CAHIER DU LAMSADE

Laboratoire d'Analyse et Modélisation de Systèmes pour l'Aide à la Décision
(Université Paris-Dauphine)
Unité de Recherche Associée au CNRS UMR 7024

POLYNOMIAL APPROXIMATION AND GRAPH-COLORING

CAHIER N° 187
janvier 2002

Mar[illegible]
Vangelis PASCHOS [2]

received: October 2001.

---

[1] L[illegible], Avenue Bernard Hirsch, 95021 C[illegible]se Cedex (France) (e-mail: [illegible]).

[2] *LAMSADE, Université Paris-Dauphine, Place du Maréchal De Lattre de Tassigny, 75775 Paris Cedex 16, France (paschos@lamsade.dauphine.fr).*

Approximation polynomiale et coloration de graphes

Résumé

Considérons un graphe $G = (V, E)$ d'ordre n. Le problème de la coloration minimum consiste à attribuer d'un nombre minimum de couleurs aux sommets de V de façon à ce que deux sommets adjacents ne soient pas coloriés avec la même couleur. Ce problème est parmi les premiers démontrés intrinsèquement difficiles et, par conséquent, il est très improbable qu'il puisse être résolu optimalement par un algorithme polynomial. Dans cet article, nous faisons un tour d'horizon des principaux algorithmes d'approximation (ceux pour lesquels des rapports d'approximation théoriques ont été étudiés) pour le problème de coloration et discutons leurs rapports d'approximation et leur complexité. Enfin, nous proposons une amélioration du rapport d'approximation pour ce problème.

Mots-clé : graphe, coloration, complexité, NP-complet, algorithme d'approximation.

Polynomial approximation and graph-coloring

Abstract

Consider a graph $G = (V, E)$ of order n. In minimum graph-coloring problem we try to color V with as few colors as possible so that no two adjacent vertices receive the same color. This problem is among the first ones proved to be intractable and hence, it is very unlikely that an optimal polynomial-time algorithm could ever be devised for it. In this paper, we survey the main polynomial time approximation algorithms (the ones for which theoretical approximability bounds have been studied) for the minimum graph-coloring and we discuss their approximation performance and their complexity. Finally, we further improve the approximation ratio for graph-coloring.

Keywords: graph, coloring, complexity, NP-complete, approximation algorithm.

Table of Contents

1 Introduction

Consider a graph $G = (V, E)$ of order n. In *minimum graph-coloring problem* (C), we wish to color V with as few colors as possible so that no two adjacent vertices receive the same color. This problem was shown to be NP-hard in Karp's original paper ([40]), and remains NP-complete even restricted to graphs of constant (independent on n) chromatic number at least 3 (for more informations about the complexity of numerous restrictions, or generalizations of C, the interested reader is also refereed to [35]). The *chromatic number* of a graph, denoted by $\chi(G)$, is the smallest number of colors that can feasibly color its vertices. A graph G is called *k-colorable* if its vertices can be legally colored by k colors, in other words if its chromatic number is at most k; it will be called *k-chromatic* if k is its chromatic number.

Since adjacent vertices are forbidden to be colored with the same color, a feasible solution of C can be seen as a partition of V into vertex-sets such that, for each one of these sets, no two of its vertices are mutually adjacent. Such sets are usually called *independent sets*. So, the optimal solution of C is a *minimum-cardinality partition into independent sets*.

Another combinatorial quantity defined on discrete structures that will be useful in the sequel is the one of *set-covering*. Given a family $\mathcal{S}$ of sets drawn from a ground set C (satisfying $\cup_{S_i \in \mathcal{S}} S_i = C$), a set-covering is a family $\mathcal{S}' \subseteq \mathcal{S}$ such that $\cup_{S_i \in \mathcal{S}'} S_i = C$.

Both quantities, independent set and set-covering, give rise to two well-known NP-hard optimization problems, namely the one of finding the *maximum-cardinality independent set of a graph*, denoted by $\alpha(G)$, and the one of finding the *minimum-cardinality set-covering of a set system*. In what follows we will denote by IS the former and by SC the latter.

Given the computational hardness of C, if one wishes to devise fast algorithms, then he/she has to develop polynomial time methods providing, for all instances, feasible solutions, the values of which are as close as possible to the value of the optimal ones. This is what in the literature is commonly called *polynomial time approximation algorithms* (PTAA).

The "goodness" of a PTAA A is measured by the so-called *approximation ratio*. Two types of ratios have been used until now: the one that, in some way, compares the value $\mathtt{A}(I)$ of the approximate solution for I, provided by A, with the value $\mathrm{OPT}(I)$ of the optimal one, and the one comparing $\mathtt{A}(I)$ not only with $\mathrm{OPT}(I)$ but also with $\mathrm{WORST}(I)$, the value of the worst feasible solution of I. Each one of these measures draws its proper working-framework into which every NP-hard problem can be analyzed and implies its proper approximation results. This is, as we shall see in the sequel, particularly true for C (in general, given an NP-hard problem, approximation results can be very different when using the one working-framework or the other). For reasons of simplicity, we will speak about *ρ-framework* to denote the former and about *δ-framework* to denote the latter. Informally:

- in the ρ-framework, the comparison between $\mathtt{A}(I)$ and $\mathrm{OPT}(I)$ is usually performed by answering the following question: *"what is the ratio between* $\mathtt{A}(I)$ *and* $\mathrm{OPT}(I)$*?"*; this question induces the so-called *approximation ratio*;
- in the δ-framework, the comparison between $\mathtt{A}(I)$, $\mathrm{OPT}(I)$ and $\mathrm{WORST}(I)$ is performed by answering the question: *"what is the least ϵ for which A is to an extent of* $(1-\epsilon)$ *like the worst algorithm, and to an extent of ϵ like the ideal one?"*; this question induces the *differential-approximation ratio*.

Let A be a PTAA for an NP-hard problem Π, let $\mathcal{I}$ be the set of instances of Π, let $\mathtt{A}(I)$ be the value of the solution provided by A on an instance $I \in \mathcal{I}$, $\mathrm{OPT}(I)$ be the value of the optimal solution for I and $\mathrm{WORST}(I)$ be the value of the worst solution of I. Moreover, let for a fixed constant M, $\mathcal{I}_M$ be the subset of $\mathcal{I}$ defined as $\mathcal{I}_M = \{I \in \mathcal{I} : \mathrm{OPT}(I) \geqslant M\}$, and let $\sigma(I)$ be the number of the feasible solutions of I.

- **Approximation ratio ρ:**
  - the *approximation ratio $\rho_{\mathtt{A}}(I)$ of the algorithm $\mathtt{A}$ on an instance $I \in \mathcal{I}$* is defined as
$$\rho_{\mathtt{A}}(I) = \frac{\mathtt{A}(I)}{\mathrm{OPT}(I)}$$
  - the *approximation ratio $\rho_{\mathtt{A}}$ of $\mathtt{A}$ for* Π is defined as
$$\rho_{\mathtt{A}} = \begin{cases} \sup\{r : \rho_{\mathtt{A}}(I) > r, \forall I \in \mathcal{I}\} & \Pi \text{ a maximization problem} \\ \inf\{r : \rho_{\mathtt{A}}(I) < r, \forall I \in \mathcal{I}\} & \Pi \text{ a minimization problem} \end{cases}$$
  - the *asymptotic approximation ratio $\rho_{\mathtt{A}}^{\infty}$ of $\mathtt{A}$ for* Π is defined as
$$\rho_{\mathtt{A}}^{\infty} = \begin{cases} \sup\{r : \exists M, \rho_{\mathtt{A}}(I) \geqslant r, \forall I \in \mathcal{I}_M\} & \Pi \text{ a maximization problem} \\ \inf\{r : \exists M, \rho_{\mathtt{A}}(I) \leqslant r, \forall I \in \mathcal{I}_M\} & \Pi \text{ a minimization problem} \end{cases}$$
  - the *approximation ratio ρ_{Π} for* Π is defined as
$$\rho_{\Pi} = \begin{cases} \max\{\rho_{\mathtt{A}} : \mathtt{A} \text{ a PTAA for } \Pi\} & \Pi \text{ a maximization problem} \\ \min\{\rho_{\mathtt{A}} : \mathtt{A} \text{ a PTAA for } \Pi\} & \Pi \text{ a minimization problem} \end{cases}$$
- **Differential-approximation ratio δ:**
  - the *differential-approximation ratio $\delta_{\mathtt{A}}(I)$ of the algorithm $\mathtt{A}$ on an instance $I \in \mathcal{I}$* is defined as
$$\delta_{\mathtt{A}}(I) = \frac{\mathrm{WORST}(I) - \mathtt{A}(I)}{\mathrm{WORST}(I) - \mathrm{OPT}(I)}$$
  - the *differential-approximation ratio $\delta_{\mathtt{A}}$ of $\mathtt{A}$ for* Π is defined as
$$\delta_{\mathtt{A}} = \sup\{r : \delta_{\mathtt{A}}(I) > r, I \in \mathcal{I}\}$$
  - the *asymptotic differential-approximation ratio $\delta_{\mathtt{A}}^{\infty}$ of $\mathtt{A}$ for* Π is defined as
$$\delta_{\mathcal{A}}^{\infty} = \lim_{k \to \infty} \sup_{\substack{I \\ \sigma(I) \geqslant k}} \left\{ \frac{\mathrm{WORST}(I) - \mathtt{A}(I)}{\mathrm{WORST}(I) - \mathrm{OPT}(I)} \right\}$$
  - the *differential approximation ratio δ_{Π} for* Π is defined as
$$\delta_{\Pi} = \max\{\rho_{\mathtt{A}} : \mathtt{A} \text{ a PTAA for } \Pi\}$$

Let us note that the best expected case for the behavior of an approximation algorithm is that it could guarantee an approximation ratio tending to 1. More formally, the ideal would be that we had a PTAA $\mathtt{A}$ receiving as inputs an instance I of an NP-hard problem and a fixed constant ϵ and guaranteeing approximation ratio (resp., differential-approximation ratio) $1 + \epsilon$ or $1 - \epsilon$ depending on if the problem at hand is a minimization or a maximization one (resp., $1 - \epsilon$, for the δ-framework), for every $\epsilon > 0$. In this case, we have, in fact, a sequence $(\mathtt{A}_\epsilon)_{\epsilon > 0}$ of approximation algorithms having the desired approximation properties just described. Such a sequence is called a *polynomial time approximation schema* (PTAS) or, in the δ-framework, *differential PTAS* (DPTAS). Moreover, we can further classify such schemata following the time-complexity of algorithm $\mathtt{A}_\epsilon$. We so speak about PTAS (DPTAS) if its complexity is $O(f(|I|)^{\Theta(1/\epsilon)})$ and about *fully PTAS* (FPTAS), or *fully DPTAS* (FDPTAS) if its complexity

is $O(g(1/\epsilon)O(|I|^k))$, where $|I|$ is the size of I, f and g polynomials not depending on $|I|$ and k a fixed constant (not depending neither on $|I|$, nor on ϵ). A PTAS, or FPTAS (resp., DPTAS, or FDPTAS) is called *asymptotic* if, for every instance I, it guarantees approximation ratio $1+\epsilon+c/\mathrm{OPT}(I)$ or $1-\epsilon-c/\mathrm{OPT}(I)$, depending on if the problem at hand is a minimization or a maximization one (resp., $1-\epsilon-c/(\mathrm{WORST}(I)-\mathrm{OPT}(I))$, for the δ-framework), for some fixed constant c and for every $\epsilon > 0$.

The approximation of C by efficient algorithms is a central problem in complexity theory. In ρ-framework, people knew quite early that a *polynomial time approximation schema* cannot exist for it, since a lower bound 2 was proved for the ratio of every PTAA supposed solving C; this result, due to Garey and Johnson, has been published in 1976. But even if the researchers conjectured that approximation ratios greater than 2 were equally unlikely, such a result was not formally produced up to the early 90's when the impossibility of approximating C by a constant ratio approximation algorithm is proved by Lund and Yannakakis. This result, motivates from then on, many researchers for searching either for approximation algorithms with improved approximation ratios, or to strengthen the existing inapproximability results. On the other hand, in δ-framework, C is better-approximable than in the ρ-one, since constant differential-approximation ratio PTAAs exist since 1994.

This paper, even if it surveys both positive and negative approximation results[1] about C in both the approximation frameworks, is rather oriented towards the positive ones. For this reason, positive results are presented and commented in detail and the underlying algorithms are specified in a kind of "pseudo-PASCAL", while the inapproximability ones are simply mentioned. Before presenting positive results, we give the intuition (or key-idea) behind them. The objective of this survey is double: to present already known results and to introduce some new ones. In order to be as short and easy to read as possible, only new results are proved, while already existing results are stated without proofs. Also, in section 3 a coloring method is discussed using a randomized algorithm that has been derandomized later by rather complicated techniques. Since both random algorithm itself and its derandomization are long, and on the other hand, paper does not deal with probabilistic methods, only the underlying idea is discussed, while we omit to specify the overall algorithm.

Let us consider a simple undirected connected graph $G = (V, E)$ of order n. Sometimes, we will denote by $|G|$ the order of G and by $V(G)$ its vertex-set. Given a set V' of vertices of a graph G, we will denote by $G[V'] = (V', E(G[V']))$ the partial subgraph of G induced by V'. For every vertex $v \in V$, we denote by $\Gamma_G(v)$ the set of neighbors of v (neighborhood of v) and by $d^\circ(v)$ the quantity $|\Gamma_G(v)|$, this quantity is classically called the degree of v; $\Delta(G) = \max_{v\in V}\{d^\circ(v)\}$ is the maximum degree of G. The notion of neighborhood can be extended to apply to a set $V' \subseteq V$; we denote by $\Gamma_G(V')$ the set of neighbors of the vertices of V', i.e., $\Gamma_G(V') = \cup_{v\in V'}\Gamma_G(v)$. Whenever no ambiguity can occur we use notation $\Gamma(v)$ instead of $\Gamma_G(v)$. Given vertices v and u, we denote by $d(v,u)$ their distance, i.e., the length (number of edges) of the shortest elementary path linking v and u (since G is undirected $d(v,u) = d(u,v)$). By K_t, we denote a complete graph on t vertices. All logarithms of the paper are to the base 2 unless otherwise noted. Finally, the following well-known expression ([10]) links $\chi(G)$ and $\alpha(G)$ and will be used later:

$$\alpha(G)\chi(G) \geqslant |G| \tag{1}$$

[1]For an NP-hard minimization (resp., maximization) problem Π, an approximation result is called *positive* if, given an approximation algorithm A, it provides an upper (resp., lower) bound on the approximation ratio guaranteed by A when used to solve Π; it is called *negative* or *inapproximability* result when it provides a lower (resp., upper) bound for the ratio of an algorithm or of a whole class of approximation algorithms for Π

Part I
Graph-coloring in standard approximation

2 How can one always color a graph with $\Delta(G)$ colors?

We first recall some standard graph-theoretic concepts used in what follows in this section (for more details one can be referred to [10, 15]). Consider a *connected graph* $G = (V, E)$. A subset $A \subseteq V$ is called *articulation set* if $G[V \setminus A]$ is not connected; an articulation set of size 1 will be called *articulation point*. A graph G is *k-connected* iff $|G| \geqslant k+1$ and iff it does not contain articulation set of size less than k. As a consequence, G is *biconnected* iff $|G| \geqslant 3$ and it does not admit articulation points (i.e., for every pair $(x, y) \in V \times V$, there exist two vertex-disjoint elementary paths between x and y). A *bloc* in G is a set $A \subseteq V$ such that $G[A]$ is connected, without articulation points, and maximal (i.e., for any $v \in V \setminus A$, $G[A \cup \{v\}]$ is either not connected, or it has at least an articulation point). Then if $|A| > 2$, $G[A]$ is biconnected, while, if $|A| = 2$, it is an *isthmus*, i.e., an edge whose removal increases the number of the connected components of the graph. Obviously, the set of blocs of G covers V. A bloc B is called *extremal* if it contains a vertex z such that for any other bloc B', either B and B' are vertex-disjoint, or z is their only common vertex.

2.1 A non-constructive theorem and its ulterior constructive proof

Theorem 1 below (known also as Brook's theorem and non-constructively proved in [17]) was constructively proved by Lovász in [42] in a rather condensed way. We give in this section a personal interpretation of this result. The proofs are given in the appendix.

Theorem 1. *([17]) If G is connected with $\Delta(G) \geqslant 3$ and if it contains no subgraph $K_{\Delta(G)+1}$, then it is $\Delta(G)$-colorable.*

Consider a graph G, and order its vertices, say $x_1, x_2, \ldots, x_n$. One can color them one-by-one in the following way: color x_1 with 1; then, color x_2 with 1, if $x_1x_2 \notin E$, with 2 otherwise and continue coloring each vertex with the smallest color it can be assigned at that stage. Denote this algorithm by `LEGAL_C` and suppose it is called with inputs G and $x_1, x_2, \ldots, x_n$. The following easy lemma holds for `LEGAL_C`.

Lemma 1. *`LEGAL_C` colors the vertices of any graph G with at most $\Delta(G) + 1$ colors, in polynomial time.*

Proof. Since $\Delta(G)$ is the maximum graph-degree, $\Delta(G) + 1$ colors are sufficient to legally color any vertex of G and all its neighbors. ∎

The key-idea of the algorithm of [42] is to construct "good" orderings of the vertices of G, so that `LEGAL_C` colors them with no more $\Delta(G)$ colors.

2.1.1 Coloring 3-connected graphs

Consider a graph $G = (V, E)$ such that: it is connected, $\Delta(G) \geqslant 3$, it does not contain a $K_{\Delta(G)+1}$ and it admits at least a pair of vertices, the removal of which does not disconnect G. Note that a 3-connected graph verifying the conditions of theorem 1 can be such a graph. Also consider the following algorithm `3C_COLOR`. We consider in what follows two types of execution for `3C_COLOR`. The first one is parameterized by a graph G, when it is executed starting from line `(1)`. The second one is parameterized by a graph G and two vertices v and u, when we suppose that lines `(1)` and `(2)` are omitted and its execution starts from line `(3)`. This is the case of execution of `3C_COLOR` in the body of algorithm `Δ_COLOR` in paragraph 2.1.3.

```
BEGIN *3C_COLOR*
(1)   let {1,...,Δ(G)} be the set of colors to be used;
(2)   choose vertices v and u such that d(v,u) = 2
          and G' ← G[V \ {v,u}] remains connected;
(3)   let w be a vertex adjacent to both v and u;
(4)   arrange the vertices of G' in a sequence (w = x1,x2,...,xn-2) such that
          each vertex xi, i ⩾ 2 is adjacent to a vertex xj, j ⩽ i;
          *this can be done breadth-first-search (BFS,[1]) starting from w*
(5)   color v and u with the color 1;
(6)   OUTPUT {1} ∪ LEGAL_C(G', xn-2, xn-3, ..., x1);
END. *3C_COLOR*
```

Lemma 2. *Let G be a 3-connected graph not containing a $K_{\Delta(G)+1}$. Then 3C_COLOR computes a legal $\Delta(G)$-coloring for G in polynomial time.*

2.1.2 Coloring connected graphs containing vertices of degree less than $\Delta(G)$

We now give another algorithm assigning at most $\Delta(G)$ colors to the vertices of a connected graph with at least one vertex of degree $< \Delta(G)$.

```
BEGIN *1V_COLOR*
(1)   let{1,...,Δ(G)} be the set of colors to be used;
(2)   choose a vertex v such that d°(v) < Δ(G);
(3)   arrange the vertices of G in a sequence (v = x1,x2,...,xn) such that
         each vertex xi, i ⩾ 2 is adjacent to a vertex xj, j ⩽ i;
         *this can be done by BFS starting from v*
(4)   OUTPUT LEGAL_C(G, xn, xn-1, ..., x1);
END. *1V_COLOR*
```

Lemma 3. *Let G be a biconnected graph, not containing a $K_{\Delta(G)+1}$, with at least one vertex of degree $< \Delta(G)$. Then 3C_COLOR computes a legal $\Delta(G)$-coloring for G in polynomial time.*

2.1.3 The overall algorithm

Given a coloring, we will call *color-permutation* the exchange of a color i to j and of j to i on all vertices colored with these colors (when j has not been used, then color-permutation between i and j becomes in using j instead of i). We are well-prepared now to specify an algorithm legally coloring the vertices of any graph G by using at most $\Delta(G)$ colors.

```
BEGIN *Δ_COLOR*
(1)   compute all the blocs of G;
(2)   FOR every bloc A of G DO
(3)      IF Δ(G[A]) < 3 THEN
(4)                     order arbitrarily the vertices of A, say x1,x2,x3;
(5)                     LEGAL_C(G[A], x1, x2, x3);
(6)                     GOTO (2);
(7)      FI
(8)      CASE |A| DO
(9)           |A| < Δ(G) + 1:  color A with colors 1,...,Δ(G) and GOTO (2);
(10)          |A| = Δ(G) + 1:  1V_COLOR(G[A]) and GOTO (2);
(11)          |A| > Δ(G) + 1:
```

```
(12)              IF G[A] is 3-connected THEN 3C_COLOR(G[A]) and GOTO (2) FI
(13)              IF G[A] is biconnected THEN
(14)                 choose arbitrarily a vertex x with d°(x) ⩾ 3;
(15)                 IF G[A \ {x}] is biconnected THEN
(16)                    let y ∈ A be such that d(x,y) = 2;
(17)                    3C_COLOR(G[A],x,y) and GOTO (2);
(18)                 FI
(19)                 IF G[A \ {x}] is not biconnected THEN
(20)                    let B1 and B2 two extremal blocs of G;
(21)                    let z1 ∈ B1 and z2 ∈ B2 adjacent to x;
(22)                    3C_COLOR(G[A],z1,z2);
(23)                 FI
(24)              FI
(25)        OD
(26)     OD
(27)     order the blocs in such a way that each bloc has at most a vertex
            in common with the blocs preceding it in the ordering;
(29)     OUTPUT the union of the colors used by permuting them if necessary;
END. *Δ_COLOR*
```

Theorem 2. *Let G be a connected graph with $\Delta(G) \geqslant 3$ and not containing subgraph $K_{\Delta(G)+1}$. Then, algorithm* Δ_COLOR *legally colors the vertices of G with at least $\Delta(G)$ colors, in polynomial time.*

2.2 A $\Delta(G)/3$ approximation ratio for graph-coloring

Suppose now that 1_COLOR is an algorithm receiving an input-graph G and deciding if G is an independent set and, if yes, coloring its vertices with one color. Obviously, this can be done in polynomial time. Also, let 2_COLOR be another algorithm deciding if G is bipartite. This can be done by starting with two colors and by coloring a vertex with one of them and its neighbors with the other one. If at the end all the vertices of G are legally colored, then it is bipartite and, moreover, a 2-coloring is discovered. Finally, consider the following PTAA for C.

```
BEGIN *APPROX_COLOR*
      1_COLOR(G);
      2_COLOR(G);
      Δ_COLOR(G);
      OUTPUT the smallest of the legal colorings produced;
END. *APPROX_COLOR*
```

Theorem 3. $\rho_{\text{APPROX\_COLOR}} \leqslant \Delta(G)/3$.

3 Towards ratios of $o(\Delta(G))$

A well-known method for solving numerous NP-hard problems approximately uses the following schema, denoted by SCHEMA in the sequel:

[Phase1] optimally solve the linear-programming relaxation of the problem (this can be done in polynomial time ([5]));

[Phase2] round this solution to a feasible (but approximate) solution for the original (integer programming) problem.

Recently an evolution of the above schema uses an extension of linear-programming relaxations, the so-called *semidefinite programming relaxations* ([3, 30]). As opposed to the formers, the latters offer rounding techniques leading to feasible integer solutions that are guaranteed to be within a specified fraction to the optimal ones. Semidefinite programming in computing approximate solutions for combinatorial problems is originally used by Goemans and Williamson ([29]) for maximum cut and maximum 2-satisfiability problems. Based upon the work of [29], Karger et al. ([39]) devise a randomized approximation algorithm for C achieving approximation ratio $o(\Delta)$. Their method can be shortly described as follows.

3.1 A semidefinite relaxation for graph-coloring

Consider a k-colorable graph. Instead of assigning integers (or colors) to the vertices of the graph, a unit vector $\vec{v}_i \in \mathbb{R}^n$ is assigned to any vertex $v_i \in V$, $i = 1, \ldots, n$. In order to capture the fact that adjacent vertices are assigned with different colors, the vectors assigned to adjacent vertices have to be different in some specific (but natural) way. This requirement leads to the *vector k-coloring*.

Given a graph $G = (V, E)$ of order n and a real $k \geqslant 1$, *a vector k-coloring of G is an assignment of unit vectors $\vec{v}_i \in \mathbb{R}^n$ to any vertex $v_i \in V$, such that for any two adjacent vertices v_i and v_j the dot product of $\vec{v}_i$ and $\vec{v}_j$ satisfies $\langle v_i, v_j \rangle \leqslant -1/(k-1)$* (in other words, the angle between the vectors corresponding to adjacent vertices must be sufficiently large).

The so-defined vector k-coloring is seen in [39] as a kind of relaxation for C that plays the role that a hypothetical fractional k-coloring would play if one used a conventional linear-programming relaxation for the problem. This is justified by the following lemma.

Lemma 4. *([39])*

- *Every k-colorable graph has a vector k-coloring.*
- *Moreover, for all positive integers k and n with $k \leqslant n+1$, there exist k unit vectors of $\mathbb{R}^n$ such that the dot product of any distinct pair is $-1/(k-1)$.*

Remark that the k vectors of the second item of lemma 4 fulfill the specification of the vector k-coloring (the emphasized proposition just above). Furthermore, one can immediately bijectively map these vectors to k distinct colors in order to produce a k-coloring of a graph G of order n.

Following [Phase1] of SCHEMA, one has now to determine a vector k-coloring of G. This can be done using the following auxiliary problem. Given a graph $G = (V, E)$ of order n, *a matrix k-coloring of G is an $n \times n$ symmetric positive semidefinite matrix M with $m_{ii} = 1$ and $m_{ij} \leqslant -1/(k-1)$ for $v_i v_j \in E$*. The key-point of the relationship between matrix and vector colorings of a graph is given by the following lemma.

Lemma 5. *([39])*

- *A graph has a vector k-coloring if and only if it has a matrix k-coloring.*
- *If a graph has a matrix k-coloring, then a vector $(k+\epsilon)$-coloring can be constructed from this matrix coloring in time polynomial in n and in $\log(\epsilon^{-1})$.*
- *Conversely, if a graph G has a vector k-coloring, then a matrix $(k+\epsilon)$-coloring of G can be constructed in time polynomial in n and in $\log(\epsilon^{-1})$.*

We have supposed at the beginning of the current paragraph that G is k-colorable. Therefore, by the second item of lemma 4, *there exists a vector k-coloring for G* and, by the first item

of lemma 5, *there exists a matrix k-coloring of G*. This matrix coloring can be constructed by solving the following semidefinite optimization problem:

$$\text{SDP} = \begin{cases} \min & \sigma \\ & M = \{m_{ij}\} \text{ positive semidefinite} \\ & m_{ij} \leqslant \sigma \text{ for } v_i v_j \in E \\ & m_{ij} = m_{ji} \\ & m_{ii} = 1 \end{cases}$$

Remark that, by definition of the matrix coloring given just above lemma 5, the solution of SDP specifies the entries of a matrix k-coloring M. Since G has a vector (and, by the first item of lemma 5, a matrix) coloring, there exists a solution to SDP with $\sigma^* = -1/(k-1)$. If one uses a linear-programming method, she/he is able to determine a feasible solution of SDP with $\sigma \leqslant -1(k+\epsilon-1)$, for some carefully chosen $\epsilon \in \mathbb{R}$. This solution, by the definition of the matrix coloring, specifies a matrix $(k+\epsilon)$-coloring of G. This, by the second item of lemma 5, can produce a vector $(k+2\epsilon)$-coloring of G. In order to simplify presentation, the error ϵ can be ignored since it can be made very small as to be irrelevant to the analysis in [39]. So, the following proposition holds.

Proposition 1. *([39]) Given a k-colorable graph G a vector k-coloring of G can be determined in polynomial time.*

3.2 The rounding phase

Once [Phase1] accomplished, one has to process [Phase2] that implies the rounding of the relaxed solution to a feasible integer one. The original rounding technique proposed, called semi-coloring in [39], is randomized and produces an assignment of colors with "relatively few" identically colored adjacent vertices. This semi-coloring is then transformed into a legal graph-coloring. More formally, *a k-semi-coloring of a graph G is an assignment of k colors to at least half of its vertices such that no two adjacent vertices receive the same color.* Dealing with semi-colorings, the following holds.

Lemma 6. *([39]) If an algorithm SEMICOLOR can k_p-semi-color any subgraph of order p of a graph G in randomized polynomial time (where k_p increases with p), then SEMICOLOR can polynomially color the whole graph G with $O(k_n \log n)$ colors.*

In other words, a semi-coloring of G can be transformed into a legal coloring by losing only a logarithmic factor with respect to the colors used for the semi-coloring. The randomized algorithm transforming vector colorings into semi-colorings is not given here. The interested reader can be referred in [39]. In any case, the following can be shown.

Proposition 2. *([39]) For every integer function $k = k(n)$, a vector k-colorable graph G can be semi-colored with at most $O(\Delta(G)^{1-(2/k)}\sqrt{\log \Delta(G)})$ colors in randomized polynomial time.*

Combination of lemma 6 with the first item of lemma 4 and with propositions 1 and 2 implies the following.

Proposition 3. *([39]) Any k-colorable graph G of order n can be colored in randomized polynomial time with at most $O(\Delta(G)^{1-(2/k)}\sqrt{\log \Delta(G)} \log n)$ colors.*

Two last points remain to be settled: (i) can the randomized method proposed be transformed into a deterministic one, and (ii) how one can make this method to run for any graph (recall that until now the graph is supposed k-colorable[2])?

[2]This means that the input of C is a graph G, an integer $k \leqslant n$ and the information that G is k-colorable.

For point (i) the answer is found in [44]. The authors propose there a clever (but long) polynomial derandomization of the algorithms in [39] achieving the same approximation guarantees than the original (random) ones. On the other hand, for point (ii), the following procedure, that will be used later also, can be used. Since the complexity of the overall k-coloring algorithm is polynomial in k, one can run it for all $k \in \{2, \ldots n\}$ in this order (multiplying so its worst-case complexity by n), and stop it for the first k for which a feasible solution is produced. As it is pointed out in [51], this thought process can be implemented by divide-and-conquer in such a way that the whole complexity of the derived algorithm will be multiplied not by k but by $\log k$. So, in the light of the above settlements, proposition 3 holds for any graph and the algorithm claimed runs in deterministic polynomial time.

Finally, run both algorithm `Δ_COLOR` and the schema dealt in the current section (parameterized by the algorithms of [39], derandomized by the method of [44], and extended so as it runs for any graph by the remark settling point (ii) above), and take finally the best of the solutions produced. Then, the following concluding theorem holds.

Theorem 4. *C can be approximately solved in polynomial time and in any graph G within approximation ratio*

$$\rho_{\mathrm{C}} \leqslant \min\left\{\frac{\Delta(G)}{\chi(G)}, O\left(\frac{\Delta(G)^{1-\frac{2}{k}}\sqrt{\log\Delta(G)}\log n}{\chi(G)}\right)\right\}.$$

4 The greedy coloring-algorithm

4.1 An excavation schema for graph-coloring

When one tries to approximately solve C, the first algorithm coming in mind is the greedy one described by the following "excavation schema" executed with parameters G and an IS-algorithm `INDEPENDENT_SET`.

```
BEGIN *EXCAVATION*
      REPEAT
            S ← INDEPENDENT_SET(G);
            color the vertices of S with the same not already used color;
            remove S from G;
      UNTIL G becomes empty;
      OUTPUT X the set of used colors;
END. *EXCAVATION*
```

Algorithm `EXCAVATION` has two interesting properties, expressed by items 1 and 2 of proposition 4 below. Item 1 is proved in [31], but has been implicitly used in [11, 37, 51] and explicitly in [31]. Item 2 is proved in [36] for the case where S is a maximum independent set and is generalized in [2] for the case where S is any independent set. Here it is mainly used in section 8.

Proposition 4.

1. *Any iterative application of algorithm `INDEPENDENT_SET` that computes an independent set of size $\iota_k(G) = \Theta(n^{1/t})$, $t > 1$, in a k-colorable graph G of order n, produces a coloring X verifying $|X| \leqslant 2n/\iota_k(G)$.*

2. $\rho_{\texttt{EXCAVATION}} \leqslant \ln n/\rho_{\texttt{INDEPENDENT\_SET}}$.

4.2 Instantiating INDEPENDENT_SET by the greedy algorithm

An interesting polynomial instantiation of the schema described above, where the independent set is found via a greedy algorithm, has been studied by Johnson in [37]. We give in what follows an outline for both the greedy IS-algorithm and the greedy C-algorithm that ensues.

```
BEGIN *GREEDY_IS*
      S ← ∅;
      REPEAT
            v ← argmin_{v_i∈V}{d°(v_i)};
            S ← S ∪ {v};
            V ← V \ ({v} ∪ Γ(v));
            G ← G[V];
      UNTIL V = ∅;
      OUTPUT S;
END. *GREEDY_IS*
```

```
BEGIN *GREEDY_C*
      REPEAT
            S ← GREEDY_IS(G);
            color the vertices of S with the same non already used color;
            V ← V \ S;
            G ← G[V];
      UNTIL V = ∅;
      OUTPUT the set X_J of used colors;
END. *GREEDY_C*
```

The complexity of algorithm GREEDY_IS is $O(|E|)$ and will be called at most n times within the REPEAT loop of algorithm GREEDY_C, each such call strictly decreasing V. Hence, the overall worst-case complexity of GREEDY_C is $O(n|E|)$.

Let us denote by $V^{(\ell)}$ the vertex-set of the current, surviving, graph at the beginning of the ℓth iteration of GREEDY_C. The key-point for the study of its approximation performance is the following. If a graph is k-colorable, then, at each iteration ℓ of GREEDY_C, $d^{\circ}(v_j) \leqslant |V| - \lceil |V|/k \rceil$, where v_j is the minimum-current-degree vertex chosen by GREEDY_IS. Therefore, the vertices colored during ℓth iteration (i.e., the constructed maximal independent set during this iteration) will be of size at least $\log_k |V|$. This leads to the following lemma, proved by Wigderson in [51].

Lemma 7. *([51]) Algorithm GREEDY_C colors any k-colorable graph G with $|X_J| \leqslant 3n/\log_k n$ colors.*

By lemma 7, the approximation ratio of algorithm GREEDY_C in a $\chi(G)$-chromatic graph becomes:

$$\rho_{\text{GREEDY\_C}} \leqslant \frac{|X_J|}{\chi(G)} \leqslant \frac{1}{\chi(G)} \frac{3n \log \chi(G)}{\log n} \leqslant \frac{n}{\log n} \tag{2}$$

leading so to the following concluding theorem.

Theorem 5. *([37])* $\rho_{\text{GREEDY\_C}} \leqslant n/\log n$.

5 Improving the ratio for graph-coloring

The improvement of the approximation ratio of C has remained open for 9 years until 1982 when Wigderson has shown in [51] how to obtain better performance guarantees. His method, outlined in what follows, is based upon the following observations:

1. *the neighborhood of any vertex in a k-colorable graph is $(k-1)$-colorable;*
2. *2-coloring is polynomial (see section 2).*

These observations (the second being a termination condition) lead, as we will see, to a nice recursive approximation strategy for C.

5.1 Coloring k-colorable graphs

We first present the following procedure, called with parameters k and G and i (where i means that G will be colored with colors $i, i+1, \ldots$), that colors a k-colorable graph with "relatively" few colors. Set, for $k = 2, 3, \ldots$, $f_k(n) = n^{\{1-(1/(k-1))\}}$.

```
BEGIN *k_COLOR*
(1)   CASE k DO
(2)       k = 2: OUTPUT 2_COLOR(G);
                 *vertices of G will be colored with colors i and i+1*
(3)       k ⩾ log|G|: color V(G) with a distinct color per vertex;
                      *vertices of G are assigned colors i, i+1,...,i+|G|-1*
(4)       k ⩽ log|G|: WHILE Δ(G) ⩾ ⌈f_k(|G|)⌉ DO
(5)                   let v be such that d°(v) = Δ(G);
(6)                   H ← G[Γ_G(v)];
(7)                   {i, i+1, ..., i+j-1} ← k_COLOR(k-1, H, i);
(8)                   color v with i+j;
(9)                   i ← i+j;
(10)                  G ← G[V \ (Γ_G(v) ∪ {v})];
(11)                  OD
(12)                  {i, i+1, ...,} ← Δ_COLOR(G);
(13)                  OUTPUT X_k_C the set of used colors;
(14)  OD
END. *k_COLOR*
```

In the initial graph, algorithm is called as `k_COLOR(k,G,1)`. Line (12) of `k_COLOR` will be executed on graphs with $\Delta(G) < \lceil f_k(|G|)\rceil$, using so less than $\lceil f_k(|G|)\rceil$ unused colors. The algorithm called in line (2) is the one deciding if a graph is bipartite, and if yes, computing a 2-coloring of its vertices.

Lemma 8. *([51]) `k_COLOR` assigns, in $O(k(n+|E|))$, the vertices of any k-colorable graph with at most $2k\lceil f_k(n)\rceil = 2k\lceil n^{(1-(1/(k-1)))}\rceil$ colors.*

5.2 Expanding `k_COLOR` to run for all graphs

As we have already mentioned at the end of paragraph 3.2, one can expand algorithm `k_COLOR` (destinated to color k-colorable graphs) to run for any graph. Recall that she/he only has to run it for all $k \in \{2, \ldots n\}$ in this order, and stop it for the first k for which procedure `k_COLOR` produces a feasible solution. This can be implemented as follows.

```
BEGIN *Ek_COLOR*
(1)   FOR p ← 1 TO ⌈logn⌉ DO
(2)       X ← k_COLOR(2^p, G, 1);
(3)       IF X is feasible THEN p_0 ← p and GOTO (5) FI;
(4)   OD
(5)   apply binary search in {2^(p_0−1) + 1, ... 2^(p_0)} to find the least k ← k_0
          for which k_COLOR produces a legal coloring;
(6)   OUTPUT X_E ← k_COLOR(k_0, G, 1);
END. *Ek_COLOR*
```

Since, given a graph G:

(i) the execution of line (2) of algorithm Ek_COLOR produces a legal coloring for all $k \geqslant \chi(G)$,

(ii) G is always $\chi(G)$-colorable, and

(iii) k_0 is the smallest k for which the execution of line (5) will produce a feasible coloring,

the following lemma holds.

Lemma 9. *([51])* $k_0 \leqslant \chi(G)$.

Combining lemmas 8 and 9, we get the following concluding theorem for the approximation performance of algorithm Ek_COLOR.

Theorem 6. *([51])* *Ek_COLOR colors, in* $O((n + |E|)\chi(G) \log \chi(G))$, *the vertices of* G *with at most* $2\chi(G)\lceil n^{(1-(1/(\chi(G)-1)))}\rceil$ *colors.*

5.3 The whole improvement

By theorem 6, $\rho_{\text{Ek\_COLOR}} \leqslant 2\lceil n^{(1-(1/(\chi(G)-1)))}\rceil$ and function $f(x) = 2\lceil n^{(1-(1/(x-1)))}\rceil$ is increasing in x. On the other hand, by lemma 7 and expression (2), $\rho_{\text{GREEDY\_C}} \leqslant 3n \log \chi(G)/(\chi(G) \log n)$ and function $g(x) = 3n \log x/(x \log n)$ is decreasing in x. Let us combine the two algorithms to produce the following final algorithm.

```
BEGIN *W_COLOR*
      X_E ← Ek_COLOR(G);
      X_J ← GREEDY_C(G);
      OUTPUT X_W = argmin{|X_E|, |X_J|};
END. *W_COLOR*
```

Of course algorithm W_COLOR composed by polynomial component-algorithms is also polynomial.

A little algebra shows that the intersection point of the curves $f(x)$ and $g(x)$ is in the neighborhood of $x = \lceil n \log \log n/2 \log n \rceil$. Algorithm Ek_COLOR is superior to GREEDY_C for $\chi(G) \leqslant \log n/\log \log n$ while, for $\chi(G) \geqslant \log n/\log \log n$ GREEDY_C is more performant than Ek_COLOR. For $\chi(G) = \log n/\log \log n$, both $\rho_{\text{Ek\_COLOR}}$ and $\rho_{\text{GREEDY\_C}}$ are of $O(n \log^2 \log n/log^2 n)$ and the following theorem concludes the section.

Theorem 7. *([51])* $\rho_{\text{W\_COLOR}} \leqslant 3n \log^2 \log n/\log^2 n$.

6 A better approximation ratio for coloring

Seven years later, Berger and Rompel, using thought processes quite close to the ones adopted in [51] (i.e., coloring first k-colorable graphs, extending the result to work for every graph, the chromatic number of which belongs to a certain interval of values, and next composing the algorithm produced with another well-working in graphs with chromatic-number values out of the interval considered), perform, in [11], a further notable improvement of the approximation ratio of C.

The key-observation in [11] is a kind of refinement of the respective observation of [37]. Recall that, as we have seen in section 4, Johnson observed that if G is k-colorable, then there exists an independent set of size at least $\lceil |V|/k \rceil$ and, consequently, any node v in this set has $d^{\circ}(v) \leqslant |V| - \lceil |V|/k \rceil$; this observation enabled him, using algorithm GREEDY_IS, to find an independent set S of $O(\log_k |V|)$ small-degree vertices to which gave the same color. The refinement lying at the heart of the method proposed in [11] is that any subset S' of S verifies $|\Gamma(S')| \leqslant |V| - \lceil |V|/k \rceil$. This allows them to somewhat modify algorithm GREEDY_IS to choose at each step a set (instead of one) of $O(\log_k |V|)$ small-degree vertices that are independent and have small neighborhood. They are so able to legally color $O[(\log_k |V|)^2]$ vertices with the same color.

Let us now outline how Berger and Rompel produce feasible colorings for k-colorable graphs. We consider, without loss of generality, that the colors are drawn from the set $\{1, 2, \ldots\}$.

6.1 Improving GREEDY_C in k-colorable graphs

Consider the following algorithm k_COLORING parameterized by an integer k, a fixed $\alpha > 0$ and a graph G. The first condition in line (10) receives TRUE if the set S is an independent set, FALSE otherwise.

```
BEGIN *k_COLORING*
(1)   IF k > min{n^(1/2), n^α} THEN OUTPUT GREEDY_C(G) FI
(2)   m ← ⌊α log_k n⌋;
(3)   c ← 1;
(4)   UNCOLORED ← V;
(5)   WHILE |UNCOLORED| ⩾ km DO
(6)         U ← UNCOLORED;
(7)         WHILE |U| ⩾ km DO
(8)               H ← G[U];
(9)               FOR S ⊆ H such that |S| = m DO
(10)                  IF S AND |Γ_H(S)| ⩽ |U| − (|U|/k) THEN GOTO (13) FI
(11)              OD
(12)              OUTPUT "G not k-colorable";
(13)              color S with color c;
(14)              U ← U \ [S ∪ Γ_H(S)];
(15)              UNCOLORED ← UNCOLORED \ S;
(16)        OD
(17)        c ← c + 1;
(18)  OD
(19)  color the UNCOLORED-vertices with colors c, ..., c+|UNCOLORED|-1;
            *assigns an unused color per UNCOLORED-vertex*
(20)  OUTPUT the colors used;
END. *k_COLORING*
```

Remark 1. Line (9) in algorithm k_COLORING can be implemented in the following way:

partition the vertices of U into $\ell = \lfloor |U|/km \rfloor$ sets B_i, $i = 1, \ldots, \ell$, such that
B_j, $j = 1, \ldots, \ell - 1$, contains km elements and
B_ℓ contains the remaining ones ($km \leqslant |B_\ell| < 2km$);
FOR $j \leftarrow 1$ TO ℓ DO FOR $S \subseteq B_j$ such that $|S| = m$ DO GOTO line (10);

By the pigeonhole principle, at least one of the sets B_i will contain a set S of size m and this set can be found by exhaustively searching the $C_{\alpha \log_k n}^{k\alpha \log_k n} = O(n)$ subsets of each B_i. Consequently, implementation of line (9) of algorithm k_COLORING can be done in polynomial time. ∎

Theorem 8. *([11]) For any $\alpha > 0$, algorithm k_COLORING colors, in $O[n^{3+3\alpha}/(k \log_k n)]$, the vertices of any k-colorable graph with $2n/(\alpha \log_k^2 n) + O(n/\log_k^3 n)$ colors.*

6.2 Modifying algorithm k_COLORING to run on all graphs

The following algorithm (parameterized by α and G) modifies, in the spirit of [51], algorithm k_COLORING to work on any graph. Algorithms 1_COLOR and 2_COLOR called by k_COLORING are as in section 2.

```
BEGIN *Ek_COLORING*
(1)   IF 1_COLOR(G) feasible THEN OUTPUT 1_COLOR(G) FI
(2)   IF 2_COLOR(G) feasible THEN OUTPUT 2_COLOR(G) FI
(3)   FOR p ← 1 TO ⌈logn⌉ DO
(4)       X ← k_COLORING(2^p, α, G);
(5)       IF X is feasible THEN p_0 ← p and GOTO (7) FI;
(6)   OD
(7)   apply binary search in {2^(p_0 − 1) + 1, ... 2^(p_0)} to find the least k ← k_0
          for which k_COLORING produces a legal coloring;
(8)   OUTPUT X_E ← k_COLORING(k_0, α, G);
END. *Ek_COLORING*
```

The complexity of algorithm Ek_COLORING is $O(n^{4+3\alpha}/\log n)$ (it calls at most $n - 2$ times k_COLORING). Moreover, with the same arguments as the ones for lemma 9, $k_0 \leqslant \chi(G)$. So, the coloring produced satisfies

$$|X_E| \leqslant \frac{2n}{\alpha \log_{\chi(G)}^2 n} + O\left(\frac{n}{\log_{\chi(G)}^3 n}\right) \tag{3}$$

6.3 The whole improvement

Recall that $\rho_{\text{Ek\_COLOR}} \leqslant 2\lceil n^{(1-(1/(\chi(G)-1)))} \rceil$ and this bound is increasing in $\chi(G)$. On the other hand, by theorem 8 and expression (3), $\rho_{\text{Ek\_COLORING}} \leqslant 2n/(\alpha \chi(G) \log_{\chi(G)}^2 n) + o(n/\log_{\chi(G)}^2 n)$ and this last bound is decreasing in $\chi(G)$. For $\chi(G) = O(\lceil \log n / \log\log n \rceil)$, both bounds are at most $O(n \log^3 \log n / \log^3 n)$. The following algorithm, BR_COLOR, combines algorithms Ek_COLOR and Ek_COLORING into an algorithmic schema for C. Its running time is of $O(\max\{n^{(4+(21/\eta))}/\log n, (n + |E|)\chi(G) \log \chi(G)\})$ ([11]).

```
BEGIN *BR_COLOR*
      fix a large η > 0;
      OUTPUT X_E ← Ek_COLORING(7/η, G);
      OUTPUT X'_E ← Ek_COLOR(G);
      OUTPUT argmin{|X_E|, |X'_E|};
END. *BR_COLOR*
```

Theorem 9. *([11])* $\rho_{\texttt{BR\_COLOR}} \leqslant \eta n \log^3 \log n / \log^3 n$.

Let us observe that algorithm BR_COLOR has a fairly serious drawback since its execution time requirements (being polynomial in n for fixed η) is exponential in η. This means that the better the approximation ratio achieved, the higher its execution time.

7 A still better approximation ratio for graph-coloring

In 1993, a further improvement of coloring's approximation ratio has been published by Halldórsson in [31]. The spirit of this work is quite similar to the previous ones (except the one of [42]), i.e., one colors a graph by excavating independent sets, but the used IS-algorithms are quite different from the greedy one used until then. It is well-known to people working on the design of approximation algorithms that very frequently the efficiency of an algorithm strongly depends on the value of the optimal solution of an instance. Some algorithms work well for small optimal values, while some other ones work better on instances with large optimal values. The key idea of [31] is to combine into an excavation schema two IS-algorithms, one of them (OPT-CLIQUE_REMOVAL) behaving efficiently in graphs with small chromatic numbers, while the other one (COLOR_IS) behaving well in graphs with large chromatic numbers. Simultaneous running, at each iteration of the excavation schema, of both algorithms and coloring largest among the independent sets computed with an unused color leads to an improved ratio for any value of $\chi(G)$.

7.1 Finding large independent sets ...

We first briefly describe the two IS-algorithms, OPT-CLIQUE_REMOVAL and COLOR_IS, used in [31] for excavating independent sets.

The idea of using the former is based upon the fact that independent sets in graphs without cliques or with small cliques are larger than independent sets in general graphs. Furthermore satisfactorily large independent sets (achieving improved approximation ratios) can be polynomially found there (see for example [49, 50], where the author deals with triangle-free graphs). Hence, given a graph G, one can reduce it by removing cliques of a certain size ℓ. In the surviving graph (that is ℓ-clique-free), in particular if its size remains large (in some sense), she/he can apply some efficient algorithm computing a "large" independent set. Excavation of such "large" independent sets is possible as long as the initial and the consecutive (surviving) graphs contain "few" disjoint cliques. Of course, if large independent sets are excavated, by item 1 of proposition 4, the graph can be colored with relatively few colors. On the other hand, if the initial graph contains many disjoint cliques, then $\alpha(G)$ must be small and, by expression (1), $\chi(G)$ must be large. In both cases the approximation ratio for C can so be improved.

Algorithm COLOR_IS originally operates in k-colorable graphs. Informally, it recursively finds an independent set S of a certain size and takes the union of S with the result of its recursive running on the graph $G[S \cap \Gamma(S)]$. This is done as long as the order of the surviving graph exceeds a fixed threshold t. As soon as the order of the surviving graph becomes smaller than t, OPT-CLIQUE_REMOVAL is called. The final independent set is the union of the independent sets recursively computed by COLOR_IS together with the one computed by OPT-CLIQUE_REMOVAL. Running COLOR_IS for any value of k and retaining the best result, one can obtain an independent set at least as good as the result of the execution COLOR_IS(χ, G), and then using item 1 of proposition 4, one can hope to compute a small coloring for G.

7.1.1 ...By removing cliques

We first present an intermediate algorithm, originally devised in [16], and used, directly or undirectly, by both OPT-CLIQUE_REMOVAL and, COLOR_IS. It is based upon the following result due to Ramsey.

Theorem 10. *For any pair (s,t) of integers, there is an integer n for which every graph of order n contains either a clique K_s, or an independent set S of size t.*

If we denote by $R(s,t)$ the minimal value of n for which the above theorem holds, then the best known upper bound for $R(s,t)$, due to Erdös and Szekeres ([25]), is $R(s,t) \leqslant C_{t-1}^{s-t+2}$. Let us set $C_{t-1}^{s-t+2} = r(s,t)$ for all positive integers s and t (by convention, if one of s and t is negative or zero, then $C_{t-1}^{s-t+2} = 1$), and let $t_s(n) = \min\{t : r(s,t) \geqslant n\}$.

Algorithm RAMSEY (parameterized by a graph G and an integer s) developed in [16] and strongly inspired by the proof of the upper bound for $R(s,t)$ ([25]) finds, in time $O(|E|)$, either a clique K of order s, or an independent set S of size t. Moreover, $r(s,t) \geqslant n$ and $s.t \geqslant c(\log n)^2$, for some constant c.

```
BEGIN *RAMSEY*
      S ← ∅;
      K ← ∅;
      WHILE |V(G)| > 1 DO
            choose v ∈ V(G);
            t ← t_s(n);
            IF |Γ(v)| ⩾ r(s − 1, t) THEN K ← K ∪ {v};
                                         G ← G[Γ(v)];
                                         s ← s − 1;
                                   ELSE S ← S ∪ {v};
                                         G ← G[V \ ({v} ∪ Γ(v))];
            FI
      OD
      OUTPUT K ← K ∪ V(G);
      OUTPUT S ← S ∪ V(G);
END. *RAMSEY*
```

It is next combined with a greedy mechanism of clique removal and the following IS-algorithm (parameterized by a graph G and an integer s) is derived in [16].

```
BEGIN *CLIQUE_REMOVAL*
      (S, K) ← RAMSEY(G, s);
      WHILE |K| ⩾ s DO
            G ← G[V \ K];
            (S, K) ← RAMSEY(G, s);
      OD
      OUTPUT S;
END. *CLIQUE_REMOVAL*
```

In order that the final coloring-algorithm is the best possible, one needs to find the value of s for which the approximation ratio of algorithm CLIQUE_REMOVAL is the best possible (in other words, the independent set computed is the largest possible). Then, one can follow the thought process originally proposed in [51], i.e., one can use binary search to define a "good" s. This leads to the following final version of CLIQUE_REMOVAL, called OPT-CLIQUE_REMOVAL and parameterized

by a graph G, of complexity $O(|E|n\log n)$, where we denote by ρ_i the approximation ratio of CLIQUE_REMOVAL(G,i).

```
BEGIN *OPT-CLIQUE_REMOVAL*
      guess an s;
      S ← CLIQUE_REMOVAL(G, s);
      IF ρ_s > (n/s^2)^(1/s) THEN RETURN S FI
      REPEAT run CLIQUE_REMOVAL(G,p) with p = 2^ℓ, ℓ = ⌈logk⌉,...;
      UNTIL the first ℓ ← ℓ_0 for which ρ_p > (n/p^2)^(1/p);
      REPEAT apply binary search in {2^(ℓ_0−1) + 1,...2^ℓ_0};
      UNTIL the least p ← p_0 for which ρ_p0 > (n/p_0^2)^(1/p_0);
      OUTPUT S ← CLIQUE_REMOVAL(G, p_0);
END. *OPT-CLIQUE_REMOVAL*
```

Lemma 10. *([31]) Running algorithm OPT-CLIQUE_REMOVAL on a graph G of order n with $\alpha(G) \geqslant tn$, $t \geqslant 1/\log n$, returns an independent set of size at least $e^{-1}n^t/t$.*

7.1.2 ...In k-colorable graphs

Let us now consider the second IS-algorithm (parameterized by an integer k and by a graph G) presented in [31] and running in k-colorable graphs. As for algorithm k_COLORING in section 6, the first condition in line (3) receives TRUE if the set S is an independent set, FALSE otherwise.

```
BEGIN *k-COLOR_IS*
(1)   IF |V| = 1 THEN OUTPUT V FI;
(2)   FOR S ⊆ V such that |S| = log_k n DO
(3)       IF S THEN
(4)          IF |G[V \ (S ∪ Γ(S))]| ⩾ nlogn/(2kloglogn)
(5)             THEN OUTPUT S ∪ k-COLOR_IS(k, G[V \ (S ∪ Γ(S))]);
(6)             ELSE I ← OPT-CLIQUE_REMOVAL(G[V \ (S ∪ Γ(S))]) ∪ S;
(7)                  IF |I| ⩾ (logn)^3/(6loglogn) THEN OUTPUT S ∪ I FI;
(8)          FI
(9)       FI
(10)  OD
(11)  OUTPUT G not k-colorable;
END. *k-COLOR_IS*
```

Note that remark 1 holds also for algorithm k-COLOR_IS. Consequently, the execution of the FOR loop of line (2) is performed in polynomial time.

We can modify algorithm k-COLOR_IS to work for all graphs (i.e., for any k) producing so the following IS-algorithm (parameterized by G) of complexity $O(n|E|\chi(G))$. The IF-condition in this algorithm receives TRUE if S_k is independent.

```
BEGIN *COLOR_IS*
      FOR k ← 1 TO n DO
          S_k ← k-COLOR_IS(k, G);
          IF S_k THEN store S_k FI
          OUTPUT the best among the S_k's stored;
      OD
END. *COLOR_IS*
```

Lemma 11. *([31]) The application of algorithm COLOR_IS in G produces an independent set of size at least $\log_{\chi(G)} n\log n/(2\max\{\log(2\chi(G)\log\log n/\log n), 1\})$.*

7.2 The overall coloring-algorithm

We are ready now to outline the overall C-algorithm of [31], the worst-case complexity of which is $O(n^2|E|)$.

```
BEGIN *H_COLOR*
      REPEAT
            SCR ← OPT-CLIQUE_REMOVAL(G);
            SCS ← COLOR_IS(G);
            S ← argmax{|SCR|, |SCS|};
            color the vertices of S with the same unused color;
            V ← V \ S;
            G ← G[V \ S];
      UNTIL V = ∅;
      OUTPUT the set of the colors used;
END. *H_COLOR*
```

Suppose first $\chi(G) \leqslant \log n/(2\log\log n)$. Then, by lemma 10, algorithm OPT-CLIQUE_REMOVAL guarantees an independent set of size

$$|\mathrm{SCR}| \geqslant e^{-1}\frac{n^{\frac{1}{\chi(G)}}}{\frac{1}{\chi(G)}} = e^{-1}\chi(G)n^{\frac{1}{\chi(G)}}.$$

By item 1 of proposition 4, an excavation schema using OPT-CLIQUE_REMOVAL (algorithm H_COLOR is instantiation of such a schema) computes a coloring of size

$$|X| = O\left(\frac{n}{e^{-1}\chi(G)n^{\frac{1}{\chi(G)}}}\right) = O\left(\frac{n^{1-\frac{1}{\chi(G)}}}{\chi(G)}\right) \tag{4}$$

On the other hand, if $\chi(G) \geqslant \log n/(2\log\log n)$, then by lemma 11, algorithm COLOR_IS guarantees an independent set of size

$$|\mathrm{SCS}| \geqslant \frac{\log^2 n}{\log\chi(G)}\frac{1}{2\max\left\{\log\left(\frac{2\chi(G)\log\log n}{\log n}\right),1\right\}}.$$

By item 1 of proposition 4, algorithm H_COLOR (using COLOR_IS) will produce a coloring with

$$|X| = O\left(\frac{n\log\chi(G)\max\left\{\log\left(\frac{2\chi(G)\log\log n}{\log n}\right),1\right\}}{\log^2 n}\right) \tag{5}$$

The corresponding ratios are the right-hand sides of expressions (4) and (5) divided by $\chi(G)$, the former being decreasing and the second increasing in $\chi(G)$. For $\chi(G) = \log n/(2\log\log n)$, the values of the two ratios are both of $O(n\log^2\log n/\log^3 n)$ and the following theorem concludes the section.

Theorem 11. *([31])* $\rho_{\mathrm{H\_COLOR}} = O(n\log^2\log n/\log^3 n)$.

8 A further improvement of the performance guarantee for the approximation of graph-coloring

Revisit for a while algorithm OPT-CLIQUE_REMOVAL of section 7. In [23], the following proposition is proved.

Proposition 5. *([23]) For every function ℓ such that, $\forall x > 0$, $0 < \ell(x) \leq \log\log x$, there exist constants κ and K such that algorithm OPT-CLIQUE_REMOVAL computes, for every graph G of order $n > \kappa$, an independent set S such that if $\alpha(G) \geqslant \ell(n) n \log\log n / \log n$, then $|S| \geqslant K \log^{\ell(n)} n$.*

In what follows, we denote by EXHAUST, an exhaustive-search algorithm for C. Without loss of generality we suppose that vertices are colored with $1, 2, \ldots$ Moreover, let K and κ be as in the quoted proposition above.

```
BEGIN *COLOR*
(1)   IF n ⩽ κ THEN OUTPUT EXHAUST(G) FI
(2)   S ← OPT-CLIQUE_REMOVAL(G);
(3)   i ← 1;
(4)   X̂ ← ∅;
(5)   V(Ĝ) ← ∅;
(6)   WHILE |S| ⩾ Klog^ℓ n DO
(7)         color S with color i;
(8)         V(Ĝ) ← V(Ĝ) ∪ S;
(9)         Ĝ ← G[V(Ĝ)];
(10)        X̂ ← X̂ ∪ {i};
(11)        i ← i + 1;
(12)        G ← G[V \ S];
(13)        IF G = ∅ THEN OUTPUT X̂ FI
(14)        S ← OPT-CLIQUE_REMOVAL(G);
(15)  OD
(16)  X̃ ← Δ_COLOR(G);
(17)  OUTPUT X ← X̂ ∪ X̃;
END. *COLOR*
```

Theorem 12.

$$\rho_{\mathrm{C}} \leqslant \max\left\{\frac{2n}{k\log^{\ell(n)-1} n}, \frac{2\Delta(G)\ell(n)\log\log n}{\log n}\right\}.$$

Proof. Obviously, if line (1) of algorithm COLOR is executed, then it returns a minimum coloring for G in polynomial time. The WHILE-loop of the algorithm above (lines (6) to (15)) is an application of EXCAVATION(G,OPT-CLIQUE_REMOVAL). Observe also that, for every iteration i of the WHILE-loop, if we denote by G_i the graph $G[V \setminus V(\hat{G})]$ ($G_1 = G$), then

$$\rho_{\text{OPT-CLIQUE\_REMOVAL}}(G_i) \geqslant \frac{K\log^{\ell(|G_i|)}|G_i|}{|G_i|} \tag{6}$$

Then by item 2 of proposition 4 and by expression (6):

$$\rho_{\text{EXCAVATION}}\left(\hat{G}\right) \leqslant \frac{\ln\left|\hat{G}\right|}{\frac{K\log^{\ell(|\hat{G}|)}|\hat{G}|}{|\hat{G}|}} = \frac{\log\left|\hat{G}\right|}{\frac{\log eK\log^{\ell(|\hat{G}|)}|\hat{G}|}{|\hat{G}|}} \leqslant \frac{\left|\hat{G}\right|}{k\log^{\ell(|\hat{G}|)-1}\left|\hat{G}\right|} \tag{7}$$

Set $\tilde{G} = G[V \setminus V(\hat{G})]$ (in other words, $\tilde{G}$ is the subgraph of G input of algorithm Δ_COLOR in line (16)). Then, by proposition 5 and expression (1), application of Δ_COLOR in $\tilde{G}$ will compute a set $\tilde{X}$ of colors verifying

$$\rho_{\Delta\_\text{COLOR}}\left(\tilde{G}\right) = \frac{\left|\tilde{X}\right|}{\chi\left(\tilde{G}\right)} \leqslant \frac{\Delta\left(\tilde{G}\right)}{\frac{\log\left|\tilde{G}\right|}{\ell\left(\left|\tilde{G}\right|\right)\log\log\left|\tilde{G}\right|}} = \frac{\Delta\left(\tilde{G}\right)\ell\left(\left|\tilde{G}\right|\right)\log\log\left|\tilde{G}\right|}{\log\left|\tilde{G}\right|} \tag{8}$$

The following holds for the set X of colors computed by algorithm COLOR:

$$\begin{aligned} |X| &= \left|\hat{X}\right| + \left|\tilde{X}\right| \leqslant \rho_{\text{EXCAVATION}}\left(\hat{G}\right)\chi\left(\hat{G}\right) + \rho_{\Delta\_\text{COLOR}}\left(\tilde{G}\right)\chi\left(\tilde{G}\right) \\ &\leqslant \max\left\{\rho_{\text{EXCAVATION}}\left(\hat{G}\right), \rho_{\Delta\_\text{COLOR}}\left(\tilde{G}\right)\right\}\left(\chi\left(\hat{G}\right) + \chi\left(\tilde{G}\right)\right) \end{aligned} \tag{9}$$

Obviously, both $\chi(\hat{G})$ and $\chi(\tilde{G})$ are smaller than $\chi(G)$. So, using expressions (7), (8 and (9), one gets

$$\begin{aligned} \rho_{\text{COLOR}} &= \frac{|X|}{\chi(G)} \leqslant \max\left\{\frac{2\left|\hat{G}\right|}{k\log^{\ell(|\hat{G}|)-1}\left|\hat{G}\right|}, \frac{2\Delta\left(\tilde{G}\right)\ell\left(\left|\hat{G}\right|\right)\log\log\left|\tilde{G}\right|}{\log\left|\tilde{G}\right|}\right\} \\ &\leqslant \max\left\{\frac{2n}{k\log^{\ell(n)-1}n}, \frac{2\Delta(G)\ell(n)\log\log n}{\log n}\right\} \end{aligned} \tag{10}$$

This completes the proof of the theorem. ∎

As we have already seen, in terms of n the best known approximation ratio for C is $O(n\log^2\log n/\log^3 n)$ (section 7) and the very tight analysis of [31] does not allow improvement of this ratio even in particular classes of graphs. Let $\ell > 5$ and suppose first that the maximum in expression (10) is realized by the term $O(n/\log^{\ell-1} n)$. Then, for graphs with $\Delta(G) = O(n/\log^{\ell-2} n)$, C is approximable within ratio $O(n/\log^{\ell-1} n)$ and theorem 12 improves the ratio of [31] by a factor $\Omega(\log^2\log n\log^{\ell-4} n)$.

Revisit now the ratio in theorem 4 and remark that the first function is decreasing in $\chi(G)$, while the second one is increasing for $\chi(G) \leqslant 2\log\Delta(G)$ and decreasing for $\chi(G) \geqslant 2\log\Delta(G)$.

- If $\chi(G) \leqslant 2\log\Delta(G)$, the ratio expression in theorem 4 attains its minimum value when the two terms are equal, in other words when $(\Delta(G))^{2/\chi(G)} = \Theta(\sqrt{\log\Delta(G)}\log n)$, i.e., when $\chi(G) = \Theta(\log\Delta(G)/\log\log n)$. In this case, the minimum and therefore the value of the ratio guaranteed by the expression in theorem 4 become $O(\Delta(G)\log\log n/\log\Delta(G))$. On the other hand, by theorem 12, when $\Delta(G) = \Omega(n/(\log^{\ell-2} n\log\log n))$, the ratio $O(\Delta(G)\log\log n/\log n)$ is always achieved independently on the values of $\chi(G)$. Hence, for $\Delta(G) \geqslant n/\log^{\ell-2} n$ and $\chi(G) \leqslant 2\log\Delta(G)$, the result of [39] (second term of the ratio in theorem 4) is improved by a factor $\log n/\log\Delta(G)$.

- If $\chi(G) \geqslant 2\log\Delta(G)$ (and $\Delta(G) \geqslant n/\log^{\ell-2} n$), then the ratio of theorem 4 is bounded above by $O((\Delta(G))^{1-(2/\log\Delta(G))}\log n/\sqrt{\log\Delta(G)})$, while the ratio of COLOR remains bounded above by $O(\Delta(G)\log\log n/\log n)$. Therefore, in this case also, algorithm COLOR dominates the one of theorem 4.

Corollary 1.

- *For $\ell \geqslant 1$, C is approximable within $O(n/\log^{\ell-1} n)$ in graphs with maximum degree at most $n/\log^{\ell-2} n\log\log n$;*

- *C is approximable within* $O(\Delta(G)\log\log n/\log n)$ *in graphs with maximum degree at least* $n/\log^{\ell-2} n \log\log n$.

Remark finally that ratio $\Delta(G)/\chi(G)$ can always be achieved if after the first line of COLOR algorithm Δ_COLOR(G) is executed and if the minimum between the solution computed by this call and the set $\hat{X} \cup \tilde{X}$ is finally retained. In this case, the ratio achieved is

$$\min\left\{\frac{\Delta(G)}{\chi(G)}, \max\left\{O\left(\frac{n}{\log^{\ell-1} n}\right), O\left(\frac{\Delta(G)\log\log n}{\log n}\right)\right\}\right\}.$$

9 Inapproximability results

9.1 Negative results via graph-theoretic gap techniques

Historically, the first negative approximation result about C is the one affirming that no PTAA can guarantee approximation ratio strictly smaller than 4/3. This result can be obtained by application of the following more general theorem.

Theorem 13. *([28]) Let* Π *be a minimization problem having all solution values in* $\mathbb{N}^+$, *and suppose that, for some fixed* $\kappa \in \mathbb{N}^+$, *the decision-problem* Π_κ: *"given an instance* I *of* Π, *is* $OPT(I) \leqslant \kappa$?" *is NP-complete. Then, unless* $P = NP$, *no PTAA* A *for* Π *can guarantee* $\rho_A < 1 + (1/\kappa)$. *As a consequence,* Π *cannot be solved by a PTAS.*

Really, consider a problem Π verifying the hypotheses of the theorem and suppose that there exists a PTAA A guaranteeing, $\forall I$, $\rho_A(I) = A(I)/\mathrm{OPT}(I) < (\kappa+1)/\kappa$. Consider also a typical instance I of Π. We will show how A can be used to solve Π_κ in polynomial time. We run A on I. Then, if:

$A(I) > \kappa + 2$, the inequality $A(I) < ((\kappa+1)/\kappa)\mathrm{OPT}(I)$ leads to $\mathrm{OPT}(I) > \kappa(\kappa+2)/(\kappa+1) > \kappa$ and A answers "no" for Π_κ;

$A(I) \leqslant \kappa$, in this case $\mathrm{OPT}(I) \leqslant A(I) \leqslant \kappa$ and A answers "yes" for Π_κ;

$A(I) = \kappa + 1$, the expression $\kappa + 1 = A(I) < ((\kappa+1)/\kappa)\mathrm{OPT}(I)$ gives $\mathrm{OPT}(I) > \kappa$ and A answers "no" for Π_κ.

Consequently, given $A(I)$ (obtained in polynomial type following the hypothesis on A), one can solve Π_κ in polynomial time, a contradiction.

The result of theorem 13 has a direct application in C. In fact, since deciding if a graph is 3-chromatic (denote this problem by C_3) is NP-complete, C cannot be approximated within a ratio 4/3, unless P = NP; so, we get the following corollary.

Corollary 2. *No PTAA for C can guarantee ratio strictly smaller than 4/3, unless* $P \neq NP$. *Consequently, C cannot be solved by a PTAS.*

Always in [28], the above result is further strengthened to apply even to asymptotic ratios and the following theorem holds.

Theorem 14. *([28]) No PTAA* A *for* Π *can guarantee* $\rho_A^\infty < 4/3$ *for C, unless* $P = NP$.

The proof of the above theorem is based upon a classical technique, that can be seen as a polynomial reduction of a problem Π' to a problem Π, establishing that if Π is constant-approximable by an algorithm A, then a *gap* between the values of $A(I)$ would allow us to correctly answer "yes" or "no" about the decision-version of Π'. For the case of theorem 14, the key-idea for the proposed

reduction is the construction of a kind of *graph-composition* (or graph-product) where, given two graphs G_1 and G_2, their composition graph $G_1[G_2]$ can be produced by replacing each vertex of G_1 by a copy of G_2 and then replacing each edge of G_1 by a complete bipartite graph joining every vertex in the copy representing to one endpoint to every vertex corresponding in the copy corresponding to the other endpoint. So, if we consider an instance G of C_3, by constructing, for a sufficiently large m, the composition $\tilde{G} = K_m[G]$, then we can prove that following the values of the coloring-solution provided by a hypothetical constant-ratio PTAA for C in $\tilde{G}$, one can correctly determine if G is 3-chromatic or not.

In [27], based upon a finer very nice gap-technique, the approximation-bound of theorem 14 is non-trivially improved and the following result is proved.

Theorem 15. *([27]) No PTAA A for Π can guarantee $\rho_A^\infty < 2$ for C, unless $P = NP$.*

9.2 Negative results via interactive proofs

The result of theorem 15 remained for long years the strongest one about C. But in 1991, the notion of *transparent proof* was introduced by Babai et al. ([7]) and has generated the exciting concept of *probabilistically checkable proofs* or *interactive proofs*. Shortly, in language-theoretic (and hence in complexity-theoretic) terms, an interactive proof system is a kind of particular conversation between a prover (P) sending a string ℓ, and a verifier (V) accepting or rejecting it; this system recognizes a language L if, **(i)** for any string ℓ of L (sent by P), V always accepts it, and **(ii)**, for any $\ell \notin L$, neither P, nor any imposter substituting P, can make V accept ℓ with probability greater than $1/3$. An alternative way of seeing this type of proofs is as maximization problems for which the objective is to find strategies (solutions) maximizing the probability that V accepts ℓ.

Interactive proofs have produced novel very fine characterizations of the set of NP languages, giving rise to the development of very sophisticated gap-techniques (drawn rather in an algebraic spirit than in a graph-theoretic one) that lead to extremely important corollaries in the seemingly unrelated (to the one of probabilistically checkable proofs) domain of polynomial approximation theory. The most important among these corollaries, that has constituted a kind of break-through for the achievement of negative answers for numerous open problems in polynomial approximation, is the one of Arora et al. ([4]) that no MAX-SNP-hard problem admits polynomial time approximation schema. The class MAX-SNP is introduced by Papadimitriou and Yannakakis in [47] and, informally, is a class of problems admitting PTAAs with constant ratio. The completeness of a particular problem in MAX-SNP, holding under a special kind of ratio-preserving reduction, called L-reduction in [47], means that if this problem admits a PTAS, then so do all the problems in MAX-SNP.

For the case of C, Lund and Yannakakis, inspired from thought processes and tools developed in [4], prove in 1992 the following theorem.

Theorem 16. *([43]) There exists $\eta > 0$ such that it is NP-hard to approximately solve C within standard-approximation ratio n^η.*

Theorem 16 does not precise values for factor η (called *hardness threshold* in the literature). Nevertheless, this result has the great merit to use interactive proof systems dealing with minimization problems, while these systems seem to be (intuitively) better-adapted for maximization ones.

Subsequent strengthenings of the above negative results, obtained thanks to characterizations of NP finer than the ones of [4], have been performed by Bellare and Sudan in [9] and increased the factor η. In [9], several such factors under several complexity theory hypotheses are provided. For instance, $\eta = 1/10$ under the hypothesis NP $\not\subseteq$ co$\tilde{\text{RP}}$, $\eta = 1/13$ under the hypothesis

coRP $\neq$ NP, $\eta = 1/14$ under the hypothesis P $\neq$ NP (for the definition of the complexity classes mentioned, see [38, 46]). In [8], further strengthenings are obtained, always using characterization of complexity classes via interactive proofs. The main results providing the best known hardness factors asymptotically different from both 0 and 1 (in any case $\eta \in]0,1[$) are the ones of the following theorem.

Theorem 17. *C cannot be approximated:*

- *within ratio $n^{1-\epsilon}$ for any constant $\epsilon > 0$ unless NP $\subseteq$ coRP ([26]);*
- *within ratio $n^{(1/5)-\epsilon}$ for any $\epsilon > 0$, assuming NP $\neq$ coRP ([8]);*
- *within ratio $n^{(1/7)-\epsilon}$ for any $\epsilon > 0$, assuming P $\neq$ NP ([8]).*

Part II
Graph-coloring and differential approximation: maximizing the number of unused colors

10 A few words about the worst-value solution for graph-coloring

The δ-framework has been axiomatized in [21](although it has marginally and occasionally been used by some researchers, cf. [6], since 1980, this use was restricted to particular problems).

Key-requirement of the δ-framework is the stability of any adopted approximation ratio with respect to the affine transformation of the objective function. In what follows, we call *affine-equivalent* problems for which the objective function of the former is an affine transformation of the objective function of the latter. Affine transformation is very natural and frequent in combinatorial optimization (the pair independent set – vertex covering is the most known but not the only example of such a transformation), and the stability of the approximation ratio under this type of transformation is not taken into account in the ρ-framework.

We give in what follows polynomial differential-approximation results for C. As we will see, in this framework C is well-approximable since one succeeds to devise constant-ratio DPTAAs. But first, let us make some remarks that will immediately introduce an information about the worst-value solution of an instance of C (recall that in differential-approximation ratio intervenes, except the values of the approximate and of the optimal solution, the value of the worst case one).

Let A be the edge-vertex incidence matrix of G. In order to define C as a mathematical program, we have to define a priori a set of eventual colors X; let $|X| = l$. The variables of the program are then (i) $\vec{y} \in I\!R^l$, the characteristic vector of the selected colors of X, and (ii) l vectors $\vec{x_i} \in I\!R^n$, $i \in \{1, \ldots, l\}$, the characteristic vectors of the independent sets corresponding to each one of the l colors. More precisely, C can be formulated as follows:

$$
\mathrm{C} = \left\{ \begin{array}{lll}
\min & \vec{1} \cdot \vec{y} & \\
 & A.\vec{x_i} \leqslant \vec{1} & \forall i \in \{1, \ldots, l\} \\
 & \vec{x_i} - y_i.\vec{1} \leqslant 0 & \forall i \in \{1, \ldots, l\} \\
 & \sum_{i=1}^{l} \vec{x_i} = \vec{1} & \\
 & \vec{y} \in \{0,1\}^l & \\
 & \vec{x_i} \in \{0,1\}^n & \forall i \in \{1, \ldots, l\}
\end{array} \right.
$$

We distinguish four blocks of constraints: the l stability constraints of $\vec{x_i}$, the l exclusion constraints meaning that if color i is not selected, then the independent set having characteristic vector $\vec{x_i}$ is empty, the partition constraint guaranteeing that every vertex is colored with exactly one color, and finally, the 0-1 usual constraints for the characteristic vectors. We can choose $n = l$, i.e., we consider that there is no more colors than vertices in G. This very simple remark supposes that we have anyway a certain initial knowledge of the problem without which we would not be able to define $\vec{y}$. Solution $\vec{y} = \vec{1}$ corresponds to the solution where we affect a distinct color per vertex, solution "unwarranted" and feasible for every graph. Consequently, one can consider that one has a resource of n colors for coloring the vertices of G. Then, if a PTAA A computes a coloring of size $\chi_A(G)$ for G, the quantity $n - \chi_A(G)$ (the enumerator of the differential approximation ratio) is exactly the number of colors A has left unused. Hence, under the differential approximation, C has a natural and picturesque interpretation as the problem of maximizing the number of unused colors.

11 A differential-approximation algorithm based upon affine-equivalence

In [19], the following thought process for the differential-approximation of C is proposed:

(i) transform C into the following affine-equivalent maximization problem AEC: "given a graph $G = (V, E)$, find a partial sub-graph H of $\bar{G}$ (the complement of G) having a maximum number of edges and such that

  (a) H is acyclic (or, equivalently, H is a forest), and

  (b) every connected component (tree) of H is included in a clique of $\bar{G}$";

(ii) devise a DPTAA for AEC and run it on $\bar{G}$;

(iii) transform the solution provided for AEC into a solution for C (thanks to the affine-equivalence, the two approximate solutions will induce the same differential-approximation ratio).

Items (ii) and (iii) of the above thought process are summarized into the following DPTAA for C where we denote by T a tree of H.

```
BEGIN *D_COLOR1*
(1)  determine Ḡ;
(2)  compute a maximum matching M in Ḡ;
(3)  H ← M;
(4)  FOR e ∈ E DO IF H∪{e} is feasible for AEC THEN H ← H∪{e} FI OD
(5)  FOR T ∈ H DO color the vertices of T with a new color OD
(6)  color the vertices of V \ V(H) using a new color per vertex;
(7)  OUTPUT the set of used colors;
END. *D_COLOR1*
```

Theorem 18. *([19]) AEC is affine-equivalent to C. Algorithm D_COLOR1 is an $O(n^{2.5})$ DPTAA achieving differential-approximation ratio 1/2 for both C and AEC. This ratio is tight for D_COLOR1.*

Despite the fact that, as we shall see, the approximation ratio induced by theorem 18 has been substantially improved in the sequel, it has its own interest since it is obtained by forwardly exploiting the notion of affine-equivalence that, as it is shown in [21] as well as in a later paper ([20]), can reveal very useful in analyzing the approximation performance of DPTAAs for several NP-hard problems.

12 From matching to 3 - independent sets

A little later in 1994, Hassin and Lahav ([34]) have improved the differential-approximation of C by proposing the following algorithm directly working on the instance of C. In what follows, we denote by i-IS an independent set of size i.

```
BEGIN *D_COLOR2*
(1)   WHILE there exists 3-IS S ⊆ V DO
(2)         color the vertices of S with a new color;
(3)         V ← V \ S;
(4)         G ← G[V];
(5)   OD
(6)   compute a maximum collection C of 2-IS in G;
(7)   FOR S ∈ C DO color S with a new color OD
(8)   color G[V \ V(C)] using a new color per vertex;
(9)   OUTPUT the set of used colors;
END. *D_COLOR2*
```

Theorem 19. *([34]) Algorithm D_COLOR2 is an $O(n^{2.5})$ differential-approximation algorithm for C achieving a differential-approximation ratio 2/3. This bound is tight for D_COLOR2.*

One can remark that there exist many similarities between algorithms D_COLOR1 and D_COLOR2. The only notable difference is that the former, instead of searching for independent sets on 2 vertices (a collection of independent sets on 2 vertices in a graph corresponds to an equal-sized matching in its complement), first searches for such sets on 3 vertices. If we keep the same spirit of these algorithms, there exists a natural way to improve the approximation ratio for C:

(i) by replacing the independent sets on 3 vertices by larger ones, and

(ii) by devising efficient approximation algorithms for the case where the surviving graph has independence number greater than, or equal to, 3.

13 Coloring graphs via set-covering

A clever improvement of the approximation ratio for C, using the solution of a particular SC problem, is presented by Halldórsson in [33]. Before outlining the devised algorithm, we need to describe a kind of local improvement, called t-improvement in [33], applied on any SC-solution.

Consider an instance $(\mathcal{S}, C)$ of SC and a set covering $\mathcal{S}' \subseteq \mathcal{S}$. A *$t$-improvement* of $\mathcal{S}'$ is formed by sets $\tilde{S}'_1, \tilde{S}'_2, \ldots, \tilde{S}'_t$ in $\mathcal{S}'$ and by sets $S_1, S_2, \ldots, S_{t-1}$ in $\mathcal{S}$ such that $\mathcal{S}'' = (\mathcal{S}' \setminus \{\tilde{S}'_1, \tilde{S}'_2, \ldots, \tilde{S}'_t\}) \cup \{S_1, S_2, \ldots, S_{t-1}\}$ is also a cover. Obviously, $|\mathcal{S}''| < |\mathcal{S}'|$. A cover is *$t$-optimal* if it contains no t-improvement. Moreover, for fixed t, t-improvements can be done in polynomial time, so does the verification of t-optimality.

In what follows, let us denote by 3SC the class of SC where the sets of $\mathcal{S}$ are all of cardinalities at most equal to 3; moreover, we recall that the 2SC, i.e., the SC with sets of cardinality at most 2, can be solved in polynomial time ([10, 28]). In fact, given an instance $I = (\mathcal{S}, C)$ of 2SC, one can construct a graph $G_I = (V, E)$ where $V = C$ and $E = \{v_i v_j : \exists S = \{c_i, c_j\} \in \mathcal{S}\}$[3]. Then, every edge-covering of G_I (i.e., every set of edges covering the vertices of V) corresponds exactly to a set-covering of the same size of I. Consequently, optimal 2SC-solution of I is a

[3]Of course, this transformation is reversible, i.e., given a graph G, one can obtain a 2SC-instance $I_G = (\mathcal{S}, C)$ setting $V = C$ and $\mathcal{S} = \{\{c_i, c_j\} : v_i v_j \in E\}$.

minimum edge-covering of G_I and finding such an edge-covering can be performed in polynomial time ([10]).

The following PTAA, where we denote by 2_SC the algorithm optimally solving 2SC in polynomial time, is proposed in [33] for 3SC. Moreover, algorithm t_IMPROVE called in line (6) of algorithm 3_SC repeatedly applies t-improvements until t-optimality.

```
BEGIN *3_SC*
(1)  find a maximal collection S_1 of mutually disjoint sets in (S, C);
(2)  C' ← C \ ∪_{S_i ∈ S_1} S_i;
(3)  S' ← {S_i ∩ C' : S_i ∈ S, S_i ∩ C' ≠ ∅};
(4)  S_2 ← 2_SC(S', C');
(5)  fix a large integer t;
(6)  OUTPUT S̃ ← t_IMPROVE(S_1 ∪ S_2);
END. *3_SC*
```

It is easy to see that the instance of SC obtained in line (3) of algorithm 3_SC is really an instance of 2SC, so it can be optimally solved in polynomial time in line (4).

If one chooses an even fixed t arbitrarily large, the analysis of algorithm 3_SC, performed in [33] produces an approximation ratio at most $(7/5)+o(4/t)$, that constitutes also an interesting improvement for the approximation ratio of 3SC. The only draw-back of this result is that the complexity of the algorithm is of $O(n^{O(t)})$ and therefore the closer to 7/5 the ratio, the higher the execution time of algorithm 3_SC; but, in any case, even of high complexity, it remains, for a fixed t, always polynomial.

The following algorithm is the one proposed for C in [33] (we use the term 4-IS to denote an independent set on 4 vertices); moreover, once more we suppose that colors are integers in $\{1, \ldots, n\}$.

```
BEGIN *D_COLOR3*
(1)  find a maximal collection S_4 of mutually disjoint 4-IS's V_i in G;
(2)  FOR i ← 1 TO |S_4| DO color the vertices of V_i with i OD
(3)  V ← V \ ∪_{V_i ∈ S_4} V_i;
(4)  G ← G[V];
(5)  find the collection S_3 of all the independent sets of G;
(6)  S̃ ← 3_SC((S_3, V));
(7)  color each member of S̃ with a new unused color;
(8)  OUTPUT the union of colors used in lines (2) and (7);
END. *D_COLOR3*
```

In algorithm D_COLOR3, since all the disjoint independent sets on 4 vertices have been removed in line (1), the surviving graph G has independence number at most 3; so, the instance $(\mathcal{S}_3, V)$ on which algorithm 3_SC is applied is really a 3SC-one (constructed in polynomial time since $|\mathcal{S}_3| = O(|V|^3)$).

Theorem 20. *([33]) Algorithm D_COLOR3 is a DPTAA for C achieving in $O(n^{O(t)})$ differential-approximation ratio 3/4.*

Let us note that, in [32], a PTAA of approximation ratio 5/7 for C is presented. This algorithm is essentially the same as algorithm D_COLOR3, but it contains no t-improvements.

14 Coloring, set-covering and semi-local optimization

Revisit for a while the t-improvement seen in section 13 and its application at line (6) of algorithm 3_SC. It is easy to see that the intuition behind such improvement is the replacement

of a constant number of sets in the current cover by a hopefully smaller number of other sets in such a way that the cover so obtained remains feasible. In [24], the following refinement of t-improvement, called *semi-local improvement*, is proposed. A *semi-local (s,t)-improvement for 3SC ($s > t$) consists of the insertion of at least s sets of size 3 and the deletion of at least t such sets from the current cover.* In addition the elements remained uncovered by the semi-local improvement are optimally covered by algorithm 2_SC. In other words in such improvement, one tries to augment the number of the 3-sets used in the cover (covering so more elements by these sets) and to reduce the number of the smaller sets used (2- and 1-sets).

Based upon semi-local improvement, a refinement of algorithm D_COLOR3 is devised in [24]. The basic idea remains the same: one greedily finds a collection of independent sets of up to a certain constant size $k+1$, colors any of them with a new color and removes all of them from the input graph; next she/he transforms the surviving graph into an instance of kSC as we have seen in section 13 and approximately solves kSC in this instance. In [24], $k = 6$. The algorithm for 6SC proposed includes three phases. The first one is totally greedy and consists of finding a maximal collection of disjoint 6-sets in the initial 6SC-instance $(\mathcal{S}, C)$. The elements covered by this collection are next removed from C and the remaining sets are updated. Of course this update will eventually create some 1-sets. The second phase is more restrictive than the first one. Here a maximal collection of disjoint 5-sets, then 4-sets is constructed but with the restriction that any such set is chosen to make part of the collection only if its choice does not increase the number of 1-sets created during the first phase. This is done greedily by considering a set and by examining if the removal of its elements will create additional one sets. In what follows, we denote by STRICT_PHASE the procedure implementing the second phase. The elements covered by collection so-constructed are removed from C and the remaining sets are updated. Finally, the third and last phase, applied in the surviving 3SC-instance, is a semi-local (2,1)-improvement. In what follows, we denote by SL_OPT21 the algorithm repeatedly applying semi-local (2,1)-improvement until no such improvement is possible.

```
BEGIN *6_SC*
      greedily choose a maximal collection S_1 of mutually disjoint 6-sets in (S,C);
      C ← C \ ∪_{S_i ∈ S_1} S_i;
      S ← {S_i ∩ C : S_i ∈ S, S_i ∩ C ≠ ∅};
      S_2 ← STRICT_PHASE(S,C);
      C ← C \ ∪_{S_i ∈ S_2} S_i;
      S ← {S_i ∩ C : S_i ∈ S, S_i ∩ C ≠ ∅};
      S_3 ← SL_OPT21(S,C);
      OUTPUT S̃ ← S_1 ∪ S_2 ∪ S_3;
END. *6_SC*
```

Based upon algorithm 6_SC, the following coloring algorithm is proposed in [24] (we always suppose that colors are integers in $\{1,\ldots,n\}$).

```
BEGIN *D_COLOR4*
(1)  find a maximal collection S_7 of mutually disjoint 7-IS's V_i in G;
(2)  FOR i ← 1 TO |S_7| DO color the vertices of V_i with i OD
(3)  V ← V \ ∪_{V_i ∈ S_7} V_i;
(4)  G ← G[V];
(5)  find the collection S_6 of all the independent sets of G;
(6)  S̃ ← 6_SC((S_6,V));
(7)  color each member of S̃ with a new unused color;
(8)  OUTPUT the union of colors used in lines (2) and (7);
END. *D_COLOR4*
```

Theorem 21. *Algorithm D_COLOR4 is a DPTAA for C achieving in $O(n^{O(t)})$ differential-approximation ratio 289/360.*

15 Unifying the above algorithms

We now show that the four D_COLOR-algorithms presented above are instantiations of the following general schema (parameterized by a graph G and an integer k). In what follows, we denote by k_SC a PTAA for kSC, i.e., the class of SC where the sets of $\mathcal{S}$ are all of sizes at most k.

```
BEGIN *DC_SCHEMA*
(1)  find a maximal collection S_{k+1} of mutually disjoint (k+1)-IS's V_i in G;
(2)  FOR i ← 1 TO |S_{k+1}| DO color the vertices of V_i with i OD
(3)  V_k ← V \ ∪_{V_i ∈ S_{k+1}} V_i;
(4)  G_k ← G[V_k];
(5)  find the collection S_k of all the independent sets of G;
(6)  S̃ ← k_SC((S_k, V_k));
(7)  color each member of S̃ with a new unused color;
(8)  OUTPUT the union X_DC of colors used in lines (2) and (7);
END. *DC_SCHEMA*
```

Obviously, if k is fixed, line (5) can be executed in polynomial time (at most $O(n^k)$). Moreover, the set-system $(\mathcal{S}_k, V)$ is an instance of kSC.

Theorem 22. *If algorithm k_SC approximately solves kSC within differential approximation ratio $\delta \leqslant k/(k+1)$, then DC_SCHEMA approximately solves C in polynomial time within differential-approximation ratio δ.*

Proof. Let χ_{k+1} and χ_k be the numbers of the colors assigned by the execution of lines (2) and (7), respectively, and denote by $\chi_{\mathrm{DC}}(G)$ the number of colors computed at line (8), i.e., the size of set X_{DC} produced in line (8).

Since algorithm k_SC achieves differential approximation ratio δ, the following holds for any kSC-instance I

$$\left|\tilde{S}\right| \leqslant (1-\delta)\mathrm{WORST}(I) + \delta\mathrm{OPT}(I) \tag{11}$$

In general, when dealing with SC, there exist two natural values that can be considered as worst solution-values for an instance $I = (\mathcal{S}, C)$: $|\mathcal{S}|$ and $|C|$. The former corresponds in taking all the sets of the family $\mathcal{S}$ and the latter in taking a subset of $\mathcal{S}$ per element of C. In order to be as restrictive as possible, it seems natural to consider

$$\mathrm{WORST}\left((\mathcal{S}, C)\right) = \min\left\{|\mathcal{S}|, |C|\right\}.$$

In instance $(\mathcal{S}_k, V_k)$ (line (6)), since all the independent sets have already been produced in line (5), singletons, corresponding to vertices of V_k are also included. Hence, a solution taking a set per vertex of V_k is feasible, and moreover, $|\mathcal{S}_k| \geqslant |V_k|$. Consequently,

$$W\left((\mathcal{S}_k, V_k)\right) = |V_k| = n - (k+1)\chi_{k+1}. \tag{12}$$

So, using expression (11) for $I = (\mathcal{S}_k, V_k)$, expression (12) and the facts: (i) the optimal set-cover of $(\mathcal{S}_k, V_k)$ is in 1–1 correspondence to the optimal coloring of G_k, and (ii), $\chi(G_k) \leqslant \chi(G)$, the following holds:

$$\chi_k \leqslant (1-\delta)\,|V_k| + \delta\mathrm{OPT}\left((\mathcal{S}_k, V_k)\right) \leqslant (1-\delta)\left(n - (k+1)\chi_{k+1}\right) + \delta\chi(G) \tag{13}$$

On the other hand, $\chi_{\text{DC}}(G) = \chi_{k+1} + \chi_k$ and using expression (13), we get

$$\begin{array}{rcl} \chi_{\text{DC}}(G) & \leqslant & \chi_{k+1} + (1-\delta)\,(n - (k+1)\chi_{k+1}) + \delta\chi(G) \\ & \Downarrow & \\ n - \chi_{\text{DC}}(G) & \geqslant & \delta\,(n - \chi(G)) + \chi_{k+1}\,(k - \delta(k+1)) \;\geqslant\; \delta\,(n - \chi(G)) \end{array}$$

for $\delta \leqslant k/(k+1)$. We so derive $(n - \chi_{\text{DC}}(G))/(n - \chi(G)) \geqslant \delta$, that concludes the proof of the theorem. ∎

The result of theorem 22 can be extended to work for any δ in the following way. Set, without loss of generality, $\mathcal{S}_{k+1} = \{S_1, S_2, \ldots, S_q\}$, the collection produced at line (1) of algorithm DC_SCHEMA; $S_i = k+1$, $i = 1, \ldots, q$. Then the following holds:

$$\text{WORST}(G) = \text{WORST}\,(\mathcal{S}_k, V_k) + q(k+1) \tag{14}$$

$$\chi_{\text{DC}}(G) = \chi_k + q \tag{15}$$

$$\chi\,((G_k) \leqslant \chi(G) \tag{16}$$

Combining expressions (14), (15) and (16), we get

$$\begin{array}{rcl} \dfrac{\text{WORST}(G) - \chi_{\text{DC}}(G)}{\text{WORST}(G) - \chi(G)} & \geqslant & \dfrac{\text{WORST}\,(\mathcal{S}_k, V_k) + q(k+1) - (\chi_k + q)}{\text{WORST}\,(\mathcal{S}_k, V_k) + q(k+1) - \chi\,(G_k)} \\ & = & \dfrac{\text{WORST}\,(\mathcal{S}_k, V_k) - \chi_k + qk}{\text{WORST}\,(\mathcal{S}_k, V_k) - \chi\,(G_k) + q(k+1)} \;\geqslant\; \min\left\{\delta, \dfrac{k}{k+1}\right\}. \end{array}$$

In all, the discussion just above derives the following theorem.

Theorem 23. *If algorithm k_SC approximately solves kSC within differential approximation ratio δ, then DC_SCHEMA approximately solves C in polynomial time within differential-approximation ratio* $\min\{\delta, k/(k+1)\}$.

15.1 Recovering algorithm D_COLOR4

Algorithm DC_SCHEMA(G,6) is nothing else than algorithm D_COLOR4. A careful lecture of the proof of theorem 4.2 (pp. 260–261) of [24] together with a preliminary remark just above the statement of theorem 4.2 (... *instead of charging cost 1 to a chosen set, now we will charge a cost* $(k-1)$ *to a chosen k-set* ...), make clear that this proof is also the one of the fact that *the differential-approximation ratio of algorithm 6_SC is bounded above by 289/360.* Then, application of theorem 22 derives the ratio claimed by theorem 21.

15.2 Recovering algorithm D_COLOR3

Suppose that DC_SCHEMA(G,3) is executed. This is exactly algorithm D_COLOR3 that calls algorithm 3_SC for which the following is proved in [33]:

$$\left|\tilde{\mathcal{S}}\right| \leqslant \frac{1}{4}\text{WORST}\,((\mathcal{S}_3, V)) + \frac{3}{4}\left|\hat{\mathcal{S}}\right|$$

where $\tilde{\mathcal{S}}$ is the solution produced at line (6) of algorithm 3_SC and $\hat{\mathcal{S}}$ is some cover of the system $(\mathcal{S}, C)$. Consequently, theorem 22 is applied with $\delta = 3/4$.

15.3 Recovering algorithm D_COLOR2

Revisit now algorithm D_COLOR2 and observe that lines (6) to (9) can be seen as the polynomial-time algorithm of [10] optimally solving 2SC, or, equivalently, minimum edge-covering in the graph G input of line (6).

Then, it is easy to see that DC_SCHEMA(G,2) is exactly D_COLOR2. Let us denote by $\mathrm{OPT}(I_G)$ the optimal 2SC-solution computed by algorithm D_COLOR2 (see footnote 3), by $S(I_G)$ some other 2SC-solution and by $\mathrm{WORST}(I_G)$ the worst-value one consisting of taking an edge per vertex of G (i.e., a set per element of C); obviously, $\mathrm{WORST}(I_G) = |G|$. We then have:

$$\mathrm{OPT}(I_G) = \frac{1}{3}\mathrm{OPT}(I_G) + \frac{2}{3}\mathrm{OPT}(I_G) \leqslant \frac{1}{3}|G| + \frac{2}{3}|S(I_G)|.$$

In other words, we are in the case of an application of theorem 22 with $\delta = 2/3$.

15.4 Recovering algorithm D_COLOR1

Observe that the subgraph H produced in line (3) is feasible for AEC. Moreover, execution of line (4) may, at worst, not change the number of the connected components of H. In this case, algorithm D_COLOR1 is nothing else than DC_SCHEMA(G,1) and the SC-instance produced at line (6) of DC_SCHEMA is an 1SC with the vertices of $V \setminus V(H)$ as ground set and the singletons containing these vertices as collection of subsets. Taking all these singletons is an optimal solution for the corresponding instance. Then, in a completely analogous way as the one of section 15.3 and denoting by X_{D1} the set of colors retained in line (7) of D_COLOR1, we get,

$$\mathrm{OPT}(G[V \setminus V(H)]) = |X_{\mathrm{D1}}| = \frac{1}{2}|V \setminus V(H)| + \frac{1}{2}\mathrm{OPT}(G[V \setminus V(H)]).$$

We are here in the case of an application of theorem 22 with $\delta = 1/2$.

16 Using Δ_COLOR to color "sparse" graphs

Consider a graph G verifying $\Delta(G) = o(n)$. Then, it is immediate to see that running algorithm Δ_COLOR in such graphs provide colorings achieving approximation ratio

$$\frac{n - o(n)}{n - \chi(G)} \underset{n\to\infty}{\longrightarrow} 1.$$

Proposition 6. *Whenever* $\Delta(G) = o(n)$, $\lim_{n\to\infty} \delta_{\Delta\_\mathrm{COLOR}} = 1$.

17 Negative results

Since C is well-approximable in δ-framework, the negative results one can obtain here have a character much less "dramatical" than the ones of the ρ-framework.

In an unpublished paper of 1993 (some of its results have later been published in [22]), the following inapproximability result has been proved for C.

Proposition 7. *Unless $P = NP$, C cannot be approximated neither by a DFPTAS, nor by an asymptotic DFPTAS guaranteeing, for every $\epsilon > 0$, differential-approximation ratio of the form $\delta = 1 - \epsilon - \{1/[n - \chi(G)]\}^{\eta}$ where η is a fixed positive constant.*

This result has been strengthened in [32] where, by a reduction from the 3-dimensional matching, it is shown that C is MAX-SNP-hard.

Theorem 24. *([32]) C is MAX-SNP-hard. Hence, it cannot be approximated by a DPTAS, unless $P = NP$.*

Let us recall that in [48], an optimization problem is called *simple* if, for every fixed constant k, its restriction to instances verifying $\mathrm{OPT}(I) \leqslant k$ can be optimally solved in polynomial time. For instance, IS or SC are simple, while C (since 3-coloring is NP-complete) or *bin-packing* are not. The notion of simplicity has a natural extension in the differential-approximation framework where we call *D-simple* an optimization problem, the restriction of which to instances verifying $|\mathrm{WORST}(I) - \mathrm{OPT}(I)| \leqslant k$ can be optimally solved in polynomial time (observe that $\sigma(I) \leqslant |\mathrm{WORST}(I) - \mathrm{OPT}(I)|$). For instance, it is easy to see that IS, SC or, even, C and bin-packing are D-simple. For D-simple problems the following theorem can be proved (see also [45]).

Theorem 25. *Every D-simple problem has a DPTAS iff it has an asymptotic DPTAS.*

Proof. Suppose Π a minimization problem, i.e., $\mathrm{WORST}(I) - \mathrm{OPT}(I) \geqslant 0$. Of course, if Π has a DPTAS, then it has an asymptotic one. Suppose that Π has an asymptotic DPTAS with a constant c (see definition of asymptotic DPTAS in section 1) and denote by `EXACT` the algorithm (parameterized by an instance I of Π and an integer $k > 0$) deciding if $|\mathrm{WORST}(I) - \mathrm{OPT}(I)| \leqslant k$ and, if yes, computing the optimal solution of I. Also, denote by `AD_SCHEMA` the asymptotic DPTAS for Π and suppose that it is parameterized by I and $\epsilon > 0$. Then, the DPTAS claimed is as follows.

```
BEGIN *D_SCHEMA*
      fix a constant ε > 0;
      IF EXACT(I, 2c/ε) successfully terminated THEN OUTPUT EXACT(I, 2c/ε) FI;
      OUTPUT L(I) ← AD_SCHEMA(I, ε/2);
END. *D_SCHEMA*
```

We now prove that `D_SCHEMA` is a DPTAS for Π. Observe first that $2c/\epsilon$ is a fixed constant. Moreover, if `EXACT`$(\mathtt{I}, 2c/\epsilon)$ has not successfully terminated, then

$$\mathrm{WORST}(I) - \mathrm{OPT}(I) > \frac{2c}{\epsilon} \Longrightarrow c \leqslant \frac{\epsilon}{2}(\mathrm{WORST}(I) - \mathrm{OPT}(I)) \tag{17}$$

In this case, execution of `AD_SCHEMA` achieves

$$\mathrm{WORST}(I) - \mathrm{L}(I) \geqslant \left(1 - \frac{\epsilon}{2}\right)(\mathrm{WORST}(I) - \mathrm{OPT}(I)) - c \tag{18}$$

Combining expressions (17) and (18), one immediately gets

$$\frac{\mathrm{WORST}(I) - \mathrm{L}(I)}{\mathrm{WORST}(I) - \mathrm{OPT}(I)} \geqslant 1 - \epsilon$$

concluding so that `D_SCHEMA` is a DPTAS for Π. ∎

Combining theorems 24 and 25, the following concluding corollary holds.

Corollary 3. *Unless $P = NP$, C cannot be solved by an asymptotic DPTAS.*

18 Final remarks

For reasons of size of the paper, we have not extensively discussed approximation results on k-chromatic graphs (the most popular of them being the 3-chromatic ones), except when such results are used to produce approximation ratios for the general coloring problem.

In the ρ-framework, positive results for k-chromatic graphs are given in [13, 39], while for the special case of $k = 3$, one can refer to [13, 14, 39]. Let us note once more that the results in [39] and [14] (ratios of values bounded above by $O(|V|^{1-(3/(k+1))} \log |V|)$ and $O(|V|^{3/14} \log^{O(1)} |V|)$, respectively) are originally guaranteed by randomized PTAAs. These algorithms have been derandomized in [44] in such a way that they still attain the same performance guarantees. Finally, the stronger inapproximability result for 3-coloring is, to our knowledge, the one of [41], where a lower bound of $5/3 - \epsilon$, $\forall \epsilon > 0$, for the ratio of every 3-coloring PTAA is provided.

When k is fixed, the results of [14, 39, 44] induce that one can legally color the vertices of a graph in polynomial time with $o(n)$ colors. Then, the differential-approximation ratios of the corresponding algorithms are $(n - o(n))/(n - k)$ and tend to 1. In other words, the differential approximation ratio for 3-coloring is asymptotically equal to 1.

Other interesting approximation issues not considered in this paper are the approximation of C in random graphs, or in special classes of graphs (for example in planar ones). Work about the former issue, as well as a certain number of references, are presented in [12, 13], while information about the latter can be found in [18].

Acknowledgment. The proofs in appendix 2 have been elaborated together with Olivier Spanjaard, PhD student at LAMSADE, University Paris-Dauphine. Cristina Bazgan and Jérôme Monnot have read the final version of this manuscript and have made many helpful and pertinent suggestions. Many thanks to all of them.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, Reading, MA, 1975.

[2] L. Alfandari and V. T. Paschos. Master-slave strategy and polynomial approximation. *Comput. Opti. Appl.*, 16:231–245, 2000.

[3] F. Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. In *Proc. FOCS'92*, pages 14–23, 1992.

[5] B. Aspvall and R. E. Stone. Khachiyan's linear programming algorithm. *J. Algorithms*, 1:1–13, 1980.

[6] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *J. Comput. System Sci.*, 21:136–153, 1980.

[7] L. Babai, L. Fortnow, L. A. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proc. STOC'91*, pages 21–31, 1991.

[8] M. Bellare, O. Goldreich, and M. Sudan. Free bits and non-approximability — towards tight results. *SIAM J. Comput.*, 27:804–915, 1998.

[9] M. Bellare and M. Sudan. Improved non-approximability results. In *Proc. STOC'94*, pages 184–193, 1994.

[10] C. Berge. *Graphs and hypergraphs.* North Holland, Amsterdam, 1973.

[11] B. Berger and J. Rompel. A better performance guarantee for approximate graph coloring. *Algorithmica*, 5:459–466, 1990.

[12] A. Blum. *Algorithms for approximate graph coloring.* PhD thesis, MIT, Laboratory for Computer Science, Cambridge Mass., USA, June 1991. Technical Report MIT/LCS/TR-506.

[13] A. Blum. New approximation algorithms for graph coloring. *J. Assoc. Comput. Mach.*, 41(3):470–516, 1994.

[14] A. Blum and D. Karger. An $\tilde{O}(n^{3/14})$-coloring for 3-colorable graphs. *Inform. Process. Lett.*, 61:49–53, 1997.

[15] B. Bollobás. *Graph theory. An introductory course.* Springer-Verlag, New York, 1979.

[16] B. B. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *BIT*, 32(2):180–196, 1992.

[17] R. L. Brooks. On coloring the nodes of a network. *Math. Proc. Cambridge Philos. Soc.*, 37:194–197, 1941.

[18] P. Crescenzi and V. Kann. A compendium of NP optimization problems. Available on www_address: `http://www.nada.kth.se/~viggo/problemlist/compendium.html`, 1995.

[19] M. Demange, P. Grisoni, and V. T. Paschos. Approximation results for the minimum graph coloring problem. *Inform. Process. Lett.*, 50:19–23, 1994.

[20] M. Demange, P. Grisoni, and V. T. Paschos. Differential approximation algorithms for some combinatorial optimization problems. *Theoret. Comput. Sci.*, 209:107–122, 1998.

[21] M. Demange and V. T. Paschos. On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoret. Comput. Sci.*, 158:117–141, 1996.

[22] M. Demange and V. T. Paschos. Asymptotic differential approximation ratio: definitions, motivations and application to some combinatorial problems. *RAIRO Oper. Res.*, 33:481–507, 1999.

[23] M. Demange and V. T. Paschos. Maximum-weight independent set is as "well-approximated" as the unweighted one. Technical Report 163, LAMSADE, Université Paris-Dauphine, 1999.

[24] R. Duh and M. Fürer. Approximation of k-set cover by semi-local optimization. In *Proc. STOC'97*, pages 256–265, 1997.

[25] P. Erdös and G. Szekeres. A combinatorial problem in geometry. *Compositio Math.*, 2:463–470, 1935.

[26] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proc. Conference on Computational Complexity*, pages 278–287, 1996.

[27] M. R. Garey and D. S. Johnson. The complexity of near-optimal graph coloring. *J. Assoc. Comput. Mach.*, 23(1):43–49, 1976.

[28] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* W. H. Freeman, San Francisco, 1979.

[29] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.*, 42(6):1115–1145, 1995.

[30] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.

[31] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Inform. Process. Lett.*, 45(1):19–23, 1993.

[32] M. M. Halldórsson. Approximating discrete collections via local improvements. In *Proc. Symposium on Discrete Algorithms*, pages 160–169, 1995.

[33] M. M. Halldórsson. Approximating k-set cover and complementary graph coloring. In *Proc. International Integer Programming and Combinatorial Optimization Conference*, number 1084 in Lecture Notes in Computer Science, pages 118–131. Springer Verlag, 1996.

[34] R. Hassin and S. Lahav. Maximizing the number of unused colors in the vertex coloring problem. *Inform. Process. Lett.*, 52:87–90, 1994.

[35] T. R. Jensen and B. Toft. *Graph coloring problems.* Wiley, 1995.

[36] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[37] D. S. Johnson. Worst-case behavior of graph-coloring algorithms. In *Proc. South-Eastern Conference on Combinatorics, Graph Theory and Computing*, pages 513–528, 1974.

[38] D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of theoretical computer science volume A: Algorithms and complexity*, pages 67–161. Elsevier, Amsterdam, 1990.

[39] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *J. Assoc. Comput. Mach.*, 45(2):246–265, 1998.

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

[41] S. Khanna, N. Linial, and S. Safra. On the hardness of approximating the chromatic number. In *Proc. Israel Symposium on Theory of Computing and Systems*, pages 250–260, 1993.

[42] L. Lovász. Three short proofs in graph theory. *J. Combin. Theory Ser. B*, 19:269–271, 1975.

[43] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. Assoc. Comput. Mach.*, 41(5):960–981, 1994.

[44] S. Mahajan and H. Ramesh. Derandomizing semidefinite programming based approximation algorithms. In *Proc. FOCS'95*, pages 162–169, 1995.

[45] J. Monnot. Familles critiques d'instances et approximation polynomiale. PhD thesis. LAMSADE, Université Paris-Dauphine, 1998.

[46] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[47] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43:425–440, 1991.

[48] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximations. *Theoret. Comput. Sci.*, 15:251–277, 1981.

[49] J. B. Shearer. A note on the independence number of triangle-free graphs. *Discrete Math.*, 46:83–87, 1983.

[50] J. B. Shearer. A note on the independence number of triangle-free graphs, ii. *J. Combin. Theory Ser. B*, 53:300–307, 1991.

[51] A. Wigderson. Improving the performance guarantee for approximate graph coloring. *J. Assoc. Comput. Mach.*, 30(4):729–735, 1983.

A Proof of the results of section 2

A.1 Proof of lemma 2

Since G is 3-connected, there always exist appropriate vertices v and u. Execution of line (4) is always possible by BFS since G' remains connected. The coloring-operation in line (6) can be successfully come up since every vertex to be colored has always less than $\Delta(G)$ neighbors already colored. Finally, concerning w, since it has two neighbors legally colored with 1, there exists at least one legal color for it. All the operations of algorithm 3C_COLOR can be performed in polynomial time.

A.2 Proof of lemma 3

Execution of line (3) is always possible by BFS since G is biconnected, consequently $G[V \setminus \{v\}]$ remains connected. In line (4) coloring is possible since every vertex to be colored has less than $\Delta(G)$ neighbors already colored. Finally, dealing with v, it can be legally colored since its degree is $\leqslant \Delta(G) - 1$. Obviously, all the operations of 1V_COLOR are performed in polynomial time.

A.3 Proof of theorem 2

Let us first note that articulation points and biconnected components of a graph can be found in polynomial time ([1]). Consequently, given any graph, its blocs can be identified in polynomial time (any edge not contained in a biconnected component is an isthmus).

Consider a bloc A of G. Then, following the unraveling of algorithm Δ_COLOR, the following cases may occur.

A.3.1 $\Delta(G[A]) < 3$

Let us first prove lemma 1. Starting from an arbitrary vertex, we apply a BFS on G by coloring every vertex and its neighbors with distinct colors. Since no vertex has more than $\Delta(G)$ neighbors, $1 + \Delta(G)$ colors are largely sufficient to legally color all the vertices of G. BFS-algorithm being polynomial ([1]), the whole process is polynomial.

Consequently, since $\Delta(G[A]) < 3 \Longrightarrow \Delta(G[A]) \leqslant 2$, following lemma 1, one can color A with 3 colors.

A.3.2 $|A| < \Delta(G) + 1$

Here also, using lemma 1, A can be colored with $\Delta(G)$ colors.

A.3.3 $|A| = \Delta(G) + 1$

Then, since by hypothesis $G[A] \neq K_{\Delta(G)+1}$, it certainly contains a vertex of degree $\Delta(G) - 1$. Hence, by lemma 3, A can be colored by algorithm 1V_COLOR using at most $\Delta(G)$ colors.

A.3.4 $G[A]$ is 3-connected

By lemma 2, algorithm 3C_COLOR colors A with at most $\Delta(G)$ colors.

A.3.5 $G[A]$ is biconnected

Consider an arbitrary vertex x with $d^\circ(x) \geqslant 3$. Then the following two cases can occur.

$G[A \setminus \{x\}]$ is biconnected. We prove by contradiction that there exists a vertex y such that $d(x,y) = 2$. Suppose that, $\forall y$, $d(x,y) < 2$. Then, no neighbor of x is linked to a vertex which is not neighbor of x also; consequently, $A = x \cup \Gamma(x)$, in other words $|A| = |x \cup \Gamma(x)| \leqslant \Delta(G) + 1$, a contradiction since $|A| > \Delta(G) + 1$.

$G[A \setminus \{x\}]$ is not biconnected. Consider then two extremal blocs B_1 and B_2 ($G[A \setminus \{x\}]$ being connected and containing at least one articulation point, such blocs always exist) and denote by $a_1 \in B_1$ and $a_2 \in B_2$ two articulation points of $G[A \setminus \{x\}]$. Since G is biconnected, there exist $z_1 \in B_1 \setminus \{a_1\}$ and $z_2 \in B_2 \setminus \{a_2\}$ adjacent to x. Suppose that one of z_1, z_2, say z_1, does not exist. Then, every path from x to any $v \in B_1 \setminus \{a_1\}$ would include a_1, contradicting so the biconnectivity of G. Finally, observe that, since $d^\circ(x) \geqslant 3$, $G[V \setminus \{z_1, z_2\}]$ remains connected.

A.4 Constructing the final coloring in line (29)

Let us now show that the union of the colorings obtained (applying some color-permutations if necessary) leads to a feasible coloring for G.

Observe first that any two blocs have at most only one vertex in common. Let us suppose the contrary, i.e., that there exist two blocs B and B' having two vertices, b and b', in common. Of course, $G[B \cup B']$ is not biconnected because, in the opposite case, B and B' would not be blocs (recall that blocs are maximal). Consequently, at least one of b, b' is articulation point. But if one removes b, $G[(B \cup B') \setminus \{b\}]$ remains connected because $G[B]$ and $G[B']$ remain connected and have b' in common; the same holds if we remove b', a contradiction.

On the other hand, we can always order the blocs in such a way that each bloc has at most one vertex in common with the union of the blocs preceding it in the ordering considered. In fact, one can construct a graph where a vertex represents a bloc and an edge links two vertices iff they represent two blocs having a vertex in common. Using the maximality condition in the definition of a bloc, one can easily see that the graph constructed is connected and acyclic, i.e., a tree. In this tree, we can apply a BFS to obtain the ordering claimed. Finally, when we integrate the vertices of a new bloc to the existing coloring, i.e., we increase the subgraph the vertices of which have been definitely colored, it is always possible to perform a color-permutation in the bloc in such a way that the common vertex (between the subgraph and the bloc), if any, has the same color in both the bloc and the subgraph.