Position-Guided Tabu Search Algorithm for the Graph Coloring Problem

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

[1] LERIA, Université d'Angers, Angers, France
{porumbel,hao}@info.univ-angers.fr
[2] LINA, Polytech'Nantes, Nantes, France
pascale.kuntz@univ-nantes.fr

Abstract. A very undesirable behavior of any heuristic algorithm is to be stuck in some specific parts of the search space, in particular in the basins of attraction of the local optima. While there are many well-studied methods to help the search process escape a basin of attraction, it seems more difficult to prevent it from looping between a limited number of basins of attraction. We introduce a Position Guided Tabu Search (PGTS) heuristic that, besides avoiding local optima, also avoids re-visiting candidate solutions in previously visited regions. A learning process, based on a metric of the search space, guides the Tabu Search toward yet unexplored regions. The results of PGTS for the graph coloring problem are competitive. It significantly improves the results of the basic Tabu Search for almost all tested difficult instances from the DIMACS Challenge Benchmark and it matches most of the results ever obtained by the best algorithms in the literature.

1 Introduction

It is well known that the performance of all heuristic algorithms is heavily influenced by the search space structure. Consequently, the design of an efficient algorithm needs to exploit, implicitly or explicitly, some features of the search space. For many heuristics, especially local searches, the difficulty is strongly influenced by the asperity of the local structures of local optima (e.g. isolated local optima, plateau structures, valley structures, etc.). A paradigmatic example of a difficult structure is the trap [1], i.e., a group of close local minima confined in a deep "well". If trapped into such a structure, even a local search with local optimum escape mechanisms can become locked looping between the local minima inside the well. Several global statistical indicators (i.e., convexity, ruggedness, smoothness, fitness distance correlation) have also been proposed to predict the performance of both local and evolutionary algorithms; we refer to [2, 3] for a summary of such measures and related issues.

Other research studies focus on the structural similarities between local optima (i.e., the "backbone" structures) or on their global arrangement (see [4] for a detailed summary of the research threads in search space analysis). Indeed, the different local optimum characteristics of the search space (the number of local

T. Stützle (Ed.): LION 2009 III, LNCS 5851, pp. 148–162, 2009.

optima, their space distribution, the topology of their basins of attraction, etc.) may be very different from one problem to another and even from one instance to another. These specific properties have been investigated for several classical problems, such as: boolean satisfiability [5,6], the 0–1 knapsack problem [7], graph coloring [8, 9, 10], graph bi-partitioning [11], the quadratic assignment problem [12], job shop or flow shop scheduling [4,13], and arc crossing minimization in graph drawing [14]. All these studies conclude that the local optimum analysis has a great potential to give a positive impact on the performance.

However, the operational integration of specific search space information in a search process remains a difficult problem. To achieve this, a heuristic needs to learn how to make better local decisions using global information available at coarser granularity levels. Moreover, the search process has usually no information on the search space before actually starting the exploration. To overcome such difficulties, the integration of a learning phase in the optimization process ("learning while optimizing") seems very promising. This approach, using ideas of reactive search [15], aims at developing an algorithm capable of performing a self-oriented exploration.

In this paper, we focus on the graph coloring problem and we present such a reactive algorithm with two central processes: (i) a classical local search based on Tabu Search (TS) [16], (ii) a learning process that investigates the best configurations visited by the first process. This learning process has the role of effectively guiding TS toward yet unexplored regions. It integrates a positional orientation system based on a metric of the search space; for this, we use a distance function that indicates how many steps TS needs to perform to go from one configuration to another.

More precisely, the Position Guided Tabu Search (PGTS) algorithm employs an extended tabu list length whenever it detects that it is exploring the proximity of a previously visited configuration, i.e., so as to avoid re-exploring the region. This strategy does not strictly prevent the algorithm from revisiting such regions, but the probability of avoiding them is strongly increased by a reinforced diversification phase associated with the extended tabu list. Here, we propose for the graph coloring problem a strategy based on a tractable distance computation, with time complexity $O(|V|)$, where V is the vertex set of the graph. We show that PGTS significantly improves the performances of the basic TS algorithm on a well-known set of DIMACS instances, and that it competes well with the best algorithms from literature.

The rest of the paper is organized as follows. Section 2 briefly outlines the graph coloring problem and its traditional TS algorithm (Tabucol). The Position Guided Tabu Search for graph coloring and the distance definition are presented in section 3. Section 4 is devoted to experimental results. Section 5 presents related work and provides elements for the generalization of PGTS to other combinatorial optimization problems, followed by some conclusions in the last section.

2 The Graph Coloring Problem and Its Classical Tabu Search Algorithm

We briefly recall the basic notions and definitions related to the graph coloring problem and to the tabu search algorithm adapted to this problem.

2.1 Definitions

Let $G = (V, E)$ be a graph with V and E being respectively the vertex and edge set. Let k be a positive integer.

Definition 1. *(Graph coloring and k-coloring) The graph G is k-colorable if and only if there exists a conflict-free vertex coloring using k colors, i.e., a function $c : V \rightarrow \{1, 2, \cdots, k\}$ such that $\forall \{i,j\} \in E, c(i) \neq c(j)$. The graph coloring problem (COL) is to determine the smallest k (the chromatic number denoted by χ_G) such that G is k-colorable.*

Definition 2. *(Color (array) based representation) We denote any function $c : V \rightarrow \{1, 2, \cdots, k\}$ by $C = (c(1), c(2), \cdots, c(|V|))$. We say that C is a candidate solution (or configuration) for the k-coloring problem (G, k).*

Moreover, C is said to be a proper (conflict-free) or legal coloring if and only if $c(i) \neq c(j)$, $\forall \{i, j\} \in E$. Otherwise, C is an improper (conflicting) coloring. A legal coloring is also referred to as a *solution* of the k-coloring problem (G, k).

Definition 3. *(Partition representation) A k-coloring $C = (c(1), c(2), \cdots, c(|V|))$ is a partition $\{C^1, C^2, \ldots, C^k\}$ of V (i.e., a set of k disjoint subsets of V covering V) such that $\forall x \in V$, $x \in C^i \Leftrightarrow c(x) = i$.*

We say that C^i is the class color i induced by the coloring C, i.e., the set of vertices having color i in C. This partition based definition is particularly effective to avoid symmetry issues arising from the color based encoding. We will see (Section 3.2) that the distance between two colorings can be calculated as a set theoretic partition distance.

Definition 4. *(Conflict number and objective function) Given a configuration C, we call conflict (or conflicting edge) any edge having both ends of the same color in C. The set of conflicts is denoted by $CE(C)$ and the number of conflicts (i.e., $|CE(C)|$—also referred to as the conflict number of C) is the objective function $f_c(C)$. A conflicting vertex is a vertex $v \in V$, for which there exists an edge $\{v, u\}$ in $CE(C)$.*

In this paper, we deal with the k-coloring problem (k-COL), i.e., given a graph G and an integer k, the goal is to determine a legal k-coloring. From an optimization perspective, the objective is to find a k-coloring minimizing the conflict number $f_c(C)$.

Algorithm 1. Basic Tabu Search Algorithm for Graph Coloring

Input: G, k
Return value: $f_c(C_{\text{best}})$ (i.e., 0 if a legal coloring is found)
C: the current coloring; C_{best}: the best coloring ever found
Begin
1. Set C a random initial configuration
2. **While** a *stopping condition* is not met
   (a) Find the best non-tabu $C' \in N(C)$ (a neighbor C' is tabu if and only if the pair (i,i'), corresponding to the move $C \xrightarrow{C(i):=i'} C'$, is marked tabu)
   (b) Set $C = C'$ (i.e., perform move $C(i) := i'$)
   (c) Mark the pair (i,i') tabu for T_ℓ iterations
   (d) **If** $(f_c(C) < f_c(C_{\text{best}}))$
       – $C_{\text{best}} = C$
End

2.2 Tabu Search for Graph Coloring

Following the general ideas of TS [16], Tabucol [17] is a classical algorithm for k-COL that moves from one configuration to another by modifying the color of a conflicting vertex. The main adaptation of the Tabu Search meta-heuristic to graph coloring consists in the fact that it does not mark as tabu a whole configuration, but only a color assignment. To check whether a specific neighbor is tabu or not, it is enough to test the tabu status of the color assignment that would generate the neighbor. The general skeleton of our Tabucol implementation is presented in Algorithm 1; the stopping condition is to find a legal coloring or to reach a maximum number of iterations (or a time limit). The most important details that need to be filled are the neighborhood relation N and the tabu list management.

Neighborhood N. Given a coloring problem $(G(V,E),k)$, the search space Ω consists of all possible colorings of G; thus $|\Omega| = |V|^k$. A simple neighborhood function $N : \Omega \to 2^\Omega - \{\emptyset\}$ can be defined as follows. For any configuration $C \in \Omega$, a neighbor C' is obtained by changing the color of a single *conflicting* vertex in C.

Tabu List Management. There are several versions of this basic algorithm in the literature, but their essential differences lie in the way they set the tabu tenure T_ℓ. In our case, it is dynamically adjusted by a function depending on the objective function (i.e., the conflict number $f_c(C) = |CE(C)|$—as in [18,19,20]), but also on the number m of the last consecutive moves that did not modify the objective function. More precisely, $T_\ell = \alpha * f_c(C) + random(A) + \left\lfloor \frac{m}{m_{\max}} \right\rfloor$, where α is a parameter taking values from $[0,1]$ and $random(A)$ indicates a function returning a random value in $\{1,2,\ldots,A\}$. In our tests, as previously published in [20], we use $\alpha = 0.6$ and $A = 10$.

The last term constitutes a reactive component only introduced to change T_ℓ when the conflict number does not change for $m_{\max}$ moves; typically, this situation appears when the search process is completely blocked cycling on a plateau. Each series of consecutive $m_{\max}$ (usually $m_{\max} = 1000$) moves leaving the conflict number unchanged increments all subsequent values of T_ℓ—but only until the conflict number changes again; such a change resets m to 0.

3 Position Guided Tabu Search Algorithm

3.1 Generic Description

The main objective of the algorithm is to discourage the search process from visiting configurations in some space regions that are already considered as explored. Taking as a basis the classical Tabu Search for graph coloring (see Algorithm 1), the new algorithm PGTS (Position Guided Tabu Search) integrates a learning component (Step 4.(c) in Algorithm 2) that processes all visited configurations and records a series of search space regions $S(C)$ that cover the whole exploration path. Ideally, these recorded regions contain all colorings that are structurally related to the visited ones.

A statistical analysis of the search space, briefly described in Section 3.3, has led us to define $S(C)$ as the closed sphere centered at C of radius R:

Definition 5. *(Sphere) Given a distance function $d : \Omega \times \Omega \longrightarrow \mathbb{N}$, a configuration $C \in \Omega$ and a radius $R \in \mathbb{N}$, the R-sphere $S(C)$ centered at C is the set of configurations $C' \in \Omega$ such that $d(C, C') \leq R$.*

Here, the distance $d(C, C')$ (see also Section 3.2) can be interpreted as the shortest path of TS steps between C and C'. More formally, $d(C, C')$ is the minimal number n for which there exist $C_0, C_1, \ldots C_n \in \Omega$ such that: $C_0 = C, C_n = C'$ and $C_{i+1} \in N(C_i)$ for all $i \in [0 \ldots n-1]$.

PGTS starts iterating as the basic TS does, but, with the learning component (Step 4.(c), Algorithm 2), it also records the center of the currently explored sphere. While the current configuration C stays in the sphere of the last recorded center C_p, we consider the search process "pivots" around point C_p. PGTS performs exactly the same computations as TS except checking the distance $d(C, C_p)$ that is performed each iteration (in Step 4.(c)).

As soon as the search leaves the current sphere, the learning component activates a global positioning orientation system. It first compares C to the list of all previously recorded configurations (procedure Already–Visited in Algorithm 2) to check whether it has already visited its proximity or not. If C is not in the sphere of a previously recorded configuration, it goes on only by changing the pivot; i.e., it replaces C_p with C and records it. Otherwise, this means the search is re-entering the sphere of a previously recorded configuration and that should be avoided. This is a situation that triggers a signal to make more substantial configuration changes: a diversification phase is needed. For this purpose, the chosen mechanism is to extend the classical tabu tenure T_ℓ with a T_c factor.

Algorithm 2. Position Guided Tabu Search

```
PROCEDURE ALREADY-VISITED
Input: current configuration C
Return value: TRUE or FALSE
1. Forall recorded configurations C_rec:
     – If d(C, C_rec) ≤ R
         • Return TRUE
2. Return FALSE
ALGORITHM POSITION-GUIDED TABU SEARCH
Input: the search space Ω
Return value: the best configuration C_best ever visited
C: the current configuration
1. Choose randomly an initial configuration C ∈ Ω
2. C_p = C (the pivot, i.e., the last recorded configuration)
3. T_c = 0  (the value by which PGTS extends the tabu tenure T_ℓ)
4. While a stopping condition is not met
   (a) Choose the best non-tabu neighbor C' in N(C)
   (b) C = C'
   (c) If d(C, Cp) > R (the Learning Component)
         – C_p = C
         – If ALREADY-VISITED(C_p)
             • Then Increment T_c
         – Else
             • T_c = 0
             • Record C_p
   (d) Mark C as tabu for T_ℓ + T_c iterations
   (e) If (f_c(C) < f_c(C_best))
         – C_best = C
   (f) If (f_c(C) < f_c(C_p))
         – Replace C_p with C in the archive
         – C_p = C (i.e., "recentering" the current sphere)
5. Return C_best
```

Using longer tabu lists makes configuration changes more diverse because the algorithm never repeats moves performed during the last $T_\ell + T_c$ iterations. As such, by varying the tabu tenure, we control the balance between diversification and intensification—a greater T_c value implies a stronger diversification of the search process. A suitable control of T_c guarantees that PGTS is permanently discovering new regions and that it can never be blocked looping only through already visited regions.

The performance of this algorithm depends on three factors: a fast procedure to compute the distance (Section 3.2), a suitable choice of the spherical radius R (Section 3.3), and a strategy to quickly check the archive (Section 3.4).

3.2 Distance Definition and Calculation Complexity

The definition of the sphere $S(C)$ in the search space Ω is based on the following distance: the minimal number of neighborhood operations that need to be applied

on a coloring so that it becomes equal with the other. This distance reflects the structural similarity between two colorings, the smaller the distance the more similar the colorings are. The equality is defined on the partition definition of a coloring (Definition 3): two colorings C_a and C_b are equal if and only if they designate the same classes of colors, i.e., if there exists a *color relabeling* σ (a bijection $\{1, 2, \ldots, k\} \xrightarrow{\sigma} \{1, 2, \ldots, k\}$) such that $C_a^i = C_b^{\sigma(i)}$, with $1 \leq i \leq k$.

The coloring distance can thus be expressed as a set-theoretic *partition distance*: the minimal number of vertices that need to be transferred from one class to another in the first partition so that the resulting partition is equal to the second. This distance function was defined since the 60ies and it can be calculated with well-studied methods—for example, see [21] for a general set-theoretic approach or [22] for the graph coloring application. Most studies consider a $O(|V| + k^3)$ algorithm based on the Hungarian method. However, we recently proved that, under certain conditions [23], this distance can be computed in $O(|V|)$ time with a enhanced method. Indeed, a fast distance computation is crucial to the PGTS algorithm as it calculates at least one distance per iteration and the time complexity of an iteration is $O(|V| + k \times f_c(C))$ (mainly due to operation 2.(a) in Algorithm 1).

The $O(|V|)$ distance calculation method is a Las Vegas algorithm (i.e., an algorithm that either reports the correct result or informs about the failure) that could calculate more than 90% of the required distances in practice: only less than 10% of cases require using the Hungarian algorithm (of complexity between $O(|V|+k^2)$ and $O(|V|+k^3)$ in the worst case). Basically, the distance is calculated with the formula $d(C_a, C_b) = |V| - s(C_a, C_b)$, where s is the complementary function of similarity, i.e., the maximum number of elements of C_a that do not need to be transfered to other C_a classes in order to transform C_a into C_b. Our algorithm goes through each element $x \in V$ and increments a matching counter between color class $C_a(x)$ of C_a and $C_b(x)$ of C_b. Denoting by $C_b^{\sigma(i)}$ the best match (with the highest counter) of class C_a^i, the similarity is at most $\sum_{1 \leq i \leq k} |C_a^i \cap C_b^{\sigma(i)}|$. The computation time can be very often reduced as PGTS does not actually require the precise value of the distance; it only has to check whether it is greater than R or not. If the aforementioned sum is less than $|V|-R$, the distance is greater than R.

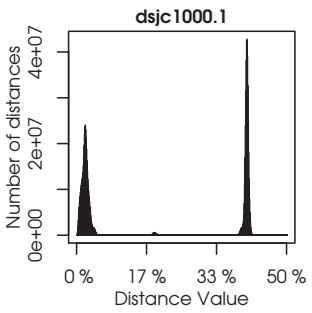
Let us note that, as the distance values are in $[0, |V|)$, we often report the distance value in terms of percentages of $|V|$.

3.3 Choice of the Spherical Radius

In the exploration process of the regions, the parameter R controls the size of the visited spheres and, indirectly, the number of recorded spheres. The extreme value $R = 0$ forces the algorithm to record all visited configurations and that compromises the solving speed (via the `Forall` loop of the Already-Visited procedure). For the other extreme value $R = |V|$, the whole search space is contained in a unique sphere (because the distance is always less than $|V|$) and the algorithm is equivalent to the basic TS.

The effective choice of R has been determined from an analysis of a classical TS scenario: start the exploration process from an initial local minimum C_0, and denote by C_0, C_1 ,$C_2, \ldots, C_n$ the best colorings it visits, i.e., the visited configurations satisfying $f_c(C_i) \leq f_c(C)$, with $0 \leq i \leq n$ (note that most of the C's can be local optima, too). We recorded all these configurations up to $n = 40000$ and we studied the possible values of the distances between each C_i and C_j with $1 \leq i, j \leq n$.

The histogram of the statistical distribution of these distance values directly showed a bimodal distribution with many occurrences of very small values (around $0.05|V|$) and of some much larger values—see an example on the right figure. There exist some distant clusters of close points; the high distances correspond to inter-cluster distances and the small ones to intra-cluster distances. If we denote a "cluster diameter" by c_d, we observed that c_d varies from $0.07|V|$ to $0.1|V|$ depending on the graph; moreover we noticed that:



- there are numerous pairs (i, j) such that $d(C_i, C_j) < c_d$;
- there are numerous pairs (i, j) such that $d(C_i, C_j) > 2c_d$;
- there are very few (less than 1%) pairs (i, j) such that $c_d < d(C_i, C_j) < 2c_d$.

It is important to note that any two visited local minima situated at a distance of more than $0.1|V|$ are not in the same cluster because, ideally, they have different backbones. We assume that this observation holds on all sequences of colorings visited by TS; the value of R is set to $0.1|V|$ on all subsequent runs. Hence, as soon as PGTS leaves the sphere $S(C)$ of a visited configuration C, it avoids to re-visit later other configurations from the same cluster.

3.4 Archive Exploration

The exploration of the archive is a tricky stage for the computation time because of the numerous distance computations. Our objective is to keep the execution time of the learning component in the same order of magnitude as the exploring component. Due to the small bound of the distance computation time, computing one distance per iteration (i.e., in Step 4.(c)) is fast. The critical stage appears when PGTS needs to check the distance from the current coloring to *all* colorings from the archive (the `Forall` loop of Step 1, procedure Already-Visited in Algorithm 2). If the archive size exceeds a certain limit, the learning component execution time can become too long.

However, the processing of the archive may be tractable if we focus the learning component only the high quality configurations.

Definition 6. *(High-quality configuration) We say that configuration $C \in \Omega$ is high-quality if and only if $f_c(C) \leq B_f$, where B_f is a fitness boundary. Otherwise, we say that C is low-quality.*

The fitness boundary B_f is automatically set by PGTS so that the total number of iterations stays in the same order of magnitude as the number of distance computations. This proved to be a good "thumb rule" for obtaining an effective algorithm. To be specific, B_f directly controls the learning overhead because the whole learning component (Step 4.(c)) is now executed only if $f(C) < B_f$. In practice, B_f varies from 5 conflicts to 20 conflicts; for some problems we can even set $B_f = \infty$ and still obtain an acceptable speed. However, the algorithm automatically lowers and raises B_f according to the balance between the number of computed distances and the number of iterations.

4 Numerical Results

In this section, we show experimentally that the learning component helps the TS algorithm to obtain several colorings never found before by any other TS algorithm [17, 18, 19, 20, 24]. In fact, PGTS competes favorably with all existing local search algorithms.

4.1 Benchmark Graphs

We carry out the comparison only on the most difficult instances from the DIMACS Challenge Benchmark [25]: (i) $dsjc1000.1$, $dsjc1000.5$, $dsjc1000.9$, $dsjc500.5$ and $dsjc250.5$—classical random graphs [26] with unknown chromatic numbers (the first number is $|V|$ and the second denotes the density); (ii) $le450.25c$ and $le450.25d$—the most difficult "Leighton graphs" [27] with $\chi = 25$ (they have at least one clique of size χ); (iii) $flat300.28$ and $flat1000.76$—the most difficult "flat" graphs [28] with χ denoted by the last number (generated by partitioning the vertex set in χ classes, and by distributing the edges only between vertices of different classes); (iv) $R1000.1$—a geometric graph constructed by picking points uniformly at random in a square and by setting an edge between all pairs of vertices situated within a certain distance.

For each graph, we present the results using a number of colors k such that the instance (G, k) is very difficult for the basic TS; most of the unselected graphs are less challenging.

4.2 Experimental Procedure

Note that PGTS is equivalent to TS in the beginning of the exploration, while the archive is almost empty. The learning process intervenes in the exploration process only after several millions of iterations, as soon as the exploration process starts to run into already explored spheres. Therefore, if the basic TS is able to solve the problem quite quickly without any guidance, PGTS does not solve it more rapidly; the objective of PGTS is visible in the long run.

In Table 1, we perform comparative tests of TS and PGTS by launching 10 independent executions with time limit of 50 hours each. Within this time limit, PGTS re-initializes its search with a random k-coloring each time it reaches 40 million iterations. All these restarts share the same archive of spheres for PGTS.

Table 1. Comparison of PGTS and basic TS for a time limit of 50 hours. Columns 1 and 2 denote the instance, the success rate (Columns 3 and 5 respectively) is the number of successful execution series out of 10; the time column presents the average number of hours needed to solve the problem (if the success rate is not 0).

Instance		PGTS		Basic TS	
Graph	K	Success rate	Time [h]	Success rate	Time [h]
$dsjc250.5$	28	10/10	< 1	10/10	< 1
$dsjc500.5$	48	2/10	35	0/10	–
$dsjc1000.1$	20	2/10	9	0/10	–
$dsjc1000.5$	87	5/10	28	0/10	–
$dsjc1000.9$	224	8/10	24	2/10	44
$flat300\_28\_0$	29	7/10	8	0/10	–
$le450\_25c$	25	4/10	11	3/10	7
$le450\_25d$	25	2/10	19	2/10	12
$flat1000\_76\_0$	86	3/10	33	0/10	–
$r1000.1c$	98	10/10	< 1	10/10	< 1

To guarantee that the comparison is unbiased, we impose the same running time limit of 50 hours for both algorithms.[1]

Generally speaking, a PGTS iteration is more computationally-expensive than a TS iteration, and, consequently, TS can perform many more iterations for the same CPU time. However, the learning process accounts for an important performance gain: in many cases in which the basic TS fails (or has a very low success rate in finding a solution, see Table 1, Column 5), PGTS (Column 3) solves the problem.

Comparison with the Best Algorithms. Table 2 reports the best results obtained by PGTS on our graph set, along with a comparison with the basic TS and with the state-of-the-art algorithms. Note that many presented k-colorings were never reported before by other local search algorithm. Among all local searches that we are aware of, only VSS and PartialCol (columns 5 and 6)—two very recent algorithms using an evolved neighborhood function and a enhanced representation, respectively, compete effectively with PGTS.

Let us mention that in the literature on graph coloring, it is a common practice to run a local search algorithm for hours in order to (try to) solve large problems. For example, the most recent coloring algorithms [24,29] use running times of 10 hours for the largest instances. Another important point is that PGTS can continually explore new regions if it is given more computation time. Consequently, it is able to find better solutions by using the additional computational resources. Notice that this is a desirable characteristic which is not given by many existing algorithms. Very often, running them beyond some time (or iteration) threshold

[1] We used a 2.7GHz processor using the C++ programming language compiled $-O2$ optimization option under Linux. The source code is the same for both algorithms, the difference is only made by the learning component that is enabled for PGTS and disabled for TS.

Table 2. Comparison of the minimum number of colors for which a solution is found by: (i) the basic TS (Column 3), (ii) the new PGTS algorithm (Column 4) and (iii) state-of-the-art algorithms (Columns 5-10). Column 2 denotes the chromatic number (? if unknown) and the best k for which a legal coloring was ever reported in the literature. The colorings we report are publicly available on the Internet: www.info.univ-angers.fr/pub/porumbel/graphs/pgts/

Graph	χ, k^*	TS	PGTS	VSS	PCol	ACol	MOR	GH	MMT
				[29]	[24]	[30]	[31]	[20]	[32]
				2008	2008	2008	1993	1999	2008
$dsjc250.5$	?, 28	28	*28*	-	-	28	28	28	28
$dsjc500.5$	?, 48	49	*48*	48	48	48	49	48	48
$dsjc1000.1$	?, 20	21	*20*	20	20	20	21	20	20
$dsjc1000.5$	?, 83	88	87	88	88	84	88	83	83
$dsjc1000.9$	?, 224	224	*224*	224	225	224	226	224	226
$le450.25c$	25, 25	25	*25*	26	25	26	25	26	25
$le450.25d$	25, 25	25	*25*	26	25	26	25	26	25
$flat300.28$	28, 32	30	29	29	28	31	31	31	31
$flat1000.76$	76, 82	87	86	87	87	84	89	83	82
$r1000.1c$	?, 98	98	*98*	–	98	-	98	–	98

will not lead to better results simply because either the algorithms are trapped in deep local optima or because they re-explore again and again the same search space areas.

5 Discussion

Here, we discuss the properties of our new approach comparing to previous ones, and propose a generalization of PGTS to other combinatorial optimization problems.

5.1 Related Work

PGTS shares some basic ideas and objectives with the feature-based Guided Local Search [33] but our solving strategy is very different. We do not use explicit penalties and we do not need to identify specific solution features to penalize the evaluation function. In fact, we implicitly use a form of penalization (by encouraging the investigation unvisited regions) but at a higher level. Our method of avoiding certain regions is very targeted, in contrast with the penalty approach that might apply the same penalty (triggered by a situation in a particular region) to some very different and distant configurations.

A drawback of PGTS, when compared to the underlying basic TS, is that it might not sufficiently explore some spherical regions that are avoided after a first visit. This point could be completed by an algorithm that investigates only the interior of the spheres of the best recorded local minima. However, the new

algorithm is still competitive even with the best known algorithms (see columns 5-10 in Table 2) from the literature.

Compared to other local search algorithms for graph coloring, one can see that PGTS resorts to a more global view of the exploration. Most previous local search algorithms focused on local level improvements, i.e., they use more powerful neighborhood relations, alternative solution encodings, specific evaluation functions, etc. Generally speaking, there are numerous such problem-specific techniques able to increase the performance of a combinatorial optimization heuristic. However, we showed that, by focusing on global-level learning techniques, one can more easily overcome the limitations to which the local-level improvement potential is inevitably exposed.

5.2 Toward a Generalization for Other Combinatorial Problems

A careful examination of the code of Position Guided Tabu Search (see Algorithm 2) shows that it contains no particular references to the coloring problem. The only required components are: a search space, a neighborhood function, an objective function and a search space metric. The performance of PGTS depends on three factors: a fast procedure to compute the distance, a suitable choice of the spherical radius R, and a strategy to quickly search the archive.

Hence, as long as there exists a distance measure whose computation time does not significantly outweighs the computation time of a TS iteration, PGTS can be applied effectively to any combinatorial problem. This search space distance should express the minimal required number of neighborhood moves to arrive from a configuration to the other. Ideally, one should be able to group in a R-sphere of a local optimum only "equivalent" local optima—i.e., configurations sharing a common "backbone" substructure.

One can find several examples of easy–to–compute distances that can also be defined in this manner by using some specific neighborhoods:

- the Hamming distance for problems with array representation using the 1-Flip neighborhood (i.e., constraint satisfaction problems with a neighborhood operator that consists in changing the value of a single variable of the current configuration),
- the Kendall tau distance [34] for problems with permutation-based representation using a neighborhood defined by adjacent transpositions (i.e., the travelling salesman problem considering a neighborhood in which a move inverses two adjacent cities—the adjacent pairwise interchange neighborhood),
- the edit distance for problems with an array representation and with the neighborhood defined using edit operations.

Concerning the archive processing time, it can be substantially reduced in at least three ways: (i) by focusing on high-quality configurations, (ii) by increasing the value of the radius R and (iii) by transforming the archive into a queue that removes the oldest element at each insert operation. In the later case, the algorithm becomes a Double Tabu Search with two lists: (1) the traditional list of the last visited configurations that are forbidden, (2) the tabu list of spheres,

used to avoid revisiting spheres visited in the recent past. The significance of the expression "recent past" would depend on the size of the queue which should be tailored according to the learning component overhead.

6 Conclusions

We have presented a new local search algorithm that uses a learning process to guide the exploration process toward unvisited search space regions. It is possible to integrate this learning process in a classical tabu search with an acceptable overhead for all combinatorial optimization problems, provided that the distance computation is not too expensive. Moreover, the new algorithm does not necessarily introduce too many auxiliary user-provided parameters because the B_f value required in archive processing can be automatically set. The R value could be determined by calculating the distances between the local minima discovered during a classical search of the search space; for the graph coloring problem, we found that these local optima are typically grouped in clusters that can be confined in R-spheres with $R = 0.1|V|$. For other problems, R might be determined by finding the maximum distance between two configurations sharing an important backbone substructure.

This algorithm enabled us to improve the results of the basic TS for all graphs for which there is at least a different algorithm that ever reported better colorings than TS. Even compared to the best known algorithms from the literature (few of them local searches), PGTS proved to be very effective. Except very few graphs, it always finds the best known coloring. Moreover, in combination with another intensification algorithm, we found for the very first time a solution with 223 colors for the well-studied $dsjc1000.9$ graph.

Acknowledgments. This work is partially supported by the CPER project "Pôle Informatique Régional" (2000-2006) and the Régional Project MILES (2007-2009). The authors are grateful to the reviewers of the paper for their useful comments and questions.

References

1. Du, D., Pardalos, P.: Handbook of Combinatorial Optimization. Springer, Heidelberg (2007)
2. Kallel, L., Naudts, B., Reeves, C.: Properties of fitness functions and search landscapes. Theoretical Aspects of Evolutionary Computing, 175–206 (2001)
3. Merz, P.: Advanced fitness landscape analysis and the performance of memetic algorithms. Evolutionary Computation 12(3), 303–325 (2004)
4. Streeter, M., Smith, S.: How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines. Journal of Artificial Intelligence Research 26, 247–287 (2006)
5. Zhang, W.: Configuration landscape analysis and backbone guided local search. Part i satisfiability and maximum satisfiability. Artificial Intelligence 158(1), 1–26 (2004)

6. Gerber, M., Hansen, P., Hertz, A.: Local optima topology for the 3-SAT problem. Cahiers du GERAD G–98–68 (1998)
7. Ryan, J.: The depth and width of local minima in discrete solution spaces. Discrete Applied Mathematics 56(1), 75–82 (1995)
8. Hertz, A., Jaumard, B., de Aragão, M.: Local optima topology for the k-coloring problem. Discrete Applied Mathematics 49(1-3), 257–280 (1994)
9. Hamiez, J., Hao, J.: An analysis of solution properties of the graph coloring problem. In: Metaheuristics computer decision-making, pp. 325–345. Kluwer, Dordrecht (2004)
10. Culberson, J., Gent, I.: Frozen development in graph coloring. Theoretical Computer Science 265, 227–264 (2001)
11. Merz, P., Freisleben, B.: Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. Evolutionary Computation 8(1), 61–91 (2000)
12. Merz, P., Freisleben, B.: Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Transactions on Evolutionary Computation 4(4), 337–352 (2000)
13. Reeves, C., Yamada, T.: Genetic algorithms, path relinking, and the flowshop sequencing problem. Evolutionary Computation 6(1), 45–60 (1998)
14. Kuntz, P., Pinaud, B., Lehn, R.: Elements for the description of fitness landscapes associated with local operators for layered drawings of directed graphs. In: Metaheuristics Computer Decision-Making, pp. 405–420. Kluwer, Dordrecht (2004)
15. Battiti, R., Brunato, R., Mascia, F.: Reactive Search and Intelligent Optimization. Springer, Heidelberg (2008)
16. Glover, F., Laguna, M.: Tabu Search. Springer, Heidelberg (1997)
17. Hertz, A., Werra, D.: Using tabu search techniques for graph coloring. Computing 39(4), 345–351 (1987)
18. Dorne, R., Hao, J.: Tabu search for graph coloring, t-colorings and set t-colorings. In: Voss, S., et al. (eds.) Meta-Heuristics Advances and Trends in Local Search Paradigms for Optimization, pp. 77–92. Kluwer, Dordrecht (1998)
19. Dorne, R., Hao, J.: A new genetic local search algorithm for graph coloring. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 745–754. Springer, Heidelberg (1998)
20. Galinier, P., Hao, J.: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 3(4), 379–397 (1999)
21. Gusfield, D.: Partition-distance a problem and class of perfect graphs arising in clustering. Information Processing Letters 82(3), 159–164 (2002)
22. Glass, C., Pruegel-Bennett, A.: A polynomially searchable exponential neighbourhood for graph colouring. Journal of the Operational Research Society 56(3), 324–330 (2005)
23. Porumbel, C., Hao, J., Kuntz, P.: An efficient algorithm for computing the partition distance. Submitted, available on request (2008)
24. Blöchliger, I., Zufferey, N.: A graph coloring heuristic using partial solutions and a reactive tabu scheme. Computers and Operations Research 35(3), 960–975 (2008)
25. Johnson, D., Trick, M.: Cliques, Coloring, and Satisfiability Second DIMACS Implementation Challenge. DIMACS series in Discrete Mathematics and Theoretical Computer Science, vol. 26. American Mathematical Society (1996)
26. Johnson, D., Aragon, C., McGeoch, L., Schevon, C.: Optimization by simulated annealing an experimental evaluation; part ii, graph coloring and number partitioning. Operations Research 39(3), 378–406 (1991)
27. Leighton, F.: A graph coloring algorithm for large scheduling problems. Journal of Research of the National Bureau of Standards 84(6), 489–503 (1979)

28. Culberson, J., Luo, F.: Exploring the k-colorable landscape with iterated greedy. In: [25], pp. 345–284
29. Hertz, A., Plumettaz, A., Zufferey, N.: Variable space search for graph coloring. Discrete Applied Mathematics 156(13), 2551–2560 (2008)
30. Galinier, P., Hertz, A., Zufferey, N.: An adaptive memory algorithm for the k-coloring problem. Discrete Applied Mathematics 156(2), 267–279 (2008)
31. Morgenstern, C.: Distributed coloration neighborhood search. In: [25], pp. 335–358
32. Malaguti, E., Monaci, M., Toth, P.: A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing 20(2), 302 (2008)
33. Voudouris, C., Tsang, E.: Guided local search. In: Glover, F., et al. (eds.) Handbook of metaheuristics, pp. 185–218. Kluwer Academic Publishers, Dordrecht (2003)
34. Kendall, M.: A new measure of rank correlation. Biometrika 30(1/2), 81–93 (1938)