A new efficient RLF-like Algorithm for the Vertex Coloring Problem

Mourchid Adegbindin*, Alain Hertz†, Martine Bellaïche*

November 2, 2015

Abstract.
The Recursive Largest First (RLF) algorithm is one of the most popular greedy heuristics for the vertex coloring problem. It sequentially builds color classes on the basis of greedy choices. In particular the first vertex placed in a color class $C$ is one with a maximum number of uncolored neighbors, and the next vertices placed in $C$ are chosen so that they have as many uncolored neighbors which cannot be placed in $C$. These greedy choices can have a significant impact on the performance of the algorithm, which explains why we propose alternative selection rules. Computational experiments on 63 difficult DIMACS instances show that the resulting new RLF-like algorithm, when compared with the standard RLF, allows to obtain a reduction of more than 50% of the gap between the number of colors used and the best known upper bound on the chromatic number. The new greedy algorithm even competes with basic metaheuristics for the vertex coloring problem.

Keywords: graph coloring, greedy algorithm.

1 Introduction

Let $G$ be an undirected graph. A vertex coloring of $G$ is the assignment of a color to every vertex such that no two adjacent vertices have the same color. The chromatic number $\chi(G)$ of $G$ is the minimum number of colors used in a vertex coloring of $G$. A stable set is a set of pairwise non adjacent vertices. Hence, a vertex coloring of $G$ is a partition of its vertex set into stable sets called color classes. The Vertex Coloring Problem (VCP) is to determine the chromatic number of a given graph. This well known NP-hard problem [4] has many real world applications in many engineering fields, including scheduling, timetabling, register allocation.

*Département de génie informatique et génie logiciel, Ecole Polytechnique de Montréal
†Département de mathématiques et de génie industriel, Ecole Polytechnique de Montréal, corresponding author, alain.hertz@gerad.ca
and frequency assignment [20]. While exact algorithms [2, 9, 11, 12, 15, 17–19] can hardly solve instances with more than 100 vertices, real world instances can have thousands of vertices, and the use of approximate algorithms, heuristics or metaheuristics is then necessary.

The best known polynomial-time algorithm for approximating $\chi(G)$ has an approximation ratio of $O(n(\log \log n)^2/(\log n)^3)$ [10], where $n$ is the number of vertices in $G$. Metaheuristics for the VCP generally produce colorings with much less colors, but without any performance guarantee. The first ones, proposed in the eighties, were based on simulated annealing [3, 14] and tabu search [13]. Nowadays, a much wider variety of metaheuristics is available, a bibliography being maintained by Chiarandini and Gualandi [6]. A vast majority of these metaheuristics solve the $k$-VCP which is, for a given integer $k$, to determine whether a graph admits a vertex coloring that uses at most $k$ colors. An upper bound on the chromatic number is therefore needed to fix an initial value for $k$ which is then decreased until no solution to the $k$-VCP can be found. Such an upper bound is typically obtained by using fast heuristics for the VCP.

The most popular fast heuristics for the VCP are based on greedy constructive procedures. These algorithms sequentially color the vertices following some rule for choosing the next vertex to color and the color to use. The best known such heuristics are the DSATUR [1] and RLF [16] algorithms. Computational studies on these algorithms [7] have shown that RLF outperforms DSATUR in terms of quality on most instances, while RLF is more time consuming with a complexity of $O(mn)$ to be compared with the $O(n^2)$ complexity of DSATUR, where $n$ is the number of vertices and $m$ the number of edges.

The aim of this paper is to propose new greedy algorithms for the VCP that can compete with basic metaheuristics. In particular, we will show that greedy choices made in the RLF algorithm can be modified in a very simple way, often with the effect of reducing the number of colors used. The new proposed RLF-like algorithms have a complexity that ranges from $O(mn)$ to $O(mn^2)$.

In the next section, we describe the standard RLF algorithm as well as some of its variations. The proposed alternative greedy choices are given in Section 3. Computational experiments are reported in Section 4, where we compare the new RLF-like algorithms with the standard RLF as well as with DSATUR and a metaheuristic.

## 2 The RLF algorithm and some variations.

The Recursive Largest First (RLF) algorithm was proposed in 1979 by F. Leighton [16]. Roughly speaking, this algorithm builds a sequence of stable sets, each one corresponding to a color class. Let $C$ be the next color class to be constructed,
let $U$ denote the set of uncolored vertices and let $W$ be the set (initially empty) of uncolored vertices with at least one neighbor in $C$. Every time a vertex in $U$ is chosen to be moved to $C$, all its neighbors in $U$ are moved from $U$ to $W$. The first vertex $v \in U$ to be included in $C$ is one with the largest number of neighbors in $U$. The rest of $C$ is built as follows: while $U$ is not empty, the next vertex to be moved from $U$ to $C$ is one having the largest number of neighbors in $W$. Ties are, if possible, broken by choosing a vertex with the smallest number of neighbors in $U$.

For a vertex $u \in U$, we denote $A_U(u)$ and $A_W(u)$ its number of neighbors in $U$ and $W$, respectively. Also, when $v$ is the first vertex placed in a color class, we denote $C_v$ the color class that contains it. Given a vertex $v$, the algorithm in Figure 1 summarizes how $C_v$ is constructed by the RLF algorithm.

**Construction of $C_v$**

**Input** A set $U$ of uncolored vertices and a vertex $v \in U$.

**Output** A stable set $C_v$ that contains $v$.

Initialize $W$ as the set of vertices in $U$ adjacent to $v$.
Remove $v$ and all its neighbors from $U$ and set $C_v \leftarrow \{v\}$.

while $U \neq \emptyset$ do

  Select a vertex $u \in U$ with largest value $A_W(u)$. In case of ties, choose one with smallest value $A_U(u)$.

  Move $u$ from $U$ to $C_v$, and move all neighbors $w \in U$ of $u$ to $W$.

end while

Figure 1: Construction of a color class.

The construction of $C_v$ can easily be implemented by updating the numbers $A_U(x)$ and $A_W(x)$ each time a vertex is removed from $U$. More precisely, $A_W(x)$ is initially (when $W = \emptyset$) set equal to 0 for all $x \in U$, and the initial values $A_U(x)$ can easily be obtained in $O(m)$ time. Then, each time a vertex $w$ is moved from $U$ to $W$, $A_W(x)$ is incremented by one unit and $A_U(x)$ is decreased by one unit for all neighbors $x \in U$ of $w$. Also, when a vertex $u \in U$ is moved from $U$ to $C_v$, $A_U(x)$ is decreased by one unit for all neighbors $x \in U$ of $u$. Hence, there are $O(m)$ such updates, and since the selection of the next vertex to be moved to $C_v$ can be done in $O(n)$ time, the construction of $C_v$ has a total complexity of $O(m + n|C_v|)$.

As mentioned above, the RLF algorithm constructs a sequence of such stable sets. It is summarized in Figure 2. Since every vertex belongs to exactly one color class, the overall complexity of the RLF algorithm is $O(km + n^2)$, where $k$
is the number of colors used. The RLF algorithm has therefore a $O(mn)$ worst case complexity.

**Algorithm RLF**

**Input** A graph $G$.

**Output** A coloring of the vertices of $G$.

$k \leftarrow 0$.

**while** $G$ contains uncolored vertices **do**

Let $U$ be the set of uncolored vertices. Set $k \leftarrow k + 1$.

Choose a vertex $v \in U$ with largest value $A_U(v)$.

Construct $C_v$ and assign color $k$ to all vertices in $C_v$.

**end while**

Figure 2: The standard RLF algorithm.

Several greedy choices are made by the RLF algorithm. The first one occurs when selecting the first vertex $v$ to be placed in a color class. Also, the selection of the next vertices to be placed with $v$ in $C_v$ is based on greedy choices. As observed by Johnson et al. [14], better results can be obtained by modifying these choices, which explains why they proposed two variations of the RLF algorithm.

**First variation: algorithm RLF***

The greedy choices made during the construction of $C_v$ aim to minimize the number of edges in the residual graph $G'$ obtained by removing the colored vertices from the original graph. Let $P$ denote the problem of finding a color class $C$ such that the number of edges in the residual graph $G'$ is minimized. The RLF* algorithm iteratively builds color classes by solving $P$ with an exact procedure.

**Second variation: algorithm XRLF**

The XRLF algorithm plays with four parameters $T, L, R, E$ in the following way.

- Each color class is build by first generating a given number $T$ of stable sets $I_1, \cdots, I_T$, and then choosing as a color class the stable set $I_i$ that induces a residual graph $G'$ with a minimum number of edges.
- The first vertex placed in each $I_i$ is chosen at random among the uncolored vertices. Then, additional vertices are added to $I_i$ until the number of vertices in $U$ is less than a fixed limit $L$. The rest of $I_i$
is obtained using an exhaustive search with always the same aim of
minimizing the number of edges in the residual graph $G'$.

- The selection of additional vertices to be added to $I_i$ (when $|U| > L$)
is done as follows: $R$ vertices $w_1, \ldots, w_R$ are chosen at random in $U$,
and a vertex $w_j$ with largest value $A_W(w_j)$ is added to $I_i$.
- Color classes are obtained in this way until the residual graph contains
less than a fixed number $E$ of vertices, in which case an exact coloring
algorithm is used to build the last color classes.

As noticed by Johnson et al [14], RLF* solves a series of NP-hard problems, but
there is no guarantee that it produces a coloring with $\chi(G)$ colors. Concerning
XRLF, different values can be assigned to the four parameters $E, R, T, L$. In
particular, if $E = L = 0, T = 1$, and $R$ is sufficiently large, then XRLF is similar
to the original RLF, while if $E = 0$ and $L = n$, then XRLF is equivalent to RLF*.

Both RLF* and XRLF combine the greedy choices of the standard RLF with
exact non polynomial-time procedures. In this paper, we rather propose RLF-like
algorithms with a polynomial-time complexity. They are obtained from
the original RLF by changing some of the greedy rules. As will be shown, these very
simple modifications make it possible to get an algorithm that produces much
better results than the original RLF, and even competes with basic metaheuristics.

3 Alternative greedy choices.

In what follows, we use the same notations as in the original RLF. In particular,
for a vertex $x \in U$, $A_W(x)$ denotes the number of neighbors of $x$ in $W$. When a
vertex $x$ is moved from $U$ to $W$, the value $A_W(x)$ is frozen in that sense that it is
not updated anymore. Hence, the value $A_W(x)$ for a vertex $x \in W$ is equal to the
last value $A_W(x)$ before the move of $x$ to $W$. We now describe two modifications
of the greedy choices made in RLF.

3.1 Alternative greedy choice for the selection of the next
vertex to be placed in $C_v$.

The first greedy choice for which we propose an alternative is the one done when
selecting a vertex $w \neq v$ to be placed in $C_v$. For a vertex $u \in U$, the value $A_W(u)$
is a kind of similarity measure between $u$ and the vertices already in $C_v$. Indeed,
it corresponds to the number of uncolored neighbors of $u$ which are also neighbors
of vertices in $C_v$. The RLF algorithm selects the vertex $u \in U$ with maximum
value $A_W(u)$. We propose another selection rule. For every vertex $u \in U$, let

\[ B(u) = \sum_{w \in W \cap N(u)} (d(w) + A_W(w)) \]
where \( N(u) \) is the set of neighbors of \( u \) and \( d(w) \) is the number of uncolored neighbors of \( w \) at the beginning of the construction of \( C_v \). The next vertex to be placed in \( C_v \) is then chosen as one with maximum value \( B(w) \). The idea behind this rule is twofold and can be explained as follows. Let \( G' \) be the graph induced by the uncolored vertices at the end of the construction of \( C_v \):

- by maximizing \( \sum_{w \in W \cap N(u)} d(w) \), we aim to favor the choice of a vertex \( u \) with mainly uncolored neighbors \( w \) of large degree in \( W \) so that the maximum degree in the residual graph \( G' \) is minimized;
- by maximizing \( \sum_{w \in W \cap N(u)} A_W(w) \), we aim to have many vertices in the residual graph \( G' \) similar to those in \( C_v \), so that the next color class can be as large as \( C_v \).

Let \( L = \{ u \in U \mid B(u) = \max_{x \in U} B(x) \} \) and \( L' = \{ u \in L \mid A_W(u) = \max_{x \in L} A_W(x) \} \). The next vertex \( u \) placed in \( C_v \) is one in \( L' \) with smallest value \( A_U(u) \). In other words, we choose a vertex \( u \) with largest value \( B(u) \), we break ties by choosing a vertex with largest value \( A_W(u) \), and if this is not sufficient, by selecting one with smallest value \( A_U(u) \).

As was the case for the values \( A_W(u) \) and \( A_U(u) \) in the original RLF algorithm, the initial values for \( B(u) \) can easily be obtained in \( O(m) \). Then, each time a vertex \( u \) is moved to \( W \), \( B(x) \) is incremented by \( A_W(x) \) units for all neighbors \( x \in U \) of \( u \). Hence, this does not change the complexity of the construction of the stable set \( C_v \). The procedure is summarized in Figure 3.

**Construction of \( C_v \) based on function \( B \)**

**Input** A set \( U \) of uncolored vertices and a vertex \( v \in U \).

**Output** A stable set \( C_v \) that contains \( v \).

Initialize \( W \) as the set of vertices in \( U \) adjacent to \( v \).

Remove \( v \) and all its neighbors from \( U \) and set \( C_v \leftarrow \{v\} \).

**while** \( U \neq \emptyset \) **do**

- Let \( L = \{ u \in U \mid B(u) = \max_{x \in U} B(x) \} \) and \( L' = \{ u \in L \mid A_W(u) = \max_{x \in L} A_W(x) \} \).
- Select a vertex \( u \in L' \) with smallest value \( A_U(u) \).
- Move \( u \) from \( U \) to \( C_v \), and move all neighbors \( w \in U \) of \( u \) to \( W \).

**end while**

Figure 3: Alternative procedure for the construction of a color class.
3.2 Alternative selection of the first vertex of a color class.

We propose to change the selection rule for the first vertex $v$ to be placed in a color class. In the RLF algorithm, $v$ is a vertex with a maximum number $A_U(v)$ of neighbors in $U$. We propose several alternatives.

(a) The first one is to build a stable set $C_v$ for every uncolored vertex $v$ and to choose one that induces a residual graph with a minimum number of edges. This gives an algorithm with total complexity $O(mn^2)$.

(b) In order to avoid this increase in complexity from $O(mn)$ to $O(mn^2)$, we propose to construct a stable set $C_v$ for a constant number $M$ of uncolored vertices $v$ having the highest values $A_U(v)$.

(c) A solution in-between is to follow alternative (b), but with $M = \lfloor pn \rfloor$ and $0 < p < 1$, which also gives an $O(mn^2)$ overall complexity, but approximately decreases the total computing time by a factor $p$ when compared with alternative (a).

4 Computational experiments.

While the proposed changes to the original RLF algorithm might seem of little importance, we show in this section that their impact on the performance of the algorithm is significant. We analyze the results obtained by eight versions of the proposed algorithm. These versions are denoted $\alpha$-RLF-$\beta$, where $\alpha = A$ or $B$, and $\beta = n, 10\%, 10, 1$:

- $\alpha = A$ means that we use the standard values $A_W(u)$ to decide which vertex $u$ is added to $C_v$, while $\alpha = B$ stands for the proposed alternative that uses values $B(u)$;
- The various values for $\beta$ indicate which strategy we follow to determine the first vertex $v$ of a color class: $\beta = n$ is for alternative (a); $\beta = 10$ or 1 are for alternative (b) with $M = 10$ and 1, respectively; $\beta = 10\%$ is for alternative (c) with $p = 0.1$.

Hence, $A$-RLF-1 is the standard RLF algorithm, both $\alpha$-RLF-1 and $\alpha$-RLF-10 have an $O(mn)$ complexity, and both $\alpha$-RLF-10% and $\alpha$-RLF-$n$ have an $O(mn^2)$ complexity. Because $\alpha$-RLF-10, $\alpha$-RLF-10% and $\alpha$-RLF-$n$ are about 10, $\frac{n}{10}$ and $n$ times slower than $\alpha$-RLF-1, we also consider the 10-RLF, 10%-RLF and $n$-RLF algorithms which consist of applying the standard RLF 10, $\frac{n}{10}$ and $n$ times, respectively, and to store only the best of the produced colorings. The $\beta$-RLF and $\alpha$-RLF-$\beta$ algorithms have therefore comparable computing times, which helps to better analyze the impact of the proposed selection rules for the first vertex of
a color class. Note that when the standard RFL is applied several times on an instance, the results may be different from one run to the other, because ties in the greedy choices are broken randomly.

We have tested the A-RLF-\(\beta\) and \(\beta\)-RLF algorithms on random graphs \(R_{n,d}\) constructed as follows: given a positive integer \(n\) and a real number \(d \in [0,1]\), \(R_{n,d}\) has \(n\) vertices and all \(\frac{n(n-1)}{2}\) ordered pairs of vertices have a probability \(d\) of being linked by an edge. Computational results on \(R_{n,d}\) graphs with \(n = 700, 800, \ldots, 1500\), and \(d = 0.1, 0.5, 0.9\) are reported in Table 1. Each result is an average on 5 instances. We indicate the average number of colors produced by each algorithm as well as the average computing times in seconds (shown in parenthesis), using a 3 GHz Intel Xeon X5675 machine with 8 GB of RAM. In Figure 4, we represent the evolution of the computing time (using an logarithmic scale) for \(d = 0.5\) and \(d = 0.9\) when the number of vertices increases. The top curve corresponds to \(10^{-8}mn^2 = 10^{-8}dn^3(n - 1)/2\), and indicates the expected shape of the curves for algorithms A-RLF-10\% and A-RLF-\(n\).

We observe that the A-RLF-\(\beta\) algorithms are faster than the \(\beta\)-RLF ones, which means that the construction of a stable set \(C_v\) for a number \(M\) of vertices \(v\) increases the computing time by a factor smaller than \(M\). But the increase is real and makes the A-RLF-10\% and A-RLF-\(n\) less attractive for large graphs. For example, while RLF finds a coloring of \(R_{1500,0.9}\) in 6 seconds, about 100 minutes are needed by A-RLF-\(n\). But the number of colors is reduced from 407.4 to 332, which represents a gain of 18\%. For comparison, applying the RLF algorithm \(n = 1500\) times on the same graph (i.e., using \(n\)-RLF) decreases the number of colors by only 7 units. These absolute and relative (in percent) gains in colors of the A-RLF-\(\beta\) and \(\beta\)-RLF algorithms with respect to the standard RLF are shown in Figure 5 for \(d = 0.5\) and 0.9. Similar curves can be obtained by comparing B-RLF-\(\beta\) with \(\beta\)-RLF. We clearly observe that the alternative selection rules (i.e., parameter \(\beta\)) for the first vertex of a color class have a very positive impact on the performance of the RLF algorithm.

We have tested the eight versions of the \(\alpha\)-RLF-\(\beta\) algorithm on the DIMACS benchmark graph coloring instances which come from various sources. For a detailed description of these instances, the reader can refer to [20]. We only report results for the seemingly most challenging instances, which are those for which the DSATUR algorithm is not able to produce a coloring with \(k\) colors, where \(k\) is the best known upper bound on \(\chi(G)\). This gives a total of 63 instances with \(36 \leq n \leq 10,000\) and \(290 \leq m \leq 990,000\).
Two measures are used to analyze the performances of the algorithms. The first one is the total absolute percent deviation (TAPD) from the best known results. More precisely, for a set $S$ of instances, let $b_s$ be the best known upper bound on the chromatic number of $s \in S$, and let $a_s$ be the number of colors produced by one of the algorithms. The TAPD of this algorithm is then defined as follows:

$$TAPD = 100 \frac{\sum_{s \in S} (a_s - b_s)}{\sum_{s \in S} b_s}.$$ 

This measure gives more importance to results on graphs with a large number of colors. For example, assume there are only two instances $s_1$ and $s_2$ in $S$ with $b_{s_1} = 10$ and $b_{s_2} = 100$. If an algorithm Algo$1$ finds a coloring of $s_1$ with 11 colors and a coloring of $s_2$ with 100 colors, then its TAPD is $\frac{100}{110} = 0.909$. If a second algorithms Algo$2$ finds $a_{s_1} = 10$ and $a_{s_2} = 110$, then its TAPD is 10 times larger, which means that Algo$1$ could appear as better than Algo$2$. Both algorithms have however similar results since they have reached the best known upper bound on one of the two instances, and have produced a coloring with 10% more colors than the best known upper bound on the other instance. To compensate such a bias, we also compute the average relative percent deviation (ARPD) which is defined as follows:

$$ARPD = \frac{100}{|S|} \sum_{s \in S} \frac{a_s - b_s}{b_s}.$$ 

For the above example, both algorithms Algo$1$ and Algo$2$ have an ARPD of 5. The detailed results of our experiments on DIMACS benchmark instances appear in Table 2. Each line in the table corresponds to a particular graph. The first columns indicate the name, the number of vertices and the number of edges of the considered graph. The next column displays the best known upper bound $k$ on the chromatic number. Note that all versions of the $\alpha$-RLF-$\beta$ algorithm possibly make random choices when choosing a first vertex $v$ for a color class $C_v$, or the next vertices to be added to $C_v$. Such choices occur when ties cannot be broken by the proposed selections rules. Each algorithm was therefore run 10 times, and we report the minimum (column min), the average (column av.) and the maximum (column max) numbers of colors used by each version of the algorithm. The last line of table 2 indicates the total number of colors, for the 63 instances. In Figure 6, we indicate the TAPDs and ARPDs associated with the best, average and worst results of each algorithm.

Insert Table 2 and Figure 6 around here.
For $\beta = 1$, we observe that the standard function $A$ proposed by Leighton produces better results than function $B$. The difference between the best and the worst results is however much smaller with $\alpha = B$ than with $\alpha = A$, which indicates that the proposed alternative greedy choice is more stable. This becomes even more evident with $\beta = 10$. Indeed, while the best minimum and average results are obtained with $\alpha = A$, the best worst case comes with $\alpha = B$. For $\beta = 10\%$, the difference in terms of total number of colors between the worst and the best case with $\alpha = A$ is 58 (2435-2377) while this difference is equal to 41 (2436-2395) with $\alpha = B$. Interestingly, by comparing the TAPDs, we observe that $A$-RLF-$n$ has a better best case than $B$-RLF-$n$, but worse average and worst cases.

The gap between the total average number of colors produced by $A$-RLF-1 and the best known upper bound $k$ is equal to 485.4 (2621.4-2136). This gap is reduced to 234 (2370-2136) with $A$-RLF-$n$, which represents an improvement of 51.8%. When comparing $B$-RLF-1 with $B$-RLF-$n$, the improvement is even larger since the difference between the total average number of colors and $k$ is reduced from 527 (2663-2136) to 227.8 (2363.6-2136), which corresponds to an improvement of 56.8%. The majority of this improvement is already obtained by setting $\beta = 10$ instead of 1. Indeed, the gain is of 30.3% for $\alpha = A$ and of 33.4% for $\alpha = B$.

The importance of modifying the greedy choices made in RLF is very clear on some instances. One of the best illustrations is given by instance school1 where the standard RLF algorithm uses 26 colors while $A$-RLF-10 and $B$-RLF-10 find colorings with only 16 colors. The best known upper bound for this instance is 14, and is reached with $\beta = 10\%$ and $\beta = n$. Another good example is instance flat300_20 where the best coloring produced by RLF uses 36 colors, while only 20 colors are used by $A$-RLF-$n$ and $B$-RLF-$n$ (which is the chromatic number of the considered graph). Note however that the standard RLF eventually produces better results than all proposed variations. For example, for instance DSJR500.1c, RLF uses 89 colors while the best coloring obtained with all proposed alternatives contains 90 colors.

Insert Table 3 around here.

It is important to mention that the improvement in quality obtained by using $\beta = 10\%$ or $\beta = n$ instead of $\beta = 1$ or $\beta = 10$ has a price. Indeed, we report in Table 4 the average computing times of the $A$-RLF-\beta algorithms (similar times are needed by the $B$-RLF-\beta algorithms). For example, for instance DSJC1000.9, the best coloring produced by RLF uses 275 colors and is obtained in 2 seconds, while
only 236 colors are used by $A$-RLF-$n$, such a coloring being obtained in about 14 minutes. Also, the optimal coloring in 100 colors of instance qg.order100 is obtained by RLF in 16 seconds, while 15 hours are needed by $A$-RLF-$n$, and 5 hours by $A$-RLF-10%. But for instances of reasonable size like flat300.20, the reduction from 36 to 20 colors mentioned above is obtained in one second. Also, for instance school1, one second is sufficient to reduce the number of used colors from 26 to 14.

It is also interesting to observe that while $A$-RLF-$n$ and $B$-RLF-$n$ show similar behaviors, they produce very different results on some instances. For example, $B$-RLF-$n$ is able to find a coloring with 92 colors for DSJR500.1c while the best coloring produced by $A$-RLF-$n$ for this instance contains 4 additional colors. On the opposite, the best coloring produced by $B$-RLF-$n$ for wap03 has 51 colors, while colorings with only 47 colors were found by $A$-RLF-$n$. In summary the two algorithms seem complementary, which explains why we now report results obtained by running both $A$-RLF-$\beta$ and $B$-RLF-$\beta$, and keeping only the best of the two produced colorings. This new algorithm, called $AB$-RLF-$\beta$ is compared to DSATUR [1] and to the Short Tabu algorithm studied in [8], which consists in taking the best result of 5 runs with 100,000 iterations of the TABUCOL algorithm of Hertz and de Werra [13]. Comparisons between these algorithms are shown in Table 3, while their TAPDs and ARPDs appear in Figure 7. The first four columns of Table 3 are the same as those in Table 2. The next columns display the best result produced by DSATUR (column DS), the minimum, average and maximum numbers of colors used by each version of the $AB$-RLF-$\beta$ algorithm, and finally the best result produced by Short_Tabu (column ST). The last line of Table 4 shows totals on the 63 instances.

We observe that while the best total number of colors used by $\alpha$-RLF-$n$ is 2342 for $\alpha = A$ and 2348 for $\alpha = B$ (see Table 2), it is reduced to 2326 by $AB$-RLF-$n$, which is even better than the total of 2331 colors produced by Short_Tabu. The best TAPDs and ARPDs of the $AB$-RLF-$\beta$ algorithms as well as those of DSATUR, RLF and Short_Tabu are shown in Figure 8. We see, for example, that while DSATUR has a TAPD of 27.95%, the standard RLF reduces it to 20.8% and the $AB$-RLF-$n$ algorithm to 8.9%, which corresponds to an additional gain of 11.9%. A perfect illustration of the effectiveness of the proposed algorithms is given by instance DSJC1000.9. The DSATUR algorithm finds a coloring with 297 colors while only 275 are necessary with RLF. With $\alpha$-RLF-$n$, we were able
to gain 39 (275-236) additional colors, which is 6 units better than the result produced by Short_Tabu. The coloring with 236 colors that we have obtained is however 14 units above the best results produced by more complex and more time-consuming metaheuristics.

5 Conclusion

The RLF algorithm is a very popular heuristic for the vertex coloring problem, mainly because it is easy to implement and has a relatively low complexity in $O(mn)$. Since various greedy choices made in RLF can have a very big impact on the performance of the algorithm, we have proposed alternative choices. Experiments have shown that much better colorings can be obtained with these alternative greedy choices. The proposed $AB$-RLF-$n$ algorithm has an $O(mn^2)$ complexity, and competes with basic metaheuristics like Short_Tabu. The difference between the number of colors used and the best known upper bound is, on average, reduced by more than 50% when compared with the standard RLF. More than 30% of this improvement can be obtained with $AB$-RLF-10 which is an $O(mn)$ algorithm, like RLF. By implementing the different versions of our algorithms, we have not sought to optimize the code, our goal being rather to demonstrate the quality gain that can be achieved by modifying the greedy choices made in the standard RLF algorithm. Better implementations based on the same ideas as those presented in [5] would certainly lead to faster algorithms. We finally note that the $AB$-RLF-$n$ algorithm is a perfect candidate for a parallel implementation since each color class is obtained by choosing among different stable sets $C_v$ (one for every uncolored vertex $v$), and these stable sets can be generated independently by different processors.

References

[1] D. Brélaz. New methods to color the vertices of a graph. Communications of the ACM 22:251–256, 1979.

[2] J.R. Brown. Chromatic scheduling and the chromatic number problem. Management Science 19:456–463, 1972.

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

[4] M. Garey and D.S. Johnson. Computer and Intractability. Freeman, San Francisco, 1979.
[5] M. Chiarandini, G. Galbiati, and S. Gualandi. Efficiency issues in the RLF heuristic for graph coloring. In Proceedings of the IX Metaheuristics International Conference, (MIC 2011), pages 461–469, 2011.

[6] M. Chiarandini and S. Gualandi. http://www.imada.sdu.dk/~marco/gcp/

[7] M. Chiarandini and T. Stuetzle. An analysis of heuristics for vertex colouring. In P. Festa, editor, Experimental Algorithms, Proceedings of the 9th International Symposium, (SEA 2010), volume 6049 of Lecture Notes in Computer Science, pages 326–337. Springer, May 2010.

[8] P. Galinier, A. Hertz and N. Zufferey. An adaptive Memory Algorithm for the k-Colouring Problem, Discrete Applied Mathematics 156:267–279, 2008.

[9] S. Gualandi and F. Malucelli. Exact Solution of Graph Coloring Problems via Constraint Programming and Column Generation. INFORMS Journal on Computing 24:81–100, 2012.

[10] M. Halldórsson. A still better performance guarantee for approximate graph coloring. Information Processing Letters 45:19–23, 1993.

[11] S. Held, W. Cook and E.C. Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. Mathematical Programming Computation 4:363–381, 2012.

[12] F. Herrmann and A. Hertz. Finding the chromatic number by means of critical graphs. ACM Journal of Experimental Algorithmics 7:1–9, 2002.

[13] A. Hertz and D. de Werra. Using Tabu Search Techniques for Graph Coloring. Computing 39:345–351, 1987.

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

[15] M. Kubale and B. Jackowski. A generalized implicit enumeration algorithm for graph coloring. Communications of the ACM 28:412–418, 1985.

[16] F.T. Leighton. A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards 84:489–503, 1979.

[17] E. Malaguti, M. Monaci and P. Toth, An Exact Approach for the Vertex Coloring Problem. Discrete Optimization 8:174–190, 2011.

[18] A. Mehrotra and M.A. Trick. A column generation approach for exact graph coloring. INFORMS Journal on Computing 8:344–354, 1996.
[19] J. Peemöller. A correction to Brélaz’s modification of Brown’s coloring algorithm. *Communications of the ACM* 26:593–597, 1983.

[20] M.A. Trick. http://mat.gsia.cmu.edu/COLOR/color.html
| $d$ | algorithm | 700  | 800  | 900  | 1000 | 1100 | 1200 | 1300 | 1400 | 1500 |
|-----|-----------|------|------|------|------|------|------|------|------|------|
| 0.1 | RLF       | 19.00 (0) | 20.60 (0) | 22.60 (0) | 24.40 (0) | 26.00 (0) | 28.00 (0) | 30.00 (0) | 31.60 (0) | 33.40 (0) |
|     | 10-RLF    | 18.40 (0) | 20.20 (0) | 22.00 (0) | 24.00 (0) | 25.80 (1) | 27.60 (1) | 29.20 (1) | 31.20 (1) | 33.00 (1) |
|     | 10%-RLF   | 18.00 (1) | 20.00 (1) | 22.00 (2) | 24.00 (3) | 25.80 (6) | 27.00 (8) | 29.00 (9) | 31.00 (13) | 33.00 (17) |
|     | n-RLF     | 18.00 (8) | 20.00 (13) | 22.00 (22) | 24.00 (33) | 25.80 (53) | 27.00 (76) | 29.00 (92) | 31.00 (130) | 33.00 (168) |
|     | A-RLF-10  | 18.00 (0) | 19.60 (0) | 21.40 (0) | 23.20 (0) | 25.00 (0) | 26.60 (0) | 28.00 (1) | 30.00 (1) | 31.80 (1) |
|     | 10%-RLF   | 17.80 (1) | 19.00 (1) | 21.00 (1) | 22.80 (2) | 24.00 (3) | 26.00 (5) | 27.20 (8) | 29.20 (10) | 31.00 (13) |
|     | n-RLF     | 17.00 (3) | 19.00 (6) | 21.00 (9) | 22.60 (15) | 24.00 (23) | 25.60 (31) | 27.00 (49) | 29.00 (84) | 30.80 (93) |
| 0.5 | RLF       | 79.80 (0) | 90.40 (0) | 99.00 (0) | 107.80 (1) | 117.60 (1) | 126.60 (1) | 135.00 (2) | 144.20 (1) | 152.80 (2) |
|     | 10-RLF    | 79.20 (1) | 88.40 (1) | 97.60 (3) | 107.00 (3) | 116.80 (6) | 125.40 (8) | 134.20 (11) | 142.80 (11) | 151.80 (15) |
|     | 10%-RLF   | 78.80 (6) | 88.20 (12) | 97.40 (23) | 106.60 (35) | 115.40 (68) | 124.40 (95) | 133.60 (142) | 142.00 (153) | 150.80 (204) |
|     | n-RLF     | 78.20 (62) | 87.80 (118) | 96.80 (260) | 105.80 (402) | 115.00 (702) | 123.60 (922) | 133.00 (1432) | 141.40 (1554) | 150.00 (1975) |
|     | A-RLF-10  | 72.20 (1) | 80.60 (1) | 89.40 (2) | 97.20 (3) | 106.20 (4) | 114.40 (5) | 121.80 (8) | 130.40 (14) | 137.60 (17) |
|     | 10%-RLF   | 69.00 (5) | 77.40 (9) | 85.00 (16) | 92.00 (33) | 100.00 (39) | 107.40 (69) | 114.80 (106) | 122.20 (142) | 129.20 (280) |
|     | n-RLF     | 67.00 (37) | 75.00 (66) | 82.40 (127) | 89.60 (258) | 97.00 (368) | 104.00 (543) | 111.40 (788) | 118.80 (1261) | 126.00 (1420) |
| 0.9 | RLF       | 207.00 (0) | 235.80 (1) | 259.40 (1) | 286.40 (2) | 308.40 (2) | 335.80 (3) | 358.60 (5) | 382.60 (5) | 407.40 (6) |
|     | 10-RLF    | 205.20 (4) | 230.80 (7) | 256.40 (12) | 280.80 (20) | 306.00 (25) | 331.00 (33) | 354.00 (45) | 379.20 (51) | 403.80 (60) |
|     | 10%-RLF   | 204.20 (28) | 230.00 (52) | 255.00 (103) | 279.60 (193) | 303.60 (259) | 328.80 (406) | 352.40 (603) | 376.40 (710) | 400.60 (906) |
|     | n-RLF     | 202.60 (280) | 229.20 (542) | 253.00 (961) | 277.00 (2056) | 302.60 (2837) | 327.60 (4074) | 351.00 (5757) | 375.00 (7667) | 398.60 (9014) |
|     | A-RLF-10  | 188.60 (4) | 210.80 (13) | 233.20 (22) | 255.80 (33) | 280.60 (40) | 301.60 (52) | 325.80 (63) | 346.80 (50) | 371.00 (63) |
|     | 10%-RLF   | 182.00 (22) | 203.20 (43) | 225.40 (90) | 245.60 (138) | 267.20 (254) | 287.00 (315) | 306.20 (554) | 325.40 (820) | 343.80 (1247) |
|     | n-RLF     | 177.20 (178) | 197.80 (318) | 217.80 (605) | 236.60 (1126) | 255.20 (1681) | 275.20 (2315) | 294.20 (3740) | 314.60 (4280) | 332.00 (5999) |

Table 1: Comparison of the A-RLF-β and β-RLF algorithms on random graphs.
Figure 4: Evolution of the computing time for random graphs.
Figure 5: Color gains of the $A$-RLF-$\beta$ and $\beta$-RLF algorithms with respect to RLF on random graphs.
| Graph    | n    | m    | k         | A-RLF-1 | B-RLF-1 | A-RLF-10 | B-RLF-10 | A-RLF-10% | B-RLF-10% | A-RLF-n | B-RLF-n |
|----------|------|------|-----------|---------|---------|----------|----------|-----------|-----------|---------|---------|
| DSJC125.1 | 125  | 736  | 5         | 6       | 6       | 6         | 6        | 6         | 6         | 6       | 6       |
| DSJC125.5 | 125  | 3,891| 17        | 5       | 20      | 20        | 5       | 20        | 20        | 5       | 20      |
| DSJC125.9 | 125  | 6,961| 44        | 48      | 49      | 49        | 45      | 46        | 47        | 46      | 47      |
| DSJC250.1 | 250  | 3,218| 8         | 9       | 9       | 9         | 9       | 9         | 9         | 9       | 9       |
| DSJC250.5 | 250  | 15,668| 28       | 34      | 34      | 35        | 36      | 36        | 36        | 36      | 36      |
| DSJC250.9 | 250  | 27,897| 72       | 83      | 83      | 85        | 85      | 85        | 85        | 85      | 85      |
| DSJC500.1 | 500  | 12,458| 12       | 14      | 14      | 15        | 15      | 15        | 15        | 15      | 15      |
| DSJC500.5 | 500  | 62,624| 48       | 59      | 59      | 60        | 61      | 61        | 62        | 62      | 62      |
| DSJC500.9 | 500  | 112,437| 126     | 151     | 152     | 155       | 155     | 155       | 155       | 155     | 155     |
| DSJR500.1 | 500  | 3,555 | 12       | 12      | 12      | 12        | 12      | 12        | 12        | 12      | 12      |
| DSJR500.5 | 500  | 58,862| 122      | 130     | 131     | 133       | 133     | 133       | 133       | 133     | 133     |
| DSJR1000.1 | 1,000| 49,629| 24       | 24      | 24      | 24        | 24      | 24        | 24        | 24      | 24      |
| DSJR1000.5 | 1,000| 249,826| 83      | 106     | 107     | 108       | 109     | 109       | 109       | 109     | 109     |
| latin_square | 900 | 307,350| 97 | 114 | 116 | 118 | 117 | 123 | 126 | 110 | 111 | 114 | 109 | 111 | 114 | 109 | 109 | 111 | 107 | 108 | 111 |
| le450_15_a | 450  | 8,168 | 16       | 16      | 16      | 16        | 16      | 16        | 16        | 16      | 16      |
| le450_15_b | 450  | 8,169 | 16       | 16      | 16      | 16        | 16      | 16        | 16        | 16      | 16      |
| le450_15_c | 450  | 16,680| 15       | 23      | 23      | 24        | 23      | 23        | 23        | 23      | 23      |
| le450_15_d | 450  | 16,750| 15       | 23      | 23      | 23        | 22      | 22        | 22        | 22      | 22      |
| le450_25_a | 450  | 28,620| 25       | 25      | 25      | 25        | 25      | 25        | 25        | 25      | 25      |
| le450_25_b | 450  | 8,263 | 25       | 25      | 25      | 25        | 25      | 25        | 25        | 25      | 25      |
| le450_25_c | 450  | 17,343| 25       | 27      | 27      | 28        | 28      | 28        | 28        | 28      | 28      |
| le450_25_d | 450  | 17,425| 25       | 28      | 28      | 29        | 29      | 29        | 29        | 29      | 29      |
| le450_5_a  | 450  | 5,714 | 5        | 7       | 7       | 8         | 7       | 7         | 7         | 7       | 7       |
| le450_5_b  | 450  | 5,734 | 5        | 7       | 7       | 7         | 5       | 5         | 5         | 5       | 5       |
| le450_5_c  | 450  | 9,803 | 5        | 5       | 5       | 6         | 6       | 6         | 6         | 6       | 6       |
| le450_5_d  | 450  | 9,757 | 5        | 5       | 5       | 7         | 7       | 7         | 7         | 7       | 7       |
| school1   | 385  | 19,095| 14       | 26      | 27      | 28        | 29      | 29        | 29        | 29      | 29      |
| school1_sh | 352  | 14,612| 14       | 22      | 23      | 24        | 24      | 24        | 24        | 24      | 24      |
| queen6_6  | 36   | 290   | 7        | 8       | 8       | 8         | 8       | 8         | 8         | 8       | 8       |
| queen7_7  | 49   | 576   | 7        | 9       | 9       | 10        | 9       | 9         | 10        | 9       | 10      |
| queen8_8  | 64   | 728   | 9        | 10      | 10      | 11        | 10      | 10        | 11        | 10      | 11      |
| queen9_9  | 81   | 2,112 | 10       | 11      | 11      | 12        | 11      | 10        | 11        | 10      | 11      |

...continued on next page
| Graph          | n   | m   | k   | A-RLF-1 min av. | B-RLF-1 min av. | A-RLF-10 min av. | B-RLF-10 min av. | A-RLF-10% min av. | B-RLF-10% min av. | A-RLF-α min av. | B-RLF-α min av. |
|---------------|-----|-----|-----|-----------------|-----------------|------------------|------------------|-------------------|-----------------|----------------|----------------|
| queen10_10    | 100 | 100 | 10  | 12.6            | 13              | 12.2             | 13               | 11.9              | 13              | 12.1           | 13              |
| queen11_11    | 121 | 11  | 11  | 13.9            | 14              | 13.9             | 14               | 13.1              | 14              | 12.7           | 13              |
| queen12_12    | 144 | 12  | 12  | 14.9            | 15              | 15.1             | 15               | 14.4              | 15              | 14.4           | 15              |
| queen13_13    | 169 | 13  | 13  | 15.9            | 16              | 15.8             | 16               | 15.1              | 16              | 15.2           | 16              |
| queen14_14    | 196 | 14  | 14  | 17.2            | 18              | 17.8             | 18               | 16.5              | 18              | 16.7           | 18              |
| queen15_15    | 225 | 15  | 15  | 18.2            | 16              | 17.1             | 18               | 17.4              | 18              | 17.1           | 18              |
| queen16_16    | 256 | 16  | 16  | 19.5            | 20              | 19.4             | 20               | 19.0              | 20              | 18.4           | 19              |
| abb313        | 1,557 | 9  | 9   | 11              | 11              | 11               | 11               | 12.1              | 11              | 11             | 11              |
| ash331        | 662 | 4   | 4   | 4               | 4               | 4.8              | 5                | 4.3               | 5               | 4.2            | 5               |
| ash608        | 1,216 | 4  | 4   | 5               | 5               | 5.2              | 6                | 4.2               | 5               | 5.1            | 6               |
| ash968        | 1,916 | 4  | 4   | 5               | 5               | 5.2              | 6                | 5                | 5               | 5.3            | 6               |
| will199       | 701 | 7   | 7   | 7               | 7               | 7.1              | 7                | 7.1               | 7               | 7.7            | 7               |
| wap01         | 2,368 | 42 | 42  | 46              | 47              | 45.2             | 46               | 45.2              | 46              | 45.2           | 46              |
| wap02         | 2,464 | 41 | 41  | 44              | 44              | 43.8             | 44               | 44.2              | 44              | 44.2           | 44              |
| wap03         | 4,730 | 44 | 44  | 50              | 51              | 50.8             | 52               | 48.9              | 51              | 49.3           | 51              |
| wap04         | 5,231 | 42 | 42  | 46              | 46              | 46.7             | 47               | 45.9              | 47              | 46.5           | 47              |
| wap05         | 905 | 50  | 50  | 50              | 51              | 50.8             | 52               | 48.9              | 51              | 49.3           | 51              |
| wap06         | 947 | 40  | 40  | 44              | 44              | 43.7             | 45               | 42.2              | 43             | 42.3           | 43             |
| wap07         | 1,809 | 42 | 42  | 45              | 48              | 45.8             | 48               | 44.9              | 48             | 45.5           | 48             |
| wap08         | 1,870 | 42 | 42  | 45              | 48              | 45.7             | 46               | 44.8              | 48             | 44.9           | 48             |
| qg.order60    | 3,600 | 60 | 60  | 60              | 60              | 60.7             | 61               | 60.6              | 60             | 60          | 60             |
| qg.order100   | 10,000 | 100 | 100 | 100             | 100             | 100.1            | 101              | 100.2             | 101            | 100.4          | 101            |
| flat300_20    | 300 | 20  | 20  | 36              | 38              | 36.8             | 38               | 32.1              | 38             | 34.4           | 38             |
| flat300_20    | 300 | 20  | 20  | 38              | 38              | 38.6             | 39               | 35.4              | 39             | 35.1           | 39             |
| flat300_20    | 300 | 20  | 20  | 37              | 37              | 37.9             | 39               | 35.7              | 39             | 35.3           | 39             |
| flat1000_50   | 1,000 | 50 | 50  | 104             | 106             | 105.4            | 106              | 95                | 106            | 90.5           | 106            |
| flat1000_60   | 1,000 | 60 | 60  | 105             | 107             | 105.7            | 108              | 95                | 106            | 90.6           | 106            |
| flat1000_76   | 1,000 | 76 | 76  | 104             | 106             | 105.2            | 106              | 96                | 106            | 91.2           | 106            |
| total         | 2,136 | 2,682 | 2,682 | 2,649         | 2,663         | 2,682            | 2,649         | 2,445            | 2,515         | 2,487          | 2,509         |

Table 2: Results of the α-RLF-β algorithms on challenging graphs
| Graph   | n   | m   | RLF | A-RLF-10 | A-RLF-10% | A-RLF-n |
|---------|-----|-----|-----|----------|-----------|---------|
| DSJC125.1 | 125 | 736 | 0   | 0        | 0         | 0       |
| DSJC125.5 | 125 | 3,891 | 0   | 0        | 0         | 0       |
| DSJC125.9 | 125 | 6,961 | 0   | 0        | 0         | 0       |
| DSJC250.1 | 250 | 3,218 | 0   | 0        | 0         | 0       |
| DSJC250.5 | 250 | 15,668 | 0   | 0        | 0         | 0       |
| DSJC250.9 | 250 | 37,897 | 0   | 1        | 2         | 0       |
| DSJC500.1 | 500 | 12,458 | 0   | 0        | 0         | 1       |
| DSJC500.5 | 500 | 62,624 | 0   | 1        | 7         | 0       |
| DSJC500.9 | 500 | 112,437 | 0   | 4        | 27        | 0       |
| DSJR500.1 | 500 | 3,555 | 0   | 0        | 0         | 1       |
| DSJR500.5 | 500 | 121,275 | 0   | 1        | 2         | 11      |
| DSJR500.9 | 500 | 58,862 | 0   | 1        | 13        | 13      |
| DSJC1000.1 | 1,000 | 49,629 | 0 | 0        | 1         | 16      |
| DSJC1000.5 | 1,000 | 249,826 | 0 | 3        | 29        | 283     |
| DSJC1000.9 | 1,000 | 449,449 | 2 | 15       | 158       | 875     |
| latin_square | 900 | 307,35 | 1   | 5        | 25        | 187     |
| le450.1 | 450 | 8,168 | 0   | 0        | 0         | 1       |
| le450.5 | 450 | 8,169 | 0   | 0        | 0         | 1       |
| le450.9 | 450 | 16,680 | 0   | 0        | 1         | 1       |
| le450.1_c | 450 | 8,260 | 0   | 0        | 0         | 1       |
| le450.5_c | 450 | 8,263 | 0   | 0        | 0         | 1       |
| le450.9_c | 450 | 17,443 | 0   | 0        | 2         | 0       |
| le450.1_d | 450 | 17,425 | 0   | 0        | 1         | 1       |
| le450.5_d | 450 | 5,714 | 0   | 0        | 0         | 1       |
| le450.9_d | 450 | 9,803 | 0   | 0        | 0         | 0       |
| le450.1_d | 450 | 9,757 | 0   | 0        | 0         | 1       |
| school1 | 285 | 19,095 | 0   | 0        | 1         | 1       |
| school1_nsh | 352 | 14,612 | 0   | 0        | 0         | 1       |
| queen6_6 | 36 | 290 | 0   | 0        | 0         | 0       |
| queen7_7 | 39 | 476 | 0   | 0        | 0         | 0       |
| queen8_12 | 96 | 1,368 | 0   | 0        | 0         | 0       |
| queen8_8 | 64 | 728 | 0   | 0        | 0         | 0       |
| queen9_9 | 81 | 2,112 | 0   | 0        | 0         | 0       |
| queen10_10 | 100 | 2,940 | 0   | 0        | 0         | 0       |
| queen11_11 | 121 | 3,960 | 0   | 0        | 0         | 0       |
| queen12_12 | 144 | 5,192 | 0   | 0        | 0         | 0       |
| queen13_13 | 169 | 6,656 | 0   | 0        | 0         | 0       |
| queen14_14 | 196 | 8,372 | 0   | 0        | 0         | 0       |
| queen15_15 | 225 | 10,360 | 0 | 0        | 0         | 0       |
| queen16_16 | 256 | 12,640 | 0 | 0        | 0         | 1       |
| ab331 | 1,557 | 53,356 | 1 | 4        | 2         | 27      |
| flat300 | 20 | 300 | 21,375 | 0 | 0 | 0 |
| flat300 | 26 | 300 | 21,633 | 0 | 0 | 0 |
| flat300 | 28 | 300 | 21,695 | 0 | 0 | 0 |
| flat1000 | 50 | 1,000 | 245,000 | 1 | 4 | 8 |
| flat1000 | 60 | 1,000 | 245,830 | 1 | 4 | 8 |
| flat1000 | 76 | 1,000 | 246,708 | 1 | 5 | 8 |
| qg.order60 | 3,600 | 212,400 | 16 | 130 | 1,7330 | 91064 |
| qg.order100 | 10,000 | 990,000 | 16 | 130 | 1,7330 | 91064 |
| flat300_20 | 300 | 21,375 | 0 | 0 | 0 | 1 |
| flat300_26 | 300 | 21,633 | 0 | 0 | 0 | 1 |
| flat300_28 | 300 | 21,695 | 0 | 0 | 0 | 1 |
| flat1000_50 | 1,000 | 245,000 | 1 | 4 | 34 | 178 |
| flat1000_60 | 1,000 | 245,830 | 1 | 4 | 34 | 178 |
| flat1000_76 | 1,000 | 246,708 | 1 | 5 | 34 | 179 |

Table 3: Average computing times (in seconds) of the A-RLF-\( \beta \) algorithms on DIMACS benchmark instances.
Figure 6: Comparisons of the $\alpha$-RLF-$\beta$ algorithms on DIMACS benchmark instances.
| Graph   | n   | m   | k   | DS | min | av. | max |
|---------|-----|-----|-----|----|-----|-----|-----|
| DSJC125.1 | 125 | 736 | 5   | 6  | 6   | 6   | 6   |
| DSJC125.5 | 125 | 3,891 | 17  | 6  | 6   | 6   | 6   |
| DSJC125.9 | 125 | 6,961 | 44  | 50 | 48  | 48  | 49  |
| DSJC250.1 | 250 | 3,218 | 8   | 10 | 9   | 9   | 10  |
| DSJC250.5 | 250 | 15,688 | 28  | 38 | 34  | 34  | 35  |
| DSJC250.9 | 250 | 27,897 | 72  | 91 | 83  | 83  | 86  |
| DSJC500.1 | 500 | 12,458 | 12  | 16 | 14  | 14  | 15  |
| DSJC500.5 | 500 | 62,624 | 48  | 67 | 59  | 59  | 60  |
| DSJC500.9 | 500 | 112,457 | 126 | 161 | 151 | 152 | 155 |
| DSJR500.1 | 500 | 3,555 | 12  | 12 | 12  | 12  | 12  |
| DSJR500.5 | 500 | 121,275 | 84  | 87 | 89  | 90  | 92  |
| DSJR500.9 | 500 | 58,867 | 120 | 130 | 130 | 131 | 134 |
| DSJC1000.1 | 1,000 | 49,629 | 20  | 26 | 24  | 24  | 24  |
| DSJC1000.5 | 1,000 | 249,826 | 83  | 114 | 106 | 107 | 108 |
| DSJC1000.9 | 1,000 | 49,449 | 222 | 297 | 275 | 279 | 283 |
| latin_square | 900 | 307,350 | 97  | 126 | 122 | 124 | 126 |
| le450_15_a | 450 | 8,168 | 15  | 16 | 16  | 16  | 16  |
| le450_15_b | 450 | 8,169 | 15  | 16 | 16  | 16  | 16  |
| le450_15_c | 450 | 16,680 | 15  | 24 | 23  | 23  | 23  |
| le450_15_d | 450 | 16,750 | 15  | 24 | 23  | 23  | 23  |
| le450_25_a | 450 | 8,260 | 25  | 25 | 25  | 25  | 25  |
| le450_25_b | 450 | 8,263 | 25  | 25 | 25  | 25  | 25  |
| le450_25_c | 450 | 17,343 | 25  | 29 | 27  | 29  | 29  |
| le450_25_d | 450 | 17,425 | 25  | 28 | 28  | 28  | 29  |
| le500_5_a | 500 | 5,714 | 5   | 10 | 7   | 7   | 7   |
| le500_5_b | 500 | 5,734 | 5   | 9  | 7   | 7   | 7   |
| le500_5_c | 500 | 9,803 | 5   | 6  | 5   | 5   | 5   |
| le500_5_d | 500 | 9,757 | 5   | 11 | 5   | 5   | 5   |
| queen6_6 | 36 | 290 | 7   | 9  | 8   | 8   | 8   |
| queen7_7 | 49 | 476 | 7   | 10 | 9   | 9   | 9   |
| queen8_8 | 96 | 1,368 | 12  | 13 | 13  | 13  | 13  |
| queen9_9 | 64 | 728 | 9   | 12 | 10  | 11  | 11  |
| queen10_10 | 81 | 2,112 | 10  | 14 | 11  | 11  | 11  |
| queen11_11 | 121 | 3,960 | 11  | 15 | 13  | 13  | 13  |
| queen12_12 | 144 | 5,192 | 12  | 15 | 14  | 14  | 14  |
| queen13_13 | 169 | 6,656 | 13  | 17 | 15  | 15  | 15  |
| queen14_14 | 196 | 8,372 | 14  | 18 | 17  | 17  | 18  |
| queen15_15 | 225 | 10,360 | 15  | 19 | 17  | 17  | 17  |
| queen16_16 | 254 | 12,641 | 16  | 21 | 19  | 20  | 20  |
| ab313 | 1,557 | 53,356 | 9  | 11  | 11  | 11  | 11  |
| ash331 | 662 | 4,185 | 4  | 5  | 4   | 4   | 4   |
| ash680 | 1,216 | 7,844 | 4  | 5  | 5   | 5   | 5   |
| ash958 | 1,916 | 12,386 | 5  | 5  | 5   | 5   | 5   |
| will199 | 701 | 6,772 | 7  | 7  | 7   | 7   | 7   |
| wap01 | 2,368 | 110,871 | 42 | 46 | 45  | 45  | 45  |
| wap02 | 2,461 | 111,742 | 42 | 44 | 44  | 44  | 44  |
| wap03 | 4,730 | 286,722 | 44 | 54 | 50  | 50  | 50  |
| wap04 | 5,251 | 294,902 | 42 | 48 | 46  | 46  | 46  |
| wap05 | 905 | 43,081 | 10 | 50 | 50  | 50  | 50  |
| wap06 | 947 | 45,571 | 40 | 46 | 44  | 44  | 44  |
| wap07 | 1,809 | 103,362 | 44 | 46 | 45  | 45  | 45  |
| wap08 | 1,870 | 104,176 | 42 | 45 | 45  | 45  | 45  |
| sqs150 | 3,600 | 212,400 | 60 | 62 | 60  | 60  | 60  |
| total | 2,136 | 2,735 | 2,576 | 2,606 | 2,466 | 2,466 | 2,360 |

Table 4: Comparison of AB-RLF-β with DSATUR and Short.Tabu.
Figure 7: Comparisons of the $AB$-RLF-$\beta$ algorithms on DIMACS benchmark instances.

Figure 8: Some TAPDs and ARPDs for the DIMACS benchmark instances.