J Math Model Algor (2008) 7:125–142
DOI 10.1007/s10852-008-9081-1

A Cooperative Search Method for the *k*-Coloring Problem

Hend Bouziri · El-Ghazali Talbi · Khaled Mellouli

Received: 1 March 2007 / Accepted: 21 December 2007 /
Published online: 6 March 2008

Abstract In this paper, a cooperative search method, based on a multi-agent structure is developed to deal with the *k*-coloring problem. Three agents coordinate using an adaptive memory, a search agent, an intensification agent and a diversification agent. We use the results of a preliminary fitness landscape study to adjust the navigation strategy in the solution space and to fix the search parameters. Our method provides competitive results and it is fast when compared with best existing techniques on instances extracted from the second DIMACS challenge.

Keywords *k*-Coloring · Fitness landscape · Cooperative method · Iterative search

Mathematics Subject Classifications (2000) 05C15 · 90C27

1 Introduction

The graph coloring problem has many applications in scheduling, timetabling and telecommunication. This problem and its various extensions are known to be NP-hard. For this reason, exact algorithms can't solve instances with more than 100 vertices. This motivates the use of metaheuristic methods.

H. Bouziri (✉)
LARODEC-ISG, ESSEC, Tunis, Tunisia
e-mail: hend.bouziri@gnet.tn

K. Mellouli
LARODEC-ISG, IHEC, Carthage, Tunisia
e-mail: khaled.mellouli@ihec.rnu.tn

E.-G. Talbi
LIFL, University of Lille 1, CNRS, INRIA, Lille, France
e-mail: el-ghazali.talbi@lifl.fr

Many iterative search techniques have been developed for the graph coloring problem. The main drawback of most of them, is that they are elaborated without a specific knowledge of the search space structure. This should explain, why these methods are efficient on some instances but provide poor results on others.

In this work, we attempt to analyse the landscape structure of the k-coloring problem before developing a resolution search method. This method learns from the landscape structure and re-uses best features in existing methods to be more effective.

In the next section, we give the formulation of the problem. In Section 3, we present some of the techniques already proposed for coloring graphs. Section 4 present the cooperative architecture of our new method. Section 5 is devoted to a preliminary fitness landscape study of different instances. In Section 6, we provide the implementation of the method according to the results of the landscape analysis. Results of the resolution of some instances of the second DIMACS challenge[1] are reported in Section 7.

2 k-Coloring Problem

Given an undirected connected graph $G = (V, E)$, the coloring of G which assigns k colors to the vertices of the graph is termed *k-coloring*. Thus a k-coloring partitions the vertices of a graph into k color classes. A *proper* coloring of a graph G, is an assignment of colors to its vertices so that no two adjacent vertices in G have the same color.

If two adjacent vertices v and w have the same color, then vertices v and w are called *conflicting vertices* and the edge linking v with w is called a *conflicting edge*. The chromatic number of a graph G is the minimum number χ for which G has a χ-coloring.

Furthermore, the k-coloring problem can be seen as a decision problem, the question to be answered is whether for some given k, a legal k-coloring exists. It is well known that the k-coloring problem for general graphs is NP-complete and that the graph coloring problem is NP-hard; only for a few special cases polynomial time algorithms are known.

3 Resolution Techniques

Too many methods are proposed for the resolution of different variants of the graph coloring problem. This is mainly due to its practical importance. We distinguish between greedy methods and iterative search methods. In this section, we focus our survey on well used methods known to be efficient in the resolution of several instances of the graph coloring problem.

[1] http://mat.gsia.cmu.edu/COLOR/instances.html.

3.1 Greedy Methods

The general idea of a greedy algorithm is to build up a solution, one element at a time, by choosing the best possible element at each iteration. To deal with the graph coloring problem, the algorithm starts with the creation of a permutation of vertices, then the first vertex in the permutation is colored with the smallest color from $\{1, .., k+1\}$, such that no conflict is created. k is the last color used and in the first step it is equal to 0. Once a vertex is colored, it is removed from the set of vertices. These two steps are repeated with the remaining set of uncolored vertices. Several different schemes have been used for the initial ordering.

Work on constructive methods still takes a great interest since search methods are time consuming. Indeed, in many practical applications, it is preferable to have a "satisfactory" solution with a short execution time than to wait in order to reach a global solution. In addition, constructive methods such as XRLF or DSATUR are very recommended as candidates to generate initial colorings in iterative search methods. In fact, they enable the search process to begin from good and fast constructed colorings.

3.1.1 DSATUR

Brélaz [2] proposed a technique that re-orders vertices at each stage, according to their saturation degree. He defines the saturation degree of a given vertex v as the number of colors already used by its adjacent vertices. Initially, the vertices are sorted by decreasing degree, and all saturations are zero, then the following routine is repeated:

1. Find the node with the highest saturation degree and assign to it the smallest color.
2. In case of a tie, the node with the highest degree (=number of neighbors that are still in the uncolored subgraph) is chosen.
3. In case of a tie, a random node is chosen.

The saturation degree is calculated dynamically as the coloring is constructed. Also, because of the random tie breaking, DSATUR is considered as a stochastic algorithm. Brélaz shows that for bipartite graphs, this method is optimal.

3.1.2 Randomized Recursive Larger First

Recursive larger first method (RLF) is based on the notion of independent sets. An independent set is the set of vertices $U \in V$ such that no two vertices of U are adjacent. The priority is assigned to independent subsets that contain the maximum number of vertices. At each iteration, the maximum independent subset is identified, and each vertex is colored by the same color, say c. In the next iteration, this subset is removed from the set of vertices then another maximum independent subset is defined and colored with the color $c+1$. The number of maximal independent subsets corresponds to the number of colors used.

XRLF is a randomized version of RLF, developed by Johnson, et al. [15]. It combines an exhaustive search with a variant of RLF algorithm. It consists of

extracting independent sets until the residual graph has less than 70 vertices, making it possible to be colored with an exact coloring algorithm.

3.2 Iterative Search Methods

Metaheuristics can operate on complete or partial solutions which can violate constraints, they use a neighborhood operator to "navigate" from a solution to another one and they stop if a satisfactory solution is found or a stopping criterion is satisfied. Various techniques have been proposed to prevent the stagnation in local optima.

These iterative search methods have proved their efficiency in the resolution of several instances of problems classified NP-hard. Three main metaheuristics are proposed to deal with the k-coloring problem: simulated annealing (SA), tabu search (TS) and genetic algorithm (GA). In the following, we present basic hybrids of metaheuristics operating on single solutions or on a population of solutions.

3.2.1 SA Combined to XRLF

In the simulated annealing procedure proposed by Chams et al. [3], both pure simulated annealing and an approach that combines XRLF and simulated annealing are discussed. In the second method, XRLF is used to construct color classes until the number of vertices in the graph is below a specified level. Simulated annealing is then applied to color the remaining vertices.

3.2.2 Distributed Coloration Neighborhood Search

Morgenstern [18] explores another strategy which starts by an initial population of proper partial k-colorings generated by XRLF. The objective is to maximize the number of colored vertices. A simple simulated annealing scheme is applied to move between partial solutions.

The neighborhood operator consists in assigning a color (say c_i) to an uncolored vertex (say v) and to remove the color on each adjacent node of the vertex v that has the color c_i, this type of move is called i-swap. The SA algorithm is run for a predefined maximal number of iterations or until a certain solution quality threshold is passed.

3.2.3 TabuCol

Herz and De Werra [12] proposed a procedure wherein a solution of a k-coloring problem is encoded by a string s of size N (number of nodes), where $s[i] \in \{1, ..., k\}$ corresponds to the color of the node v_i. The goal is to minimize the number of edges connecting two nodes having the same color, a solution is then evaluated by:

$$f(s) = \sum_{j=1}^{k} |E_j|, \tag{1}$$

where E_j is the set of edges of the graph G having both endpoints in V_j, with V_j is the set of nodes having the color c_j, $j \in \{1, ..., k\}$.

The tabu search procedure started by assigning k colors randomly to the n vertices, if this constitutes a proper k-coloring then the search process is restarted with $(k-1)$ colors.

A neighboring solution s' of a solution s is obtained by coloring a conflicting node v_i with a color c_j different from its current color c_i. The tabu algorithm consists in generating a set of such neighbors, and then selecting the neighbor with the fewest conflicts. After a move of vertex v from a color class V_i to V_j, a tabu activation rule forbids the move that returns v to V_i for a number of iterations called *tabu tenure*. The procedure stops when $f(s) = 0$ or if a pre-specified number of iterations is reached without finding a legal coloring for a given k.

The algorithm provides very interesting results even for large problems. Experiments show that it outperforms simulated annealing on several instances.

3.2.4 EA with a Knowledge-Augmented Crossover

In the implementation of Ferland and Fleurent [6] of the evolutionary algorithm, a knowledge-augmented crossover operator is used. It consists of defining a set of conflicting nodes. An offspring from two parents is generated by coloring the node in one parent using the color supplied by the other parent if it is conflict free for the later.

When a node is in conflict in both parents, the algorithm selects the least frequently used color among adjacent nodes in either parent. Each of the remaining nodes is colored using colors of one of the parents selected randomly as in the uniform crossover. A mutation operator is applied to each generated individual.

Ferland and Fleurent also use the tabu search algorithm as a mutation operator in the genetic algorithm. Experiments show that this genetic hybrid performs slightly better than the multi-start tabu search.

Moreover, the authors propose to extract stable sets by a modified version of Stabulus (defined by Friden et al. in [7]) with a different objective function, then to run TabuCol to color the residual graph. This technique improves clearly the results on random large graphs.

3.2.5 Greedy Randomized Adaptive Search Procedure

This method (GRASP for short), proposed by Laguna and Marti [17], consists of two phases: construction phase and improvement phase. In the construction phase, the GRASP uses a randomized version of the RLF to construct initial colorings and uses the objective function to rank all candidate solutions. The best coloring is then selected.

The improvement step consists of a local search procedure. Its input is the graph $G = (V, E)$ and a valid c-coloring $V_1, .., V_c$. The output is the k-coloring $V_1, .., V_k$ with $k \leq c$. The goal is to reduce the size of the current coloring. GRASP repeats these two steps and finally returns the best solution encountered.

Experiments show that if the amount of time in the improvement time phase is limited, metaheuristics (namely tabu search and simulated annealing) are not able significantly to improve the performance of the overall method when compared with simple descent mechanisms.

3.2.6 Hybrid Evolutionary Algorithm

This algorithm (HEA for short) is proposed by Galinier and Hao [8]. It begins with creating an initial population of solutions. A local search operator is then applied to each individual of the population for a given number of iterations. At each generation, two parents are selected to perform a specialized crossover operator called GPX (Greedy Partition Crossover), the generated child is improved by the use of the local search again. The population is updated by including this new individual in a replacement phase. This procedure is repeated until a stopping criterion is satisfied.

Results show that HEA outperforms previous methods for many instances of the DIMACS challenge for the graph coloring problem. For all the tested instances HEA is faster in finding the colorings.

3.2.7 An Adaptive Memory Algorithm

To follow the crossover idea proposed in HEA, Galinier proposed, later in [9], an adaptive memory algorithm (AMACOL). It is a hybrid evolutionary heuristic that used a central memory containing pieces of solutions. During each generation, the algorithm creates an offspring solution s, applies a local search operator on s and updates the central memory with the resulting solution. In the context of k-coloring, stable sets corresponds to pieces of solutions. The results are comparable to those obtained by the HEA technique.

4 A New Cooperative Search Method

We have concluded from the survey of main existing approaches proposed for coloring graphes, the following observations:

- Hybrids that combine evolutionary and local search are the best among iterative search methods in the resolution of the graph coloring problems. The experiments, on several schemes of hybrids, showed the benefits of working on a population of solutions and evolving with it during the search process. Furthermore, the use of an appropriate crossover enhances considerably the efficiency of the *evolution*. This is basically the contribution of Galinier in his HEA method.
- Constructive methods such as XRLF or DSATUR are very recommended as candidates to generate initial colorings in iterative search methods, especially DSATUR. In fact, they enable the search process to begin from good and fast constructed colorings. Experiments on several methods show that using DSATUR to generate initial solutions can improve the search process in time and quality.
- Every search process is an alternate between diversification and intensification. This differs according to the considered problem and even to the treated instance. For this, we consider that, the architecture proposed in [1] is a good illustration of this alternate. Thus an iterative search process involves three agents that interact using a memory to store the history of the search process.

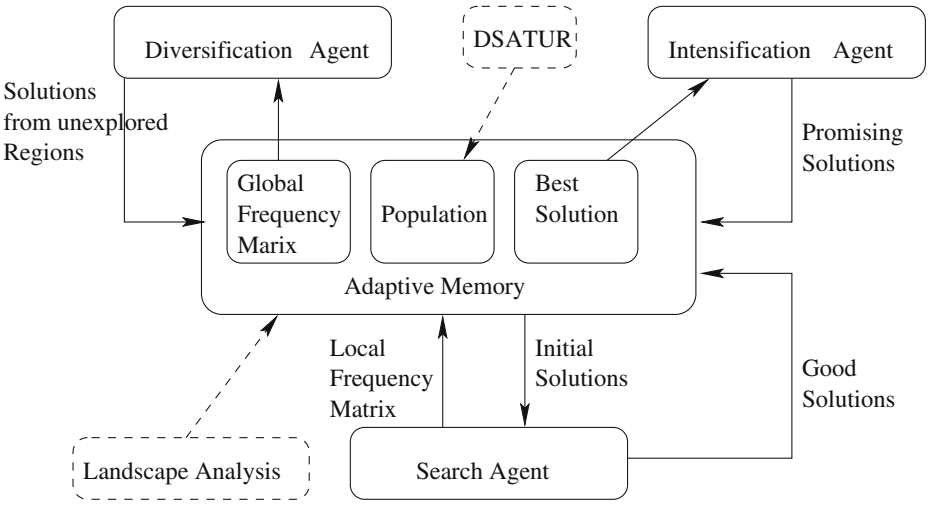

Fig. 1 A cooperative search method for the graph coloring problem

Consequently, we will focus our study on the elaboration of a cooperative search method based (CSM for short) operating on a multi-agents structure for the resolution of the k-coloring problem. It is composed of three agents: the intensification agent, the search agent and the diversification agent. To implement the agents, we have adapted existing methods that have proved their efficiency in previous studies on many hard instances of the graph coloring problem.

The chosen cooperative multi-agents architecture, called CO-SEARCH, is proposed originally by Bachelet and Talbi [1] for the quadratic assignment problem. Weinberg [19] re-used this model to solve the frequency assignment problem modelled as a set T-coloring problem.

The three agents interact using an adaptive memory as it is illustrated by Fig. 1. Since in a search technique, our purpose is to *navigate* efficiently across the different types of relief to reach the *lowest* point, we claim that it is essential to know which *structure* we will navigate, otherwise, the search will be done *blindly*. So, before fixing our *navigation strategy*, we perform a preliminary analysis of the k-coloring fitness landscape. This is precisely the goal of the next section.

5 A Preliminary Fitness Landscape Analysis

The fitness landscape is basically defined by three parameters:

1. The representative space corresponding to the set of points that our cooperative search method (CSM) tries to visit. These points correspond to the space of k-colorings, given k the number of colors and a graph G. To avoid the *solution symmetry* in the search, we consider one solution as the set of colorings generating the same partition of vertices.
2. The neighboring operator that allows the move from a point to another one. In our investigation of the graph coloring landscape, we use the neighborhood

operator that changes a color of a node with another one. Thus, at each move a node is removed from a class color to be assigned to another one.

3. The fitness function which assigns a value to each point of the search space [16]. The goal of the search process, in our study, is to obtain a *proper* coloring, given a graph G and k colors to be assigned to each node. Thus the fitness function is given by the formula (1). The value of the fitness at each point corresponds to its altitude in the landscape.

Many attempts have already been performed to analyse the search space of the graph coloring problem such as the work of Hertz et al. [14]. In our analysis, we use some statistical tools to provide an idea as clear as possible of the structure of the k-coloring landscape.

5.1 Experimental Setup

Three families of graphs were chosen for our computational testing: Leighton graphs, flat graphs and random graphs. The instances are extracted from the second DIMACS benchmark challenge.

Columns 1 to 5, in Table 1, show for each studied graph, respectively, its name, the number of vertices, the number of edges, its chromatic number (or its best known lower bound when the chromatic number is unknown) [5] and the best coloring found in the literature.

Table 1 The experimental protocol

Instances	Nodes	Edges	χ values	Best k
DSJC125-1	125	1,472	≤ 5	5
DSJC125-5	125	7,782	≤ 10	17
DSJC250-9	125	13,922	≤ 30	44
DSJC250-1	250	64,336	≤ 8	8
DSJC250-5	250	31,336	≤ 11	28
DSJC250-9	250	55,794	≤ 35	72
DSJC500-1	500	24,916	≤ 12	6
DSJC500-5	500	125,248	≤ 16	49
DSJC500-9	500	224,874	≤ 42	127
flat300-28-0	300	21,695	28	31
le450-5a	450	5,714	5	5
le450-5b	450	5,734	5	5
le450-5c	450	9,803	5	5
le450-5d	450	9,757	5	5
le450-15a	450	8,168	15	15
le450-15b	450	8,169	15	15
le450-15c	450	16,680	15	15
le450-15d	450	16,750	15	15
le450-25a	450	8,260	25	25
le450-25b	450	8,263	25	25
le450-25c	450	17,343	25	25
le450-25d	450	17,425	25	25

For each graph G of Table 1, we define an instance by the pair (G, k), where k corresponds to column 5 of the table (the number of colors used to solve the k-coloring problem). We start experiments by the generation of an initial population of random solutions for each instance . Then we apply a rapid descent method on each individual to get a local population. Experiments are performed on 100 solutions generated randomly and on the 100 corresponding local solutions. At each step, we choose the best solution among neighboring ones.

Furthermore, we perform a correlation structure study to analyze the trajectories on the fitness landscapes associated to different instances. This analysis starts by the generation of a random point (a random k-coloring of the graph). The walk is then performed by the application of the neighborhood operator at each step. During this process, the fitness of the resulting points are computed and recorded.

5.2 Fitness Diversity

The fitness of a solution (a k-coloring) corresponds to its altitude in the landscape. Then we can study the "relief" of the fitness landscape by measuring the fitness diversity. The coefficient of variation cv is often used in statistic to compare the variability of populations in term of dispersions.

The *sample coefficient of variation* is the sample standard deviation (s_f) divided by the sample mean ($\bar{f}$), where n is the sample (population) size:

$$cv = \frac{s_f}{\bar{f}},$$

with $s_f = \sqrt{\frac{\sum_{i=1}^{n}(f_i - \bar{f})^2}{n-1}}$ and $\bar{f} = \frac{\sum_{i=1}^{n} f_i}{n}$. In these formula, f_i corresponds to the fitness of the k-coloring c_i. In the context of the fitness landscape, the coefficient of variation should compare the dispersion of "fitnesses" relatives to different instances. For a given instance (which corresponds to a given landscape), more the cv is near zero, less the fitnesses are dispersed.

In Fig. 2, we illustrate the coefficient of variation of local solutions relative to the instances of Table 1. We can see that almost all coefficients of variation are

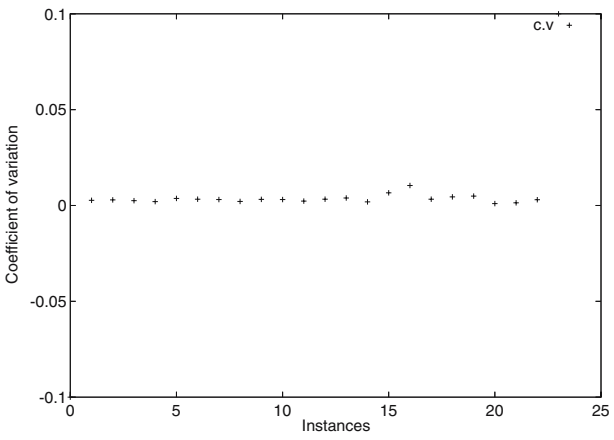

Fig. 2 Coefficient of variation $c.v.$ of different instances

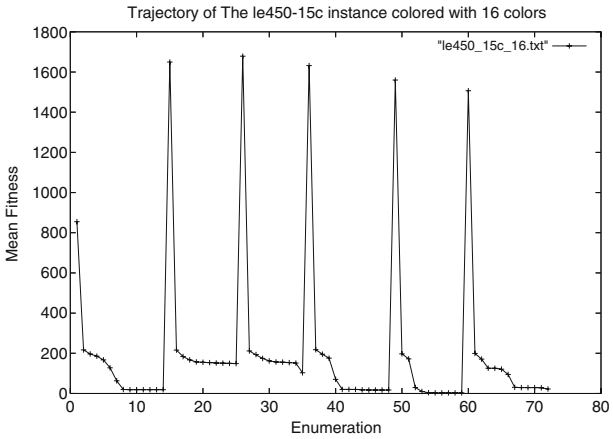

Fig. 3 The trajectory of the multi-starting tabu search algorithm in coloring le450-15c graph with 16 colors

near zero. This shows that for all the tested instances, the fitness are gathered near their mean. This should be interpreted by the fact that locally, optimal solutions (not necessarily proper ones) have the same fitness. We can conclude then that the landscape structure of the graph coloring problem can be seen as a set of plateaux.

This kind of landscape confirms the importance of re-starting from different points when dealing with coloring problems. Thus, it is essential to our method to contain a diversification tool to *escape* to other more promising regions in the landscape.

Indeed, many authors such as in [10], noticed that the objective function decreases dramatically in the early stages of the search and the search procedure generally uses most of its time trying to eliminate the last conflicting nodes.

Figure 3 shows that when coloring the le450-15c graph for example, the search drops rapidly and stagnates. It requires four restarting points to reach the region where an optimal solution is detected. It is clear that if we omit the re-starting procedure, the search will stagnate for long time in the same region.

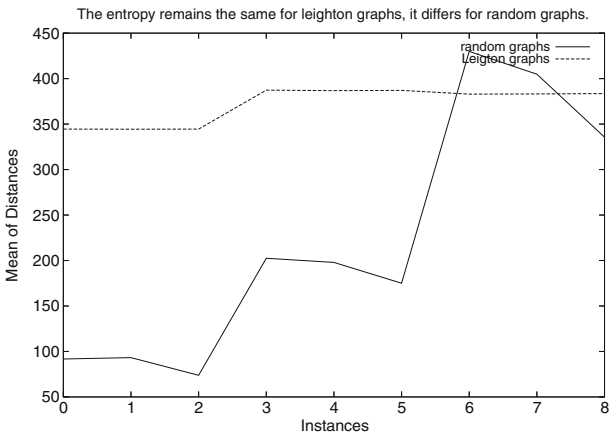

Fig. 4 The mean of distances variation for random and Leighton graphs

5.3 Coloring Diversity

A solution differs from another one by its partitioning of nodes into color classes. Thus, to measure the diversity of solutions (colorings) in the landscape, we use the mean of distances $\bar{d}$ between all possible pairs of colorings $d(c_i, c_j)$ in the landscape. Given n the number of individuals in a population P, $\bar{d}$ is given by:

$$\bar{d} = \frac{2}{n(n-1)} \sum_{i=1} \sum_{j<i} d\left(c_i, c_j\right).$$

Where the distance d between colorings c_1 and c_2 is the minimal number of node moves between color classes needed to obtain c_2 from c_1.

Results reveal, as Fig. 4 shows, that the mean of distances remains basically the same for Leighton graphs and changes with instances for random graphs. This can be interpreted by the fact that for Leighton graphs (which are structured) same sets of nodes seem to be always together (frozen same sets) and distances between solutions remain the same.

Hamiez and Hao [11] and Culberson [4] showed the existence of a particular set of vertices, that are always in the same color class, when solutions are generated. This set is called the *frozen same set*. Culberson [4] mentioned that frozen same sets can be detected rapidly by greedy algorithms. This insights us to use random initial solutions when dealing with random graphs and to use greedy methods when we want to generate initial colorings for Leighton graphs.

To confirm these assumptions, we compare the behavior of the search by the use of the DSATUR method as initial solution generating method and its behavior when using a random initial population. We perform the tests on a Leighton graph (le450-5c) and two random graphs (DSJC125-5 and DSJC125-9).

Figure 5 clearly shows that the use of the DSATUR as initial population generation method improves the results of the search agent. However, for random graphs in Fig. 6, we obtain similar results when using DSATUR or generating randomly initial solutions.

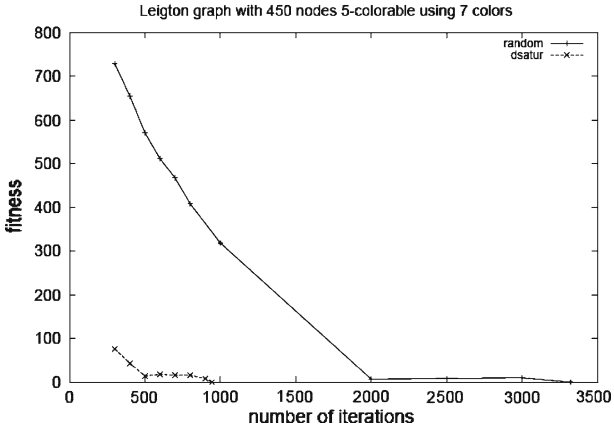

Fig. 5 The comparison of the algorithm trajectory when using the DSATUR or random generation of initial solutions for the le450-5c graph

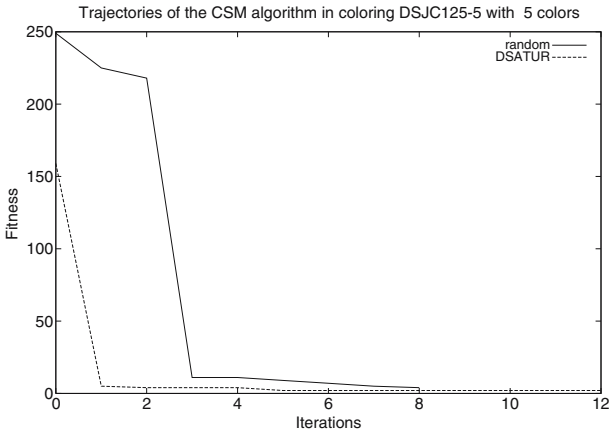

Fig. 6 The comparison of the algorithm trajectory when using the DSATUR or random generation of initial solutions for random graphs DSJC125-5

In this sense, many authors, as in [10], noticed that the use of a greedy method to generate initial solutions can be avoided since, in many cases, random initial solutions lead to similar results in time and qualities. But they do not explain this behavior; nor for which graphs it is true. They just concluded it by experiments.

5.4 Search Walk Length

Hordijk [13] defines the correlation length as the largest time lag i for which the correlation between two points i steps apart is still statistically significant. That is the quicker the correlogram drops to zero, the less the correlation length is. Table 2 shows the correlation length for some k-coloring instances. More the correlation length is high, the flatter the landscape is. This means that it needs longer search walk to be explored.

Table 2 Correlation length of some instances

Graph	Correlation length
le450-5c	130
le450-15c	214
le450-25a	838
le450-25b	396
le450-25c	249
flat300-28-0	202
DSJC125-1	65
DSJC125-5	151
DSJC125-9	370
DSJC250-5	678
DSJC250-9	150
DSJC500-1	452
DSJC500-5	847

6 The Implementation of the Cooperative Search Method

Now that the general structure of the k-coloring landscape is defined, we can derive how the method defined in Fig. 1 has to work.

6.1 The Global Navigation Strategy

The steps followed by our cooperative search method are illustrated by the Algorithm 1. If the graph is structured, we begin by a modified version of DSATUR. The version is modified to provide a complete coloring even if it isn't proper. However, if the graph is random, the initial population of solutions is generated randomly.

Algorithm 1 The cooperative search method

Input: A graph G=(N,E), k the number of colors.
$pop_i \longleftarrow$ constructed either randomly or by DSATUR
$S^* \longleftarrow$ the best individual in pop_i
while the stopping criterion is not satisfied **do**
  if Stagnation **then**
    $pop_i \longleftarrow$ Diversification(pop_v)
  end if
  $pop_v \longleftarrow$ Search(pop_i)
  if $\bar{d}(popv)$ is high **then**
    Intensification(pop_v)
  end if
  $SV \longleftarrow$ the best individual in pop_v
  if fitness (SV) $\leq$ fitness (S^*) **then**
    $S^* \longleftarrow SV$
  end if
  $pop_i \longleftarrow pop_v$
end while
Return(S^*)

When the search agent leads to the same quality population for a given number of iterations (we reach a stagnation phase), we have to restart the navigation process in an other region, this is precisely the diversification agent mission.

At each iteration, a local population is provided by the use of the search agent. If the members of the population are well diversified ($\bar{d}$ is high), we recombine the qualities of the current population to improve the search agent work. This constitutes the intensification agent mission.

Once the global search strategy is defined, we can specify in detail the role of each component of our cooperative search method presented in Fig. 1.

6.2 The Adaptive Memory

The adaptive memory plays an important role in coordinating the different agents of the model. It stores information that the method requires to navigate efficiently in the landscape. This memory contains the following information.

- The constraint matrix which is consulted to see whether two nodes are connected by an edge.
- The *frequency matrix* F which preserves the most frequently visited regions. Each cell $F(i, j)$ contains the number of times the node i and the node j are assigned to the same color class. When the search process starts, all the cells of the frequency matrix are initialized to zero. Then, each time a node i is assigned to a color class c, every cell $F(i, j)$ is incremented by one if j have also the color c. A such implementation of the frequency matrix can avoid symmetrical regions that represent same partitions of nodes.
- The mean of distances of the current population is computed dynamically to measure its diversity. Also, the fitnesses are stored in successive iterations to detect stagnation phases.

6.3 The Search Agent

We choose to use the tabu search technique as search agent. This choice is motivated by two factors. The first one, is that this technique uses an efficient local diversification tool, the tabu list, that enables the search to be guided in a rugged structure. The second factor, is that the method is simple to be implemented and shows good performance when tested in several works [8–10, 12].

The neighborhood operator consists of moving a node from a color class to another one without checking if the node is in conflict or not. For the implementation of the tabu list, we use a matrix of $N \times k$, where N is the number of nodes in the graph and k the number of color classes. When the search procedure starts, all the cells of the matrix are initialized to zero. Each time a node i is colored with the color j in the tabu search, the cell (i, j) in the matrix is incremented by one. When the value of the cell (i, j) reaches the size of the tabu list, it is set to zero.

The search agent explores the local region for a maximal number of iterations corresponding to the walk length L, which depends on the correlation length of the corresponding instance (as it was mentioned in Section 5.4). By experimentation, we fix L as:

$$L \simeq correlationlength \times 100.$$

The aspiration criterion in our implementation is simple, it consists on suspending the tabu status of a move if it leads to a coloring better than the best one encountred during the search process.

6.4 The Diversification Agent

In the preliminary landscape study, we have concluded that the common point between k-coloring landscapes is that they are composed of rugged plateaux.

For this reason, we need a tool to leave current plateau, each time we find that the search is *stagnated* without improving the results.

The role of the diversification agent is to lead the search to reach unexplored plateaux. The problem here is that the good qualities found till now, will be lost if we choose to restart the search process randomly, thus, we say that the aim of the diversification agent, is *to diversify not to randomize*. Consequently, we choose to preserve the maximal sized color classes, in the best solutions.

To complete the construction of the new starting solutions, the agent consults the frequency matrix F. Each uncolored node i (not affected to the maintained partitions) is assigned to the same partition as the node j such that $F(i, j)$ is minimal, where $j \neq i$.

6.5 The Intensification Agent

This agent is implicated to improve local solutions found by the search agent. If the region is promised (high mean distance with near optimal solution), we intensify the search process by the use of a genetic procedure. We use a *standard* genetic schema.

At each generation, the genetic algorithm uses a probabilistic selection to choose the two parents. The GPX crossover operator is used to provide a *child* which will replace the worst individual in the population. The goal is to preserve good qualities in parent solutions. The evolutionary process operates on fixed sized population and stops when a "homogenous" population is reached.

7 Experiments and Result Analysis

In this section, we report the results provided by our cooperative search method (CSM) on some of the instances already studied in Section 5.

Results are gathered in Table 3. For a given instance, column 1 to 4, correspond, respectively to the name of the graph, the chromatic number (the lower bound, if the chromatic number is unknown), the number of used colors k, the number of iterations of the CSM algorithm (see Algorithm 1).

Because of the random feature of our cooperative search method, we ran it for five times for each k-coloring instance and we report, in column 5 of the table, the fitness of the best coloring. We note that the number of iterations that requires our method is small, despite the combinatorial feature of the problem.

The comparison of our results with the best ones is reported in Table 4. For a given graph, column 1 to 4, correspond, respectively to the name of the graph, the lower number of colors used in the literature, the number of the colors provided by HEA [8] and the number of colors provided by our method CSM.

The reported results show clearly that our method provides colorings that are competitive with the best existing methods. In fact, for example, Galinier needed 490,000 iterations to color DSJC250-5 with 28 colors. However, CSM needed 1,000 iterations to color the same graph with 29 colors at a cost of only two conflicting edges. In the case of Leighton graphs, CSM colored the graph le450-25c with 26 colors after only 40 iterations and HEA needed 800,000 iterations to color the same graph with the same number of colors. We compare our results with those of Galinier,

Table 3 Results of the cooperative search method

Graph	χ values	k values	*#iter*	Best f
le450-5c	5	5	100	0
le450-15c	15	15	100	76
		16	68	0
le450-25a	25	25	6	0
le450-25b	25	25	12	0
le450-25c	25	25	1,000	2
		26	40	0
flat300-28-0	28	31	1,000	11
		32	20	0
DSJC125-1	5	5	160	0
DSJC125-5	12	16	50	9
		17	100	0
DSJC125-9	30	42	200	2
		43	100	1
		44	2	0
DSJC250-5	13	29	1,000	2
		30	50	0
DSJC250-9	35	71	1,000	2
		72	100	0
DSJC500-1	6	12	100	25
		13	75	3
		14	2	0
DSJC500-5	16	49	100	25
		50	100	6
		51	5	0

because he developed the most recently efficient method. Also, his method is a hybrid local search. The reported results show clearly our method provides colorings that are competitive with the best existing methods.

Furthermore, we see that they are very close to outperform best colorings on some instances.

Table 4 Comparison of the results of the cooperative search method with the best known ones

Graph	Best k	HEA	CSM
le450-5c	5	5	5
le450-15c	15	15	16
le450-25a	25	–	25
le450-25b	25	–	25
le450-25c	25	26	26
flat300-28-0	31	31	32
DSJC125-1	5	–	5
DSJC125-5	17	–	17
DSJC125-9	44	–	44
DSJC250-5	28	–	30
DSJC250-9	72	–	72
DSJC500-1	12	–	14
DSJC500-5	48	49	51

8 Conclusion

In this paper, we have developed a new cooperative search method to solve the *k*-coloring problem. It is based on the cooperation between three agents: the search agent, the intensification agent and the diversification agent. Results on several instances extracted from the second DIMACS challenge, are competitive with the best known ones and are obtained in lower time. This effectiveness can be explained by three factors.

1. The cooperative structure, since to find a solution in a search space, we need to alternate our search strategy between looking for good regions, intensifying the search if the region is promising and diversifying the search if we are blocked in the same region. In addition, CSM operates on a population of *k*-colorings.
2. Avoiding solution symmetry either in the study of the fitness landscape and in the implementation of diversification structures, namely the frequency matrix and the tabu tenure.
3. The preliminary fitness landscape study helped us in determining the global navigation strategy, in the generation of initial solutions and in fixing the search length.

As perspectives, the robustness and the effectiveness of our approach motivate the use of similar approaches to deal with other combinatorial optimization problems. The agents will be chosen according to specified problem and the navigation strategy has to be adapted to the corresponding fitness landscape structure. Another promising search axis in this field is to implement the cooperative multi agents structure on a distributed system.

References

1. Bachelet, V.: Métaheuristiques parallèles hybrides: application au problème d'affectation quadratique. PhD Thesis, Université des Sciences et Technologies de Lille, France, Janvier (2000)
2. Brélaz, D.: New methods to color vertices of a graph. Commun. ACM **22**, 251–256 (1979)
3. Chams, M., Hertz, A., de Werra, D.: Some experiments with simulated annealing for coloring graphs. Eur. J. Oper. Res. **32**(2), 260–266 (1987)
4. Culberson, J.: Frozen development in graph coloring. Technical Report APES-19-2000, APES Research Group (2000) February 1
5. Desrosiers, C., Galinier, P., Hertz, A.: Efficient Algorithms for Finding Critical Subgraphs. Les Cahiers de GEARD G-2004-31 (2004) April
6. Fleurent, C., Ferland, J.A.: Genetic and hybrid algorithms for graph coloring. Ann. Oper. Res. **63**, 437–461 (1995)
7. Friden, C., Hertz, A., de Werra, D.: Stabulus, a technique for finding stable sets in large graphs with tabu search. Computing **42**, 35–44 (1989)
8. Galinier, P., Hao, J.K.: Hybrid evolutionary algorithm for graph coloring. J. Comb. Optim. **3**(4), 379–397 (1999)
9. Galinier, P., Hertz, A., Zufferey, N.: An adaptive memory algorithm for the *k*-colouring problem. Les cahiers de GERAD G-2003-35, GERAD, Montréal (2003)
10. Galinier, P., Hertz, A.: A survey of local search methods for graph coloring. Les cahiers de GIRAD G-2004-32, GERAD, Montréal (2004)
11. Hamiez, J.P., Hao, J.K: An analysis of solution properties of the graph coloring problem MIC'2001. 4th Metaheuristics International Conference, Porto, Portugal pp. 16–20 (2001) July
12. Hertz, A., de Werra, D.: Using tabu search techniques for graph coloring. Computing **39**, 345–351 (1987)

13. Hordijk, W.: A mesure of landscapes. Technical report 95-045-049, Santa Fe Institute, Santa Fe, New Mexico, USA (1995) May
14. Hertz, A., Jaumard, B., de Aragao, M.P.: Local optima topology for the *k*-coloring problem. Discrete Appl. Math. **49**, 257–280 (1994)
15. Johnson, D.S., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: an experimental evaluation: part II, graph coloring and number partitioning. Oper. Res. **39**(3), 378–406 (1991)
16. Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. Santa Fe Institute, Working Paper 95-02-022 (1995)
17. Laguna, M., Marti, R.: A GRASP for coloring sparse graphs. Technical Report (1999) November 17
18. Morgenstern, C.: Distributed coloration neighborhood search. Discrete Mathematics and Theoretical computer science, Am. Math. Soc. **26**, 335–358 (1996)
19. Weinberg, B.: Analyse et résolution approchée de problèmes d'optimisation combinatoire: application au problème de coloration de graphe. Ph.D. Thesis, Université des Sciences et Technologies de Lille, France (2004)