrows ideas from both variable neighborhood search [6] and iterated local search [14], in Section 2. In Section 3, we describe how the same algorithm with just a few modifications can be applied to find lower bounds to the problem. Finally, in Section 4 we report about the computational results attained by the devised heuristic algorithm. The comparison with the benchmarks from the literature indicates that our algorithm finds 27 new best results on 38 instances and it is never outperformed.

The following definitions will be useful for the rest of the paper. Vertices linked by an edge in G are said to be *adjacent*. For a vertex set $X \subseteq V$ the subgraph of G *induced* by X is $G[X] = (X, \{uv \in E \mid u, v \in X\})$. The set of vertices adjacent to v in the graph induced by $X \cup v$ is denoted by $A_X(v) = \{w \in X \mid vw \in E(G)\}$. The degree of a vertex v induced by X is the size of $A_X(v)$ and is denoted by $d_X(v)$. The *saturation degree* of a vertex v induced by X is the number of different colors used in X and is denoted by $\delta_X(v)$. The edge density of a graph $G = (V, E)$ is computed as $\rho_G = |E|/\binom{|V|}{2}$.

2 A local search based primal heuristic

We describe a local search algorithm to find a proper coloring that gives an upper bound to the chromatic sum. First, we give a high level description and then present details on its parts. For an introduction to local search methods see [7, 16].

A candidate solution is any coloring φ (proper or not) that uses k colors and assigns exactly one color to each vertex. The evaluation function to minimize is

$$f(\varphi) = \sum_{i=1}^{k} i \cdot |C_i| + M \cdot |E(C_i)|$$

where $E(C_i)$ is the set of edges in C_i and M is a large positive natural number.

We define two neighborhood operators. They change a candidate solution φ acting on a pair $\langle v, j \rangle$ with $v \in V$ assigned to color $i = \varphi(v)$, and j a color different from i.

- The *one-exchange neighborhood operator* changes in φ the color of v from i to j. The color j is selected between 1 and $k + 1$, hence this operator may eventually increase the number of colors used by φ.
- The *swap neighborhood operator* changes in φ the color of v from i to j and the color of all vertices in $A_{C_j}(v)$ from j to i. This change is illustrated in Figure 1. The color j is any color from 1 to k already in use.

The overall search procedure is illustrated by means of *generalized local search machines* [7] in Figure 2. The initial state CH computes a complete coloring not necessarily proper by random assignment of colors to vertices or by means of the MDSAT or RLF heuristics. Afterwards, it performs swap changes until no further improvement can be obtained. The state type BI_1 computes the best swap change. It then applies the change only if it improves the current solution. The condition predicate IM is true if the solution has been improved and false otherwise. A solution that arrives to the state type BI_2 from the state BI_1 is thus a local optimum with respect to the swap neighborhood but not necessarily a proper coloring. A similar process occurs at the state type BI_2 that computes the best one-exchange and applies it, if it improves the current solution. The solution that enters in state SI is a proper coloring (this claim

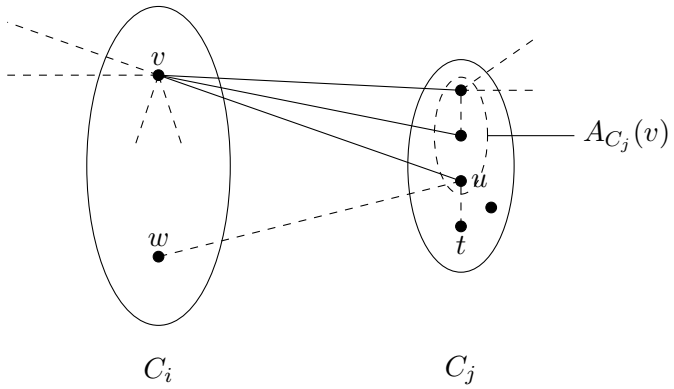

Figure 1: Two color classes C_i and C_j with $i < j$. Depending on the situation we may create new conflicts, eliminate existing conflicts, or both.

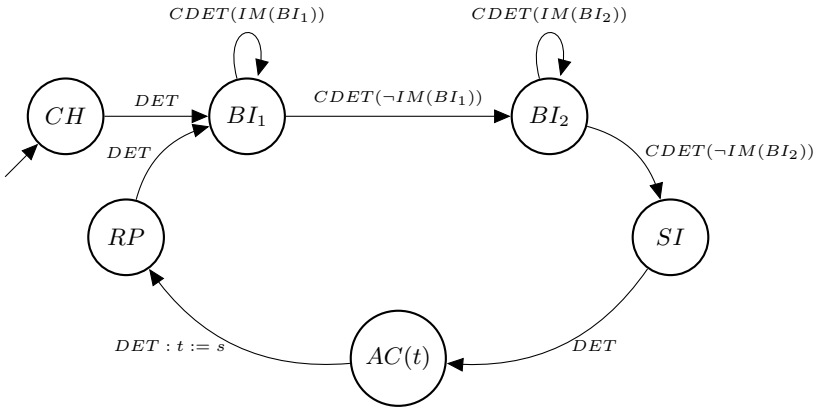

Figure 2: A generalized local search machine for MSCP.

will be justified below). In SI two operations are performed: (i) assign all vertices to their respective smallest legal color; (ii) permute the color labels to transform the coloring in a sorted coloring, in which vertices in the largest class are assigned to color 1, those in the second largest class to color 2, and so on. Note that operation (i) is the only one in the whole algorithm that can decrease the number of used colors.

The state type AC implements an acceptance criterion that compares the entering solution with the solution stored in t and outputs in solution s the best of the two. The state type RP applies a random perturbation to the solution consisting in changing the color to a fraction p of vertices randomly chosen from V. The new color for these vertices is chosen at random from the numbers $[1..k + 1]$.

The whole procedure terminates when a time limit, a number of total machine iterations or a number of non improving iterations has been reached. In this last case, we count as iteration any change in the current solution. The output is the solution stored in t.

Implementation details

In the following we refine the description of the procedures associated with the state types of the local search machine of Figure 2.

Initial solution (CH) The state CH generates an initial solution. We consider three alternatives: a completely random assignment of colors to vertices and two construction heuristic strategies.

Minimum Impact heuristics The *DSATUR* heuristic [2] chooses at each iteration the uncolored vertex with maximum saturation degree. The *MDSAT* algorithms in [13] enhance this rule by choosing the uncolored vertex that minimizes the impact on the saturation degree of the adjacent vertices.

In detail, let v be an uncolored vertex of a partial coloring, c its smallest legal color and $U(v) \subseteq A_V(v)$ the set of uncolored vertices adjacent to v. The impact of coloring v with c is measured by the number $\delta^+(v)$ of vertices in $U(v)$ whose saturation degree would increase and by the number $\delta^=(v) = |U(v)| - \delta^+(v)$. The MDSAT heuristics define 5 different ways of deciding the next vertex to color while trying to minimize the impact. In detail, they choose the vertex that achieves the following criteria in lexicographic order: (1) $(\max\{\delta^=\}, \min\{\delta^+\})$, (2) $(\min\{\delta^+\}, \max\{\delta^=\})$, (3) $(\min\{\delta^+/\delta^=\})$, (4) $(\min\{\delta^+/\delta^=\}, \min\{\delta^+\})$, (5) $(\min\{\delta^+/\delta^=\}, \max\{\delta^=\})$. Ties are broken by choosing the vertex with the largest label.

In each iteration MDSAT goes through all uncolored vertices and for each of them it computes the impact. This implies looking at all neighbors of the vertex and checking if the saturation degree will decrease or not. This extra cost makes MDSAT computationally worse than DSATUR. We implemented the MDSAT algorithms with a complexity of $O(n \cdot (n + m))$, with $m = |E|$. Finally, before output, we sort in decreasing order the color classes. In the experimental analysis, we will denote the instantiation of these heuristics more concisely by MDS($\cdot$), making explicit the vertex selection criterion used.

Recursive largest first (RLF) This heuristic looks at MSCP from the partitioning perspective. Given the sorting property of solutions a good strategy to construct a coloring with a small sum number might be to iteratively find a maximum independent set and remove its vertices from the graph. Clearly, finding the maximum independent set is itself an NP-hard problem. The RLF heuristic for graph coloring [12] implements a similar strategy finding maximal independent sets.

In [13] the RLF heuristic is enhanced with further rules and shown to perform as the best of the MDSAT heuristics. Here, we include in our analysis only the standard RLF for graph coloring. It has worst case complexity of $O(n^3)$.

Best improvement on swap (BI_1) The best swap is determined after evaluating all pairs $\langle v, j\rangle$, with $v \in V, j \in \{1 \dots k\}$ and $\varphi(v) < j$. Each pair is assessed by evaluating $\Delta = f(\varphi') - f(\varphi)$, where φ' is the new solution obtained after the corresponding swap. The computation of Δ can be decomposed into

$$\Delta(\langle v, j\rangle) = \Delta_1(\langle v, j\rangle) + M \cdot \Delta_2(\langle v, j\rangle)$$

where Δ_1 is the change in the color sum and Δ_2 is the change in the number of conflicting edges. These last two components can be computed as

$$\begin{aligned}
\Delta_1(\langle v, j\rangle) = &\quad (d_{C_j}(v) - 1) \cdot (i - j) \\
\Delta_2(\langle v, j\rangle) = &\sum_{u \in A_{C_j}(v)} (d_{C_i}(u) - 1) - d_{C_i}(v) - \sum_{u \in A_{C_j}(v)} d_T(u).
\end{aligned}$$

where $T = C_j \setminus A_{C_j}(v)$.

For a vertex v and a color l, a crucial operation is retrieving all vertices adjacent to v with color l, i.e., $A_{C_l}(v)$. To this goal, we keep an array of lists LL with $|V| \times k$ indices, where we record $A_{C_l}(v)$ for each (v, l).

The neighborhood examination is executed by considering all pairs of color classes (i, j) with $i < j$ and for each i considering all vertices $v \in C_i$. Each vertex is thus considered exactly once. The computation of Δ is dominated by the last term of Δ_2 that costs $O(|V|^2)$. Examining and evaluating the whole neighborhood takes therefore $O(|V|^3)$ time.

Note that when a vertex v is moved from one color class to another, LL must be changed for all the vertices adjacent to v. The complexity of this operation depends on the data structure that implements

the lists LL. We use a hash-map, where insertions and deletions are performed in amortized constant time. Hence, updating LL for a vertex that changes color has amortized complexity $O(|V|)$.

Best improvement on one-exchange (BI_2) The best one-exchange is found by considering all pairs $\langle v, j\rangle$. Each pair is evaluated by computing $\Delta = f(\varphi') - f(\varphi)$, where φ' is the new solution obtained after the corresponding one-exchange. As above, the computation of Δ can be decomposed into its two components Δ_1 and Δ_2. In this case, we have

$$\Delta_1(\langle v, j\rangle) = \quad j - i$$

$$\Delta_2(\langle v, j\rangle) = \quad M \cdot (d_{C_j}(v) - d_{C_i}(v)),$$

thus we spend time $O(|V|k)$ to find the best one-exchange, and $O(|V|)$ to update LL.

We can guarantee that a solution that is local optimum with respect to the one-exchange neighborhood is a proper coloring by setting the value of M to a value that makes always preferable an improvement in Δ_2 over an improvement in Δ_1. Thus, M has to be set larger than the best possible Δ_1. The largest change that can happen with the one-exchange neighborhood is when moving a vertex v from the color class k to the color class 1. In this case, we have $\Delta_1 = 1 - k$. Since in our definition of one-exchange we allow to move a vertex into an empty color class, k can be as large as $|V|$. Hence, setting $M \geq |V|$ will guarantee that for any coloring that is not proper, a conflict repair change, possibly opening a new color class, will always outweigh a worsening in the color sum.

Reassignment and sorting (SI) Reassigning to all vertices the smallest legal color can be done in linear time if an auxiliary array SC of size $|V|$ is maintained, where we record at index u the smallest legal color for vertex u. The value of SC is linked to the lists in LL. Precisely, $SC[u] = j$ if and only if the list $LL[u][j]$ is empty and the lists $LL[u][i]$ for all $1 \leq i < j$ are not empty. We can therefore maintain SC updated throughout the search operations while updating LL. This adds however an extra cost of $O(k)$ for each update of LL. Thus, when a vertex changes color, the cost of updating LL goes from the previously mentioned amortized $O(|V|)$ to $O(|V|k)$.

Sorting the color classes in decreasing order is done by keeping an additional data structure consisting of an array of lists CC indexed by color number i, $1 \leq i \leq k$, each list containing the vertices assigned to color class i. Sorting CC in decreasing order takes $O(k \log k)$. Consequently, we recolor the vertices in $O(|V|)$ and for each vertex update LL and SC in $O(|V|k)$.

The overall complexity of reassignment and sorting is dominated by recoloring and updating LL and SC, and is therefore $O(|V|^2 k)$.

Random perturbation (RP) We select the value p representing the fraction of vertices that change color uniformly at random from $[0.01; 0.05]$. Further, we choose uniformly at random $\lfloor p|V| \rfloor$ numbers in $[1..|V|]$. Finally, to each of these vertices we assign a new color chosen uniformly at random from $[1..k+1]$. Including the update of LL and SC the whole perturbation has a worst case of $O(|V|^2 k)$.

3 A local search heuristic for lower bounds

Given a clique H of G the best way to color it for MSCP uses the first $|H|$ colors of $[1..|V|]$. If we decompose the graph into disjoint cliques, and color each clique without considering the edges between them, then we obtain a lower bound to the chromatic sum. Formally, a *clique decomposition* $\mathcal{H}$ is a partition of the vertices V into $H_1, \ldots, H_l$, $H_i \cap H_j = \emptyset$, for all $i \neq j$, $\bigcup_i H_i = V$ and such that the induced subgraphs $G[H_1], \ldots, G[H_l]$ are all cliques. Each clique H_i, $1 \leq i \leq l$, has associated a best possible color sum of $|H_i|(|H_i|+1)/2$. A clique decomposition $\mathcal{H}$ of G gives therefore a lower bound for $\Sigma(G)$, i.e., $LB(\mathcal{H}) = \sum_{i=1}^{l} |H_i|(|H_i|+1)/2$. The best lower bound is then found by solving the problem

$$LB^* = \max\{LB(\mathcal{H}) \mid \mathcal{H} \text{ is a clique decomposition of } \mathcal{G}\}. \qquad \text{(LBP)}$$

Note that LB^* might be strictly smaller than $\Sigma(G)$.

This way of computing a lower bound is exploited in [17] where an approximation of LB^* is determined heuristically by applying the greedy algorithms from [13] on the complementary graph $\overline{G}$. Computational results show that the procedure leads to the best known lower bounds. Similarly, we approximate LB^* by applying our local search algorithm from Figure 2 in the previous section on the complementary graph. The procedure requires a few adjustments that are described next.

Candidate solutions are now colorings φ (proper or not) of the complement graph $\overline{G}$ corresponding to clique decompositions in the original graph G. The color classes $C_i, 1 \leq i \leq k$ associated to φ may contain edges, i.e., they may not be cliques in the original graph. The evaluation function accounts for both color sum and violations of the clique requirement. We define it as

$$f(\varphi) = \sum_{i=1}^{k} \frac{x_i(x_i+1)}{2} - M \cdot |\overline{E}(C_i)|$$

where x_i are the sizes of the color classes in $\overline{G}$ and M is a positive natural number. The goal of the local search is maximizing this function.

We decompose again Δ into a contribution Δ_1 for the change in color sum and a contribution Δ_2 for the number of conflicting edges. We write $\Delta = \Delta_1 - M\Delta_2$, thus improvements are given by $\Delta_1 > 0$ and $\Delta_2 < 0$. The value of Δ_1 has now to be updated for swaps to

$$\Delta_1(\langle v, j\rangle) = x_i(d_{C_j}(v) - 1) + x_j(1 - d_{C_j}(v)) + d_{C_j}(v)(d_{C_j}(v) - 2) + 1$$

and for one-exchanges to

$$\Delta_1(\langle v, j\rangle) = -x_i + (x_j + 1).$$

Further, we have to reconsider the value of M to ensure that after BI_2 we have a local optimum with respect to the one-exchange that is a proper coloring in $\overline{G}$ and a clique decomposition in G. Following the same reasoning as above we consider the largest improvement of Δ_1 for a one exchange, which occurs when we move a vertex from the smallest color class to the largest one. The extreme situation is when we have only two color classes, one containing one single vertex and another containing $|V| - 1$ vertices and we move the vertex from the former to the latter. In this case we have $\Delta_1 = |V| - 1$. Hence by setting $M \geq |V|$ we guarantee that a one-exchange diminishing the number of conflicting edges will always be preferred. For colorings that are not proper there is always a one-exchange with $\Delta_2 < 0$ since we can eventually open a new color class.

Finally, in the state type SI it is not anymore necessary to transform the coloring in a sorted coloring.

4 Experimental results

All algorithms are implemented in C++ and compiled with GNU g++4.4.5 using flag -O3. All tests are conducted on a Intel Core i7 processor 2.93Ghz with 8192MB cache L2, 4 GB RAM and running Ubuntu 10.10 operating system.

4.1 Construction heuristics

Construction heuristics are worth studying on their own since they constitute the fastest way to produce a proper coloring addressing the objective of MSCP. To test the construction heuristics we generated five uniform random graphs for all nine combinations of the characteristics: size $\{500, 1000, 2000\}$ and edge density $\{0.1, 0.5, 0.9\}$. We ran once each heuristic on the instances. Within each instance we rank the results separately for the two criteria, number of colors and color sum. For each criterion we compute separately the 95% confidence intervals of the mean rank value by means of the Friedman Rank Sum test for unreplicated two-way designs with Bonferroni correction [4]. The result of both analyses is visualized in Figure 3. We can conclude that MDS(3) and MDS(5) dominate the other heuristics with

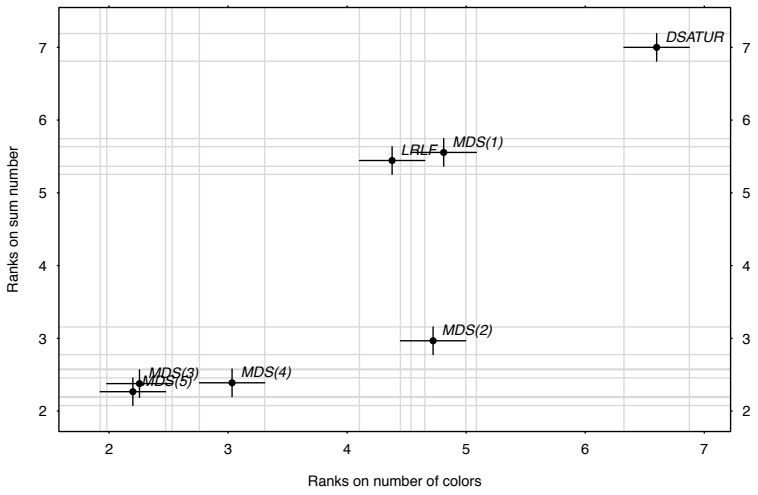

Figure 3: All-pairwise comparisons of construction heuristics on uniform random graphs. Two algorithms are significantly different in statistical terms in one of the two criteria if their intervals in the corresponding axis do not overlap.

statistical significance. Further analyses unveiled that these conclusions hold also within each class of instances at specific combinations of the size and density characteristics. An analysis of execution time is reported in Figure 4. The figure shows that all versions of MDSAT perform the same in terms of time and that MDSAT takes more time than both RLF and DSATUR. The version of RLF we considered was Lazy RLF, a fast implementation of RLF. With respect to growth the linear trend in the log-log plot indicates that all heuristics are polynomial in the size of the instance. On the largest and most dense graphs it takes more than 100 seconds for MDSAT to terminate.

4.2 Primal heuristics and lower bounds

We test three algorithms to find upper bounds.

RMDS(5) is the control, it repeats MDS(5) $|V|$ times starting from a different vertex every time.

RND+LS implements the local search machine of Figure 2 using a random assignment of colors in CH.

MDS(5)+LS implements the local search machine of Figure 2 using MDS(5) in CH.

In addition, we aim at comparing the results of these three algorithms with those in the literature. For this task we use as test instances a selection of the DIMACS instances for graph coloring [8]. Unfortunately, in the references with which we intend to compare [1, 5, 9, 13], some details to make the experiments reproducible are missing. Therefore, we decided, arbitrarily, to use one hour of computation time on the machine indicated. Since this termination condition is likely to observe algorithms at different stages of their convergence depending on the size of the instance, we also include in the comparison the results after a common number of full iterations of the local search machine in Figure 2. This number was determined by number of iterations achieved in one hour on the most onerous instance, `DSJC1000.9`, and amounts to 879 iterations. We distinguish the two versions of the algorithms by the suffixes `1h` and `879`.

We report the analysis of the comparison in Figure 5, left, and the numerical details in Table 1. One single run per instance was performed for a total of 38 instances. We do not consider algorithms from the literature in the analysis of Figure 5 since this would have required reducing the sample of instances to just 7. The analysis in Figure 5 is conducted in the same ways as for the construction heuristics. We

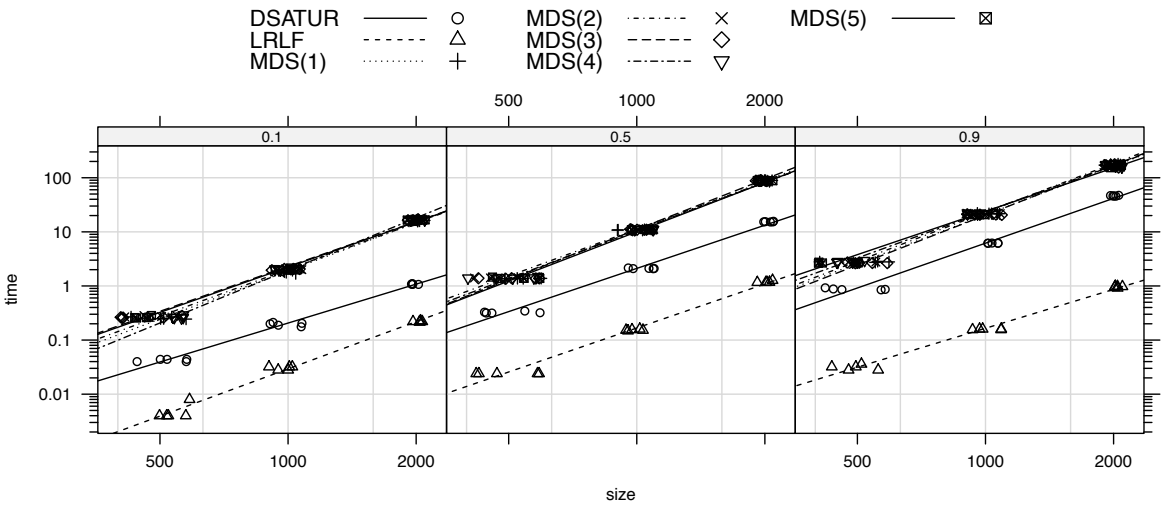

Figure 4: The growth in runtime for the construction heuristics. A log-log transformation is applied to both axes. A superimposed line represents the linear regression on the data points. A random perturbation is applied to the data points on the x-axis to make them visible.

can conclude with statistical significance that after 879 iterations MDS(5)+LS is better than RND+LS and RMDS(5) in terms of color sum. The results after one hour are clearly better than after 879 iterations both in terms of color sum and number of colors used. At this point the contribution of MDSAT as starting heuristic becomes insignificant. Comparing the numerical results of MDS(5)+LS-879 against those published in the literature and reported in Table 1, we see that out of 38 instances MDS(5)+LS-879 attains better color sums on 27 instances and it never performs worse. Moreover, the entity of the improvement grows with instance size. The largest value registered is 6362.

We compared our dual local search algorithm for computing lower bounds, denoted MDS(5)+LS, against the re-implementation of the algorithm in [17], denoted RMDS(n). This algorithm runs all five MDSAT heuristics each of them restarted using all vertices in V as the initial vertex. We run our MDS(5)+LS for one hour on all 38 instances, while we let RMDS(n) run to completion, which took usually less than one hour. In Figure 5, right, the two algorithms are not significantly different according to the Friedman Rank Sum test. However a Wilcoxon Signed Rank test, which is possible in this special case of only two algorithms and that ranks differences taking into account their entities, gives a p-value of 0.01443 indicating that RMDS(n) performs significantly better on the set of instances. In Table 5, we computed the differences between the upper bounds found after one hour and the lower bounds. Among the five instances that are closed, three where already closed in [17], and `zeroin.i.2` and `zeroin.i.3` are here closed for the first time.

5 Conclusions

We introduced a new local search algorithm for the minimum sum coloring problem. The algorithm combines ideas of variable neighborhood search and iterated local search. It oscillates between proper and unproper colorings and leaves k free to vary during the search. We assessed experimentally the algorithm under the criterion of color sum at two termination conditions. We observed that if long running times are possible, the algorithm can be simplified by using a random color assignment as initial solution. Otherwise, a greedy heuristics is profitable. The comparison with the results from the literature reveal that our new algorithm attains new best results on 27 out of 38 instances and it is never outperformed.

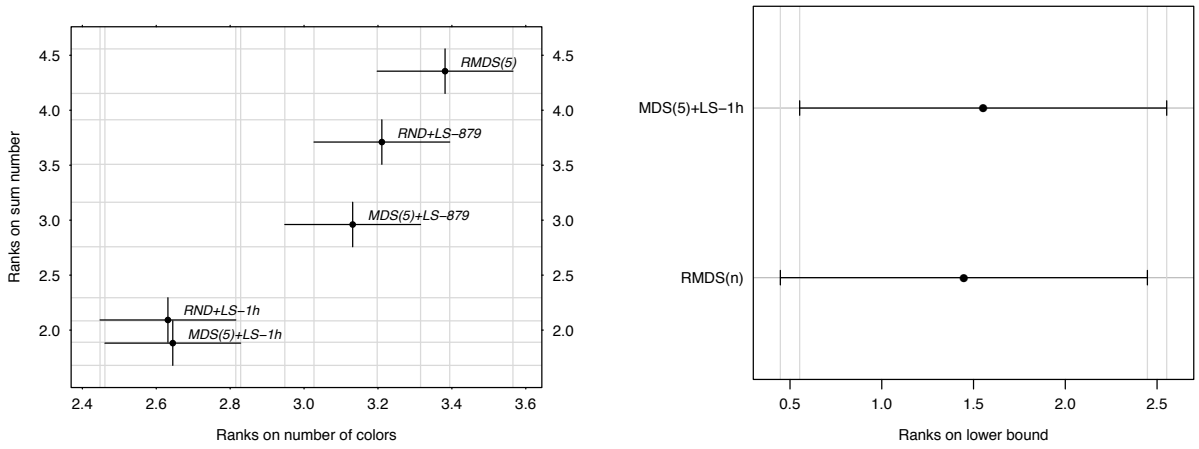

Figure 5: All-pairwise comparisons for local search algorithms for upper bounds (left) and lower bounds (right) on 38 DIMACS instances. Two algorithms are significantly different in statistical terms if their intervals in the corresponding axis do not overlap.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
instance	RMDS(n)	MDS(5)+LSc-1h	[1]	[5]	[13]	[9]	RMDS(5)	RND+LS-879	RND+LS-1h	MDS(5)+LS-879	MDS(5)+LS-1h	diff@879	diff@1h	UB-LB
DSJC125.1	238	238	344		352		338	338	326	331	326	-13	-18	88
DSJC125.5	504	493	1103		1141		1071	1057	1016	1050	1015	-53	-88	511
DSJC125.9	1600	1621	2631		2653		2624	2569	2512	2542	2511	-89	-120	890
DSJC250.1	537	521	1046		1068		1040	1032	978	1038	977	-8	-69	440
DSJC250.5	1150	1128	3779		3658		3619	3561	3295	3556	3281	-102	-377	2131
DSJC250.9	3972	3779	9198		8942		8861	8577	8430	8620	8412	-322	-530	4440
DSJC500.1	1163	1143	3205		3229		3153	3159	2965	3101	2951	-104	-254	1788
DSJC500.5	2616	2565			12717		12482	12379	11747	12427	11717	-290	-1000	9101
DSJC500.9	10074	9731			32713		32323	31333	30818	31187	30872	-1526	-1841	20798
DSJC1000.1	2499	2456			10276		10182	10423	10165	10155	10123	-121	-153	7624
DSJC1000.5	5787	5660			45408		44422	43744	43067	44382	43614	-1026	-1794	37827
DSJC1000.9	23863	23208			119111		117258	112593	112593	112749	112749	-6362	-6362	88886
2-Insertions_3	55	55		62			62	62	62	62	62	0	0	7
3-Insertions_3	84	84		92			92	94	92	92	92	0	0	8
anna	272	273				281	299	276	276	276	276	-5	-5	3
david	234	234				243	254	238	237	238	237	-5	-6	3
fpsol2.i.1	3402	3151		3405			3473	3403	3403	3403	3403	-2	-2	1
games120	442	442		446		460	449	446	443	444	443	-2	-3	1
huck	243	243		243		243	246	243	243	243	243	0	0	0
inithx.i.1	3581	3486		3679			3846	3676	3676	3676	3676	-3	-3	95
jean	216	216				218	223	217	217	217	217	-1	-1	1
miles250	316	318		343		347	338	330	325	331	325	-12	-18	7
miles500	677	686		755		762	717	726	712	724	712	-31	-43	26
mug100_1	186	188		211			203	204	202	203	202	-8	-9	14
mug100_25	183	186		214			203	204	202	203	202	-11	-12	16
mug88_1	163	164		190			180	180	178	179	178	-11	-12	14
mug88_25	161	162		187			179	180	178	179	178	-8	-9	16
myciel3	16	16		21		21	21	21	21	21	21	0	0	5
myciel4	34	34		45		45	46	47	45	45	45	0	0	11
myciel5	70	70		93		93	96	97	93	93	93	0	0	23
myciel6	142	142		189		189	194	200	193	189	189	0	0	47
myciel7	286	286		381		382	389	396	385	381	381	0	0	95
queen5_5	75	75				75	75	75	75	75	75	0	0	0
queen6_6	126	126		138		138	142	138	138	138	138	0	0	12
queen7_7	196	196				196	196	202	196	196	196	0	0	0
queen8_8	288	288				302	307	296	291	300	291	-2	-11	3
zeroin.i.2	1004	1004		1013			1029	1004	1004	1004	1004	-9	-9	0
zeroin.i.3	998	998		1007			1023	998	998	998	998	-9	-9	0

Table 1: Numerical results in terms of color sum on the DIMACS instances. Columns 2 and 3 report lower bounds. Columns 4–7 report results from the literature and columns 8–12 the results of our algorithms. Column 13 (14) gives the difference between column 11 (12) and the best of columns 4–7. The last column gives the difference between column 12 and the best of columns 2–3.

References

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

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

[3] M. Chiarandini, I. Dumitrescu, and T. Stützle. Stochastic local search algorithms for the graph colouring problem. In T. F. Gonzalez, editor, *Handbook of Approximation Algorithms and Metaheuristics*, Computer & Information Science Series, pages 63.1–63.17. Chapman & Hall/CRC, Boca Raton, FL, USA, 2007.

[4] W. Conover. *Practical Nonparametric Statistics*. John Wiley & Sons, New York, NY, USA, third edition, 1999.

[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. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130(3):449–467, 2001.

[7] H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 2004.

[8] D. S. Johnson and M. Trick, editors. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, Providence, RI, USA, 1996.

[9] Z. Kokosiński and K. Kwarciany. On sum coloring of graphs with parallel genetic algorithms. In *Proceedings of the 8th international conference on Adaptive and Natural Computing Algorithms*, pages 211–219. Springer, 2007.

[10] E. Kubicka. The chromatic sum of graphs; history and recent developments. *The International Journal of Mathematical Sciences*, 30:1563–1573, 2004.

[11] E. Kubicka and A. J. Schwenk. An introduction to chromatic sums. In *Proceedings of the 17th conference on ACM Annual Computer Science Conference*, CSC '89, pages 39–45, New York, NY, USA, 1989. ACM.

[12] 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.

[13] Y. Li, C. Lucet, A. Moukrim, and K. Sghiouer. Greedy Algorithms for the Minimum Sum Coloring Problem. In *Logistique et transports*, pages 1–6, Sousse Tunisia, 03 2009. Available as hal-00451266 at Hyper Articles en Ligne.

[14] H. R. Lourenço, O. Martin, and T. Stützle. Iterated local search. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 321–353. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[15] M. Malafiejski. Sum coloring of graphs. In M. Kubale, editor, *Graph Colorings*, volume 352 of *Contemporary Mathematics*, pages 55–65. AMS, 2004.

[16] W. Michiels, E. Aarts, and J. Korst. *Theoretical Aspects of Local Search*. Monographs in Theoretical Computer Science. Springer, 2007.

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