Journal of Heuristics, 1: 105–128 (1995)

Embedding a Sequential Procedure Within an Evolutionary Algorithm for Coloring Problems in Graphs

DANIEL COSTA
Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, fax: +41-21-693 42 50, phone: +41-21-693 25 68, email: costa@dma.epfl.ch

ALAIN HERTZ
Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, fax: +41-21-693 42 50, phone: +41-21-693 25 68, email: hertz@dma.epfl.ch

OLIVIER DUBUIS
Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, fax: +41-21-693 42 50, phone: +41-21-693 25 68

Abstract

We present in this article an evolutionary procedure for solving general optimization problems. The procedure combines efficiently the mechanism of a simple descent method and of genetic algorithms. In order to explore the solution space properly, periods of optimization are interspersed with phases of interaction and diversification. An adaptation of this search principle to coloring problems in graphs is discussed. More precisely, given a graph G, we are interested in finding the largest induced subgraph of G that can be colored with a fixed number q of colors. When q is larger or equal to the chromatic number of G, then the problem amounts to finding an ordinary coloring of the vertices of G.

Key Words: chromatic number, combinatorial optimization, evolutionary methods, graph coloring, q-stability number, sequential methods

1. Introduction

Let $G = (V, E)$ be an undirected graph with vertex set V and edge set E. A *stable* set is a subset $S \subseteq V$ of vertices whose elements are pairwise nonadjacent—that is, $\{x, y\} \in S \Rightarrow [x, y] \notin E$. A partial *q-coloring* of G is a partition $(V_1, V_2, \ldots, V_q)$ of a subset $V' \subseteq V$ of vertices into q disjoint stable sets. By assigning a color $c(x) = i$ to each vertex $x \in V_i$, a partial q-coloring can be seen as a coloring in q colors of a subset $V' \subseteq V$ of vertices such that no two adjacent vertices receive the same color. The size of a partial q-coloring is the cardinality of the set $V' = V_1 \cup V_2 \cup \ldots \cup V_q$. The maximum size of a partial q-coloring of G is denoted $\alpha_q(G)$ and is called the *q-stability number* of G. The partial q-coloring problem amounts to finding a partial q-coloring of maximum size in G. For $q = 1$, we have the well-known stable set problem. A partial q-coloring of size $\alpha_q(G)$ will be called optimal in the remainder of this article. It is to be observed that the stable set problem and the partial q-coloring problem are of equivalent complexity in that

there is a one-to-one correspondence between the partial q-colorings of G and the stable sets of a larger graph $G' = G + K_q$, which is constructed on the basis of G by making q copies of G and by adding all possible edges between the q copies of each vertex of G (Berge, 1989).

A partial q-coloring of size $n = |V|$ (that is, all vertices are colored) is commonly called a q-coloring of G. The problem in this case amounts to finding a q-coloring with q as small as possible. The minimum q for which a q-coloring exists is called the *chromatic number* of G and is denoted $\chi(G)$. Thus $\chi(G) = \min\{q \mid \alpha_q(G) = n\}$.

The problems of finding an optimal partial q-coloring (especially a stable set) or the chromatic number of a graph are among the most studied problems in the field of graph theory. These problems have a large variety of real-life applications in the area of scheduling and timetabling. Unfortunately both of these problems have been shown to be NP-hard (Garey and Johnson, 1979), and thus it is believed that there is no polynomial algorithm for solving these problems in an arbitrary graph. It follows that only small problem instances can be solved exactly within a reasonable amount of time in the general case (Dubois and de Werra, 1994; Friden, Hertz, and de Werra, 1990; Mannino and Sassano, 1992). This is why many researchers have concentrated their work on heuristic methods rather than on exact methods in order to solve practical problems of large size (Fleurent and Ferland, 1995; Friden, Hertz, and de Werra, 1989; Gendreau, 1994; Hertz and de Werra, 1987; Johnson, et al., 1991; Morgenstern, 1990).

One standard iterative approach for solving the partial q-coloring problem consists of developing a procedure that seeks a partial q-coloring of size k in G, where k is a fixed integer. An initial partial q-coloring of size k can be found by means of a greedy algorithm or simply by setting $k = q$. The value of k is increased by one unit, and the procedure is applied iteratively as long as it succeeds. When the whole process terminates, the current value $k - 1$ is a lower bound on $\alpha_q(G)$. A similar algorithm can be used for finding the chromatic number of a graph G. Starting with an initial q-coloring of G, we try to reduce q by one unit iteratively using a procedure whose function is to find a q-coloring in G. It follows that the resulting $q + 1$ value is an upper bound on $\chi(G)$.

The two problems we are interested in are as follows. Given a graph G and two positive integers q and k, find either a partial q-coloring of size k in G or a q-coloring in G. In our terminology, we understand that the value q is different in these two problems.

In the next section we sketch the basic ideas of three general concepts for searching through the solution space of a combinatorial optimization problem. Then we indicate how two different solution methods can be combined to define a general procedure in which cooperation phases are interspersed with periods of optimization. More precisely, we focus on a simple hybrid algorithm based on two well-known methods—namely, a standard descent method and a genetic algorithm. Adaptations of this mixed procedure to the problem of finding a partial q-coloring of size k and a q-coloring in a graph G will be presented in Sections 3 and 4, respectively. In Section 5 numerical results are given for randomly generated graphs, and some experiments are performed to evaluate the quality of the estimates for $\chi(G)$ and $\alpha(G)$ that we shall consider. Section 6 concludes with general remarks on this work and directions for future research.

2. General Search Principles for Combinatorial Optimization

Generally speaking, the problems of finding a partial q-coloring of size k and a q-coloring in a graph G belong both to the class of combinatorial optimization problems (COPs). Given a finite set X of feasible solutions s and an objective function f assigning to each solution $s \in X$ a value $f(s)$, the goal is to find a solution $s^* \in X$ for which $f(s)$ is minimum. Restriction to function minimization is without loss of generality since $\max f(s) = -\min(-f(s))$. When designing a solution method for a COP, either we build a method that is specific to the problem under study, or we adapt some general existing schemes. In this latter case three different search principles can be defined for finding good solutions in X.

2.1. A Constructive Approach

Constructive methods build feasible solutions of the form $s = (x_1, x_2, \ldots, x_n)$ by starting from an "empty" solution $s(0)$ and by inserting repeatedly a component x_i into the current partial solution $s(i-1) = (x_1, x_2, \ldots, x_{i-1})$. An example of such a method is the well-known greedy algorithm that completes at best the partial solution in a rather myopic way—that is, without taking care of all the consequences the current completion may have on the final solution. A greedy coloring (or partial coloring) algorithm proceeds by adding vertices one after the other (as long as it is possible) to the portion of the graph already colored. Each newly adjoined vertex is assigned the smallest color not allocated to one of its neighbors. Various constructive algorithms of this type have been defined for tackling the coloring problem (Brélaz, 1979; Culberson, 1992) and the stable set problem (Gendreau, 1994). The advantage of a constructive method is that it is very fast. However, it is generally not competitive from the viewpoint of the quality of the final solution obtained.

2.2. A Sequential Approach

Sequential methods start from an initial solution $s \in X$ (for example, obtained by a constructive method) and move step by step to a neighbor solution $s' \in X$ by performing some elementary modifications $m \in M_s$. M_s refers to the set of acceptable modifications at solution s. A solution s' obtained from s via an acceptable modification m will be denoted $s' = s \oplus m$. The neighborhood of a solution s can be defined more formally by $N(s) = \{\tilde{s} \in X \mid \exists m \in M_s: \tilde{s} = s \oplus m\}$. The process is repeated until some stopping criterion is met. The simplest example of a sequential approach is the steepest-descent method, which is formulated in Figure 1. A recent study (Hertz, 1995) has described the class of graphs for which a steepest-descent method is guaranteed to find a maximum stable set independently of the initial solution $s \in X$.

Sometimes it is too time-consuming to scan the entire neighborhood, and thus only a sample $V^* \subset N(s)$ is examined at each step. When such a shortcut is accepted, we will use the term of descent method instead of steepest-descent method in the remainder of this article. The danger with a descent method is to be trapped in a local minimum of the objective function. To overcome the limitation of local optimality, sequential methods with some

```
choose an initial solution s ∈ X;
stop := false;
While not stop do
    find a best s' ∈ N(s) (i.e., f(s') ≤ f(s̃) ∀s̃ ∈ N(s));
    If f(s') ≥ f(s), then stop := true else s := s';
```

Figure 1. The steepest-descent method.

additional features need to be defined. Simulated annealing (SA) and tabu search (TS) are two such methods. The difference between them lies in how the step from one solution to a neighbor one is selected. SA simulates the cooling of a collection of atoms by exploiting properties of statistical mechanics, while TS has recourse to notions of artificial intelligence for applying some learning rules. Adaptations of these two procedures to the problems considered in this paper can be found in Chams, Hertz, and de Werra (1987), Friden, Hertz, and de Werra (1989), Hertz and de Werra (1987), and Johnson et al. (1991). For more detailed explanations on how SA and TS work, the reader is referred to Eglese (1990), Kirkpatrick, Gelatt, and Vecchi (1983), and Reeves (1993), and Glover, Taillard, and de Werra (1993), Hertz, Taillard and de Werra (1995), and Reeves (1993), respectively.

2.3. *An Evolutionary Approach*

Evolutionary methods deal with a population of solutions $\mathcal{P}$ instead of a single (partial) solution like constructive and sequential methods. The central idea is to use collective properties of a group of distinguishable solutions for searching in X. Once an initial population of solutions $\mathcal{P}_0$ has been generated, it is improved by a cyclic two-step evolution process consisting of a *cooperation* step and a *self-adaptation* step. Each cycle is called a *generation*. In the cooperation step, solutions of the current population exchange information with the aim of producing new solutions that inherit good attributes. In the self-adaptation step, solutions change their internal structure without any interaction with the other members of the population. We say that an evolutionary method goes through a diversity crisis or converges prematurely when its population contains a high percentage of replicates of the same solution. A number of mechanisms need to be incorporated to prevent this outcome.

- *Genetic algorithms* (Davis, 1991; Liepins and Hilliard, 1989) are evolutionary methods based on genetics and biological mechanisms of natural selection. They are composed of three different operators that use probabilistic rules. The cooperation step is governed by a reproduction operator and a crossover operator. The reproduction operator eliminates the average worst solutions and gives more importance to the best ones by duplicating them. Then the crossover operator introduces a diversity within the population by mating pairs of solutions in order to give birth to new offspring. The self-adaptation step plays a secondary role in GAs. It is performed by a mutation operator that changes arbitrarily, with a low probability, each component x_i of all solutions $x = (x_1, x_2, \ldots, x_n)$ of the current population $\mathcal{P}$.
- *Scatter search* (Glover, 1993, 1994) simulates an evolution that is not related to the field of genetics. There is no reproduction and mutation operator, and the manner of combining

solutions is different from the use of a genetic crossover in the sense that groups of more than two solutions are taken into account in the mating process. The self-adaptation step is made up of a "rounding" mechanism that consists essentially of modifying the solutions in order to make them feasible.

- *Ant systems* (Colorni, Dorigo, and Maniezzo, 1991a, 1991b, and 1992) define an evolutionary method that imitates the behavior of an ant colony having to perform some specific tasks. In the cooperation phase each solution is examined with the aim of updating a global memory keeping track of important structures of X, which have been successfully exploited in the past. The self-adaptation step uses a problem-specific constructive method to create a new population of solutions on the basis of the global memory.

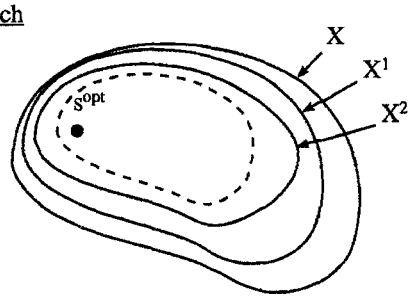
Although the field of evolutionary methods has kept growing and evolving since the first developments in the 1970s there exist few evolutionary algorithms for solving problems related to graph coloring. The only adaptations we are aware of are described in Carter and Park (1993), Davis (1991), and Fleurent and Ferland (1995).

The three search principles sketched in this section are illustrated in Figure 2. The constructive approach considers a subset of X that gets smaller at each step, the sequential approach defines a path through X, and the evolutionary approach manipulates a set of solutions located in different promising areas of X.

So far a great number of successful achievements in the field of general heuristics for combinatorial optimization has occurred by adapting sequential algorithms like *SA* and *TS*. Recent studies have shown that pure evolutionary methods are not very competitive when applied to highly constrained problems. However, it has been observed that the use of an evolutionary approach is well suited to complement and improve on existing procedures. One of the most promising innovations in the field of general heuristics is the insertion of sequential search techniques into an evolutionary method. The combination of these two different search procedures yields a *hybrid* algorithm the performance of which is generally significantly better than the one of both procedures running separately (Costa, 1995; Thangiah, Osman, and Sun, 1994). In a hybrid algorithm the role of the sequential procedure is to explore thoroughly distinct areas of X, whereas the evolutionary procedure provides global guidance through X.

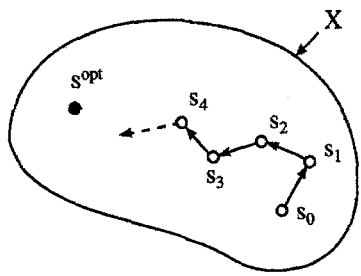
Following the ideas presented in Grefenstette (1987), and Mühlenbein, Gorges-Schleuter, and Krämer (1988), we focus on a hybrid algorithm combining a simple descent method and GAs. The general scheme of the evolutionary descent method (EDM) we have retained is given in Figure 3. It differs from most of the hybrid algorithms that have been developed recently in the sense that it deals with a steepest-descent method instead of a refined sequential method that accepts nonimproving moves (Costa, 1995; Fleurent and Ferland, 1994, 1995; Moscato, 1993). We will show that it is sufficient to use a simple sequential method in ahybrid algorithm whose crossover operator is adequately designed. The reproduction step of standard GAs has been partially omitted in our implementation of EDM because the risk of premature convergence is elevated in a hybrid algorithm whose individual optimization phase is deterministic. Indeed, a descent method applied to two identical solutions yields two identical local optima. The worse solution in the population at the end of the current generation is merely replaced by the best solution found at the end of the previous generation. In order to save computation time at each step of the descent method, it has been decided to

a) A constructive approach

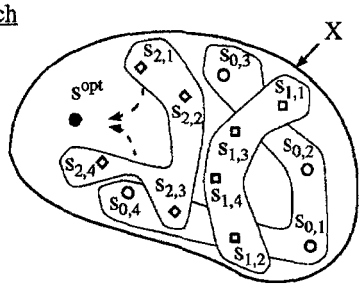


$X^i = \{s=(x_1, x_2, ..., x_n) \in X \mid x_1, x_2, ..., x_i \text{ are fixed}\}\ i=1, ..., n$

b) A sequential approach



c) An evolutionary approach



$\mathcal{P}_{gen} = \{s_{gen,i} \in X;\ i=1, ..., m=4\}\ gen=0, 1, 2, ...$

Figure 2. Three search principles for combinatorial optimization.

move directly to the first neighbor solution encountered, which is better than the current one (if any). The definition of an efficient crossover is the crucial point when implementing a GA. A good knowledge of the problem under study is required. ETD is stopped after a fixed number $ngen_{max}$ of generations if no satisfactory solution has been reached before.

In the next section we present an adaptation of EDM for finding a partial q-coloring of size k in a graph G. A new crossover operator will be defined and some refinements will be incorporated into EDM to increase its efficiency by achieving short term and long term diversification.

choose a random initial population $\mathcal{P}_0 := \{s_0^1, s_0^2, \ldots, s_0^m\}$
gen := 0
While stopping criterion = false **do**
  for $i := 1$ **to** m **do**
    create s_{gen+1}^i by applying crossover to s_{gen}^i and s_{gen}^{i+1} (*the addition is taken modulo n*)
  For $i := 1$ **to** m **do**
    apply mutation to s_{gen+1}^i
    improve s_{gen+1}^i with a steepest-descent method
  $\mathcal{P}_{gen+1} := \{s_{gen+1}^1, s_{gen+1}^2, \ldots, s_{gen+1}^m\}$
  replace the worst solution in $\mathcal{P}_{gen+1}$ by the best one in $\mathcal{P}_{gen}$
  gen := gen + 1

Figure 3. An evolutionary descent method.

3. An Evolutionary Descent Method for Finding a Partial q-Coloring of Size k

Given a graph $G = (V, E)$, q colors, and a fixed integer k, the set of feasible solutions X contains all partitions $s = (V_0, V_1, \ldots, V_q)$ of V into $q + 1$ subsets such that $|V_1| + \ldots + |V_q| = k$. Vertices in V_0 are left uncolored, whereas those in V_i receive color i. The neighborhood of a solution s is composed of all solutions that can be reached from s by exchanging the class of two vertices or by moving a colored vertex from one color class to another.

To reduce the risk of premature convergence, short-term diversification is performed at each generation of EDM by using a dynamic objective function and a nonstandard mutation operator. Edges whose endpoints are in the same color class will be referred to as *conflicting edges*. Let w_e be a weight associated to an edge e and $E(Vi)$ the collection of edges with both endpoints in V_i. The cost $f(s)$ of a feasible solution is computed dynamically on the basis of the set of conflicting edges:

$$f(s) = \sum_{i=1}^{q} \sum_{e \in E(V_i)} w_e.$$

Initially, we set $w_e = 1$ $\forall e \in E$. In order not to manipulate the same conflicting edges for too long, it has been decided to increase by an amount Δw the weight w_e of all the conflicting edges in the m colorings s_{gen}^i at the end of each generation and to reset all the w_e values to 1 every $ngen_w$ generations in EDM. A solution $s \in X$ is clearly a partial q-coloring of G if and only if $f(s) = 0$—that is, V_i is a stable set $\forall i = 1, \ldots, q$.

The mutation operator consists of replacing, with a probability p_{mut}, a solution by one of its neighbors chosen randomly. Usually p_{mut} is held constant throughout a run of a genetic algorithm. However, following the ideas presented in (Fogarty, 1989), it has been observed that EDM performs significantly better if the probability of mutation is changed once in a while. The variation scheme shown in Figure 4 has been retained in our adaptation of EDM. The probability of mutation is left at a constant level p_{mut}^0 for $ngen_{mut}$ generations, then it is increased, or alternately, decreased linearly for r_{mut} generations.

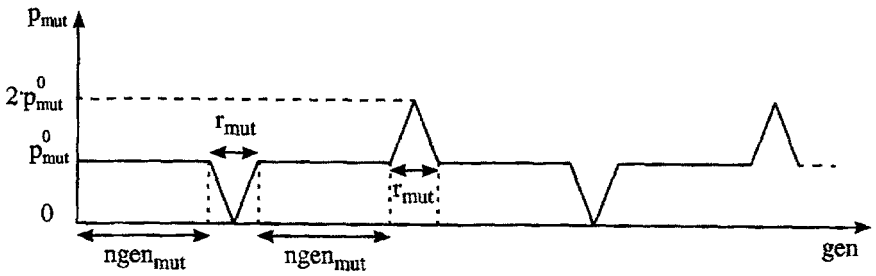


Figure 4. A variation scheme for the probability of mutation p_{mut}.

In order to make the most of the characteristics of the best solution s^* found so far, it has been decided to insert s^* in the set of parent solutions before every cooperation phase of EDM. For this purpose we replace the worst solution in the population $\mathcal{P}_{gen}$ at the beginning of each generation by the best one in the population $\mathcal{P}_{gen-1}$ at the beginning of the previous generation. The merits of a steady-state model where some solutions are kept from one generation to another are discussed in Davis (1991) and Syswerda (1989).

Long-term diversification is achieved within EDM by changing periodically the structure of the solution space X to drive the search into new regions. The size k of the feasible partial q-colorings considered is increased during r_{chX} generations every $ngen_{chX}$ generation ($r_{chX} < ngen_{chX}$). According to preliminary experiments, increasing k by $\lceil q/2 \rceil$ units yields good results.

Previous experiments with the most commonly cited crossover operators did not yield effective results in the context of graph coloring. Fleurent and Ferland (1995) developed a graph-adapted recombination operator that appeared to be significantly better than the ordinary one-point, two-point, and uniform crossover operators. The recombination operator we have retained in our study is an adaptation of the so called *union crossover*, which have been defined in Costa (1995) for solving large-size scheduling problems. Roughly speaking the union crossover starts by building a "megasolution" that inherits all the information contained in the parent solutions s_1 and s_2. The megasolution is then transformed into a feasible solution by sequentially removing the most costly redundant information it contains. Figure 5 shows formally how two partial q-colorings s_1 and s_2 give birth to a new solution s_3 based on the above mechanism. The color of a vertex v in s_i is denoted $c_i(v)$. If v is not colored in s_i, then $c_i(v) = 0$. It is important to point out that s_2 has an influence larger than s_1 on the offspring s_3 since it determines the color class of a vertex that is colored in both parent solutions. Vertices that are colored in only one parent solution receive a color $\tilde{c} \in \{1, \ldots, q\}$ that causes a minimum number of conflicting edges in the solution s_3 achieved so far. This coloring is done by simply following the given indexing of the vertices. A more strategic variant can be created by examining vertices in an appropriate order. The path relinking concepts proposed by Glover (1994) could be of some utility at this level. Experiments in this direction were not conducted. Feasibility of s_3 is achieved

Input $s_1, s_2 \in X$
Output $s_3 = (V_0, V_1, \ldots, V_q) \in X$

$V_i := \emptyset \ \forall i = 0, \ldots, q$;
For every vertex v such that $c_1(v) > 0$ and $c_2(v) > 0$ **do** $V_{c_2(v)} := V_{c_2(v)} \cup \{v\}$
For every other vertex v **do**
  If $c_1(v) > 0$ **or** $c_2(v) > 0$ **then**
    $V_{\tilde{c}(v)} := V_{\tilde{c}(v)} \cup \{v\}$ where $\tilde{c}(v) \in \{1, \ldots, q\}$ is the color yielding the smallest number of conflicting edges adjacent to v in the current solution s_3
  else $V_0 := V_0 \cup \{v\}$;
$c_3(v) := i \ \forall v \in V_i \ (i = 0, \ldots, q)$;
While $|V_1| + \ldots + |V_q| > k$ **do**
  Find the vertex v^+ adjacent to the highest number of conflicting edges in s_3;
  $V_{c_3(v^+)} := V_{c_3(v^+)} \setminus \{v^+\}$
  $c_3(v^+) := 0$;

Figure 5. A union crossover for the partial q-coloring problem.

by repeatedly uncoloring the vertices adjacent to the greatest number of conflicting edges until we are left with exactly k colored vertices.

4. A Special Case: Finding a q-Coloring

Given that all vertices have to be colored when looking for a q-coloring of G, the feasible solutions $s \in X$ are all partitions $s = (V_1, \ldots, V_q)$ of the vertex set V into q subsets (that is, $V_0 := \emptyset$). Thus finding a q-coloring in $G = (V, E)$ is a special case of the problem studied in the previous section where k is set equal to $|V|$. Except for the crossover operator and the long-term diversification process, which need to be reconsidered, the adaptation of EDM presented in Section 3 is well suited for tackling the q-coloring problem.

In this case the union crossover needs to be specialized because all vertices are colored in both parent solutions. Let $c_i(v)$ be the color of vertex v in the coloring s_i and $nce_i(v, d)$ be the number of conflicting edges located at a distance d from v in s_i. By definition the distance between a vertex v and an edge $e = [x, y]$ is the least number of edges in a path from v to an endpoint of e (that is, x or y). To each vertex v, we associate a penalty $p_i(v)$, which measures how "close" vertex v is to conflicting edges in s_i. More formally,

$$p_i(v) = \sum_{d=0}^{2} \omega_d \cdot nce_i(v, d),$$

where ω_d is a weight balancing the importance of conflicting edges located at different distances from a vertex v ($\omega_0 > \omega_1 > \omega_2$). On the basis of preliminary experiments, we have decided not to consider distances d greater than 2 because no significant improvements were observed while more computational effort was required for evaluating the penalties $p_i(v)$.

Each vertex v in the offspring s_3 to be created will be colored either with color $c_1(v)$ or with color $c_2(v)$. Preference is given to the color $c_i(v)$ with the smallest penalty $p_i(v)$ in s_1 or s_2. If $p_1(v) = p_2(v)$, then we examine the vertices of v already colored in the current partial coloring s_3 and we choose the color $c \in \{c_1(v), c_2(v)\}$ minimizing $nadj(v, c)$, where $nadj(v, c)$ denotes the number of vertices adjacent to v that are colored with color c in s_3. If this criterion is not enough to decide between $c_1(v)$ and $c_2(v)$ (that is, $nadj(v, c_1(v)) = nadj(v, c_2(v))$), then we select the color of vertex v arbitrarily among $\{c_i(v), c_2(v)\}$. A formal description of this graph coloring adapted crossover operator is given in Figure 6.

EDM is redirected to different portions of the search space X by performing regularly two long-term diversification phases. The first phase consists in modifying slightly the structure of X like in the partial q-coloring problem. We have decided to reduce the number q of possible colors by one for r_{chX} generations every $ngen_{chX}$ generation. The iterated greedy algorithm presented in Culberson (1992) governs the second long-term diversification phase of EDM. Let $\hat{s}$ be a q-coloring of G (that is, $f(\hat{s}) = 0$). A permutation π of the vertices $v \in V$ is said *$\hat{s}$-compatible* if it rearranges the color classes of $\hat{s}$ in the following way: $c(v_{\pi(i)}) = c(v_{\pi(j)} = c \Rightarrow c(v_{\pi(k)}) = c \; \forall i \leq k \leq j$. It can be shown easily that a greedy algorithm for graph coloring based on a $\hat{s}$-compatible permutation produces a new acceptable coloring s' using q or fewer colors. This property can be extended to feasible colorings $s \in X$. By applying a greedy algorithm to a s-compatible permutation, we produce a new feasible coloring s' in at most q colors such that $f(s') \leq f(s)$. The second long-term diversification phase is activated once every $ngen_{igr}$ generation, and it consists in applying iteratively a greedy algorithm of this type to each solution $s \in \mathcal{P}$ produced by the descent method in EDM. Starting with a solution $s' = s \in \mathcal{P}$ a sequence of r_{igr} different feasible solutions s' is created by generating successively $nperm_{igr}$ s'-compatible permutations π. Following the observations made in Culberson (1992), the permutations π we generate are either random or based on the order or the size of the classes V_i $(1 \leq i \leq q)$ of the current solution s'. The best solution produced in this way replaces the initial solution s.

Input $s_1, s_2 \in X$
Output $s_3 = (V_1, \ldots, V_q) \in X$

$nadj(v, c) = 0 \; \forall v \in V, \forall c = 1, \ldots, q;$
$V_c := \emptyset \; \forall c = 1, \ldots, q;$

For every vertex $v \in V$ **do**
  If $p_1(v) < p_2(v)$ **then** $c_3(v) = c_1(v)$
  else If $p_2(v) < p_1(v)$ **then** $c_3(v) = c_2(v)$
  else (*$p_1(v) = p_2(v)$*)
    If $nadj(v, c_1(v)) < nadj(v, c_2(v))$ **then** $c_3(v) = c_1(v)$
    else If $nadj(v, c_2(v)) < nadj(v, c_1(v))$ **then** $c_3(v) = c_2(v)$
    else (*$nadj(v, c_1(v)) = nadj(v, c_2(v))$*)
      $c_3(v) = c_1(v)$ or $c_2(v)$ arbitrarily;
  $nadj(v', c_3(v)) = nadj(v', c_3(v)) + 1$ for every vertex v' adjacent to v;
  $V_{c_3(v)} = V_{c_3(v)} \cup \{v\}$;

Figure 6. A crossover operator for the q-coloring problem.

As observed in Chams, Hertz, and de Werra (1987), Fleurent and Ferland (1995), Hertz and de Werra (1987), and Johnson et al. (1991), it is difficult, if not impossible, to find a q-coloring of a large graph G (more than 300 nodes) with q close to $\chi(G)$ by applying directly a given algorithm $\mathfrak{A}$ on the graph G. The approach proposed in Chams, Hertz, and de Werra (1987), Fleurent and Ferland (1995), Hertz and de Werra (1987), and Johnson et al. (1991) consists in constructing consecutively color sets (that is, stable sets) of G that are as large as possible until we are left with at most a certain number n_{left} of vertices. The algorithm $\mathfrak{A}$ is then invoked to color the remaining vertices. In this article we generalize this approach by removing consecutively partial q'-colorings ($1 \leq q' < q$) instead of stable sets ($q' = 1$) of G. The removal of partial q'-colorings as well as the coloring of the residual graph will be performed by the two adaptations of EDM presented in this article. In order to reduce the number of edges in the remaining uncolored subgraph, we have observed that it is advantageous to remove partial q'-colorings of G that are connected to as many noncolored vertices as possible. The external degree of a partial q-coloring is defined as the total number of edges with exactly one colored endpoint. If EDM has to choose between two feasible partial q'-colorings of equal value, then preference will be given to the one with the largest external degree (Johnson et al., 1991; Fleurent and Ferland, 1995). Moreover, once a legal partial q'-coloring has been found ($f(s) = 0$) we run EDM for $ngen^+$ additional generations with the aim of finding new legal partial q'-colorings that are better from the external degree point of view.

5. Computational Experiments

5.1. Random Graphs

Randomly generated graphs are used to test the efficiency and examine the average behavior of the adaptations of EDM presented in the two previous sections. A random graph $G_{n,p}$ is a graph with n vertices such that there exists an edge with a probability p between any pair of vertices, independently of the existence or nonexistence of any other edge. This family of graphs has been deeply studied with respect to coloring, especially for $p = 0.5$. There exists some asymptotic results but they are of little practical use in evaluating algorithm performance (Bollobas, 1985). Probabilistic estimates of $\alpha_q(G_{n,p})$ and $\chi(G_{n,p})$ can be obtained easily as shown in Johri and Matula (1982). Let

$$E(G_{n,p}, j) = \binom{n}{j} \cdot (1 - p)^{\binom{j}{2}}$$

be the expected number of stable sets of size j in $G_{n,p}$. The size $\alpha(G_{n,p})$ of the largest stable set in $G_{n,p}$ can be estimated by computing the maximum value of j such that $E(G_{n,p}, j)$ is greater than one—that is, $\alpha(G_{n,p}) = max(j \mid E(G_{n,p}, j) \geq 1)$. By sequentially deleting an independent set of maximum expected size, we obtain the following estimates of $\alpha_q(G_{n,p})$ and $\chi(G_{n,p})$:

1. Estimate of $\alpha_q(G_{n,p})$:
   $\tilde{\alpha}_q(G_{n,p}) := 0;$
   $n' := n;$
   For $r := 1$ **to** q **do**
      $\tilde{\alpha}_q(G_{n,p}) := \tilde{\alpha}_q(G_{n,p}) + \tilde{\alpha}(G_{n',p});$
      $n' := n' - \alpha(G_{n',p});$
2. Estimate of $\chi(G_{n,p})$:
   $\tilde{\chi}(G_{n,p}) := 0;$
   $n' := n;$
   While $n' > 0$ **do**
      $\tilde{\chi}(G_{n,p}) = \tilde{\chi}(G_{n,p}) + 1;$
      $n' := n' - \tilde{\alpha}(G_{n',p});$

These estimates assume that the edge density remains constant after each stable set deletion. Although it is not true in general, these estimates also assume that an optimal partial q-coloring of G, or $\chi(G)$-coloring, is made up of optimal stable sets. A counterexample is provided in Figure 7. Since the graph G contains a clique of size 3, there exists at least one uncolored vertex in every partial 2-coloring of G ($\Rightarrow \alpha_2(G) \leq n - 1 = 6$). It follows that the partial 2-coloring of size 6 of Figure 7 is optimal. This 2-coloring is made up of two stable sets of size 3, whereas the stability number of G is 4. It can easily be seen that there exists no partial 2-coloring of size larger than 5 that contains the unique optimal stable set of G. Given an edge density p and a number q of colors, a number n of vertices will be referred to as q-stability break point, respectively a chromatic break point, if $\tilde{\alpha}_q(G_{n',p})$, respectively $\tilde{\chi}(G_{n',p})$, increases as n' goes from $n - 1$ to n.

The algorithms described in Sections 3 and 4 were implemented in Pascal and run on a Silicon Graphics Workstation (9 Mflops). Computing times are reported in CPU seconds of this machine. For each value of $p \in \{0, 4, 0.5, 0.6\}$ a sample of 20 $G_{100,p}$ graphs, 10 $G_{300,p}$ graphs, 5 $G_{500,p}$ graphs, and 2 $G_{1000,p}$ graphs is taken into account in our experiments. Each class of graphs is generated using a combination of multiplicative linear congruential generators that is described in detail in L'Ecuyer (1988). For $p = 0.5$ we consider the same random graphs as Fleurent and Ferland (1995). Note that the sample contains one $G_{500,0.5}$ and one $G_{1000,0.5}$ graph that have been used in Johnson et al. (1991) for testing various algorithms. All the results reported in the subsequent tables are average results obtained after one run of EDM for each graph in a class $G_{n,p}$.

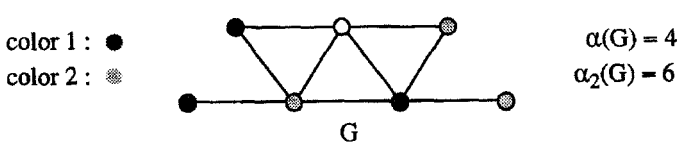


Figure 7. An optimal partial 2-coloring that does not contain an optimal stable set.

5.2. *Adjustment of the Parameters*

Various experiments have been carried out with the computer code in order to select appropriate values of the parameters governing the search in our two adaptations of EDM. The following values have yielded good results on average: $ngen_{mut} = 60$, $r_{mut} = 10$, $ngen_{chX} = 100$, $r_{chX} = 10$, (*partial q-coloring*), $ngen_{chX} = 200$, $r_{chX} = 60$ (*q-coloring*), $ngen_{igr} = 100$, $nperm_{igr} = 50$, $ngen^{+} = 100$. The other parameters need to be set more carefully.

Let us examine first the weights w_0, w_1, and w_2, which define the crossover operator of the q-coloring problem. For this purpose we consider the problem of finding colorings in $\tilde{\chi}(G_{100,0.5}) = 16$ colors to the 20 graphs in the class $G_{100,0.5}$. The results we have obtained with four sets of weights are sketched in Table 1 ($m = 10$, $p^0_{mut} = 0.04$, $\Delta w = 0.03$). *CPU*, *ngen*, Δf, *nstep*, and *entr* are average values that denote, respectively, the CPU time needed to achieve a coloring, the number of generations of EDM, the increase (in percent) of the objective function f after a crossover phase, the number of steps needed by the descent method for finding a local optimum, and the entropy of the population. The entropy is a value in the interval [0, 1] that measures the diversity of a population of solutions (Fleurent and Ferland, 1995). A nil entropy characterizes a population of identical colorings whereas an entropy close to 1 indicates that the colors of vertices are uniformly distributed within the population.

We observe that it is worth, in terms of the CPU time and the number of generations *ngen*, to take account of the conflicting edges located at a distance 1 and 2 from a vertex when computing its penalty. By giving some importance to these remote conflicts, we reduce considerably the increase of f after performing the crossover operator, but we lose diversity within the population. We see that the higher Δf is the more steps the descent method has to perform in order to achieve a local optimum. The best results have been obtained by considering all conflicting edges up to a distance 2 and by differentiating slightly the edges located at a distance 1 and 2. In the subsequent experiments we will use exclusively the set (100, 3, 1).

The simultaneous influence of p^0_{mut} and Δw on the performance of EDM is studied below for 300 and 500 vertex graphs of density $p = 0.5$. The results we have obtained for three test problems with a population $\mathcal{P}$ of size $m = 10$ are presented in Tables 2, 3, and 4. Similar results that we do not report here have been observed for other values of p, m, and n. The values shown in the following tables correspond to the average CPU time needed to achieve a solution. If EDM does not manage to find a solution in less than $ngen_{max} = 5{,}000$ for all the graphs in a class, then the success rate of EDM is indicated in parenthesis.

We see that the parameters Δw and p^0_{mut} have the same effect on the performance of EDM in that, for a fixed value of Δw (respectively p^0_{mut}), the elapsed CPU time decreases

Table 1. Influence of the weights w_0, w_1, and w_2.

(w_0, w_1, w_2)	*CPU* (sec)	*ngen*	Δf (%)	*nstep*	*entr*
(1, 0, 0)	4.20	28	350.2	38.6	0.53
(10, 1, 0)	3.05	23	198.3	25.9	0.40
(100, 10, 1)	2.30	17	164.1	24.5	0.39
(100, 3, 1)	1.90	15	169.0	23.6	0.38

Table 2. Variation of Δw and p^0_{mut} in the partial 2-coloring problem ($\rightarrow \tilde{\alpha}_2(G_{500,0.5}) = 26$).

$\Delta w \backslash p^0_{mut}$	0	0.02	0.04	0.06	0.08	0.10
0	(0/5)	740.5 (2/5)	432.0	628.6	379.2	502.2
0.01	607.0 (1/5)	295.6	100.2	131.0	226.2	469.8
0.03	437.8	130.6	77.0	131.8	128.2	232.4
0.05	190.6	82.8	73.2	119.8	253.4	372.2
0.07	185.6	51.6	85.0	68.4	131.0	445.4
0.09	200.2	81.8	55.0	174.6	220.0	217.8

Table 3. Variation of Δw and p^0_{mut} in the partial 3-coloring problem ($\rightarrow \tilde{\alpha}_3(G_{500,0.5}) = 39$).

$\Delta w \backslash p^0_{mut}$	0	0.02	0.04	0.06	0.08	0.10
0	(0/5)	(0/5)	1,367.7 (3/5)	3,731.5 (2/5)	1,200.0 (1/5)	(0/5)
0.01	(0/5)	943.2	412.6	2,249.5 (4/5)	1,108.0 (1/5)	(0/5)
0.03	535.5 (2/5)	161.6	492.8	1,386.3 (3/5)	3,505.0 (1/5)	(0/5)
0.05	533.4	253.4	326.8	2,621.0	4,135.0 (2/5)	(0/5)
0.07	157.7 (4/5)	209.8	302.0 (5/5)	2,610.0 (3/5)	5,384.0 (3/5)	(0/5)
0.09	340.8 (5/5)	248.0	367.6 (5/5)	2,293.2 (4/5)	4,728.0 (1/5)	(0/5)

Table 4. Variation of Δw and p^0_{mut} in the 35-coloring problem ($\rightarrow \tilde{\chi}(G_{300,0.5}) = 35$).

$\Delta w \backslash p^0_{mut}$	0	0.02	0.04	0.06	0.08	0.10
0	(0/10)	1,198.9 (6/10)	1,316.9 (6/10)	1,289.5	846.1 (9/10)	795.0 (8/10)
0.01	535.3 (9/10)	279.0	280.5	338.2	1,007.0	1,130.4 (5/10)
0.03	349.2	180.9	192.7	370.4	910.7 (9/10)	1,353.0 (7/10)
0.05	362.8	215.3	208.5	273.8	1,059.9	1,522.8 (9/10)
0.07	474.6	213.3	194.3	422.7	720.7 (9/10)	1,570.0 (9/10)
0.09	344.4	136.6	227.0	360.6	670.1 (9/10)	1,454.9 (8/10)

first and then increases when p^0_{mut} (respectively Δw) grows. The best results are obtained with intermediate values of Δw and p^0_{mut}—namely, $0.03 \leq \Delta w \leq 0.07$ and $0.02 \leq p^0_{mut} \leq 0.06$. For smaller and larger values of these parameters EDM does not manage to find conflict free solutions for all the graphs in the class. Based on these observations it has been decided to set $p^0_{mut} = 0.04$ and $\Delta w = 0.03$ in the subsequent experiments.

To find an appropriate value for the size m of the population $\mathcal{P}$ we investigate the behavior of EDM with m ranging from 2 to 20. Tables 5 and 6 show the results produced by EDM when applied to the problems of finding a partial 2-coloring and a partial 3-coloring in $G_{500,0.5}$, as well as a 35-coloring in $G_{300,0.5}$ and a 50-coloring in $G_{500,0.5}$. In the latter case we have decided to use EDM to first remove stable sets in the original graph until we are left with at most $n_{left} = 300$ vertices to color. Once again each run of EDM is stopped after $ngen_{max} = 5{,}000$ generations if no conflict-free solution has been found before. The average number of generations and the average CPU time required to find a conflict-free solution are reported in Tables 5 and 6.

We observe that the number of generations decreases with m varying from 2 to 10 and then gets rather unpredictable for larger values of m. We think that the size of the population does not have a significant influence on the quality of the results when it is large enough. Beyond a certain value of m, the extra information we gain by adding some individuals

Table 5 Variation of m in the partial q-coloring problem.

	$\rightarrow \tilde{\alpha}_2(G_{500,0.5}) = 26$		$\rightarrow \tilde{\alpha}_3(G_{500,0.5}) = 39$	
m	*ngen*	*CPU*	*ngen*	*CPU*
2	1,659 (4/5)	152.7 (4/5)	1,203 (2/5)	189.0 (2/5)
4	603	109.2	1,053	334.8
6	159	47.6	980	474.6
8	132	49.6	873	635.0
10	113	59.4	372	382.0
12	130	78.4	747	968.4
14	134	100.2	1,383	2,159.6
16	78	73.0	1,060	1,933.0
18	84	82.8	2,127	4,515.8
20	58	68.4	1,796	4,290.4

Table 6. Variation of m in the q-coloring problem.

	$\rightarrow \tilde{\chi}(G_{300,0.5}) = 35$		$\rightarrow \tilde{\chi}(G_{500,0.5}) = 50$	
m	*ngen*	*CPU*	*ngen*	*CPU*
2	1,558	204.8	4,940 (1/5)	1,886.0 (1/5)
4	920	239.9	2,444 (3/5)	2,733.7 (3/5)
6	662	261.9	4,238 (4/5)	4,237.0 (4/5)
8	375	198.4	1,331	3,908.2
10	285	192.0	1,701	4,964.2
12	237	195.4	1,883	6,994.6
14	181	178.5	1,382	6,655.2
16	240	268.2	1,593	7,631.4
18	226	284.2	865	8,361.6
20	173	245.8	1,591	10,045.2

in the population does not really help. As we use a nonparallel machine, the CPU time is clearly proportional to the size of the population m and the number of generations *ngen*. From a computational point of view the best results have been obtained with m ranging from 6 to 14.

Hereafter we report the results obtained by EDM on the samples of random graphs described above. The size m of the population has been set to 10. EDM is stopped if it fails to find a conflict-free q-coloring, respectively a conflict-free q-coloring, in less than $ngen_{max}$ = 5,000, respectively $ngen_{max}$ = 50,000 generations. In this case the success rate of EDM is shown in parenthesis and the average CPU times reported do not take account of the unsuccessful runs of EDM.

5.3. Results

Let us deal at first with the problem of finding an optimal stable set in a graph. For this purpose we run the branch and bound algorithm TABARIS, which is described in Friden,

Hertz, and de Werra (1990). TABARIS is able to find an optimal set for each of the graphs considered, except for the two graphs in the class $G_{1000,0.4}$ where too much computing time is necessary.

Table 7 reports, for every class $G_{n,p}$, the estimate of the stability number presented above, the minimum, maximum, and average stability number as well as the average CPU time needed by TABARIS and EDM to achieve an optimal stable set. In the class $G_{1000,0.4}$ the estimate $\tilde{\alpha}(G_{n,p})$ replaces the stability number, which is unknown. As the variation of the stability number within a class decreases considerably when the size n increases, it follows that the stability number of the two graphs in the class $G_{1000,0.4}$ is very likely to be equal to 19. We observe that EDM finds an optimal stable set in each graph where the stability number is known in a reasonable computational effort when compared to TABARIS. Contrary to EDM whose running time seems to depend essentially on the size n, the running time of TABARIS grows exponentially with n and decreases exponentially with p. More specifically, for a given density p, the difficulty of EDM in finding an optimal stable set grows as n gets closer to its lower 1-stability break point.

Table 8 is devoted to the search of partial 2- and 3-colorings of maximum expected size in 300, 500 and 1,000 vertex graphs. Graphs on 100 nodes have been omitted for two reasons. First the q-stability number in a class $G_{100,p}$ is not unique (see Table 7), and thus the estimate $\tilde{\alpha}_q(G_{n,p})$ is not sharp. Second, random graphs on 100 vertices can be colored optimally almost instantaneously by any sophisticated sequential or evolutionary procedure.

Table 8 shows that the success rate of EDM in finding partial 2- and 3-colorings of maximum expected size is rather low. Only ten out of the eighteen runs of EDM achieve a partial q-coloring of size $k = \tilde{\alpha}_q(G_{n,p})$ for every graph in a class $G_{n,p}$. For graphs on 300 vertices we are convinced that EDM fails only when $\tilde{\alpha}_q(G_{n,p})$ is larger than $\alpha_{dq}(G_{n,p})$. For example, it is not possible to color with two colors twenty nodes in every $G_{300,0.6}$ graph when four out of these ten graphs have a stability number equal to 9 (see Table 7). For

Table 7. Optimal stable sets in 100, 300, 500, and 1,000 vertex graphs.

			$\alpha(G_{n,p})$				
n	p	$\tilde{\alpha}(G_{n,p})$	Average	Minimum	Maximum	CPU (TABARIS)	CPU (EDM)
100	0.4	12	11.35	10	13	<0.1	0.1
	0.5	9	9.10	9	10	<0.1	0.1
	0.6	8	7.15	7	8	<0.1	<0.1
300	0.4	15	15.10	15	16	149.3	2.4
	0.5	12	12.0	12	12	17.7	9.7
	0.6	10	9.60	9	10	3.6	5.8
500	0.4	17	17.0	17	17	6,556.0	58.1
	0.5	13	13.0	13	13	414.8	14.2
	0.6	11	10.2	10	11	40.2	11.6
1000	0.4	19	—	—	—	—	633.5
	0.5	15	15.0	15	15	44,209.5	578.5
	0.6	12	12.0	12	12	1,795.5	788.6

Table 8. Partial q-colorings of size k in 300, 500, and 1,000 vertex graphs.

		$q = 2$			$q = 3$		
n	p	$\tilde{\alpha}_2(G_{n,p})$	k	*CPU*	$\tilde{\alpha}_3(G_{n,p})$	k	*CPU*
300	0.4	30	30	50.7	45	45	303.0 (8/10)
	0.5	24	24	28.9 (9/10)	36	36	252.9 (9/10)
	0.6	20	19	21.8 (6/10)	29	28	32.0 (6/10)
			18	2.1		27	9.7
500	0.4	34	34	220.4	51	51	2,485.6 (4/5)
						50	1,444.5
	0.5	26	26	59.4	39	39	382.0
	0.6	22	21	9.0 (1/5)	32	31	49.5 (1/5)
			20	16.4		30	40.6
1,000	0.4	38	38	4,372.5	57	57	10,817.6
	0.5	30	30	4,317.4	45	45	6,133.4 (1/2)
						44	7,036.8
	0.6	24	24	5,128.1 (1/2)	36	35	8,611.6 (1/2)
			23	134.5		34	639.3

larger graphs we cannot be so categorical, especially for graphs on 1,000 vertices that are likely to contain partial q-colorings of expected maximum size. Larger partial q-colorings could eventually be obtained by increasing the maximum number of generations performed by EDM. To our knowledge there exists no exact algorithm for computing the q-stability number of a graph, and thus it is hard to evaluate the performance of EDM from a solution quality viewpoint.

Generally speaking the difficulty in finding partial q-colorings of size $\tilde{\alpha}_q(G_{n,p})$ grows with p and n. However, it depends also, for given values of q and p, of the remoteness between n and its lower q-stability break point. Tables 7 and 8 show that the running time of EDM grows considerably with q, whereas it has been observed that the number of generations does not vary significantly. This time increase is due to the descent method in EDM, which takes much more time to achieve local optima at each generation. The larger q is and the larger the neighborhood of a partial q-coloring is.

Tables 9 and 10 summarize the results we have obtained with EDM in an attempt of coloring the graphs considered in this article by using a number of colors q as small as possible. The results in Table 10 have been obtained by the combined method mentioned at the end of Section 4. Partial q-colorings are removed repeatedly until the residual graph is left with at most $n_{left} = 300$ vertices. Last row of Table 9 shows that the adaptation of EDM to the q-coloring problem is not sufficient to obtain colorings in 50 colors for every $G_{500,0.5}$ graph. Moreover the successful runs require more time than the combined method when applied to the same problem (see Table 10).

For a given n, we have observed that the elapsed CPU time per generation in EDM decreases as p increases. This means that the descent method finds local optima in dense graphs faster than in sparse graphs, even though the neighborhood is larger in dense graphs.

Table 9. q-colorings of 100, 300, and 500 vertex graphs.

n	p	$\tilde{\chi}(G_{n,p})$	q	*CPU*
100	0.4	14	14	<0.1
			13	0.7
			12	40.8
	0.5	16	16	1.9
			15	17.5
			14	6.0 (1/20)
	0.6	19	19	3.1
			18	165.1 (14/20)
300	0.4	28	28	201.7
			27	603.3
			26	10,348.2 (5/10)
	0.5	35	35	192.0
			34	741.2
			33	11,920.1 (7/10)
	0.6	42	42	5,252.6
			41	35,724.4 (5/10)
500	0.5	50	50	35,940.0 (2/5)

Table 10. q-colorings of 500 and 1,000 vertex graphs.

				CPU		
n	p	$\tilde{\chi}(G_{n,p})$	q	$q' = 1$	$q' = 2$	$q' = 3$
500	0.4	40	40	2,805.8	5,647.4	15,314.6
			39	5,277.2	7,594.4	16,963.6
	0.5	50	50	4,964.2	6,036.2	11,396.8
			49	11,397.8	11,152.4	17,680.0
	0.6	62	62	27,122.4	28,603.8	26,416.2
1,000	0.4	68	68	35,801.0	65,903.5	156,906.0
			67	41,106.0	67,972.0	162,817.5
	0.5	85	85	37,845.5	70,789.5	147,805.0
	0.6	106	108	69,079.0 (1/2)	120,925.0 (1/2)	—
			109	69,296.0	98,278.5	185,199.0

With the exception of the $G_{1000,0.5}$ and $G_{1000,0.6}$ graphs, we see that EDM always finds colorings in a number of colors smaller than the probabilistic estimate $\tilde{\chi}(G_{n,p})$. Only the two $G_{1000,0.6}$ graphs could not be colored in $\tilde{\chi}(G_{n,p})$ colors. EDM succeeds in coloring both of them with 108 colors but it uses two different strategies, namely $q' = 1$ for one graph and $q' = 2$ for the other. Because of the large expected running time, experiments with $q' = 3$ were not conducted. When using a combined method for coloring large graphs, we note that it is not worth extracting q'-colorings with $q' > 1$ since EDM generally requires much more computational time without producing better results.

Table 11. Results on $G_{n,0.5}$ graphs.

n	Number of Graphs in $G_{n,0.5}$	$\tilde{\chi}(G_{n,0.5})$	Hertz and de Werra (1987)		Fleurent and Ferland (1995)		Our Results (EDM)	
			q^*	$\bar{q}$	q^*	$\bar{q}$	q^*	$\bar{q}$
100	20	16	16	—	15	14.95	15	14.95
300	10	35	35	—	34	33.5	34	33.3
500	5	50	50	—	49	49.0	49	49.0
1,000	2	85	87	—	84	84.0	85	85.0

To our knowledge no experiments with colorings have been conducted on classes of random graphs of density different from 0.5. For every class $G_{n,0.5}$ the results in Tables 9 and 10 appear to be better than those reported in Hertz and de Werra (1987). Fleurent and Ferland (1995) developed in a recent study an evolutionary algorithm for coloring random graphs and Leigthon graphs. Their algorithm differs significantly from EDM in that they use

- A steady-state population where only two solutions are changed at each generation,
- An improvement of TABUCOL (Hertz and de Werra, 1987), instead of a simple descent method, for reaching good colorings at each generation,
- A modification of STABULUS (Friden, Hertz, and de Werra, 1989) for extracting stable sets as large as possible when coloring graphs with more than 300 nodes.

Despite these refinements the results for $G_{n,0.5}$ graphs which are presented in Fleurent and Ferland (1995) are very similar to those produced by EDM (as a reminder, we deal with the same set of $G_{n,0.5}$ graphs). The results obtained by Hertz and de Werra (1987) and Fleurent and Ferland (1995) as well as our results are summarized in Table 11. The value q^* denotes the smallest number of colors for which all graphs of the same class can be colored without a failure. $\bar{q}$ denotes the average number of colors needed to color the graphs in the class when trying with q^* or less colors. No running time is reported in Table 11 because all algorithms have been implemented on different computers.

It appears that the algorithm of Fleurent and Ferland (1995) succeeds in coloring the two $G_{1000,0.5}$ graphs with eighty-four colors, whereas EDM achieves colorings with one color more. On the other hand, we notice that EDM finds a larger number of 33-colorings in the class $G_{300,0.5}$. Different hybrid schemes were investigated by Fleurent and Ferland (1995). It is important to point out that EDM clearly outperforms their algorithm when TABUCOL is replaced by a descent method (like in EDM). For example, when searching for a 34-coloring in a $G_{300,0.5}$ graph, no feasible colorings with less than ten conflicts could be achieved in Fleurent and Ferland (1995), whereas every graph in the class $G_{300,0.5}$ could be colored in thirty-four colors with EDM.

5.4. Practical Considerations on the Estimates $\tilde{\alpha}(G_{n,p})$ and $\tilde{\chi}(G_{n,p})$

In this subsection we generate a sample of 100 random graphs for each value of n and for two given densities $p = 0.4$ and $p = 0.6$. Then we use the algorithms TABARIS (Friden,

Hertz, and de Werra, 1990) and EPCOT (Dubois and de Werra, 1994) to compute respectively the stability number $\alpha(G_{n,p})$ and the chromatic number $\chi(G_{n,p})$ of each graph. Both of these algorithms are based on an implicit enumeration of all possible solutions of the considered problem.

The estimate $\tilde{\alpha}(G_{n,p})$ and the mean stability number $\alpha(G_{n,p})$ obtained by TABARIS are plotted in the graphics of Figure 8 for each value of n in the interval (20,200). For $p = 0.4$ we see that $\tilde{\alpha}(G_{n,p})$ overestimates $\alpha(G_{n,p})$ for almost every n up to 100. Other experiments with small values of n ($n \leq 150$) have shown that the smaller p is and the more $\tilde{\alpha}(G_{n,p})$ overestimates $\alpha(G_{n,p})$. As expected, when n grows, the slope of the curve $\alpha(G_{n,p})$ decreases and the difference between two consecutive 1-stability break points increases. This means that the error of the estimate $\tilde{\alpha}(G_{n,p})$ decreases with n. More generally, we can say that the larger the number of edges in $G_{n,p}$, the more accurate $\tilde{\alpha}(G_{n,p})$ is. This empirical consideration is confirmed in Johri and Matula (1982) from a probabilistic point of view. The density function of $\alpha(G_{n,p})$ is shown to have a peaked behavior for large graphs.

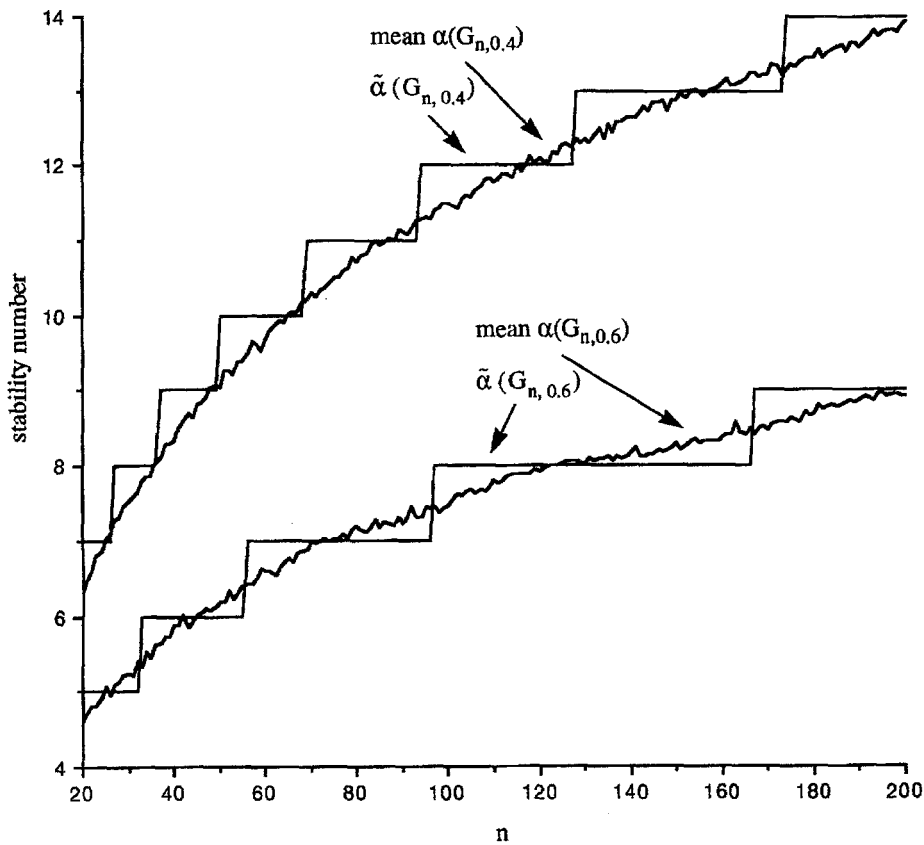


Figure 8. Comparison of $\tilde{\alpha}(G_{n,p})$ and $\alpha(G_{n,p})$ for $p = 0.4$ and $p = 0.6$.

In Figure 9 we plot the estimate $\tilde{\chi}(G_{n,p})$ and the mean chromatic number $\chi(G_{n,p})$ obtained by EPCOT for values n up to 80. The graphics we obtain provide evidence that the estimate $\tilde{\chi}(G_{n,p})$ is not very good. For a small density p, given that $\tilde{\alpha}(G_{n,p})$ often overestimates $\alpha(G_{n,p})$ for small values of n, we expected $\tilde{\chi}(G_{n,p})$ to be significantly smaller than the mean chromatic number $\chi(G_{n,p})$. Surprisingly, an opposite behavior is observed: $\tilde{\chi}(G_{n,p})$ generally overestimates $\chi(G_{n,p})$, and the gap between $\chi(G_{n,p})$ and $\tilde{\chi}(G_{n,p})$ increases when p decreases. In fact, the approach that consists of repeatedly removing a maximum stable set in a graph for computing an estimate of the chromatic number is not very appropriate. The following explanation can be given. The process often ends up with the removal of a large number of stable sets of size 1 or 2. We observed that this number tends to increase when the density p decreases. However, optimal colorings are generally made up of balanced stable sets and thus they rarely contain stable sets of size 1 or 2 and of maximum size. This fact is predicted by Morgenstern (1990) when he shows that the most likely color class size distribution has a normal form.

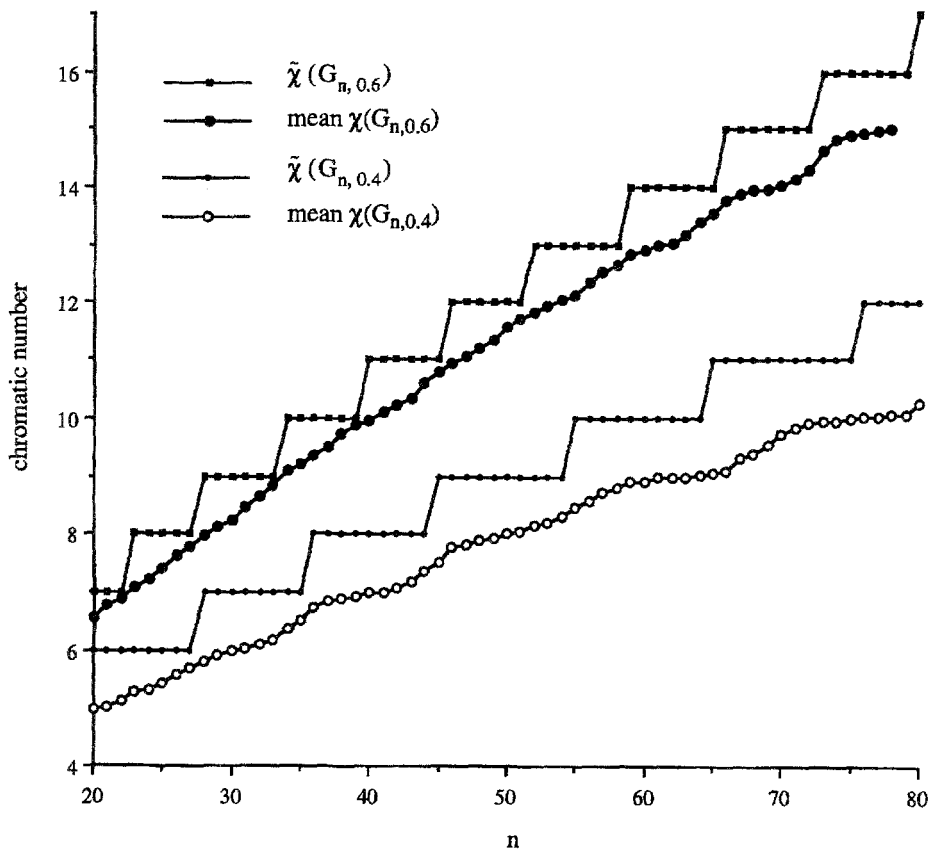


Figure 9. Comparison of $\tilde{\chi}(G_{n,p})$ and $\chi(G_{n,p})$ for $p = 0.4$ and $p = 0.6$.

6. Concluding Remarks

Despite the impressive abilities of today's computers, many COPs still exist that require a heuristic search approach in the general case. In this article we show that an intelligent embedding of a sequential procedure within an evolutionary algorithm framework results in an efficient algorithm for solving problems related to graph coloring. The potential of the hybrid method we have developed is investigated on various classes of graphs by considering two well-known combinatorial optimization problems—namely, the search for an optimal partial q-coloring and the search for the chromatic number. The results we have obtained are very satisfactory. Indeed, we managed to match most of the results obtained by a similar hybrid algorithm (Fleurent and Ferland, 1995) governed by a sophisticated adaptation of tabu search instead of a simple descent method. Our study shows that it is not essential to have recourse to a smart sequential method in a hybrid algorithm when the diversification scheme and the crossover operator are properly designed.

Our experience with the two adaptations of EDM led us to the following general comments:

- EDM is an easily implementable algorithm that can be modified, with little effort, to tackle a variety of assignment problems where there are competing objectives and multiple resources.
- Due to its asynchronicity and intrinsic parallel nature, EDM is well suited for parallel computation. The use of a parallel MIMD machine would have reduced the execution time of the algorithm by a factor equal to the size of the population approximately. Investigations to evaluate the speed-up of the algorithm have not been carried out in this study. Evolutionary algorithms form an emerging framework in computer programming that could challenge in the near future very sophisticated algorithms. The results we have obtained with EDM are encouraging for future research.

Acknowledgments

The authors would like to thank Professor Fred Glover for his careful reading of an earlier version of this article, Charles Fleurent for having provided the random graph generator used in a recent study (Fleurent and Ferland, 1995), Nicolas Dubois for the drawing of Figure 9, and two anonymous referees for their pertinent comments. Support of the Fonds National de la Recherche Scientifique under grant No. FN 21-36173.92 is gratefully acknowledged.

References

Berge, C. (1989). Minimax relations for the partial q-colorings of a graph. *Discrete Mathematics*, 74, 3–14.

Bollobas, B. (1985). *Random Graphs*. New York: Academic Press.

Brélaz, D. (1979). New methods to color vertices of a graph. *Communications of the ACM*, 22, 251–256.

Carter, B., and Park, K. (1993). How good are genetic algorithms at finding large cliques? An experimental study. Technical Report BU-CS-93-015, Boston University.

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

Colorni, A., Dorigo, M., and Maniezzo, V. (1991a). Distributed optimization by ant colonies. In F. Varela and P. Bourgine (Eds.), *Proceedings of the First European Conference on Artificial Life* (pp. 134–142). Paris: Elsevier.

Colorni, A., Dorigo, M., and Maniezzo, V. (1991b). An autocatalytic process. Technical Report 91-016, Politecnico di Milano, Dipartimento di Elettronica, Milano.

Colorni, A., Dorigo, M., and Maniezzo, V. (1992). An investigation of some properties of an ant algorithm. In R. Männer and B. Manderick (Eds.), *Proceedings of the Second Conference on Parallel Problem Solving from Nature* (pp. 509–520). Brussels: Elsevier.

Costa, D. (1995). An evolutionary tabu search algorithm and the NHL scheduling problem. *INFOR* 33(3), 161–178.

Culberson, J.C. (1992). Iterated greedy graph coloring and the difficulty landscape. Technical Report TR 92-07, University of Alberta, Department of Computing Science, Edmonton.

Davis, L. (1991). *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.

Dubois, N., and de Werra, D. (1994). EPCOT—an efficient procedure for coloring optimally with tabu search. *Computers and Mathematics with Applications* 25, 35–45.

Eglese, R.W. (1990). Simulated annealing: A tool for operational research. *European Journal of Operational Research*, 46, 271–281.

Fleurent, C., and Ferland, J.A. (1994). Genetic hybrids for the quadratic assignment problem. In P.M. Pardalos and Wolkowicz (Eds.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (Vol. 16) (pp. 173–188).

Fleurent, C., and Ferland, J.A. (1995). Genetic and hybrid algorithms for graph coloring. *Annals of Operations Research*.

Fogarty, T.C. (1989). Varying the probability of mutation in the genetic algorithm. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms*. George Mason University: Morgan Kaufmann.

Friden, C., Hertz, A., and de Werra, D. (1989). STABULUS: A technique for finding stable sets in large graphs with tabu search. *Computing*, 42, 35–44.

Friden, C., Hertz, A., and de Werra, D. (1990). TABARIS: An exact algorithm based on tabu search for finding a maximum independent set in a graph. *Computers and Operations Research*, 17(5), 437–445.

Garey, M.R., and Johnson, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: Freedman.

Gendreau, M. (1994). An appraisal of greedy heuristics for the maximum clique problem. Technical Report, Université de Montréal, Centre de Recherche sur les Transports.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

Glover, F. (1993). Scatter search and star-paths: Beyond the genetic metaphor. Technical Report, University of Colorado, School of Business, Boulder.

Glover, F. (1994). Genetic algorithms and scatter search: Unsuspected potentials. *Statistics and Computing*, 131–140.

Glover, F., Taillard, E., and de Werra, D. (1993). A user's guide to tabu search. *Annals of Operations Research*, 41, 3–38.

Grefenstette, J.J. (1987). Incorporating problem scpecific knowledge into genetic algorithms. In L. Davis (Ed.), *Genetic Algorithms and Simulated Annealing* (pp. 42–60). Cambridge: Morgan Kaufmann.

Hertz, A. (1995). Polynomially solvable cases for the maximum stable set problem. *Discrete Applied Mathematics* 60, 195–210.

Hertz, A., and de Werra, D. (1987). Using tabu search for graph coloring. *Computing*, 39, 345–351.

Hertz, A., Taillard, E., and de Werra, D. (1995). Tabu search. In J.K. Lenstra (Ed.), *Local Search in Combinatorial Optimization*. Wiley.

Johnson, D.S., Aragon, C.R., McGeoch, L.A., and Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation. Part II, graph coloring and number partitioning. *Operations Research*, 39, 378–406.

Johri, A., and Matula, D.W. (1982). Probabilistic bounds and heuristic algorithms for coloring large random graphs. Technical Report 82-CSE-06, Southern Methodist University, Department of Computing Science, Dallas.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, 220, 671–680.

L'Ecuyer, P. (1988). Efficient and Portable Combined Random Number Generators. *Communications of the ACM*, 31, 742–774.

Liepins, G.E., and Hilliard, M.R. (1989). Genetic algorithms: Foundations and applications. *Annals of Operations Research*, 21, 31–58.

Mannino, C., and Sassano, A. (1992). An exact algorithm for the maximum stable set problem. Working Paper No. 334, IASI-CNR, Roma.

Morgenstern, C. (1990). Algorithms for General Graph Coloring. Doctoral dissertation, University of New Mexico, Department of Computer Science, New Mexico.

Moscato, P. (1993). An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. *Annals of Operations Research*, 41, 85–121.

Mühlenbein, H., Gorges-Schleuter, M., and Krämer, O. (1988). Evolution algorithms in combinatorial optimization. *Parallel Computing*, 7, 65–85.

Reeves, C.R. (Ed.). (1993). *Modern Heuristic Techniques for Conbinatorial Optimization*. Oxford: Blackwell Scientific.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 2–9). San Mateo, CA: Morgan Kaufmann.

Thangiah, S.R., Osman, I.H., and Sun, T. (1994). Hybrid genetic algorithm, simulated annealing and tabu search methods for vehicle routing problems with time windows. Research Report, Artificial Intelligence and Robotics Laboratory, Computer Science Department, Slippery Rock University, Slippery Rock, USA.