Informed Reactive Tabu Search for Graph Coloring

Daniel Cosmin Porumbel [1)], Jin-Kao Hao [2)] and Pascale Kuntz [3)]

1) Univ. Lille-Nord de France, UArtois, LGI2A, Technoparc Futura, 62400 Béthune, France

2) Université d'Angers, LERIA, 2 Bd Lavoisier, 49045 Angers, France

3) Université de Nantes, LINA, rue Christian Pauc, 44306 Nantes, France

April 17, 2013

Abstract

Tabu Search (TS) has always been a very popular algorithm for graph coloring, both as a stand-alone optimizer as well as a subroutine inside population-based hybrid methods. We present two TS extensions that could allow previous TS algorithms to improve their behavior at almost no additional computational cost. First, we integrate two new evaluation functions which employ supplementary (structural or dynamic) information in addition to the conventional objective function (the number of edges with both ends of the same color). These new evaluation functions allow the search process to differentiate configurations not distinguished by the conventional evaluation function. The second extension concerns a reactive mechanism for improving the tabu list management. Theoretical arguments show that this reactive component completely eliminates the risk of getting blocked looping on plateaus. Numerical experiments show that the two proposed TS extensions can be very useful inside a stand-alone TS optimizer, as well as inside TS subroutines of state-of-the-art hybrid methods.

1 Introduction

The graph (vertex) coloring problem (*Coloring*) requires finding the chromatic number of a graph, i.e. the minimum number of colors needed to construct a coloring without conflicts (with no edge having both ends of the same color). The graph k-coloring problem (k-*Coloring*), the decision version of *Coloring*, is to decide whether or not there is a coloring with no conflicts using k colors. *Coloring* is one of the 21 fundamental computer science problems whose NP-completeness was proved in the early 1970s [25]. Except a few special cases (trees, bipartite graphs, etc.), no polynomial algorithm can solve or approximate *Coloring* within a constant factor unless $P = NP$. In practice, *Coloring* and k-*Coloring* constitute a convenient and powerful model for formulating numerous applications, as for example: frequency assignment in cellular networks, timetabling, register allocation in compilers, scheduling problems, and many others—see references in [28, 3, 33].

Since the early 1970s, numerous coloring algorithms have been developed and important progress has been made. Exact algorithms do not usually deal with graphs of more than 100 vertices [28], and so, heuristic algorithms are often employed for coloring larger graphs. Existing (meta-)heuristic coloring algorithms belong to five main solving approaches: (i) sequential

constructive heuristics (e.g. some of the best are RLX and XRLF [23]), (ii) local search (tabu search [20, 12, 3], simulated annealing [5, 23], iterative local search [30, 7, 6], variable neighborhood search [1, 19], local search with forward checking [35], distance and position guided search [33], etc.), (iii) population-based hybrid algorithms [29, 13, 10, 15, 17, 27, 26, 32, 34], (iv) neural networks approaches [22, 36], (v) swarm intelligence algorithms [8, 11, 4, 31], (vi) independent set extraction [38], and (vii) distributed and hybrid quantum annealing [37].

Among these coloring approaches, one of the most effective and most popular choices is represented by Tabu Search or by algorithms incorporating Tabu Search. Many of the best-known results available today have been obtained with hybrid methods that employ Tabu Search as a local optimizer [15, 17, 27, 26, 32]. One observes that classical TS coloring algorithms are quite simple and "lightweight": they do not make use of "heavy machineries" such as linear programming relaxations, distributed computing, ant swarms, evolutionary computing, niching or fitness sharing techniques, etc. One wonders then whether a TS coloring algorithm can be further boosted by improving some of its key ingredients, while keeping it reasonably simple.

This paper introduces two TS extensions: well-informed evaluation functions and a simple-but-effective reactive tabu list. The classical objective function (the number of conflicts) is essential because it defines the landscape and guides the search process (together with the neighborhood). However it makes no distinction among numerous k-colorings with the same number of conflicts but with a different potential of leading to a solution. As such, we investigate two new evaluation functions that use supplementary information: (i) the degrees of the conflicting vertices, and (ii) the frequencies of color changes during a first stage of the execution.

Regarding the tabu list tunning, we observed that the existing well-established techniques can still be improved, as looping problems can still occur at times. For this reason, we introduce a new reactive technique[2] that allows the search process to avoid getting blocked indefinitely looping on plateaus. The resulting algorithm (hereafter referred to as IRTS—Informed Reactive Tabu Search) reaches good results as a stand alone algorithm, and it can also be very useful as a local optimizer in state-of-the-art hybrid methods.

The rest of the paper is organized as follows. In Section 2, we give preliminary definitions and we present in detail the TS variant we use. Sections 3 and 4 are devoted to the new evaluation functions and to the reactive tabu list, respectively. In Section 5, we perform an experimental study of the proposed ideas, followed by conclusions in the last section.

2 Generic Tabu Search for k-*Coloring*

2.1 Preliminary Definitions

Let $G = (V, E)$ be a graph with V and E being respectively the vertex and edge sets. We recall the following definitions.

Definition 1. *(*k-Coloring *and* Coloring*) Given a graph* $G = (V, E)$ *and a positive integer* k*, the graph* k-coloring problem *requires deciding if* G *is* k*-colorable, i.e. if there is a function* $C : V \rightarrow \{1, 2, \cdots, k\}$ *such that* $C(v) \neq C(v') \; \forall \{v, v'\} \in E$*. The* graph coloring problem *requires finding the chromatic number of* G*, i.e. the smallest* k *such that* G *is* k*-colorable.*

It is clear that k-*Coloring* and *Coloring* are two tightly related problems. A possible approach to deal with *Coloring* consists of solving a series of increasingly-difficult k-*Coloring*

problems: start with a very large initial k and iteratively decrement k after solving each k-*Coloring* instance. The k-coloring problem becomes more and more difficult until it can no longer be solved by the algorithm. The best k for which k-*Coloring* is solved constitutes an upper bound to the chromatic number of G.

In the following, we represent a coloring function $C : V \rightarrow \{1, 2, \cdots, k\}$ as an array $C = [C(1), C(2), \ldots, C(|V|)]$. We say that C is a k-coloring for G, or a configuration.

Definition 2. *(Conflicts) The conflicts (the conflicting edges) of a k-coloring C are given by $\overline{\mathscr{E}}(C) = \{\{v, v'\} \in E : C(v) = C(v')\}$. The set of conflicting vertices is: $\overline{\mathscr{V}}(C) = \{v \in V : \exists v' \in V$ s.t. $\{v, v'\} \in \overline{\mathscr{E}}(C)\}$.*

The *conflict number* of C is denoted by $|\overline{\mathscr{E}}(C)|$, or simply by $|\overline{\mathscr{E}}|$, i.e. the argument C is omitted when no confusion is possible. A coloring C is conflict-free if and only if $|\overline{\mathscr{E}}| = 0$.

Definition 3. *(Conflicting degree of a vertex) Let v be a vertex, d_v its degree, and C a configuration. We define $\overline{\mathscr{V}}_v = \{v' \in V | \{v, v'\} \in \overline{\mathscr{E}}(C)\}$. We call $\frac{|\overline{\mathscr{V}}_v|}{d_v}$ the conflicting degree of v under C.*

It is easy to see that $0 \leq |\overline{\mathscr{V}}_v| \leq d_v \ \forall v \in V$. The minimal value $|\overline{\mathscr{V}}_v| = 0$ is reached for non-conflicting vertices; $|\overline{\mathscr{V}}_v| = d_v$ indicates that vertex v is conflicting with all its neighbors. Moreover, the following relation holds for any coloring: $2|\overline{\mathscr{E}}| = \sum_{v \in V} |\overline{\mathscr{V}}_v|$.

Finally, notice that we solve k-*Coloring* as an optimization problem. Given a k-*Coloring* instance (G, k), the optimization problem is determined by the couple as (S, f_c), where: (i) S is the search space composed of all the $|V|^k$ possible k-colorings; (ii) $f_c : S \rightarrow I\!N$ is the objective function counting the number of conflicts using formula below.

$$f_c(C) = \left|\overline{\mathscr{E}}(C)\right|, \forall C \in S \tag{1}$$

Accordingly, any configuration $C^* \in S$ such that $f(C^*) = |\overline{\mathscr{E}}(C^*)| = 0$ corresponds to a conflict-free or legal k-coloring, a *solution* of the given k-*Coloring* instance.

2.2 Key Components of the TS Coloring Procedure

Tabu Search [18] was first applied to k-*Coloring* in 1987, leading to the well-known Tabucol algorithm [20]. Since then, the approach has inspired several other TS variants and important progress has been made. We here discuss two of the most important improvements, as they are tightly related to our study. First, [13] introduced an efficient incremental evaluation technique for streamlining the calculations. Secondly, more elaborate tabu-list management techniques have been developed [10, 15, 9]. A historical presentation of Tabucol as well as a comprehensive analysis of some of the best local search algorithms can be found in [16]. Experimental comparisons of TS with other coloring heuristics are also available in the literature [21, 6]. Finally, one should be aware that other TS variants *do* exist (e.g. the PartialCol algorithm [3] using partial colorings), but these variants are not directly related to our approach.

2.2.1 Fast neighborhood evaluation via calculation streamlining

Given a k-coloring C, a neighboring k-coloring C' can be obtained by simply changing the color $C(v)$ of a *conflicting* vertex v into a new color $C'(v)$. We denote by $\langle v, C'(v) \rangle$ the

transition from C to C'. By focusing on conflicting vertices, this neighborhood helps the search process to concentrate on influential moves and to avoid irrelevant ones, i.e. changing the color of a non-conflicting vertex would not directly improve the objective function f_c. Furthermore, this neighborhood $N(C)$ has a cardinal of $|N(C)| = (k-1)|\overline{\mathscr{V}}|$, which is considerably lower than the cardinal of a neighborhood in which *any* vertex could change its color (i.e. $(k-1)|V|$).

To rapidly choose the best next coloring from $N(C)$, we use a $|V| \times k$ table Γ such that $\Gamma_{v,C'(v)}$ indicates the number of conflicts of vertex v if v would receive color $C'(v)$. As such, $\Gamma_{v,C'(v)} - \Gamma_{v,C(v)}$ represents the f_c variation that would be induced by changing the color of v from $C(v)$ to $C'(v)$. Since we only deal with conflicting vertices, the best move is searched by going through all elements $\Gamma_{v,C'(v)}$ with $v \in \overline{\mathscr{V}}$ (i.e. $(k-1) \cdot |\overline{\mathscr{V}}|$ elements). After performing a move, Γ can be updated in $O(|V|)$ time (because only columns $C(v)$ and $C(v')$ might require updating in Γ). This incremental evaluation technique proves to be essential for a practically viable examination of the complete neighborhood.

2.2.2 Tabu list management

A tabu list is commonly regarded as a first-in-first-out structure recording recent configurations or recent moves. In our case, it is more convenient to implement it using a $|V| \times k$ table T in which each element $T_{v,C'(v)}$corresponds to a possible move $\langle v, C'(v)\rangle$. Each time a move $\langle v, C'(v)\rangle$ is performed, v receives the new color $C'(v)$ and the lost color of v becomes forbidden (tabu) for the next tl (tabu tenure) iterations. In practice, each element of T records the current iteration number plus the tabu tenure tl. Consequently, in order to check out whether or not a move $\langle v, C'(v)\rangle$ is tabu, it is enough to compare $T_{v,C'(v)}$ to the current iteration counter.

A well-established tenure [15, 10] in graph coloring is: $tl = \alpha \cdot |\overline{\mathscr{E}}| + random(1, A)$. The term $random(1, A)$ makes reference to a routine returning a random integer between 1 and A; it imposes a quality independent tenure to all moves. The value of $\alpha \cdot |\overline{\mathscr{E}}|$ depends on the current coloring: this term aims at penalizing moves associated to lower quality configurations. Our new reactive component for tunning the tabu list is introduced in Section 4.

Aspiration criterion Notice that it would be quite unreasonably to forbid a move that would lead to a new best coloring. In this case, we make use of an aspiration criterion that removes the tabu status of the corresponding move.

2.3 TS Formal Specification

Algorithm 1 presents the general TS k-coloring procedure, including the components presented above. For a given k-coloring instance (G, k), our TS variant starts out with an initial (random) k-coloring C. The main steps of an iteration are: (A) use Γ to pick up an acceptable move (i.e. non-tabu, or tabu but satisfying the aspiration criterion) that minimizes the number of conflicts, (B) set the current color of v as tabu, (C) execute the move and (D) update Γ accordingly. The process stops when a legal coloring is found or when a time (or iteration) limit is reached. The new evaluation functions will be integrated in this algorithm at Step 5.A, see Section 3.3 below. The reactive tabu list will be integrated at Step 5.B.

Algorithm 1 Template of the classical Tabu Search

Input: graph G, integer k;
Return: $f_c(C^*)$ the best conflict number ever found;
Variables: C (current coloring), C^* (best coloring found so far), T (tabu Table), tl (tabu tenure), Γ (table of move values), iter_counter (the current iteration);
Begin

1. $T = 0_{|V|\times k}$ /*no move is tabu when starting*/
2. $C =$ a random initial k-coloring
3. $C^* = C$
4. Initialize Γ /*see §2.2.1*/
5. **while** ($f_c(C) > 0$ **and** time/iteration limit not reached)
   A. Use Γ to select a best neighbor C' in $O(k|\overline{\mathscr{V}}|)$: if more moves lead to a best conflict number, C' is selected using a probabilistic choice guided by the evaluation function (see Section 3.3)
   B. $T_{v,C(v)} =$ iter_counter $+ tl$ /*mark $\langle v, C(v)\rangle$ tabu, see §2.2.2 and §4*/
   C. $C(v) = C'(v)$
   D. Update Γ /*in $O(|V|)$ time, see §2.2.1*/
   E. **if** $(f_c(C) < f_c(C^*))$ **then** $C^* = C$ /*better coloring found*/
6. **return** $f_c(C^*)$

End

3 New Evaluation Functions

The function f_c (Equation (1)) is commonly used as the evaluation and objective function. Since f_c only counts the number of conflicts, it has an inherent inconvenience: it makes no distinction between all configurations with the same conflict number. Indeed, these configurations are equivalent for f_c even if they may have different potential for leading to a legal coloring.

To overcome this difficulty, we propose to enrich f_c with additional information. We introduce a heuristic function $h : S \to [0, 1)$ and combine it with f_c by the following simple linear form, leading to a new evaluation function $\widetilde{f}$:

$$\widetilde{f}\,(C) = f_c\,(C) - h\,(C) \tag{2}$$

where h is a heuristic that can discriminate configurations equivalent in terms of f_c, as exemplified in Sections 3.1 and 3.2 below. By considering $h(C) < 1$ for any coloring C , we obtain $\lceil \widetilde{f}(C) \rceil = f_c(C)$.

3.1 A Degree-Based Evaluation Function

Let us first consider the example from Figure 1 showing two 3-colorings C_1 and C_2 of a small graph. The edges in conflict are respectively $\{a, b\}$ for C_1 and $\{a, c\}$ for C_2. Consequently, $|\overline{\mathscr{E}}\,(C_1)\,| = |\overline{\mathscr{E}}\,(C_2)\,| = 1$ and the two configurations are thus equivalent for f_c. However, it is easier to solve the 3-coloring problem from C_1 than from C_2.

Indeed, since the degree of b is small, one can assign to b a color not used by its neighbors (*i.e.* black or white) to solve the $\{a, b\}$ conflict on C_1. That can be done in one step and it does not introduce any other conflicts. Solving the $\{a, c\}$ conflict on C_2 is more difficult because any color change on vertex a or c would perturb one of its more numerous neighbors. Intuitively,

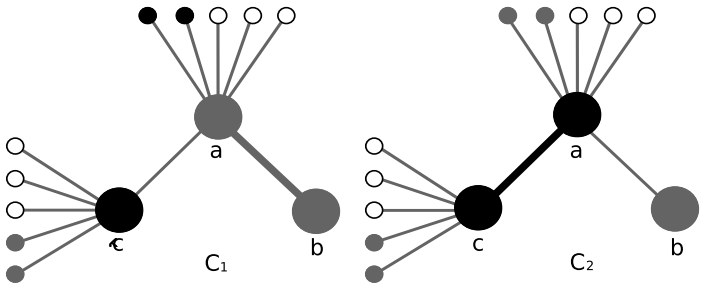

Figure 1: Two 3-colorings C_1 and C_2 with one conflict (marked with a thicker line). It is easier to solve the gray one (C_1, left) than the black one (C_2, right) even if both configurations have just a single conflict.

the more neighbors a vertex has, the more difficult it is to change its color without perturbing the rest of the configuration.

More precisely, for each conflicting vertex, we can use its degree to define a penalty term: the higher its degree is, the smaller the heuristic term (h) is, and the higher the value of the evaluation function becomes, see also (4) below. In order to take all the conflicting vertices into account, we use the following heuristic h_1 to define the penalty of k-coloring C:

$$h_1(C) = \frac{1}{|V|} \sum_{v \in \overline{\mathscr{V}}} |\overline{\mathscr{V}}_v| \cdot \frac{1}{d_v} \tag{3}$$

According to the notation of conflicting degree of vertices (Section 2.1, Definition 3), we see that $h_1(C)$ gives the total of the conflicting degrees of all the conflicting vertices of C. Our first degree-based evaluation function $\widetilde{f}_1$ can now be defined as follows:

$$\widetilde{f}_1(C) = f_c(C) - h_1(C) \tag{4}$$

The only role of the $\frac{1}{|V|}$ coefficient in (3) is to keep the value of h_1 in $[0, 1)$. Therefore, $\widetilde{f}_1$ preserves the f_c ordering: $\widetilde{f}_1(C) < \widetilde{f}_1(C')$ whenever $f_c(C) < f_c(C')$. The new evaluation function should only help the search process choose between neighbors with the same conflict number; it should not introduce penalties that outweigh the number of conflicts.

3.2 An Evaluation Function Based on Search History

We propose here a second heuristic h_2 which takes into account information collected during the search process. The considered information is the number of color changes per vertex. Basically, if a vertex v changes its color frequently, the evaluation should give more weight to v, so as to help the search process fix its color.

More precisely, let us consider a first stage of the search with the basic evaluation function f_c. For each vertex v, we compute a frequency coefficient $freq(v)$ which is a scaling of the number of color changes applied on v during the first stage. The heuristic h_2 and the second new evaluation function can now be defined as follows:

$$\widetilde{f}_2(C) = f_c(C) - h_2(C) = f_c(C) - \sum_{v \in \overline{\mathscr{V}}} |\overline{\mathscr{V}}_v| \cdot \frac{1}{freq(v)} \tag{5}$$

Given two colorings with the same conflict number, the evaluation function $\widetilde{f}_2$ prefers the one whose conflicting vertices had lower frequencies of color change in the past. We consider the vertices with high frequencies of color changes to be more critical: the new function encourages the search process to first assign a color to these vertices, by giving them more weight in the evaluation. As such, using $\widetilde{f}_2$ results in more frequent color changes on vertices that were "quite fixed" during the first stage; it has an implicit diversification effect.

3.3 Specific TS Integration and Comparison with Related Literature

To be able to integrate the calculation of h_1 or h_2 in TS, the additional computational overhead should be limited—TS needs to remain a very fast algorithm, see also Section 2.2.1. For this purpose, IRTS uses the following formula to compute any of the two heuristics (denoted by h).

$$h(C) = \sum_{v \in V} |\overline{\mathscr{V}}_v| \cdot h_v = \sum_{\{v,v'\} \in \overline{\mathscr{E}}} (h_v + h_{v'}),$$

where h_v is the penalty associated with vertex v, i.e. $h_v = \frac{1}{|V| \cdot d_v}$ for the first function, or $h_v = \frac{1}{freq(v)}$ for the second. Using (2), we observe that any new function $\widetilde{f}(C)$ can be calculated with the formula $f_c(C) - h(C) = \sum_{\{v,v'\} \in \overline{\mathscr{E}}} 1 - \sum_{\{v,v'\} \in \overline{\mathscr{E}}} (h_v + h_{v'})$. Before launching the search process, one constructs a table $\widetilde{\mathcal{T}}$ so that $\widetilde{f}(C) = \sum_{\{v,v'\} \in \overline{\mathscr{E}}} \widetilde{\mathcal{T}}_{vv'}$. The only difference between the calculation of f_c and $\widetilde{f}$ is the initial value of $\widetilde{\mathcal{T}}$. To be specific, one uses $\widetilde{\mathcal{T}}_{vv'} = 1 \; \forall v, v' \in V$ to compute f_c, or $\widetilde{\mathcal{T}}_{vv'} = 1 - h_v - h_{v'}$ to compute $\widetilde{f}$. Matrix $\widetilde{\mathcal{T}}$ can be integrated in the streamlined calculation of Γ (see Section 2.2.1), so as to perform an incremental evaluation.

A different integration issue arises from the inherent imperfections of the heuristics. The fact that $\widetilde{f}\,(C_1) < \widetilde{f}\,(C_2)$ should ideally imply that the probability to reach a legal coloring is greater when the search runs from C_1 than when it runs from C_2. Unfortunately, due to the complexity of the fitness landscape, this probability is unknown; as such, the computation of an exact evaluation is very difficult (h_1 and h_2 are only heuristics).

Such imperfections render slightly more complicated the use of $\widetilde{f}$ in Tabu Search. If one directly inserts a very discriminative $\widetilde{f}$ in Step 5.A. of Algorithm 1, the number of choices for iteratively selecting the next coloring C' could be very limited—leading to insufficient diversification. As mentioned in Step 5.A. of Algorithm 1, the next coloring C' is selected from among those that minimise the number of conflicts $\lceil \widetilde{f}\,(C_i) \rceil$; if more moves satisfy this condition, we perform a tie-breaking choice guided by $\widetilde{f}$. Consider the set of all neighboring colorings C_i minimizing the number of conflicts. The next coloring C' is selected from this set: each C_i can be chosen with a probability proportional to $h(C_i)$. This choice can be seen as an instance of a fitness-proportional selection in the evolutionary computing terminology.

IRTS seems to be the first algorithm that uses new evaluation functions within a Tabucol-based TS, making appeal to a specific integration methodology. However, degree information was also used in a different form in the constructive heuristic introduced by [18]; the authors proposed to wait to color low degree dependent vertices until all other vertices are colored. The Impasse coloring algorithm [29, 27] is based on a different encoding, but it also intends to first color high degree vertices, leaving more vertices of smaller degree in the uncolored class. Completely different evaluation functions can also be found in the literature, as for example $\hat{f}_{\text{dsjc}} = -\sum_{i=1}^{k} |C^i| + \sum_{i=1}^{k} 2|C^i||\overline{\mathscr{E}}_i|$, where C^i is the set of vertices having color i and $\overline{\mathscr{E}}_i$ is the set of conflicts of color i [23].

4 A Simple-but-Effective Tabu Tuning Technique

It is well known that the tabu list must be managed with care. In Section 2.2.2, we indicated a well-established formula for calibrating the tabu tenure: $tl = \alpha \cdot |\overline{\mathscr{E}}| + random(1, A)$. We here provide the rationale for integrating an additional reactive component.

Let us consider a plateau with n configurations $C_1, C_2, ..., C_n$ having the same conflict number $|\overline{\mathscr{E}}|$. A tabu list of length at least n is necessary to break a cycle of length n: $C_1 \rightarrow C_2 ... \rightarrow C_n \rightarrow C_1$. If the classical Tabu tenure verifies $\max tl < n$, the Tabu mechanism can no longer stop the search process from repeatedly performing the above cycle. As such, a larger-than-expected plateau is able to completely block the algorithm. Indeed, experiments confirm that the classical TS reaches sooner or later a point beyond which it can not longer make any progress (see Section 5.2.2).

To overcome this difficulty, we consider a reactive tabu list: when the conflict number stays constant during a number of iterations P_{max}, we increment the length of the tabu tenure for all subsequent iterations. In other words, when we observe P_{max} consecutive transitions $C_1 \rightarrow C_2 ... \rightarrow C_{P_{\max}}$ such that $f_c(C_1) = f_c(C_2) = \cdots = f_c(C_{P_{\max}})$, the tabu tenure becomes $tl + 1$ for the forthcoming iterations. If the conflict number stays constant for another $P_{\max}$ iterations, the tabu tenure becomes $tl+2$; after another $P_{\max}$ iterations it becomes $tl+3$, etc. The tabu tenure is thus continually incremented as long as the conflict number remains constant. We reset it to the original tl value only when f_c changes again. This way, we guarantee that, sooner or later, the tabu list is increased to a value that can break a cycle of any length. Theoretically, in the worst case, tl would be incremented indefinitely and this would lead to a tabu list that contains all possible moves. In this exceptional case, a random move is performed and we consider that the resulting random walk could *not* lead to periodic cycles.

Let us remark that using a large tabu list all the time would have a negative effect; outside large plateaus, it could encourage the algorithm to leave promising regions too early. Here, the algorithm learns from its own search evolution, and resorts to a larger tabu list only when it is necessary. This simple mechanism enabled TS to solve important looping problems without affecting its performance outside plateaus. More discussions about the practical impact of our reactive tuning are given in the experimental part (Section 5.2.2).

To our knowledge, while reactive tabu techniques were often used in other problems [2], they have not been typically applied to graph coloring. However, in the context of a different TS variant (based on partial configurations [3]), the local search process is considered trapped if the objective function fluctuation stays a long period bellow a certain threshold; this threshold is set by a separate tuning phase along with two other parameters of the reactive scheme. IRTS takes this approach even further by only using one parameter ($P_{\max}$) that can be tunned quite easily (see Section 5.1.2). More distantly related, the algorithm from [9] uses some different ideas to detect looping, focusing on the identification of vertices causing loops.

5 Experiments and Discussions

In this section, we report empirical results of the Informed Reactive TS algorithm on the complete set of DIMACS coloring benchmarks.

G	k	G	k	G	k	G	k
$dsjc125.1$	5	$r125.1$	5	$le450.5a$	5	$flat300.20.0$	20
$dsjc125.5$	17	$r125.5$	36	$le450.5b$	5	$flat300.26$	26
$dsjc125.9$	44	$r125.1c$	46	$le450.5c$	5	$flat1000.50$	50
$dsjc250.1$	8	$r250.1$	8	$le450.5d$	5	$flat1000.60$	60
$dsjc250.5$	28	$r1000.1$	20	$le450.15a$	15	$school1$	14
$dsjc250.9$	72	$r1000.1c$	98	$le450.15b$	15	$school1.nsh$	14
		$dsjr500.1$	12	$le450.25a$	25		
				$le450.25b$	25		

Table 1: Easy DIMACS k-coloring instances. For these graphs, numerous other papers report legal colorings with exactly the same values of k, but there is no mention of a solution using $k-1$ colors (for $flat$X.Y and leX.Y graphs, we even know the chromatic number).

5.1 Experimental Conditions

5.1.1 The DIMACS benchmark

The Dimacs competition benchmark [24] is composed of 46 graphs from the following families: (i) random graphs $dsjc$X.Y with X vertices and density Y; (ii) flat graphs, generated by partitioning the vertex set into K_p classes and by adding edges only between vertices of different classes ($flat$X.Y, where X denotes $|V|$ and Y is the chromatic number K_p); (iii) Leighton graphs with 450 vertices and with known chromatic number Y (leX.Y, they have a clique of size Y); (iv) two families of random geometrical graphs generated by picking points uniformly at random in the unit square and by joining any 2 points situated within a certain distance ($dsjr$X.Y and rX.Y, where X is $|V|$ and Y is the distance threshold); (v) huge random graphs ($C2000.5$ and $C4000.5$) with up to 4 million edges; (vi) class scheduling graphs ($school1$, $school1.nsh$) and a latin square graph ($latin\_square\_10$).

Easy instances and hard instances Table 1 presents the easy graph k-coloring instances; they can be rapidly solved by all our TS versions with a success rate of 100% . In what follows, we focus on the rest of 19 ***hard*** instances, as most coloring research papers do.

5.1.2 Parameters

Recall that IRTS requires only three parameters: A, α and P_{max}. The first two are inherited from previous TS versions, as they are utilized to compute the tabu tenure ($tl = \alpha \cdot |\overline{\mathscr{E}}| + random(1, A)$, see Section 2.2.2). We simply used the following values already reported in the literature [15]: $A = 10$ and $\alpha = 0.6$.

The third parameter P_{max} represents the number of iterations with constant conflict number necessary to assume the search process is stuck, i.e the reactive component is only activated after P_{max} iterations with no conflict number variation (see Section 4). We set $P_{\text{max}} = 1000$, but there are many safe values one could use. We empirically observed that each time the conflict number stayed constant for 1000 iterations, it remained so indefinitely (with the classical tunning). Larger values of P_{max} would only render the reactive reaction less prompt. Smaller

values can trigger reactive reactions more often than necessary, thus risking to influence the search process without proper reason.

5.2 Influence of the Proposed TS Extensions

Let us first present preliminary experiments and arguments regarding the positive impact of the new evaluation functions and of the reactive tabu list. While the new functions aim at better guiding TS toward promising regions, the objective of the reactive tabu list is to unlock the search process from looping. As such, the two proposed TS extensions are completely independent and complementary to one another.

5.2.1 Impact of the new functions w.r.t. instance characteristics

Our preliminary experiments showed that the influence of the new function $\widetilde{f}_1$ is more visible on certain instances than on others. Generally speaking, the best improvement is seen on the *most difficult* instances and on certain specific graph classes. An example of good performance is given by the random geometrical graphs ($dsjr$X.Y and rX.Y); f_c is clearly dominated by $\widetilde{f}_1$ in all our experiments—see also Table 2 and Table 3 below.

The explanation of this performance variation lies in the structure of the graphs, more exactly in the degree variation. For the geometrical graphs mentioned above, the maximum degree can be by an order of magnitude higher than the minimum degree (e.g. for $dsjr500.5$) and this makes any degree-based differentiation very effective. Indeed, we observed that the average graph class effectiveness of $\widetilde{f}_1$ can be ranked according to the degree variation, from the highest to the lowest: geometrical graphs, Leighton graphs, random graphs, flat graphs. An extreme case is the Latin square graph which is regular (i.e. constant degree): the new degree-based evaluation function brings no new distinction between vertices.

5.2.2 Influence of the reactive tunning

To evaluate the influence of the new reactive tabu list, we analyzed and compared the classical tabu tuning (Section 2.2.2) with the reactive tabu tuning. Recall that the objective of the reactive component is to avoid looping on k-colorings with the same conflict number. Using several representative graphs from each important family, we performed between 20 and 100 executions of the TS algorithm equipped with the classical tabu tuning and we counted how many of them got stuck before reaching 20 million iterations. Here, we consider that the search process is stuck from the moment when the conflict number does no longer variate. The conclusion of *this experiment was very clear*: more than 90% of all executions got stuck looping on a plateau before reaching 20 million iterations.

By introducing the reactive part, these looping problems are solved and the TS algorithm could successfully escape all plateaus. This allows the search process to effectively take profit from longer running times—without reactive tuning, there would be no much use to run IRTS more than 20 million iterations (regardless of the evaluation function). We empirically observed that the reactive component is triggered only a few times in millions of iterations, and so, there is virtually no negative interference with other algorithm components. Another positive point of our reactive scheme is the simplicity: only one parameter $P_{\max}$ is needed and it can be easily assigned a safe value—i.e. any value X satisfying the property that if the conflict number $|\overline{\mathscr{E}}|$ stays constant for X iterations, $|\overline{\mathscr{E}}|$ remains constant indefinitely.

5.3 Results on Short Term Stopping Conditions

To assess the potential impact of IRTS inside more sophisticated hybrid algorithms, we consider a limit of 1.000.000 iterations and we analyse the running profile along these iterations. A local search coloring algorithm can visit millions of colorings per minute, and so, the chosen iteration limit corresponds to the relatively short running times typically allowed for internal subroutines of hybrid algorithms.

Table 2 reports the average conflict numbers along these 1.000.000 iterations with all three functions on 10 representative graphs from different families. Rather then reporting only the minimum conflict number, this average information offers a better view of the global behavior of the algorithm over the considered number of iterations. We observe that the average conflict number is always smaller for $\widetilde{f}_1$ than for $\widetilde{f}_c$. The conclusion is that the new evaluation function usually allows the search process to visit more rapidly colorings of higher quality.

Graph	k	Average Number of Conflicts			Statistical Confirmation	
		$\widetilde{f}_1$	$\widetilde{f}_2$	f_c	$[\widetilde{f}_1] \not\equiv [f_c]$	$[\widetilde{f}_2] \not\equiv [f_c]$
$dsjc250.5$	28	8.138	7.133	9.878	Yes	Yes
$dsjc500.5$	48	24.9	28.5	26.6	Yes	Yes
$dsjc1000.5$	87	35	32.2	37.7	Yes	Yes
$dsjr500.5$	122	5.106	8.875	10.46	Yes	Yes
$r1000.5$	234	16.02	25.76	29.09	Yes	Yes
$le450\_25c$	25	9.38	9.524	12.97	Yes	Yes
$le450\_25d$	25	9.183	13.94	13.74	Yes	No
$flat300\_28\_0$	30	21.63	23.87	22.13	No	No
$flat1000\_76\_0$	86	30.39	29.02	32.04	Yes	Yes

Table 2: Average conflict number of the colorings visited during the first 1000000 iterations with all three functions. In most cases, the modified functions, in particular $\widetilde{f}_1$, allow the algorithm to visit colorings with (statistically) fewer conflicts.

Indeed, the differences between these averages were confirmed by a statistical test. We considered the null hypothesis that the average of the conflict numbers obtained with $\widetilde{f}_1$ (or $\widetilde{f}_2$ respectively) is equal to the average obtained with f_c. Using a very confident level of significance of $\alpha = 0.1\%$, this hypothesis was rejected in most of the cases, confirming that most reported differences are statistically significant—see the last two columns of Table 2.

In the best cases, the conflict numbers of the configurations visited by $\widetilde{f}_1$-IRTS can even be half of those visited by f_c-IRTS. The second function $\widetilde{f}_2$ also shows an improvement on two thirds of instances, but with a smaller amplitude (recall that f_c-IRTS is equivalent to $\widetilde{f}_2$-IRTS in the first half of the search, during the first 500.000 iterations).

5.4 Long-term Results and Comparison with the Best Algorithms

In this section, we provide detailed results (Table 3) with all three evaluation functions for all hard instances. For each instance, we perform 10 executions (launched from different random colorings) and we report the success rate, as well as the average computing effort for finding

a conflict-free coloring. We allow the algorithm a maximum time limit of 10 hours.[1] In the coloring literature, it is a common practice to use time limits of several hours; this is also the case for many state-of-the-art algorithms from Table 4. However, one should be aware that it is difficult to make local search algorithms substantially improve their empirical performance by allowing more than one hour of running time (if there are plateaus looping problems).

Graph	k	$\widetilde{f}_1$-RCTS			f_c-RCTS			$\widetilde{f}_2$-RCTS		
		#Hits [/ 10]	#Iters [10^6]	Time [h]	#Hits [/ 10]	#Iters [10^6]	Time [h]	#Hits [/ 10]	#Iters [10^6]	Time [h]
$dsjc500.1$	12	10/10	96	$< \frac{1}{4}$	10/10	64	$< \frac{1}{4}$	10/10	64	$< \frac{1}{4}$
$dsjc500.5$	48	1/10	1352	6.33	0/10	—	—	0/10	—	—
$dsjc500.9$	126	9/10	360	1.67	10/10	457	2.4	10/10	466	2.5
$dsjc1000.1$	21	10/10	1	$< \frac{1}{4}$	10/10	2	$< \frac{1}{4}$	10/10	2	$< \frac{1}{4}$
$dsjc1000.5$	87	1/10	873	7	0/10	—	—	0/10	—	—
$dsjc1000.9$	224	4/10	420	7.5	1/10	321	5	3/10	492	7.33
$dsjr500.1c$	85	5/10	470	1.2	1/10	7	$< \frac{1}{4}$	1/10	7	$< \frac{1}{4}$
$dsjr500.5$	122	7/10	469	2.4	0/10	—	—	0/10	—	—
$r250.5$	65	10/10	99	$< \frac{1}{4}$	0/10	—	—	0/10	—	—
$r1000.5$	237	2/10	1059	7.5	0/10	—	—	0/10	—	—
$le450.15c$	16	8/10	11	$< \frac{1}{4}$	9/10	17	$< \frac{1}{4}$	9/10	17	$< \frac{1}{4}$
$le450.15d$	16	10/10	1	$< \frac{1}{4}$	10/10	< 1	$< \frac{1}{4}$	10/10	< 1	$< \frac{1}{4}$
$le450.25c$	25	9/10	621	1.1	6/10	572	1.2	6/10	203	1.2
$le450.25d$	25	9/10	937	2	2/10	1895	4.5	2/10	1895	4.5
$flat1000.76$	87	10/10	290	2.3	10/10	265	2	10/10	265	2
$flat300.28$	30	4/10	1183	4.5	6/10	538	2.33	8/10	737	3.1
$latin\_square$	100	6/10	641	3	4/10	1005	5	5/10	1141	5.5
$C2000.5$	162	4/10	237	3.63	1/10	601	9.5	2/10	477	8
$C4000.5$	305	4/10	88	4.75	2/10	85	4.5	2/10	98	5.5

Table 3: Detailed results of IRTS with a time limit of 10 hours for all three evaluation functions. $\widetilde{f}_1$-IRTS finds better solutions than f_c-IRTS on 25% of graphs and has a improved success rate on another 25% of graphs. The difference between $\widetilde{f}_2$-IRTS and f_c-IRTS can only be visible on instances that require more than 5 hours (the last three and $dsjc1000.9$).

The first two columns of Table 3 denote the k-*Coloring* instance, i.e. the graph and the number of colors k. For each evaluation function, we provide the success rate (Columns 3,6,9) and the average computation efforts needed to solve the instance: the average number of iterations in millions (Columns 4, 7, 10) and the average time in hours (Columns 5, 8, 11).

First, we observe that $\widetilde{f}_1$-IRTS reaches better k-colorings (i.e. with a smaller k than f_c-IRTS) for more than 25% of the instances—i.e. 5 graphs our of 19. Furthermore, we observe that $\widetilde{f}_1$-IRTS obtains a success rate twice as good as f_c-IRTS for another 25% of graphs. Thus, one can say that the new evaluation function $\widetilde{f}_1$ brings important improvement on half of the instances (i.e. 10 out of 19).

The most important progress can be observed on graph classes with a large degree variation. Indeed, on random geometrical graphs, $\widetilde{f}_1$-IRTS can quickly (in less than one hour) reach upper

[1] We used a 2.8 GHz Xeon processor. The programs were written in C++ and compiled with the -O2 optimization option—the gcc compiler, version 4.1.2 under Linux (kernel version 2.6).

bounds that f_c-IRTS did not find in 10 hours—see graph $r250.5$. Furthermore, $\widetilde{f}_1$-IRTS solves the difficult instance ($dsjr500.5$, 122) in less than two hours and with a stable success rate. Out of the best fifteen algorithms from the literature (see also Table 4), this instance was previously solved only by very few (much more sophisticated and very recent) algorithms [27, 32, 26]. Similar observations apply to $r1000.5$ or $r250.5$: to the best of our knowledge, IRTS is the first local search capable of finding high quality upper bounds for these graphs.

The advantage of the evaluation function $\widetilde{f}_2$ over f_c is less pronounced on certain graphs. This is not very surprising, as $\widetilde{f}_2$-IRTS diverges from f_c-IRTS only in the second half of the search, after 5 hours of computation. Consequently, the values reported by f_c-IRTS and $\widetilde{f}_2$-IRTS are completely identical on instances that never require more than 5 hours. By considering now only the 4 instances that often require more than 5 hours (the last three and $dsjc1000.9$), one observes that $\widetilde{f}_2$ has an improved success rate for 3 instances out of 4.

5.4.1 Direct comparison with state-of-the-art algorithms

Graph		Local Search Algorithms						Population-based Hybrid Algorithms							Ants	
	IRTS	Ils [7, 30] 2002	Vns [1] 2003	Als [9] 2008	PCol [3] 2008	Vss [19] 2008	TS-Div/Int [33] 2010	Dcns [29] 1996	Hga [14] 1996	Hea [15] 1999	Amcol [17] 2008	Mmt [27] 2008	MCol [26] 2010	EvoDiv [32] 2010	Abac [4] 2008	ALSC [31] 2009
$dsjc500.1$	12	12	—	13	12	12	12	—	—	—	12	12	12	12	13	12
$dsjc500.5$	48	49	49	50	48	48	48	49	49	48	48	48	48	48	50	48
$dsjc500.9$	126	126	—	128	126	126	126	—	—	—	126	127	126	126	127	127
$dsjc1000.1$	21	—	—	21	20	20	20	—	—	20	20	20	20	20	21	20
$dsjc1000.5$	87	89	90	89	89	88	85	89	84	83	84	83	83	83	91	86
$dsjc1000.9$	224	—	—	230	225	224	223	226	—	224	224	224	223	223	229	225
$r250.5$	65	—	—	—	66	—	—	65	69	—	—	65	65	65	–	–
$r1000.5$	237	—	—	—	248	—	—	241	268	—	—	234	245	237	–	–
$dsjr500.1c$	85	—	—	—	85	85	—	85	85	—	86	85	85	85	85	85
$dsjr500.5$	122	124	—	—	126	125	—	123	130	—	125	122	122	122	128	125
$le450.15c$	16	15	15	—	15	15	15	15	15	15	15	15	15	15	15	15
$le450.15d$	16	15	15	—	15	15	15	15	15	—	15	15	15	15	–	15
$le450.25c$	25	26	—	—	25	26	25	25	—	26	26	25	25	25	26	26
$le450.25d$	25	26	—	—	25	26	25	25	—	—	26	25	25	25	26	26
$flat300.28$	30	31	31	—	28	28	28	31	33	31	31	31	29	29	–	29
$flat1000.76$	87	—	89	—	88	86	85	89	84	83	84	82	82	82	–	85
$latin\_square$	100	99	—	—	—	—	—	98	106	—	104	101	99	98	100	–
$C2000.5$	162	—	—	—	—	—	—	165	169	—	—	—	148	148	–	–
$C4000.5$	305	—	—	—	—	—	—	—	313	—	—	—	272	271	–	–

Table 4: Upper bounds reached by IRTS (with a time limit of 10 hours) compared to those reported by 15 state-of-the-art papers. Certain bounds from Columns 3–17 represent the best performance of more than one algorithm, with very diverse stopping conditions.

Table 4 contrasts the results[2] of IRTS with the best results of fifteen other algorithms (six local search approaches, seven evolutionary hybrid algorithms and two ant-based methods). IRTS competes well with most previous local search methods: it reaches three upper bounds that were never found before with local search (on geometrical graphs, see $dsjr500.5$, $r1000.5$, or $r250.5$). Regarding the hybrid approaches, one should be aware that these algorithms outperform *most* local search methods on *several* graphs. For instance, on $dsjc1000.5$, all (recent) hybrid methods reach upper bounds with 3-4 colors less than any existing local search (except

[2]The colorings reported by IRTS are publicly available at `www.info.univ-angers.fr/pub/porumbel/graphs/irts/`

TS-Div/Int). The advantage of IRTS is that it is quite simple and it can be directly employed as a local optimizer in other hybrid algorithms (we already used a degree-based function in [32]).

6 Conclusions

This paper proposed two extensions of the classical Tabu Search template for vertex graph coloring: new well-informed evaluation functions and a simple-but-effective tabu list. The new functions integrate additional information related to the graph structure (degrees of conflicting vertices, for $\widetilde{f}_1$), or dynamic knowledge acquired along the search (frequencies of color changes, for $\widetilde{f}_2$). The most important positive effect of the evaluation function $\widetilde{f}_1$ can be observed on the (geometrical) graphs with the largest degree variation. Furthermore, the proposed reactive tabu list completely eliminates the risk of looping indefinitely on plateaus; as such, TS is effectively allowed to take profit from longer computing time.

Tabu Search is routinely used for graph coloring, both as a stand-alone algorithm, as well as a local optimizer inside state-of-the-art hybrid methods. This paper shows how integrating new evaluation functions in Tabu Search can help it perform better in both cases. Compared to previous local search algorithms, IRTS is capable of finding several upper bounds that were never reached before. Furthermore, the proposed TS extensions could be profitably used by the TS internal routine of any existing hybrid algorithm, as their induced overhead is negligible (see Sections 3.3 and 4).

Finally, let us comment that the idea of using artificial well-informed evaluation functions, although often partially overlooked, could be very useful for meta-heuristics in general. A carefully designed evaluation function, using problem-specific knowledge, would permit to enhance our capacity of solving hard and large combinatorial optimization problems.

Acknowledgments: This work is partially supported by projects MILES (2007-2009), Radapop (2009-2013) and LigeRO (2009-2013) from the Région "Pays de la Loire", France.

References

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

[2] R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6:126–126, 1994.

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

[4] T.N. Bui, T.V.H. Nguyen, C.M. Patel, and K.A.T. Phan. An ant-based algorithm for coloring graphs. *Discrete Applied Mathematics*, 156(2):190–200, 2008.

[5] M. Chams, A. Hertz, and D. de Werra. Some experiments with simulated annealing for coloring graphs. *European Journal of Operational Research*, 32(2):260–266, 1987.

[6] 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*, pages 63–1–63–17. Chapman & Hall/CRC, Boca Raton, FL, USA., 2007.

[7] M. Chiarandini and T Stützle. An application of iterated local search to graph coloring. In D. S. Johnson et al., editors, *Computational Symposium on Graph Coloring and its Generalizations*, pages 112–125, 2002.

[8] D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48(3):295–305, 1997.

[9] I. Devarenne, Mabed H., and A. Caminada. Intelligent neighborhood exploration in local search heuristics. In *Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 144–150. IEEE Computer Society, 2006.

[10] R. Dorne and J.K. Hao. A new genetic local search algorithm for graph coloring. In *PPSN 98*, volume 1498 of *LNCS*, pages 745–754. Springer, 1998.

[11] K.A. Dowsland and J.M. Thompson. An improved ant colony optimisation heuristic for graph colouring. *Discrete Applied Mathematics*, 156(3):313–324, 2008.

[12] N. Dubois and D. de Werra. Epcot: An efficient procedure for coloring optimally with tabu search. *Computers & Mathematics with Applications*, 25(10-11):35 – 45, 1993.

[13] C. Fleurent and J.A. Ferland. Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*, 63(3):437–461, 1996.

[14] C. Fleurent and J.A. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In *Cliques, Coloring, and Satisfiability Second DIMACS Implementation Challenge* [24], pages 619–652.

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

[16] P. Galinier and A. Hertz. A survey of local search methods for graph coloring. *Computers and operations research*, 33(9):2547–2562, 2006.

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

[18] F. Glover, M. Parker, and J. Ryan. Coloring by tabu branch and bound. In *Cliques, Coloring, and Satisfiability Second DIMACS Implementation Challenge* [24], pages 285–307.

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

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

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

[22] A. Jagota. An adaptive, multiple restarts neural network algorithm for graph coloring. *European Journal of Operational Research*, 93(2):257–270, 1996.

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

[24] D.S. Johnson and M. Trick. *Cliques, Coloring, and Satisfiability Second DIMACS Implementation Challenge*, volume 26 of *DIMACS series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.

[25] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[26] Z. Lü and J.K. Hao. A memetic algorithm for graph coloring. *European Journal of Operational Research*, 203(2):241–250, 2010.

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

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

[29] C. Morgenstern. Distributed coloration neighborhood search. In *Cliques, Coloring, and Satisfiability Second DIMACS Implementation Challenge* [24], pages 335–358.

[30] L. Paquete and T. Stützle. An experimental investigation of iterated local search for coloring graphs. in s. cagnoni et al., editors, *evoworkshops*, volume 2279 of *LNCS*, pages 121–130. springer, 2002.

[31] M. Plumettaz, D. Schindl, and N. Zufferey. Ant Local Search and its efficient adaptation to graph colouring. *Journal of the Operational Research Society*, 61(5):819–826, 2009.

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

[33] C.D. Porumbel, J.K. Hao, and P. Kuntz. A search space "cartography" for guiding graph coloring heuristics. *Computers & Operations Research*, 37:769–778, 2010.

[34] D. C. Porumbel, J.K. Hao, and P. Kuntz. Spacing memetic algorithms. In *GECCO*, pages 1061–1068, 2011.

[35] S. Prestwich. Coloration neighbourhood search with forward checking. *Annals of Mathematics and Artificial Intelligence*, 34(4):327–340, 2002.

[36] P.M. Talaván and J. Yáñez. The graph coloring problem: A neuronal network approach. *European Journal of Operational Research*, 191(1):100–111, 2008.

[37] O. Titiloye and A. Crispin. Graph Coloring with a Distributed Hybrid Quantum Annealing Algorithm. J. O'Shea, N. Nguyen, K. Crockett, R. Howlett, L. Jain, editors, *KES-AMSTA*, volume 6682 of *LNCS*, pages 553–562. springer, 2011.

[38] Q. Wu and J.K. Hao. Coloring large graphs based on independent set extraction *Computers and Operations Research* 39: 283-290, 2012