Scatter Search for Graph Coloring*

Jean-Philippe Hamiez\(^1\) and Jin-Kao Hao\(^2\)

\(^1\) LGI2P, École des Mines d'Alès, EERJE
Parc Scientifique Georges Besse
F-30035 Nîmes Cedex 01, France
hamiez@site-eerie.ema.fr

\(^2\) LERIA, Université d'Angers
2, Bd. Lavoisier
F-49045 Angers Cedex 01, France
Jin-Kao.Hao@univ-angers.fr

Abstract. In this paper, we present a first scatter search approach for the Graph Coloring Problem (GCP). The evolutionary strategy scatter search operates on a set of configurations by combining two or more elements. New configurations are improved before replacing others according to their quality (fitness), and sometimes, to their diversity. Scatter search has been applied recently to some combinatorial optimization problems with promising results. Nevertheless, it seems that no attempt of scatter search has been published for the GCP. This paper presents such an investigation and reports experimental results on some well-studied DIMACS graphs.

1 Introduction

Scatter search \([13,14]\) is an evolutionary approach related to the tabu search metaheuristic \([12]\). It is based on strategies proposed in the 1960s for combining decision rules and constraints. This approach has only been applied recently to a few optimization problems. Applications of this method include, e.g., vehicle routing \([27]\) and unconstrained optimization \([23]\); see also \([13]\) for more references.

Like other population-based methods, scatter search uses combination of configurations to generate new configurations which can replace others in the population. But the way combinations and replacements are made differs from the traditional strategies used in genetic algorithms. Combinations operate on multiple parents, and replacements rely on the improvement of a fitness function as well as the improvement of the population diversity. Furthermore, scatter search generally works with a small set of configurations and uses deterministic heuristics as much as possible in place of randomization to make a decision.

Graph \(k\)-coloring can be stated as follows: given an undirected graph \(G\) with a set \(V\) of vertices and a set \(E\) of edges connecting vertices, \(k\)-coloring

\* This work was partially supported by the Sino-French Joint Laboratory in Computer Science, Control and Applied Mathematics (LiAMA) and the Sino-French Advanced Research Programme (PRA).
$G$ means finding a partition of $V$ into $k$ classes $V_1, \ldots, V_k$, called color classes, such that no couple of vertices $(u, v) \in E$ belongs to the same color class. Formally, \{ $V_1, \ldots, V_k$ \} is a valid $k$-coloring of the graph $G = (V, E)$ if $\forall i \in [1..k]$ and $\forall u \in V_i$, $\exists v \in V_i / (u, v) \in E$. The graph coloring problem (GCP for short) is the optimization problem associated with $k$-coloring. It aims at searching for the minimal $k$ such that a proper $k$-coloring exists. This minimum is the chromatic number $\chi(G)$ of graph $G$.

$k$-coloring and the GCP are well-known NP-hard problems [20] and only small problem instances can be solved exactly within a reasonable amount of time in the general case [5]. It is also hard even to approximate the chromatic number of a graph. In [25], it is proved that for some $\epsilon > 0$, approximating the chromatic number within a factor of $n^\epsilon$ is NP-hard. Indeed, one of the best known approximation algorithm [15] provides an extremely poor performance guarantee\(^1\) of $O(n(\log \log n)^2/(\log n)^3)$ for a graph with $n$ vertices.

Graph coloring has many real applications, e.g., timetable construction [24] or frequency assignment [9]. There are many resolution methods for this problem: greedy constructive approaches, e.g., DSATUR [1] and the Recursive Largest First algorithm [24], hybrid strategies like HCA [4,8] and those proposed in [6, 26], local search meta-heuristics, e.g., simulated annealing [18] or tabu search [3, 16], neural network attempts [17], ... See [2] for a more extensive list of other references about the GCP.

Despite the fact that the literature on graph coloring is always growing, there exists, to our knowledge, no approach relying on scatter search for the GCP. The goal of our study is then to provide an experimental investigation of scatter search applied to the GCP. The paper is organized as follows: Sect. 2 recalls the general template of scatter search; Sect. 3 presents our first scatter search algorithm for graph coloring; next section gives the results we obtained on some of the well-known DIMACS benchmark graphs [19], before concluding.

### 2 General Design of Scatter Search

We briefly recall here the components of scatter search; fundamental concepts and motivations are described in [13]. See also [14] for an exhaustive illustration of these components on a non-linear optimization problem.

1. **Diversification Generation Method.** This step is used first to initialize the population and, eventually, to rebuild a subset of the population during the search. Configurations are built in order to respect a maximal diversity\(^2\). See [21], e.g., for an illustration of an appropriate generator for a 0-1 knapsack problem;

\(^1\) The performance guarantee is the maximum ratio, taken over all inputs, of the color size over the chromatic number.

\(^2\) One of the lessons provided in [22] suggests to “consider the use of frequency memory to develop effective diversification”.
2. **Improvement Method.** New configurations, obtained in step 1 or by combination, are improved in quality;

3. **Reference Set Update Method.** Improved configurations are checked for replacing others in a reference set (RefSet for short) according to their quality (line 2 in Algorithm 1) or their diversity (line 3). RefSet consists of \( b \) best evaluated configurations (BestSet) and \( d \) most diverse configurations (DivSet). See [23] for other update methods;

4. **Subset Generation Method.** This step produces subsets of configurations (with two elements or more) from the reference set to be combined;

5. **Configuration Combination Method.** Subsets of configurations built in step 4 are combined, generally using a problem-dependent combination operator. Laguna and Armentano [22] also suggest that “the use of multiple combination methods can be effective”.

**Algorithm 1:** Scatter search outline

```
begin
  P ← \emptyset /* Start with an empty population \( P^* \) */
  while \( P \) is not full do
    Build a new configuration \( c \) with the Diversification generation method
    Apply the Improvement method to \( c \) to obtain \( c^* \)
    if \( c^* \notin \ P \) then \( P \leftarrow P \cup \{c^*\} \)
    Build BestSet by selecting \( / \) removing in \( P \) the \( b \) best evaluated configurations
    Build DivSet by selecting in \( P \) the \( d \) most diverse configurations
    RefSet ← BestSet ∪ DivSet
  [1] while RefSet contains new elements do
    for each subset \( i \) built by applying the Subset generation method on RefSet do
      Use the Configuration combination method with \( i \) to obtain \( c_i \)
      Apply the Improvement method on \( c_i \) to obtain \( c_i^* \)
      if \( c_i^* \notin \text{RefSet} \) and evaluation of \( c_i^* \) is better than the worst evaluated element \( c_{worst} \) in BestSet then
        BestSet ← BestSet ∪ \{\( c_i^* \)\}; BestSet ← BestSet \_ \{c_{worst}\}
      [2] else if \( c_i^* \notin \text{DivSet} \) and \( c_i^* \) add diversity to DivSet then
        DivSet ← DivSet ∪ \{\( c_i^* \)\}
        Remove the element with minimal diversity in DivSet
  [3] end
end
```

Algorithm 1 gives an overall view of one single iteration of a generic scatter search procedure, and thus, the way the components are linked\(^4\). The main loop 1 controls the termination of the procedure; the process stops when the population evolves no more in quality. Note that some scatter search procedures also use

\(^3\) Note that some scatter search procedures only use the quality criterion to update \( \text{RefSet} \). In this case, remove line 3.

\(^4\) See [10] for helpful information on practical implementation.
the evolution of the RefSet diversity as a stop criterion. More iterations can be done by restarting loop 1 with a new set of configurations composed of the b best evaluated configurations found by the previous iteration and d new diverse configurations built using the diversification generation method.

3 Scatter Search for Graph Coloring

In this section we describe the five components of our scatter search procedure dedicated to the graph coloring problem. These components are organized in the same way as in Algorithm 1. Let us first describe some concepts useful for the understanding of the overall procedure and its composing elements.

Configuration: a configuration c is any partition \{V_1, \ldots, V_k\} of V into k subsets. \(V_i(i \in [1..k])\) is an independent set if \(\forall u, v \in V_i, (u, v) \notin E\). c is a proper k-coloring if each \(V_i \in c\) is an independent set. We will refer to partial coloring for configurations in which some vertices are not assigned a color.

Evaluation Function: two configurations can be compared in terms of quality using an evaluation (or a fitness) function \(f: f(c) = |\{(u, v) \in V_i(V_i \in c, i \in [1..k])/(u, v) \in E\}|\). In other words, \(f\) counts the edges having both endpoints in the same color class. Solving a k-coloring instance means finding a particular configuration \(c^*\) such as \(f(c^*) = 0\).

General Resolution Strategy for Graph Coloring: k-coloring aims at finding a complete assignment of k colors to the vertices that satisfies all the constraints. Such an assignment is said consistent (proper k-coloring). The generalized GCP can then be stated as solving successive k-colorings with decreasing values of k until no proper k-coloring can be obtained.

3.1 Diversification Generation Method

We generate conflicting configurations with k colors by means of random independent sets built using Algorithm 2.

Algorithm 2: A diversification generation method for the GCP

\begin{algorithm}
\begin{algorithmic}
\State \textbf{begin}
\For {i = 1 to k, by 1}
\State A $\leftarrow \emptyset$; \(V_i \leftarrow \emptyset\)
\While {\(V \neq \emptyset\)}
\Comment{[1]}
\State Choose (randomly) a vertex \(v \in V\)
\State \(V \leftarrow V - \{v\}\); \(V_i \leftarrow V_i \cup \{v\}\)
\For {each \(u \in V/(u, v) \in E\)}
\State \(A \leftarrow A \cup \{u\}\); \(V \leftarrow V - \{u\}\)
\EndFor
\State \(V \leftarrow A\)
\EndWhile
\EndFor
\State \textbf{end}
\end{algorithmic}
\end{algorithm}
Randomization is used here (line 1) only to insure diversity. Other meaningful choice rules can easily replace it. For instance, the vertices can be selected in decreasing order of their saturation degree, like in DSATUR [1].

### 3.2 Improvement Method

We use a tabu search algorithm to improve new configurations. This algorithm iteratively makes best 1-moves, changing the current color of a conflicting vertex\(^5\) to another one, until achieving a proper coloring. “Best moves” are those, which minimize the difference between the fitness of the configuration before the move is made and the fitness of the configuration after the move is performed. In case of multiple best 1-moves, choose one randomly. A tabu move leading to a configuration better than the best configuration found so far, within the same execution of the improvement method or within the overall scatter search procedure, is always accepted (aspiration criterion).

The tabu tenure \(l\) is dynamically computed by the formula proposed in [3]:

\[
l = \alpha \times |E_c| + \text{random}(g)
\]

where \(E_c\) is the set of conflicting edges in configuration \(c\). \(\text{random}(g)\) is a function which returns an integer value uniformly chosen in \([1,g]\). \(\alpha\) weights the number of conflicting edges.

A move \(m\) can be characterized by a triplet \((u, V_{old}, V_{new})\), \(u \in V\), \(V_{old}\) and \(V_{new}\) being, respectively, the previous and the new colors of the conflicting vertex \(u\). So, when a move \(m\) is performed, assigning \(u\) to the color class \(V_{old}\) is forbidden for the next \(l\) iterations by introducing the \((u, V_{old})\) couple in the tabu list.

The algorithm stops when a solution is obtained or when a maximum number of moves have been carried out without finding a solution. Algorithm 3 gives an outline of the procedure, which is extracted from an effective generic tabu search [3] designed for various coloring problems (\(k\)-coloring, GCP, \(T\)-coloring and set \(T\)-coloring).

#### Algorithm 3: A tabu search for graph coloring

begin
Let \(c\) be the configuration to improve
\(TL \leftarrow \emptyset\) /* Initialize the tabu list \(TL\) to empty */
\(c^* \leftarrow c\)
while \(f(c^*) > 0\) and not Stop condition do
\(c\) by performing a best 1-move \(m(u, V_{old}, V_{new})\)
\(TL \leftarrow TL \cup (u, V_{old})\)
if \(f(c) < f(c^*)\) then
\(c^* \leftarrow c\)
end
\(^5\) A vertex \(u \in V\) is said conflicting if \(\exists v \in V; (u, v) \in E\).
3.3 Reference Set Update Method

We use the same reference set update method as the one described in Algorithm 1 (Sect. 2):

1. An improved configuration \( c_{new} \) replaces another one in \( BestSet \) if its evaluation is better than the worst evaluated configuration \( c_{worst} \) in \( BestSet \).
2. If \( c_{new} \) is not kept due to its quality (step 1), it is checked for diversity. \( c_{new} \) replaces the configuration with lowest diversity \( c_{worst} \) in \( DivSet \) if \( c_{new} \) has a higher diversity than \( c_{worst} \).

To compute the diversity \( D(c_1, c_2) \) between two configurations \( c_1 \) and \( c_2 \), we use the distance measure introduced in [8]. The distance between \( c_1 \) and \( c_2 \) is the minimum number of 1-moves necessary to transform \( c_1 \) into \( c_2 \). Let us call it the move distance \( (D_M) \).

Note that the Hamming distance \( D_H(c_1, c_2) \) is not well adapted to compare, in terms of diversity, two configurations \( c_1 \) and \( c_2 \) of the GCP. This is due to the definition itself of this distance. For two strings (configurations) \( c_1 \) and \( c_2 \) of the same length, \( D_H(c_1, c_2) \) is the number of different positions in the two strings. To illustrate the difference between these two distances, let us consider the example of Fig. 1.

Left drawing gives a representation of the traditional assignment strategy in which colors are assigned to vertices. In Fig. 1(a), and according to the definition of the Hamming distance, \( D_H(c_1, c_2) = 10 \). So, \( c_1 \) is quite far from \( c_2 \). Right drawing shows a representation of the partition approach for the same configurations. In this case, the labeling of the colors is indifferent. In Fig. 1(b), moving the vertex 7 from the class \( V_2 \) to \( V_3 \) in \( c_1 \) leads to \( c_2 \) (up to a permutation of the colors). Then, \( D_M(c_1, c_2) = 1 \).

![](image)

**Fig. 1.** Comparing Hamming and move distances

Note that the diversity is a measure over all the configurations in RefSet. So, it is updated each time a new configuration is added to RefSet.

3.4 Subset Generation Method

The smallest subsets (denoted by 2-subsets) consist of all the couples of configurations in RefSet. Intermediate subsets (\( a \)-subsets, \( a \in \{3, 4\} \)) are built by
augmenting each \((a-1)\)-subset with the best evaluated configuration not included in the subset. Finally, \(c\)-subsets \((c \in \left[5, \|Re\set\|\right])\) contain the best \(c\) elements. In the scatter search template, each subset is generated only once, while, in the context of genetic algorithms, combinations based on the same subset are allowed. See [10], from which this description is extracted, for motivations about this method.

### 3.5 Configuration Combination Method

Combination may be viewed as a generalization upon multiple parents of classical crossovers which usually operate with only two parents. We used a generalization of the powerful greedy partition crossover (GPX), proposed in [8] within an evolutionary algorithm. GPX has been especially developed for the graph coloring problem with results reaching, and sometimes improving, those of the best known algorithms for the GCP.

Given a subset \(p\) generated by the subset generation method, the generalized combination operator builds the \(k\) color classes of the offspring one by one. First, choose arbitrarily (and temporarily remove) a configuration \(c \in p\). Temporarily remove from \(c\) a minimal set \(A_c\) of conflicting vertices such as \(c\) becomes a partial proper \(k\)-coloring. Next, fill in a free color class \(V_{ij}(i \in \left[1..k\right])\) of the offspring with all conflict-free vertices of the color class \(V^c\in c\) having maximum cardinality (break ties randomly) and remove these vertices from all configurations in \(p\). Then, make \(A_c\) available for \(c\) and repeat these steps until the \(k\) color classes of the offspring contain at least one vertex. Finally, to complete the new configuration if necessary, assign to each free vertex in the offspring a color such that it minimizes the conflicts over the graph.

The chosen configuration \(c\) is temporarily removed from \(p\) to balance the origins of the color classes of the new configuration; all removed configurations become enable when \(p\) is empty. Fig. 2 summarizes the combination mechanism. See also [8] for an illustration of GPX on two parents.

---

**Fig. 2.** Generalization of GPX
4 Preliminary Results

We give in this section some preliminary results obtained with our scatter search algorithm (called SSGC hereafter). Results of SSGC are reported for some of the DIMACS benchmark graphs \cite{19}\footnote{Available via FTP from: dimacs.rutgers.edu/pub/challenge/graph/benchmarks/.} together with those of the best-known algorithms and those of a generic tabu search algorithm:

- Morgenstern \cite{26}, two local search algorithms based on particular neighborhoods (denoted by 3a and 3b in Table 1), and a distributed population-based algorithm (3c);
- Funabiki and Higashino \cite{7}, a descent algorithm with various heuristics mixed with a greedy construction stage and the search for a maximum clique (denoted by MIPS,CLR);
- Dorne and Hao \cite{3}, a generic tabu search (GTS for short) which solves successive $k$-colorings, $k$ decreasing as long as a proper $k$-coloring is found.

Columns 5-7 in Table 1 summarize the best-known results given in the above papers for some DIMACS graphs: the smallest coloring ever obtained, the method which produced such a coloring, and the best computing time required. Columns 1-4 recall the characteristics of each graph: its name, number of vertices and edges, and its chromatic number (or its best known lower bound when unknown).

Results of GTS are reported in Table 2 (columns 3-5), including mean number of moves. The last four columns give the results of our scatter search algorithm\footnote{SSGC is implemented in C (CC compiler with -O5 flag) and runs on a Sun Ultra 1 (256 RAM, 143 MHz).}. Time entries and number of moves and combinations are averaged over five to ten runs. A maximum of 100000 moves were allowed for the improvement method. We only make, for each execution, one iteration of scatter search, i.e., main loop 1 in Algorithm 1 was performed on a single reference set. The $\alpha$ and $g$ parameters (Sect. 3.2) used in computing the dynamic tabu tenure $l$ (1) were empirically determined, respectively 2 and 10 at most. The size of the starting population was no more than 20 configurations and the $b$ and $d$ parameters (Sect. 2) were both empirically fixed to 10 at most. For the flat300.20.0 and dje125.5 graphs, the improvement method was allowed to perform 1\% of random walks, i.e.: moving a random vertex into a random color class. Information about computing time (in seconds) in Table 1 and 2 is only for indicative purpose because results were obtained on different machines. “-” marks signal that no result is available.
Table 1. Best known results for some instances of the 2nd DIMACS challenge

<table>
<thead>
<tr>
<th>Graph name</th>
<th>[V]</th>
<th>[E]</th>
<th>χ</th>
<th>k</th>
<th>Method</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>school1</td>
<td>385</td>
<td>19095</td>
<td>14</td>
<td>14</td>
<td>MIPS_CLR</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>school1_nsh</td>
<td>352</td>
<td>14612</td>
<td>14</td>
<td>14</td>
<td>MIPS_CLR</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>dsjrv00.1c</td>
<td>500</td>
<td>121275</td>
<td>84</td>
<td>85</td>
<td>3c</td>
<td>1240</td>
</tr>
<tr>
<td>r125.5</td>
<td>125</td>
<td>3838</td>
<td>36</td>
<td>36</td>
<td>3c</td>
<td>&lt; 1</td>
</tr>
<tr>
<td>r250.1c</td>
<td>250</td>
<td>30227</td>
<td>64</td>
<td>64</td>
<td>3c</td>
<td>0</td>
</tr>
<tr>
<td>r250.5</td>
<td>250</td>
<td>14849</td>
<td>65</td>
<td>65</td>
<td>3c</td>
<td>181</td>
</tr>
<tr>
<td>r1000.1c</td>
<td>1000</td>
<td>485090</td>
<td>≥ 90</td>
<td>98</td>
<td>3c</td>
<td>1240</td>
</tr>
<tr>
<td>r1000.5</td>
<td>1000</td>
<td>238267</td>
<td>≥ 234</td>
<td>237</td>
<td>MIPS_CLR</td>
<td>3266</td>
</tr>
<tr>
<td>le450_15a</td>
<td>450</td>
<td>8168</td>
<td>15</td>
<td>15</td>
<td>3b</td>
<td>&lt; 60</td>
</tr>
<tr>
<td>le450_15b</td>
<td>450</td>
<td>8169</td>
<td>15</td>
<td>15</td>
<td>3b</td>
<td>&lt; 60</td>
</tr>
<tr>
<td>le450_15c</td>
<td>450</td>
<td>16680</td>
<td>15</td>
<td>15</td>
<td>3b</td>
<td>126</td>
</tr>
<tr>
<td>le450_15d</td>
<td>450</td>
<td>16750</td>
<td>15</td>
<td>15</td>
<td>3b</td>
<td>80</td>
</tr>
<tr>
<td>flat300_20_0</td>
<td>300</td>
<td>21375</td>
<td>20</td>
<td>20</td>
<td>3a</td>
<td>1</td>
</tr>
<tr>
<td>dsjc125.5</td>
<td>125</td>
<td>3891</td>
<td>≥ 10</td>
<td>17</td>
<td>3c</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 2. Results of GTS and SSGC

<table>
<thead>
<tr>
<th>Graph name</th>
<th>χ</th>
<th>GTS</th>
<th>SSGC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>k</td>
<td>Moves</td>
</tr>
<tr>
<td>school1</td>
<td>14</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>school1_nsh</td>
<td>14</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>dsjrv00.1c</td>
<td>84</td>
<td>85</td>
<td>21000</td>
</tr>
<tr>
<td>r125.5</td>
<td>36</td>
<td>36</td>
<td>147000</td>
</tr>
<tr>
<td>r250.1c</td>
<td>64</td>
<td>64</td>
<td>462</td>
</tr>
<tr>
<td>r250.5</td>
<td>65</td>
<td>65</td>
<td>7800</td>
</tr>
<tr>
<td>r1000.1c</td>
<td>≥ 90</td>
<td>98</td>
<td>1623000</td>
</tr>
<tr>
<td>r1000.5</td>
<td>≥ 234</td>
<td>242</td>
<td>6027000</td>
</tr>
<tr>
<td>le450_15a</td>
<td>15</td>
<td>15</td>
<td>103000</td>
</tr>
<tr>
<td>le450_15b</td>
<td>15</td>
<td>15</td>
<td>33000</td>
</tr>
<tr>
<td>le450_15c</td>
<td>15</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>le450_15d</td>
<td>15</td>
<td>16</td>
<td>-</td>
</tr>
<tr>
<td>flat300_20_0</td>
<td>20</td>
<td>20</td>
<td>33000</td>
</tr>
<tr>
<td>dsjc125.5</td>
<td>≥ 10</td>
<td>17</td>
<td>348000</td>
</tr>
</tbody>
</table>
From Table 1 and 2 we can make several remarks. First, SSGC manages to reach the results of the best-known algorithms (column \( k \)), except on the r1000.5 graph for which a 237-coloring has been published recently. The sophisticated algorithm used to reach this coloring includes, among other components, the search for a maximum clique. Nevertheless, SSGC obtains here a better coloring than GTS. Note that the previous best result for this graph was reported in [26] with 241 colors. Our scatter search approach also improves in quality on the results obtained with tabu search (GTS) on four graphs. This means that tabu search, the improvement method we used, surely benefits from the other general components of scatter search (Sect. 2).

Nevertheless, SSGC seems to be quite slow (regarding the number of moves) to reach the same results as those of GTS. This can partially be explained by the time spent in building the initial reference set, which include the improvement of all the configurations in the population. Furthermore, the generation method used in SSGC (random independent sets, see Sect. 3.1) differs from the one performed by GTS which is based on DSATUR [1]. It would certainly be interesting to follow the approach used in GTS to build the initial reference set. Another reason to the slowness of SSGC may be allotted with the diversity update step which occurs each time a new configuration \( c_{\text{new}} \) is inserted in \( \text{RefSet} \). One possible way to speed up this step could be to regulate it according to a minimal diversity \( D_0 \) (fixed parameter or dynamically computed). In other words, \( c_{\text{new}} \) will be added to \( \text{DivSet} \) only if its minimal diversity is greater than \( D_0 \).

**Table 3. Results of SSGC using a descent improvement method**

<table>
<thead>
<tr>
<th>Graph name</th>
<th>( \chi )</th>
<th>( k )</th>
<th>Moves</th>
<th>Combinations</th>
<th>Time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>school1</td>
<td>14</td>
<td>14</td>
<td>900</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>school1_test</td>
<td>14</td>
<td>14</td>
<td>408</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>r125.5</td>
<td>36</td>
<td>36</td>
<td>7</td>
<td>66</td>
<td>7</td>
</tr>
<tr>
<td>r250.1c</td>
<td>64</td>
<td>64</td>
<td>83600</td>
<td>37499</td>
<td>–</td>
</tr>
<tr>
<td>r250.5</td>
<td>65</td>
<td>65</td>
<td>25</td>
<td>580</td>
<td>452</td>
</tr>
</tbody>
</table>

We also experimented scatter search with a less elaborated improvement method, the simplest descent heuristic, which makes best 1-moves until achieving no improvement. This approach was only able to solve a few “easy” instances (see Table 3) implying that the global efficiency of scatter search is greatly dependent on the particular efficiency of its improvement algorithm.

5 Conclusions and Perspectives

We have developed a first adaptation of the scatter search method to the graph coloring problem. The preliminary results we obtained on some well-studied DIMACS graphs using this technique show that scatter search may be effective
for some of them since SSGC reached most of the results of the best-known approaches. SSGC obtains also better results than those of a generic tabu search on some graphs.

At the same time, we observed that scatter search is quite slow due, essentially, to the numerous components it uses. Efforts must then be made to develop effective procedures to make the scatter search quicker. Another difficulty is to identify meaningful rules when choices are needed.

We are now working on replacing some random choices made in SSGC by deterministic rules. A first modification have been carried out in the diversification generation method (Sect. 3.1) by selecting vertices to be included in the same independent set according to Brélaz heuristics [1]. Vertices are inserted in color classes in decreasing order of their saturation degrees. Ties are broken using the connection degree and, only if needed, randomness. Few results are available since tests are currently running. However, the modified procedure was able to solve the school1, school2, and r125.5 graphs even while the improvement method was left out. In other words, solutions were found only by combining sets of configurations. This suggests that the generalization of the effective greedy partition crossover may be a suitable combination operator.

This research has highlighted the need for further investigation to make scatter search more effective for graph coloring. By using the generic tabu search in the initialization step for building the reference set, one may speed up the overall procedure. Another possible way could be to use maximal independent sets in place of selecting vertices at random to identify independent sets. Indeed, this later way shares the same objective as the greedy partition crossover. Lessons provided in [22] could also be useful for further attempts. Finally, some choice rules proposed in [11] may be used as fast heuristics either to generate the initial population or to color free vertices after each combination.

References

Optim. 3(4) (1999) 379–397
10. Glover, F.: A template for scatter search and path relinking. In Hao, J.K., Lutton,
Notes in Computer Science 1363 Springer-Verlag (1998) 13–54
Tech. Report (February 2001)
ors) Theory and applications of evolutionary computation: recent trends Springer-
Verlag (to appear)
39 (1987) 345–351
lated annealing: an experimental evaluation. II. Graph coloring and number parti-
19. Johnson, D.S., Trick, M.A. (editors): Cliques, coloring, and satisfiability: 2nd DI-
MACS implementation challenge, 1993. DIMACS Series in Discr. Math. and The-
(1972) 85–103
of applied optimization Oxford Academic Press (to appear)
22. Laguna, M., Armento, V.A.: Lessons from applying and experimenting with scatter
search. Tech. Report (March 2001)
23. Laguna, M., Martí, R.: Experimental testing of advanced scatter search designs for
Nat. Bur. Stand. 84 (1979) 489–506
25. Lund, C., Yannakakis, M.: On the hardness of approximating minimization prob-
357
27. Rego, C.: Integrating advanced principles of tabu search for the vehicle routing
problem. Working paper, Faculty of Sciences, University of Lisbon (1999)