



**HAL**  
open science

# An axiomatic analysis of concordance-discordance relations

Vangelis Paschos

► **To cite this version:**

| Vangelis Paschos. An axiomatic analysis of concordance-discordance relations. 2007. hal-00909467

**HAL Id: hal-00909467**

**<https://hal.science/hal-00909467>**

Preprint submitted on 26 Nov 2013

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

**CAHIER DU LAMSADE**

**270**

Octobre 2007

An overview on polynomial approximation of  
NP-hard problems

Vangelis Th. Paschos

# An overview on polynomial approximation of NP-hard problems\*

Vangelis Th. Paschos  
LAMSADE, CNRS UMR 7024 and University Paris-Dauphine  
Place du Maréchal de Lattre de Tassigny, 75775, Paris Cedex 16, France  
paschos@lamsade.dauphine.fr

October 29, 2007

## Abstract

The fact that it is very unlikely that a polynomial time algorithm could ever be devised for optimally solving **NP**-hard problems, strongly motivates both researchers and practitioners in trying to heuristically solving such problems, by making a trade-off between computational time and solution's quality. In other words, heuristic computation consists of trying to find in reasonable time, not the best solution but one solution which is "near" the optimal one. Among classes of heuristic methods for **NP**-hard problems, the *polynomial approximation algorithms* aim at solving a given **NP**-hard problem *in polynomial time* by computing *feasible solutions* that are, under some predefined criterion, *as near as possible to the optimal ones*. The polynomial approximation theory deals with the study of such algorithms. This survey presents and analyzes in a first time approximation algorithms for some classical examples of **NP**-hard problems. In a second time, it shows how classical notions and tools of complexity theory, such as polynomial reductions, can be matched with polynomial approximation in order to devise structural results for **NP**-hard optimization problems. Finally, it presents a quick description of what it is commonly called inapproximability results. Such results provide limits on the approximability of the problems tackled.

## 1 What is polynomial approximation and why we do it?

It is widely believed that no polynomial algorithm can be ever devised for optimally solving **NP**-hard problems. So, several approaches, more or less satisfactory, can be followed when one tries to solve them. Roughly these approaches belong to one of the following two families.

*Exact (optimal) algorithms*, that compute optimal solutions for the problems but they run in exponential time; such algorithms are based upon either search tree-based methods (branch-and-bound, branch-and-cut, branch-and-price, etc.), or upon dynamic programming, or ...

*Heuristic methods*, that run faster than the exact algorithms and compute sub-optimal solutions (that are, sometimes, unfeasible); the most notorious among these methods are:

- the polyhedral methods,
- the metaheuristics (simulated annealing, genetic algorithms, tabou, etc.),
- the *polynomial approximation algorithms with (a priori) performance guarantees*.

---

\*Full text of an invited talk at Balkan Conference on Operational Research, BALCOR'07, Serbia, 2007

The goal of this overview is to make a general presentation of this last category of heuristic methods, that are the polynomial approximation algorithms.

What is polynomial approximation? Roughly speaking, this is the art to achieve, in polynomial time, feasible solutions with objective value as close as possible (in some predefined sense) to the optimal value. How can we do it? Hopefully, the reader will see it in what follows.

Why do we use polynomial approximation rather than other heuristic methods? There are several reasons for that. The main such reasons are, to our opinion, both “operational” and “structural”.

A central operational reason is that there exist problems, describing natural situations, either requiring that *feasible* solutions must be obtained quickly (i.e., in polynomial time) and this fact must be guaranteed a priori, or where optimum is misdefined, or senseless. Also, it is sometimes necessary in the context these problems are handled, that a decision maker has an a priori estimation of the performance of the algorithms used to solve them.

On the other hand, the main “structural” reason is that polynomial approximation is a natural extension of complexity theory to combinatorial optimization and largely contributes to the enrichment of both domains.

The aim of the study of polynomial approximation of combinatorial optimization problems is to characterize the ability of a specific problem to be “well-solved” in polynomial time. This is done by determining both upper and lower bounds of approximability, i.e., by exhibiting specific approximation algorithms achieving a given level of approximation, on one side, and, on the other side, showing that no algorithm can possibly provide a better approximation level, until somewhat strange and unexpected happens, i.e., until a very highly improbable complexity hypothesis (e.g.,  $\mathbf{P} = \mathbf{NP}$ ) holds.

Existence of polynomial approximation as scientific area has started from the seminal paper by [46]. Since then, this research programme is one of the most active research programmes in operational research, combinatorial optimization and theoretical computer science.

## 2 Preliminaries

The object of polynomial approximation is the class of the so-called **NPO** problems. Informally, this is the class of optimization problems the “decision”-counterparts of which are in **NP**. Let us recall that, given an optimization problem, it is immediate to produce its decision version in the following way:

- if the optimization goal of the problem is “max”, then its decision version becomes: “does there exist a solution with value *greater than, or equal to, K?*”, where  $K$  is some constant that, for the decision version, is part of the instance;
- if the optimization goal of the problem is “min”, then its decision version becomes: “does there exist a solution with value *smaller than, or equal to, K?*”, where  $K$  is as in the previous item.

Solving a decision problem  $\Pi$  becomes to correctly answering by *yes* or *no* to the question defining  $\Pi$ . For example, for the decision version of MAX INDEPENDENT SET<sup>1</sup>: “given a graph  $G$  and a constant  $K$ , does there exist an independent set of  $G$  of size at least  $K$ ?”, solving it becomes to correctly answering if  $G$  has an independent set of size at least  $K$ , or not.

Formally, an **NPO** problem  $\Pi$  is a four-tuple  $(\mathcal{I}, \text{Sol}, m, \text{goal})$  such that:

- $\mathcal{I}$  is the set of instances (recognizable in polynomial time);

---

<sup>1</sup>We properly define this problem in Section 4.1.

- for  $I \in \mathcal{I}$ ,  $\text{Sol}(I)$  is the set of feasible solutions of  $I$ ; feasibility of any solution is decidable in polynomial time;
- for any  $I \in \mathcal{I}$ , at least one feasible solution can be computed in polynomial time;
- the objective value  $m(I, S)$  of any solution  $S$ , is computable in polynomial time;
- $\text{goal} \in \{\min, \max\}$ .

We now briefly present some basic notions in polynomial approximation theory. More details can be found in [5, 44, 66, 72].

Given an instance  $I$  of a combinatorial maximization (resp., minimization) problem  $\Pi = (\mathcal{I}, \text{Sol}, m, \text{goal})$ ,  $\omega(I)$ ,  $m_{\mathbf{A}}(I, S)$  and  $\text{opt}(I)$  will denote the values of the worst solution of  $I$  (in the sense of the objective function), the approximated one,  $S$  (provided by some polynomial time approximation algorithm  $\mathbf{A}$  supposed to feasibly solve problem  $\Pi$ ), and the optimal one, respectively. The worst solution of  $I$  is the optimal solution for  $I$  with respect to the **NPO** problem  $\Pi' = (\mathcal{I}, \text{Sol}, m, \text{goal}')$  where:

$$\text{goal}' = \begin{cases} \max & \text{if goal} = \min \\ \min & \text{if goal} = \max \end{cases}$$

There exist mainly two paradigms dealing with polynomial approximation.

*Standard approximation.* The quality of an approximation algorithm  $\mathbf{A}$  is expressed by the ratio  $\rho_{\mathbf{A}}(I) = m_{\mathbf{A}}(I, S)/\text{opt}(I)$ . Note that approximation ratio for minimization problems is in  $[1, \infty)$ , while for maximization ones this ratio is in  $(0, 1]$ .

*Differential approximation.* The quality of an approximation algorithm  $\mathbf{A}$  is expressed by the ratio  $\delta_{\mathbf{A}}(I) = |\omega(I) - m_{\mathbf{A}}(I, S)|/|\omega(I) - \beta(I)|$ . The value of the differential ratio is always in  $[0, 1]$ , independently on the optimization goal of the problem.

For both paradigms, the closer the ratios to 1, the better the performance of the approximation algorithm  $\mathbf{A}$ . Let us note also that, as we will see, results obtained adopting the one or the other of the two paradigms are very often different even for the same problem.

The rest of the paper is organized as follows. In Section 3 the main approximability classes are defined. In Section 4 we analyze polynomial approximation algorithms for several well-known hard optimization problems. In Section 5, we show how the main tool of complexity theory, that are polynomial reductions, can be adapted to the framework of polynomial approximation, in order to produce structural results. In Section 6 we say a few words for the limits of approximability, i.e., for inapproximability of **NP**-hard problems. Providing inapproximability results stating that a specific problem is not approximable within better than some approximation level, is crucial (although somewhat far from the classical operational researchers concerns; for this reason this part of the paper will be short) for characterizing approximability of **NP**-hard problems and for understanding structure of them. Finally, in Section 7 we present a quick overview on completeness result in approximation classes.

For shortness, only the problems discussed in details in this paper will be defined. For the other ones, the interested reader can be referred to [35]. Also, for notions and definitions from graph theory, one can be referred to [16].

### 3 Approximation classes

According to the best approximation ratios known for them, **NP**-hard problems are classified to *approximability classes*. These classes create a kind of hierarchy in the class of the **NP**-hard

problems. The most known among them (going from the pessimistic to the optimistic ones) are the following classes (for any standard-approximation class **C**, **DC** denotes the respective differential class).

**Exp-APX** and **Exp-DAPX**. Classes of problems for which the best ratio known is exponential (or the inverse of an exponential) with the size of their instance. Notorious member of **Exp-APX** is MIN TSP. On the other hand, no natural combinatorial optimization problem is still known to be in **Exp-DAPX**.

**Poly-APX** and **Poly-DAPX**. Classes of problems for which the best ratio known is polynomial (or the inverse of a polynomial) with the size of their instance. MAX INDEPENDENT SET, MAX CLIQUE, MIN COLORING, etc., belong to **Poly-APX**, while MAX INDEPENDENT SET, MAX CLIQUE, MIN VERTEX COVER, MIN SET COVER, etc., belong to **Poly-DAPX**.

**Log-APX** and **Log-DAPX**. Classes of problems for which the best ratio known is some logarithm (or the inverse of some logarithm) of the size of their instance. MIN SET COVER and MIN DOMINATING SET are the most notorious representatives of **Log-APX**. On the other hand, no natural combinatorial problem is known to be in **Log-DAPX**.

**APX** and **DAPX**. Here start the “more optimistic” approximability classes. **APX** and **DAPX** are the classes of problems approximable within ratios that are fixed constants. MIN VERTEX COVER, MIN METRIC TSP<sup>2</sup>, BIN PACKING, MAX TSP, etc., belong to **APX**, while MIN TSP, MAX TSP, MIN COLORING, etc., belong to **DAPX**.

**PTAS** and **DPTAS**. Classes of problems admitting *polynomial time approximation schemata*. A polynomial time approximation schema, is a sequence of algorithms  $A_\epsilon$  achieving ratio  $1 + \epsilon$ , for every  $\epsilon > 0$  ( $1 - \epsilon$  for maximization problems, or for the differential paradigm), in time which is polynomial with the size of the instance but exponential with  $1/\epsilon$ . MAX PLANAR INDEPENDENT SET<sup>3</sup>, MIN PLANAR VERTEX COVER<sup>4</sup>, MIN EUCLIDEAN TSP<sup>5</sup>, etc., are in **PTAS**. On the other hand, MAX INDEPENDENT SET, MIN PLANAR VERTEX COVER, BIN PACKING, etc., are known to belong to **DPTAS**.

**FPTAS** and **DFPTAS**. Classes of problems admitting *fully polynomial time approximation schemata*. A fully polynomial time approximation schema is a polynomial time approximation schema that, furthermore, is polynomial with  $1/\epsilon$  also. KNAPSACK is in both **FPTAS** and **DFPTAS**.

Let us also mention the existence of another approximability class denoted by **0-DAPX** (defined in [13]) that is meaningful only for the differential approximation paradigm. **0-DAPX** is the class of problems for which any polynomial algorithm returns a worst solution on at least one of their instances. In other words, for problems in **0-DAPX**, their differential approximation ratio is *equal* to 0. MIN INDEPENDENT DOMINATING SET is known to be in **0-DAPX**.

Finally, let us note that, by the way approximability classes are defined (i.e., as functions of the instance size), there exist indeed a continuum of such classes.

Figures 1 and 2 illustrate the approximability classes landscapes for standard and differential approximability paradigms, respectively.

---

<sup>2</sup>MIN TSP is complete graphs whose edge-weights verify the triangle-inequality.

<sup>3</sup>MAX INDEPENDENT SET in planar graphs.

<sup>4</sup>MIN VERTEX COVER is planar graphs.

<sup>5</sup>MIN METRIC TSP in (0,1)-plane.

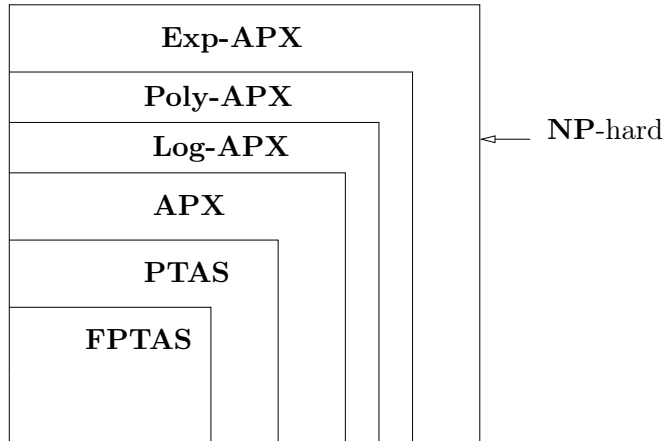


Figure 1: Standard approximability classes (under the assumption  $\mathbf{P} \neq \mathbf{NP}$ ).

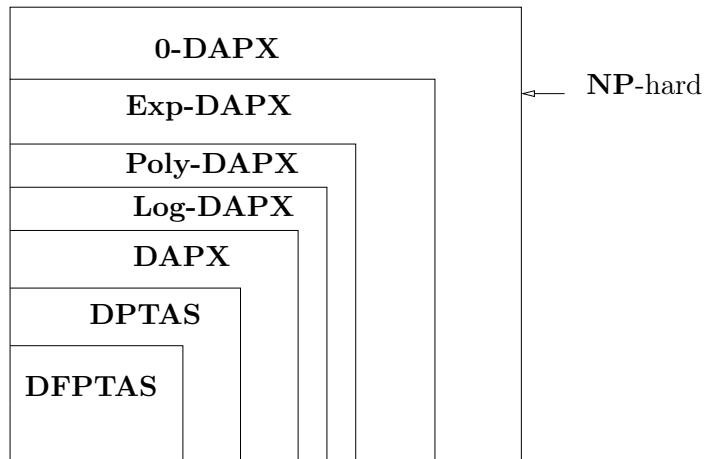


Figure 2: Differential classes (under the assumption  $\mathbf{P} \neq \mathbf{NP}$ ).

Dealing with the classes defined above, the following inclusions hold:

$$\begin{aligned}
 \mathbf{PO} &\subset \mathbf{FPTAS} \subset \mathbf{PTAS} \subset \mathbf{APX} \subset \mathbf{Log-APX} \subset \mathbf{Poly-APX} \subset \mathbf{Exp-APX} \subset \mathbf{NPO} \\
 \mathbf{PO} &\subset \mathbf{DFPTAS} \subset \mathbf{DPTAS} \subset \mathbf{DAPX} \subset \mathbf{Log-DAPX} \subset \mathbf{Poly-DAPX} \subset \mathbf{Exp-DAPX} \\
 &\subset \mathbf{0-DAPX} \subset \mathbf{NPO}
 \end{aligned}$$

These inclusions are strict unless  $\mathbf{P} = \mathbf{NP}$ .

Indeed, for any of these classes, there exist natural problems that belong to each of them but not to the immediately smaller one. For instance, for the standard paradigm:

$$\begin{aligned}
 \text{KNAPSACK} &\in \mathbf{FPTAS} \setminus \mathbf{PO} \\
 \text{MAX PLANAR INDEPENDENT SET} &\in \mathbf{PTAS} \setminus \mathbf{FPTAS} \\
 \text{MIN VERTEX COVER} &\in \mathbf{APX} \setminus \mathbf{PTAS} \\
 \text{MIN SET COVER} &\in \mathbf{Log-APX} \setminus \mathbf{APX} \\
 \text{MAX INDEPENDENT SET} &\in \mathbf{Poly-APX} \setminus \mathbf{Log-APX} \\
 \text{MIN TSP} &\in \mathbf{Exp-APX} \setminus \mathbf{Poly-APX}
 \end{aligned}$$

## 4 Approximation algorithms for several NP-hard problems

### 4.1 MAX INDEPENDENT SET

Given a graph  $G(V, E)$ , MAX INDEPENDENT SET consists of determining a maximum-size set  $V' \subseteq V$  such that, for every  $(u, v) \in V' \times V'$ ,  $(u, v) \notin E$ .

Let us first consider the integer-linear formulation of MAX INDEPENDENT SET and its linear relaxation, denoted by MAX INDEPENDENT SET-R, where given a graph  $G(V, E)$ ,  $A$  denotes its incidence matrix:

$$\text{MAX INDEPENDENT SET} = \begin{cases} \max & \vec{1} \cdot \vec{x} \\ & A\vec{x} \leq \vec{1} \\ & \vec{x} \in \{0, 1\}^n \end{cases} \quad (1)$$

$$\text{MAX INDEPENDENT SET-R} = \begin{cases} \max & \vec{1} \cdot \vec{x} \\ & A\vec{x} \leq \vec{1} \\ & \vec{x} \in (\mathbb{Q}^n)^+ \end{cases} \quad (2)$$

The following seminal theorem, due to [57] gives a very interesting characterization for the basic optimal solution of (2).

**Theorem 1.** ([57]) *The basic optimal solution of MAX INDEPENDENT SET-R is semi-integral, i.e., it assigns to the variables values from  $\{0, 1, 1/2\}$ . If  $V_0$ ,  $V_1$  and  $V_{1/2}$  are the subsets of  $V$  associated with 0, 1 et  $1/2$ , respectively, then there exists a maximum independent set  $S^*$  such that:*

1.  $V_1 \subseteq S^*$
2.  $V_0 \subseteq V \setminus S^*$

A basic corollary of Theorem 1 is that in order to solve MAX INDEPENDENT SET, one can first solve its linear relaxation MAX INDEPENDENT SET-R (this can be done in polynomial time [3, 37, 48, 69]) and store  $V_1$  and then solve MAX INDEPENDENT SET in some way in  $G[V_{1/2}]$ , i.e., the subgraph of  $G$  induced by  $V_{1/2}$ .

Indeed, solution of MAX INDEPENDENT SET-R provides sets  $V_0$ ,  $V_1$  and  $V_{1/2}$  that form a partition of  $V$ . Furthermore, by the constraint set of this program, edges can exist into  $V_0$  and  $V_{1/2}$  and between  $V_1$  and  $V_0$  and  $V_0$  and  $V_{1/2}$ , but not between  $V_1$  and  $V_{1/2}$  (see also Figure 3 where thick lines indicate the possible existence of edges between vertex-sets). So, the union of  $V_1$  (that is an independent set per se) and of an independent set of  $G[V_{1/2}]$  is an independent set for the whole of  $G$ .

Consider now the following algorithm, due to [43], denoted by IS:

1. solve MAX INDEPENDENT SET-R in order to determine  $V_0$ ,  $V_1$  et  $V_{1/2}$ ;
2. color  $G[V_{1/2}]$  with at most  $\Delta(G[V_{1/2}])$  colors (where, for a graph  $G$ ,  $\Delta(G)$  denotes its maximum degree) using the algorithm by [53]; let  $\hat{S}$  be the largest color
3. output  $S = V_1 \cup \hat{S}$

**Theorem 2.** *Algorithm IS achieves approximation ratio  $2/\Delta(G)$  for MAX INDEPENDENT SET.*

**Proof.** Let us first recall, that the vertices of a graph  $G$  can be feasibly colored<sup>6</sup> with at most  $\Delta(G)$  colors in polynomial time ([53]). This result, is, in fact, the constructive proof of an existential theorem about such coloring originally stated by [18].

---

<sup>6</sup> Given a graph  $G(V, E)$ , a coloring of  $V$  consists of coloring the vertices of  $V$  in such a way that no two adjacent vertices receive the same color; in other words, a color has to be an independent set and a coloring of  $V$  is, indeed, a partition of  $V$  into independent sets.



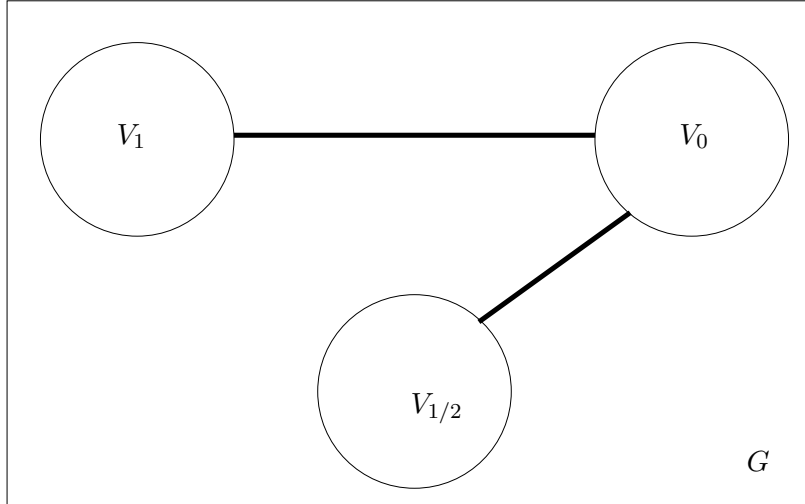


Figure 3: A graph  $G$ , input of MAX INDEPENDENT SET, and the possibility of existence of edges between sets  $V_0$ ,  $V_1$  and  $V_{1/2}$  computed by the solution of MAX INDEPENDENT SET-R.

Fix a maximum independent set  $S^*$  of  $G$  that contains  $V_1$  (by item 1 in Theorem 1 this is always possible). Since  $\hat{S}$  is the largest of the at most  $\Delta(G[V_{1/2}])$  colors (independent sets) produced in step 2 of Algorithm IS, its size satisfies:

$$|\hat{S}| \geq \frac{|V_{1/2}|}{\Delta(G[V_{1/2}])} \quad (3)$$

The size of  $S$  returned by the algorithm at step 3 is given by:

$$m(S, G) = |S| = |V_1| + |\hat{S}| \stackrel{(3)}{\geq} |V_1| + \frac{|V_{1/2}|}{\Delta(G[V_{1/2}])} \geq |V_1| + \frac{|V_{1/2}|}{\Delta(G)} \quad (4)$$

Finally, denote by  $S_{1/2}^*$  a maximum independent set in  $G[V_{1/2}]$ . Observe that the value of the optimal solution for MAX INDEPENDENT SET-R in  $G[V_{1/2}]$  is equal to  $|V_{1/2}|/2$ . Since MAX INDEPENDENT SET is a maximization problem, the objective value of (integer) MAX INDEPENDENT SET is bounded above by the objective value of (continuous) MAX INDEPENDENT SET-R. Hence:

$$|S_{1/2}^*| \leq \frac{|V_{1/2}|}{2} \quad (5)$$

Denote by  $\alpha(G)$ , the size of  $S^*$ . Then, obviously, the following holds:

$$\text{opt}(G) = |S^*| = |V_1| + |S_{1/2}^*| \stackrel{(5)}{\leq} |V_1| + \frac{|V_{1/2}|}{2} \quad (6)$$

Putting together (4) and (6) we get:

$$\frac{m(S, G)}{\text{opt}(G)} \geq \frac{2}{\Delta(G)}$$

that completes the proof. ■

A slight improvement of the ratio claimed in Theorem 2 appears in [63].

An immediate corollary of Theorem 2 is that MAX INDEPENDENT SET belongs to **Poly-APX**. Also, since the value  $\omega(G)$  of a worst solution of the problem is 0 (i.e., we can consider the empty vertex-set as a feasible MAX INDEPENDENT SET-solution), standard- and differential-approximation ratios coincide. So, MAX INDEPENDENT SET belongs to **Poly-DAPX** also. Finally, let us note that the strongest approximation results known for this problem are the following:

- MAX INDEPENDENT SET is asymptotically approximable (i.e., for large  $\Delta(G)$  within ratio  $k/\Delta(G)$ , for every fixed constant  $k$  ([28, 64]);
- MAX INDEPENDENT SET is approximable within ratio  $O(\log^2 n/n)$  ([39]);
- MAX INDEPENDENT SET is approximable within  $O(\log n/\Delta(G) \log \log n)$  ([29]);
- MAX INDEPENDENT SET is inapproximable within better than  $O(n^{\epsilon-1})$  for any  $\epsilon > 0$ , unless  $\mathbf{P} = \mathbf{NP}$  ([42]).

## 4.2 MIN SET COVER

Given a ground set  $C = \{c_1, \dots, c_n\}$  and a collection  $\mathcal{S} = \{S_1, \dots, S_m\} \subset 2^C$  of subsets of  $C$ , MIN SET COVER consists of determining a minimum-size sub-collection  $\mathcal{S}' \subseteq \mathcal{S}$  that covers  $C$ , i.e., such that  $\cup_{S \in \mathcal{S}'} S = C$ .

Let us consider the following natural greedy algorithm for MIN SET COVER denoted by **GREEDYSC**:

1. set:  $\mathcal{S}' = \mathcal{S}' \cup \{S\}$ , where  $S \in \operatorname{argmax}_{S_i \in \mathcal{S}} \{|S_i|\}$  ( $\mathcal{S}'$  is assumed to be empty at the beginning of the algorithm);
2. update  $I(\mathcal{S}, C)$  by setting:  $\mathcal{S} = \mathcal{S} \setminus \{S\}$ ,  $C = C \setminus S$  and, for  $S_j \in \mathcal{S}$ ,  $S_j = S_j \setminus S$ ;
3. repeat steps 2 and 2 until  $C = \emptyset$ ;
4. output  $\mathcal{S}'$ .

This algorithm has been independently analyzed by [46, 52]. Its version for weighted MIN SET COVER where we ask for determining a set cover of minimum total weight has been analyzed by [20]. It is easy to see that Algorithm **GREEDYSC** runs in polynomial time.

**Theorem 3.** *The standard-approximation ratio of Algorithm **GREEDYSC** is bounded above by  $1 + \ln \Delta$ , where  $\Delta = \max_{S \in \mathcal{S}} \{|S|\}$ .*

**Proof.** Denote by  $I_i(\mathcal{S}_i, C_i)$  the surviving instance at the first moment residual cardinalities of sets in  $\mathcal{S}$  are at most  $i$ ; denote by  $m(I_i, \mathcal{S}')$  the number of sets of residual cardinality  $i$  placed in  $\mathcal{S}'$  and note that  $m(I_\Delta, \mathcal{S}') = m(I, \mathcal{S}')$ . Following these notations:

$$C_\Delta = C \tag{7}$$

$$m(I, \mathcal{S}') = \sum_{i=1}^{\Delta} m(I_i, \mathcal{S}') \tag{8}$$

For  $i = 1, \dots, \Delta$ , we have the following  $\Delta$ -line equation-system:

$$|C_i| = \sum_{k=1}^i k \times m(I_k, \mathcal{S}') \tag{9}$$

where any of the above equations expresses the facts that:

- anytime a set  $S$  is chosen to be part of  $\mathcal{S}'$ , Algorithm **GREEDYSC** removes from  $C$  the elements of  $S$ ;
- for any  $i$  the remaining ground set  $C_i$  to be covered, is covered by sets of cardinality at most equal to  $i$  chosen later by the algorithm.

Multiplying the  $\Delta$ th line of (9) by  $1/\Delta$  and, for the other lines, line  $i$  by  $1/(i(i+1))$  and taking into account (7), (8) and the fact that:

$$\frac{1}{i(i+1)} = \frac{1}{i} - \frac{1}{i+1}$$

we finally obtain:

$$\left( \sum_{i=1}^{\Delta-1} \frac{|C_i|}{i(i+1)} \right) + \frac{|C|}{\Delta} = \sum_{i=1}^{\Delta} m(I_i, \mathcal{S}') = m(I, \mathcal{S}') \quad (10)$$

Consider now an optimal solution  $\mathcal{S}^*$  of  $I(\mathcal{S}, C)$  and let  $\mathcal{S}_i^*$  be an optimal solution for  $I_i(\mathcal{S}_i, C_i)$ ,  $i = 1, \dots, \Delta$ . Elements of  $\mathcal{S}^*$  covering  $C_i$  are always present and form a feasible solution for  $I_i$ . Therefore:

$$\text{opt}(I_i) = |\mathcal{S}_i^*| \leq |\mathcal{S}^*| = \text{opt}(I) \quad (11)$$

$$|C_i| \leq i \times \text{opt}(I_i) \quad (12)$$

where (12) expresses the fact that in order to cover  $|C_i|$  elements by sets covering at most  $i$  of them, at least  $|C_i|/i$  such sets will be needed. Putting (10), (11) and (12) together, we get:

$$m(I, \mathcal{S}') \leq \text{opt}(I) \times \sum_{i=1}^{\Delta} \frac{1}{i} \leq \text{opt}(I)(1 + \ln \Delta)$$

that is the ratio claimed. ■

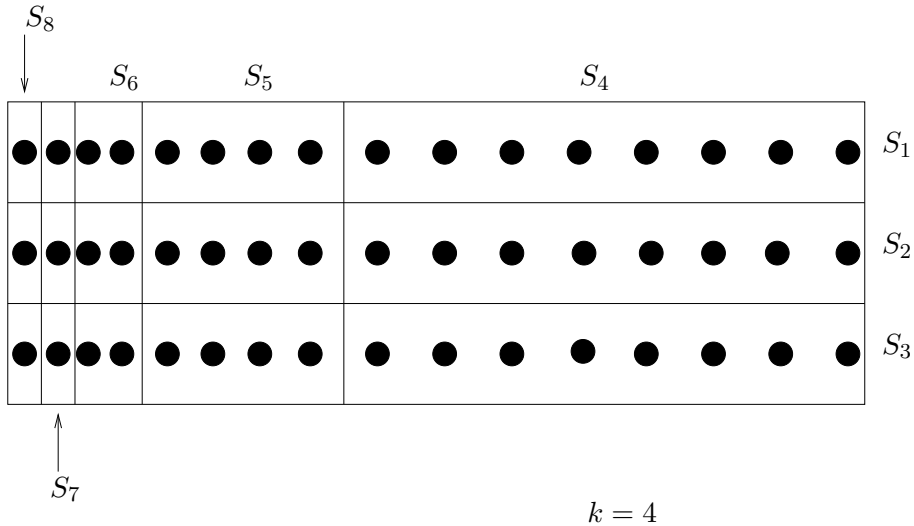


Figure 4: On the tightness of **GREEDYSC** with  $k = 4$ .

Let us now show that ratio proved in Theorem 3 is asymptotically tight. Fix some  $k$  and consider the instance of Figure 4, designed for  $k = 4$ . We are given a set  $C$  of  $3 \times 2^k$  elements:  $C = \{1, 2, \dots, 3 \times 2^k\}$  and a family  $\mathcal{S} = \{S_1, S_2, \dots, S_{k+4}\}$  of  $k + 4$  subsets of  $C$ . For simplicity, consider that elements of  $C$  are entries of a  $3 \times (k + 1)$  matrix. Then, sets in  $\mathcal{S}$  are as follows:

- three disjoint sets:  $S_1, S_2$  et  $S_3$ , each one containing the  $2^k$  elements of each of the three lines of the matrix; they form a partition on  $C$ , i.e.,  $C = S_1 \cup S_2 \cup S_3$  et  $S_i \cap S_j = \emptyset$ ,  $i, j = 1, 2, 3$  ;
- $k + 1$  sets  $S_4, \dots, S_{k+4}$ , of respective sizes:  $3 \times 2^{k-1}, 3 \times 2^{k-2}, \dots, 3 \times 2^1, 3 \times 2^0$  and 3 contain, respectively, the points of the  $2^{k-1}, 2^{k-2}, \dots, 2^1, 2^0$  and 1 columns.

It is easy to see that Algorithm **GREEDYSC** will choose the  $k + 1$  sets  $S_4, \dots, S_{k+4}$ , in this order, returning a set cover of size  $k + 1$ , while the optimal set cover is the family  $\{S_1, S_2, S_3\}$  of size 3. Note finally that, for the instance considered,  $\Delta = 3 \times 2^{k-1}$ . Consequently, approximation ratio achieved here is  $(k + 1)/3 = O(\ln(3 \times 2^{k-1}))$ .

More recently, another analysis of Algorithm **GREEDYSC** has been presented by [71] providing a tight ratio of  $O(\log |C|)$ . On the other hand, as stated in [67] (see [31] for an informal proof), it is impossible to approximate **MIN SET COVER** within better than  $O(\log |C|) - O(\log \log |C|)$ , unless a highly unlikely complexity classes relationship is true. Note finally that the result of Theorem 3 has been slightly improved down to  $\ln \Delta + (5/6)$  by [36].

A direct corollary of Theorem 3 (or alternatively of [71]) and of [67] is that **MIN SET COVER** definitely belongs to **Log-APX**.

Dealing with the differential approximation paradigm, **MIN SET COVER** is approximable within differential ratio bounded below by  $1.365/\Delta$  ([12]) and inapproximable as is ([27]). So, it belongs to **Poly-DAPX**.

### 4.3 MIN VERTEX COVER

Given a graph  $G(V, E)$ , **MIN VERTEX COVER** consists of determining a minimum-size set  $V' \subseteq V$  such that, for every  $(u, v) \in E$ , at least one among  $u$  and  $v$  belongs to  $V'$ .

Consider the following algorithm, denoted by **MATCHING**:

1. compute a maximal matching<sup>7</sup>  $M$  in  $G$ ;
2. return the set  $C$  of the endpoints of the edges in  $M$ .

Algorithm **MATCHING** is polynomial since a maximal matching can be easily computed by picking an edge  $e$ , deleting it from  $G$  together with any edge sharing an endpoint with  $e$  and iterating these operations until no edge survives in  $G$ .

**Theorem 4.** *Algorithm **MATCHING** is 2-standard-approximation algorithm for **MIN VERTEX COVER**.*

**Proof.** Let us first prove that  $C$  is a vertex cover for  $G$ . The endpoints of any edge  $e$  of  $M$  cover  $e$  itself and any other edge sharing a common endpoint with  $e$ . Denote by  $V(M)$  the set of endpoints of the edges of  $M$ . Since  $M$  has been built to be maximal, the edges in  $M$  have common endpoints with any edge in  $E \setminus M$ , so,  $V(M)$  cover both  $M$  and  $E \setminus M$ , i.e., the whole of  $E$ .

Set  $m = |M|$ ; then:

$$|V(M)| = m(G, C) = 2m \tag{13}$$

On the other hand, since edges in  $M$  do not pairwise share common endpoints, any solution (a fortiori an optimal one) must use at least  $m$  vertices in order to cover them (one vertex per edge of  $M$ ). So, denoting by  $\tau(G)$  the cardinality of a minimum vertex cover in  $G$ , we have:

$$\tau(G) = \text{opt}(G) \geq m \tag{14}$$

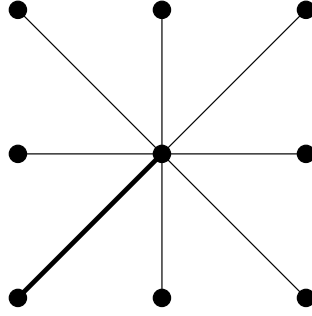


Figure 5: Tightness of the ratio achieved by Algorithm `MATCHING`.

Putting (13) and (14) together, we immediately get:  $m(G, C)/\tau(G) \leq 2$  as claimed. ■

We now show that the ration claimed in Theorem 4 is tight. Consider the graph of Figure 5. Thick edge is a maximal matching for this graph and, consequently, the solution taken by Algorithm `MATCHING` will contain two vertices (the two endpoints of the thick edge). On the other hand, the center of the star suffices to cover all its edges.

One of the most known open problems in polynomial approximation is the improvement of ratio 2 for `MIN VERTEX COVER`. A lot of unsuccessful effort has been provided for such improvement until now. All this long effort has only produced ratios of the form:

- $2 - (\log \log n / \log n)$  ([10, 54]);
- $2 - (2 \ln \ln n / \ln n)$  ([40]);
- $2 - (\log \log \Delta(G) / \log \Delta(G))$  ([40]).

Unfortunately, a more recent result by [50] gives strong evidence that this improvement should be impossible.

In the differential approximation paradigm, `MIN VERTEX COVER` is equiapproximable with `MAX INDEPENDENT SET` ([27]). So, it definitely belongs to **Poly-DAPX**.

#### 4.4 MIN TSP

Given a complete graph on  $n$  vertices, denoted by  $K_n$ , with positive weights on its edges, `MIN TSP` consists of determining a Hamiltonian tour<sup>8</sup> of  $K_n$  of minimum total cost.

Let us note that, with respect to the differential paradigm, computing a worst solution for `MIN TSP` is not trivial at all. On the opposite of the problems seen until now in the previous sections that had “trivial” worst solutions (the empty set for `MAX INDEPENDENT SET`, the whole family  $\mathcal{S}$  for `MIN SET COVER`, or the whole vertex-set  $V$  of the input-graph for `MIN VERTEX COVER`) worst solution of `MIN TSP` is an optimal solution of `MAX TSP`, where we wish to determine a maximum total-cost Hamiltonian cycle. This problem is also **NP-hard**. So, determining a worst solution of `MIN TSP` is as hard as to determine an optimal solution.

Consider the following very well-known algorithm for `MIN TSP`, denoted by `2_OPT`, originally devised by [25], where  $d(i, j)$  denotes the weight of edge  $(v_i, v_j)$ :

1. construct some Hamiltonian tour  $T$  (this can be done, for example, by the nearest-neighbor heuristic);

---

<sup>7</sup>Given a graph  $G(V, E)$ , a matching is a subset  $M \subseteq E$  such that no two edges in  $M$  share a common endpoint; a matching is maximal (for inclusion) if it cannot be augmented remaining a matching.

<sup>8</sup> A simple cycle passing through all the vertices of  $K_n$ .

2\_OPT

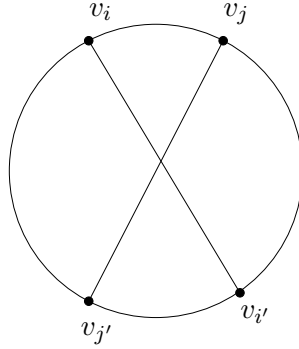


Figure 6: Algorithm 2\_OPT

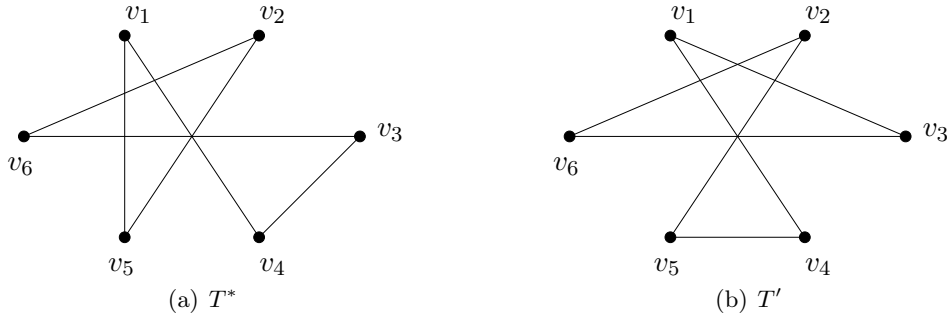


Figure 7: Tours  $T^*$  and  $T'$  of the proof of Theorem 5 for a  $K_6$ .

2. consider two edges  $(v_i, v_j)$  and  $(v_{i'}, v_{j'})$  of  $T$ ; if  $d(i, j) + d(i', j') > d(i, i') + d(j, j')$ , then replace  $(v_i, v_j)$  and  $(v_{i'}, v_{j'})$  in  $T$  by  $(v_i, v_{i'})$  and  $(v_j, v_{j'})$  (Figure 6) i.e., produce a new Hamiltonian tour  $T \setminus \{(v_i, v_j), (v_{i'}, v_{j'})\} \cup \{(v_i, v_{i'}), (v_j, v_{j'})\}$ ;
3. repeat step 2 until no swap is possible;
4. output  $T$  the finally produced tour.

Algorithm 2\_OPT is polynomial when, for instance,  $d_{\max}$  the maximum edge-weight is bounded above by a polynomial of  $n$ . For other cases where 2\_OPT is polynomial, see [56].

**Theorem 5.** ([56]) *Algorithm 2\_OPT achieves differential-approximation ratio bounded below by  $1/2$ .*

**Proof.** Assume that  $T$  is represented as the set of its edges, i.e.:

$$T = \{(v_1, v_2), \dots, (v_i, v_{i+1}), \dots, (v_n, v_1)\}$$

and denote by  $T^*$  an optimal tour. Let  $s^*(i)$  be the index of the successor of  $v_i$  in  $T^*$ . So,  $s^*(i) + 1$  is the index of the successor of  $v_{s^*(i)}$  in  $T$  (mod  $n$ ) (in other words, if  $s^*(i) = j$ , then  $s^*(i) + 1 = j + 1$ ).

The tour  $T$  computed by 2\_OPT is a local optimum for the 2-exchange of edges in the sense that every interchange between two non-intersecting edges of  $T$  and two non-intersecting edges

of  $E \setminus T$  will produce a tour of total distance at least equal to  $d(T)$ , where  $d(T)$  denotes the total weight of  $T$ . This implies in particular that,  $\forall i \in \{1, \dots, n\}$ :

$$d(i, i+1) + d(s^*(i), s^*(i)+1) \leq d(i, s^*(i)) + d(i+1, s^*(i)+1) \quad (15)$$

Observe now that (see also Figure 7), denoting by  $T_\omega$  a worst tour in  $K_n$  the following hold:

$$\bigcup_{i=1, \dots, n} \{(v_i, v_{i+1})\} = \bigcup_{i=1, \dots, n} \{(v_{s^*(i)}, v_{s^*(i)+1})\} = T \quad (16)$$

$$\bigcup_{i=1, \dots, n} \{(v_i, v_{s^*(i)})\} = T^* \quad (17)$$

$$\bigcup_{i=1, \dots, n} \{(v_{i+1}, v_{s^*(i)+1})\} = \text{some feasible tour } T' \text{ better than } T_\omega \quad (18)$$

Add inequalities in (15), for  $i = 1, \dots, n$ :

$$\sum_{i=1}^n (d(i, i+1) + d(s^*(i), s^*(i)+1)) \leq \sum_{i=1}^n (d(i, s^*(i)) + d(i+1, s^*(i)+1)) \quad (19)$$

Putting (19) together with (16), (17) and (18) we get:

$$(16) \implies \sum_{i=1}^n d(i, i+1) + \sum_{i=1}^n d[s^*(i), s^*(i)+1] = 2m(K_n, T)$$

$$(17) \implies \sum_{i=1}^n d[i, s^*(i)] = \text{opt}(K_n)$$

$$(18) \implies \sum_{i=1}^n d[i+1, s^*(i)+1] = d(T') \leq \omega(K_n)$$

Then, some easy algebra, and by taking into account that the differential ratio for a minimization problem increases with  $\omega$ , leads to:

$$\frac{\omega(K_n) - m(K_n)}{\omega(K_n) - \text{opt}(K_n)} > \frac{1}{2}$$

that completes the proof of the theorem. ■

We now show that ratio  $1/2$  is tight for Algorithm 2\_OPT. Consider a  $K_{2n+8}$ ,  $n \geq 0$ , set  $V = \{i : i = 1, \dots, 2n+8\}$ , set:

$$\begin{aligned} d(2k+1, 2k+2) &= 1 & k = 0, 1, \dots, n+3 \\ d(2k+1, 2k+4) &= 1 & k = 0, 1, \dots, n+2 \\ d(2n+7, 2) &= 1 \end{aligned}$$

and set the distances of all the remaining edges to 2.

Consider the tour  $T = \{(i, (i+1)) : i = 1, \dots, 2n+7\} \cup \{(2n+8), 1\}$  and observe that it is a local optimum for the 2-exchange on  $K_{2n+8}$ . Indeed, let  $(i, (i+1))$  and  $(j, (j+1))$  be two edges of  $T$ . We can assume w.l.o.g. that  $2 = d(i, i+1) \geq d(j, j+1)$ , otherwise, the cost of  $T$  cannot be improved. Therefore,  $i = 2k$  for some  $k$ . In fact, in order that the cost of  $T$  is improved, there exist two possible configurations, namely  $d(j, j+1) = 2$  and  $d(i, j) = d(j, j+1) = d(i+1, j+1) = 1$ . Then, the following assertions hold:

- if  $d(j, j + 1) = 2$ , then  $j = 2k'$ , for some  $k'$ , and, according to the construction of  $K_{2n+8}$ ,  $d(i, j) = 2$  (since  $i$  and  $j$  are even), and  $d(i + 1, j + 1) = 2$  (since  $i + 1$  and  $j + 1$  are odd); so the 2-exchange does not yield a better solution;
- if  $d(i, j) = d(j, j + 1) = d(i + 1, j + 1) = 1$ , then according to the construction of  $K_{2n+8}$  we will have  $j = 2k' + 1$  and  $k' = k + 1$ ; so, we lead a contradiction since  $1 = d(i + 1, j + 1) = 2$ .

Furthermore, one can easily see that the tour:

$$T^* = \{(2k + 1)(2k + 2) : k = 0, \dots, n + 3\} \cup \{(2k + 1)(2k + 4) : k = 0, \dots, n + 2\} \cup \{(2n + 7)2\}$$

is an optimal tour of value  $\beta(K_{2n+8}) = 2n + 8$  (all its edges have distance 1) and that the tour:

$$T_\omega = \{(2k + 2)(2k + 3) : k = 0, \dots, n + 2\} \cup \{(2k + 2)(2k + 5) : k = 0, \dots, n + 1\} \\ \cup \{(2n + 8)1, (2n + 6)1, (2n + 8)3\}$$

realizes a worst solution for  $K_{2n+8}$  with value  $\omega(K_{2n+8}) = 4n + 16$  (all its edges have distance 2).

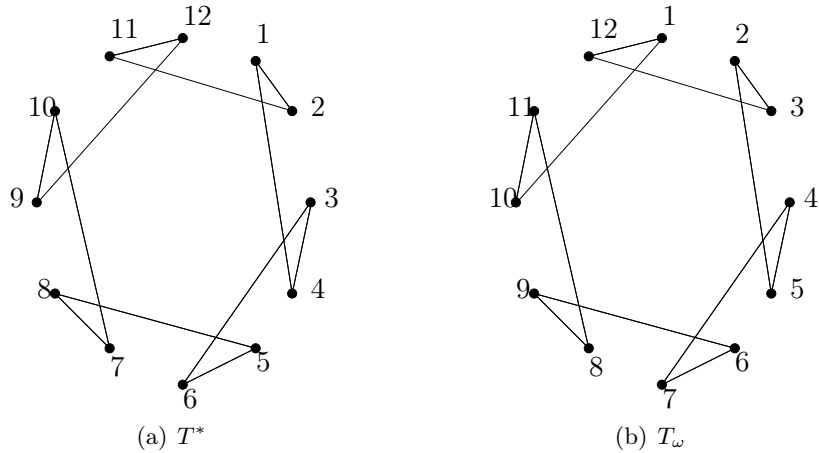


Figure 8: Tightness of the 2\_OPT approximation ratio for  $n = 1$ .

Consider a  $K_{12}$  constructed as described just above (for  $n = 2$ ). Here,  $d(1, 2) = d(3, 4) = d(5, 6) = d(7, 8) = d(9, 10) = d(11, 12) = d(1, 4) = d(6, 3) = d(5, 8) = d(7, 10) = d(9, 12) = d(11, 2) = 1$ , while all the other edges are of distance 2. In Figures 8(a) and 8(b),  $T^*$  and  $T_\omega$ , respectively, are shown ( $T = \{1, \dots, 11, 12, 1\}$ ). Hence, in  $K_{2n+8}$  considered, the differential-approximation ratio of 2\_OPT is equal to  $1/2$ .

The best differential-approximation ratio for MIN TSP is  $3/4$  ([30]) but it does not admit a polynomial time differential-approximation schema ([56, 55]). So, MIN TSP definitely belongs to **DAPX**. Furthermore, it is proved in [56] that MIN TSP and MIN METRIC TSP are equiapproximable for the differential approximation.

Dealing with the standard approximation paradigm MIN TSP is in **Exp-APX** (see Section 6). On the other hand, MIN METRIC TSP is approximable within standard-approximation ratio  $3/2$  by the celebrated Christofides algorithm ([19]) while the most famous relaxation of MIN METRIC TSP, that is when edge-weights are either 1 or 2 is approximable within  $8/7$  ([17]). Finally, MIN EUCLIDEAN TSP is in **PTAS** ([1]).

#### 4.5 MIN COLORING

Given a graph  $G(V, E)$ , MIN COLORING consists of determining the minimum number of colors (i.e., of independent sets, see also footnote 6), that feasibly color the vertices of  $G$ .



The worst-solution value for MIN COLORING is equal to  $n$ , since coloring any vertex of the input graph with its own color produces a feasible coloring. Furthermore, this coloring cannot be augmented without producing empty colors.

Consider the following algorithm for MIN COLORING, denoted by COLOR and devised by [41] (see also [65]):

1. find an independent set  $S$  of size 3 in  $G$ ; color its vertices with a new color and remove it from  $G$ ;
2. repeat step 1 until no independent set of size 3 is found;
3. determine a maximum family of disjoint independent sets of size 2 in (the surviving) graph  $G$  and color the vertices of each of them with a new color;
4. color the remaining vertices of  $G$  using as new colors as the vertices to be colored;
5. output  $C$  the union of colors used at steps 2, 3 and 4.

Observe first that Algorithm COLOR runs in polynomial time. Indeed, step 1 is greedy. For a graph of order<sup>9</sup>  $n$ , all the independent sets of size 3 can be found in time  $O(n^3)$  by exhaustive search. Step 2 can be performed in polynomial time since it amounts to a maximum matching computation that is polynomial ([61]). Indeed, at step 3, the maximum independent set of the surviving graph  $G$  has size at most 2. Consider  $\bar{G}$  that is the complementary<sup>10</sup> of  $G$ . Any independent set of size 2 in  $G$  becomes an edge in  $\bar{G}$  and maximum family of disjoint independent sets of size 2 in  $G$  is exactly a maximum matching in  $\bar{G}$ . So, computation in step 3 is nothing else than computation of a maximum matching.

**Lemma 1.** *Steps 3 and 4 of Algorithm COLOR optimally color a graph  $G$  with  $\alpha(G) = 2$ .*

**Proof.** Since  $\alpha(G) = 2$ , colors in  $G$  are either independent sets of size 2, or singletons. Fix some coloring  $C$  using  $x$  colors of size 2 and  $y$  colors that are single vertices. If  $n$  is the order of  $G$ , we have:

$$|C| = x + y \tag{20}$$

$$n = 2x + y \tag{21}$$

By (20) and (21),  $|C| = n - x$ . Hence, the greater  $x$ , the better the coloring  $C$  and a minimum coloring corresponds to a maximum  $x$ . This is exactly what Step 3 of Algorithm COLOR does. ■

We are ready now to prove the following theorem.

**Theorem 6.** *([41]) Algorithm COLOR is 2/3-differential-approximation algorithm for MIN COLORING.*

**Proof.** We prove the theorem by induction on  $n$  the size of the input graph.

If  $n = 1$  then Algorithm COLOR optimally colors it with one color. Assume that theorem's statement remains true for  $n \leq k$  and consider a graph  $G$  of order  $n = k + 1$ . We distinguish two cases.

If  $G$  does not contain an independent set of size greater than 2, then by Lemma 1, Algorithm COLOR computes an optimal coloring for  $G$ .

---

<sup>9</sup>The order of a graph is the cardinality  $|V|$  of its vertices.

<sup>10</sup>Given a graph  $G(V, E)$ , the complementary graph  $\bar{G}(V, \bar{E})$  of  $G$  is the graph having the same set of vertices  $V$  as  $G$  and  $\bar{E} = \{(v_i, v_j) : i \neq j \text{ and } (v_i, v_j) \notin E\}$ .

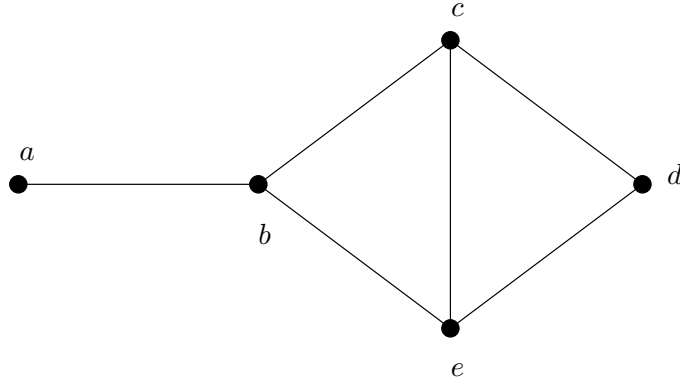


Figure 9: The complement of a graph where differential ratio  $2/3$  is attained by Algorithm **COLOR**.

Assume now that  $G$  is such that  $\alpha(G) \geq 3$  and denote by  $\chi(G)$  the chromatic number (i.e., the cardinality of a minimum coloring) of  $G$ . Then, obviously, at least an independent set  $S$  has been found by **COLOR** at step 1 and:

$$\chi(G[V \setminus S]) \leq \chi(G) \quad (22)$$

Consider the graph  $G[V \setminus S]$  of order  $n - 3$  and its coloring  $C \setminus S$ . This graph is colored with  $|C| - 1$  colors and, by the induction hypothesis:

$$n - 3 - |C \setminus S| \geq \frac{2}{3}(n - 3 - \chi(G[V \setminus S])) \quad (23)$$

Combining (22) and (23) we get:

$$n - |C| = n - |C \setminus S| - 1 \geq \frac{2}{3}(n - 3 - \chi(G[V \setminus S])) + 2 \geq \frac{2}{3}(n - \chi(G)) \quad (24)$$

Taking into account that  $\omega(G) = n$ , (24) directly derives the differential ratio claimed. ■

We now prove that differential ratio  $2/3$  is tight for Algorithm **COLOR**. Consider a graph  $G$  the complement of which is shown in Figure 9. It is easy to see that in  $G$ , Algorithm **COLOR** would produce  $C = \{\{a\}, \{d\}, \{b, c, e\}\}$ , while the optimal coloring is  $C^* = \{\{a, b\}, \{c, d, e\}\}$ . Taking into account that  $\omega(G) = 5$ , ratio  $2/3$  for this instance is immediately proved.

Dealing with standard paradigm, the best known approximation ratios for **MIN COLORING** are:

- $O(n(\log \log n)^2 / \log^3 n)$  ([38]);
- $\Delta \log \log n / \log \Delta$  ([29]).

On the other hand, it is inapproximable within better than:

- $n^{1-\epsilon}$  for any constant  $\epsilon > 0$ , unless  $\mathbf{NP} \subseteq \mathbf{coRP}$  ([34]);
- $n^{(1/5)-\epsilon}$  for any  $\epsilon > 0$ , assuming that  $\mathbf{NP} \neq \mathbf{coRP}$  ([15]);
- $n^{(1/7)-\epsilon}$  for any  $\epsilon > 0$  unless  $\mathbf{P} \neq \mathbf{NP}$  ([15]).

More about the complexity classes mentioned in the results above can be found in [59].

#### 4.6 MIN WEIGHTED BIPARTITE COLORING

Consider a vertex-weighted graph  $G(V, E)$  and denote by  $w_i$  the weight of vertex  $v_i \in V$ . For any subset  $V' \subseteq V$  define its weight  $w(V')$  by:

$$w(V') = \max \{w_i : v_i \in V'\} \quad (25)$$

MIN WEIGHTED COLORING consists of determining a coloring  $C = (S_1, \dots, S_k)$  of  $G$  minimizing the quantity:

$$m(G, C) = \sum_{i=1}^k w(S_i) \quad (26)$$

where, for  $i = 1, \dots, k$ ,  $w(S_i)$  is defined as in (25).

MIN WEIGHTED COLORING is obviously **NP**-hard in general graphs since setting  $w_i = 1$ ,  $v_i \in V$ , it becomes the classical MIN COLORING problem. However, it is proved in [26] that it is **NP**-hard, even in bipartite graphs (MIN WEIGHTED BIPARTITE COLORING). Let us note that MIN COLORING is polynomial in these graphs since they are 2-colorable.

In what follows in this section, we present a polynomial time differential-approximation schema originally developed by [26].

Consider a vertex-weighted bipartite graph  $B(U, D, E)$  and the following algorithm denoted by BIC:

1. range the vertices of  $B$  in decreasing order with respect to their weights;
2. fix an  $\epsilon > 0$  and set  $\eta = \lceil 1/\epsilon \rceil$ ; set  $S_U = \{v_{4\eta+3}, \dots, v_n\} \cap U$  and  $S_D = \{v_{4\eta+3}, \dots, v_n\} \cap D$ ;
3. compute an optimal weighted coloring  $\tilde{C}$  in  $B' = B[\{v_1, \dots, v_{4\eta+2}\}]$ ;
4. output  $C = S_U \cup S_D \cup \tilde{C}$ .

Since the graph  $B'$  of step 3 has fixed size, computation of  $\tilde{C}$  can be performed in polynomial time by an exhaustive search. So, Algorithm BIC is polynomial.

Denote by  $C^* = (S_1^*, S_2^*, \dots, S_p^*)$  an optimal MIN WEIGHTED BIPARTITE COLORING-solution of  $B$  and let  $w_1 = w_{i_1} \geq w_{i_2} \geq \dots \geq w_{i_p}$  be the weights of its colors. Remark also that:

$$\begin{aligned} \omega(B) &= \sum_{v_i \in U \cup D} w(v_i) \\ \text{opt}(B) &= w_{i_1} + w_{i_2} + \dots + w_{i_p} \\ \omega(B') &= \sum_{i=1}^{4\eta+2} w_i \end{aligned} \quad (27)$$

$$\omega(B') \leq \omega(B) \quad (28)$$

The proof of the existence of a polynomial time differential-approximation schema for MIN WEIGHTED BIPARTITE COLORING is based upon the following two lemmata.

**Lemma 2.**  $|\tilde{C}| \leq 2\eta + 2$ .

**Proof.** Note first that it cannot exist more than 2 colors that are singletons in  $\tilde{C}$ . A contrario, at least two of them are in  $U$  or in  $D$ . By concatenating them into a single color we reduce the objective value (26) of  $\tilde{C}$ .

Denote by  $x$  the number of colors that are singletons and by  $y$  the number of the other colors of  $\tilde{C}$ . Then, obviously,  $x + 2y \leq 4\eta + 2$  and, as mentioned just before,  $x \leq 2$ ; henceforth:

$$2x + 2y \leq 4\eta + 4 \implies |\tilde{C}| = x + y \leq 2\eta + 2$$

that proves the lemma. ■

**Lemma 3.**  $m(B', \tilde{C}) = \text{opt}(B') \leq \text{opt}(B)$ .

**Proof.** Just remark that coloring  $(S_1^* \cap V(B'), S_2^* \cap V(B'), \dots, S_p^* \cap V(B'))$  is feasible for  $B'$  and it is only a part of  $C^*$ . ■

We are ready now to prove the following theorem.

**Theorem 7.** ([26]) *Algorithm BIC is polynomial time differential-approximation schema for MIN WEIGHTED BIPARTITE COLORING.*

**Proof.** Using (27) and Lemma 2 we have:

$$\omega(B') - \text{opt}(B') = \sum_{i=1}^{4\eta+2} w_i - \sum_{j=1}^{|\tilde{C}|} w_{i_j} \geq 2\eta w_{4\eta+2} \geq \frac{2}{\epsilon} w_{4\eta+2} \quad (29)$$

On the other hand:

$$w(S_U) \leq w_{4\eta+2} \quad (30)$$

$$w(S_D) \leq w_{4\eta+2} \quad (31)$$

From (29), (30) and (31), we get:

$$\begin{aligned} m(B, C) &= w(S_U) + w(S_D) + \text{opt}(B') \\ &= (1 - \epsilon)\text{opt}(B') + \epsilon \left( \text{opt}(B') + \frac{1}{\epsilon} w(S_U) + \frac{1}{\epsilon} w(S_D) \right) \\ &\leq (1 - \epsilon)\text{opt}(B') + \epsilon \underbrace{\left( \text{opt}(B') + \frac{2}{\epsilon} w_{4\eta+2} \right)}_{\leq \text{opt}(B') \leq \omega(B)} \leq (1 - \epsilon)\text{opt}(B) + \epsilon\omega(B) \end{aligned}$$

that gives the schema claimed. ■

As it is shown in [26], MIN WEIGHTED BIPARTITE COLORING cannot be solved by fully polynomial time differential-approximation schema. On the other hand, it is approximable within standard-ratio slightly better than  $4/3$  and inapproximable within standard ratio  $8/7$ , unless  $\mathbf{P} = \mathbf{NP}$  ([26]).

#### 4.7 KNAPSACK

We finish Section 4 by handling one of the most famous problems in combinatorial optimization, that is KNAPSACK. An instance  $I$  of knapsack is the specification of two vectors  $\vec{a}$  and  $\vec{c}$  and of a constant  $b$  and can be defined in terms of an integer-linear program as follows:

$$I = \left\{ \begin{array}{l} \max \quad \vec{a} \cdot \vec{x} \\ \vec{c} \cdot \vec{x} \leq b \end{array} \right.$$

Consider the following algorithm for KNAPSACK presented by [45]:

1. fix an  $\epsilon > 0$  and build the instance  $I' = ((a'_i, c_i)_{i=1, \dots, n}, b)$  with  $a'_i = \lfloor a_i n / (a_{\max} \epsilon) \rfloor$ ;
2. output  $S := \text{DYNAMICPROGRAMMING}(I')$ .

This dynamic programming algorithm is a classical example of how polynomial time approximation schemata are constructed. In fact, the most common technique for them consists first of scaling down data in such a way that the new instance becomes polynomial, then of solving it and, finally, of proving that the solution obtained corresponds to a feasible solution of the initial instance whose value is “very close” to the optimal value.

Step 2 above runs in  $O(n^2 a'_{\max} \log c_{\max}) = O((n^3 \log c_{\max})/\epsilon)$  ([45]). So the whole running time of the algorithm is polynomial.

**Theorem 8.**  $\text{KNAPSACK} \in \text{FPTAS}$ .

**Proof.** Let  $S^*$  be an optimal solution of  $I$ . Obviously,  $S^*$  is feasible for  $I'$ . Let:

$$t = \frac{a_{\max}\epsilon}{n} \quad (32)$$

Then, for every  $i = 1, \dots, n$ :

$$a'_i = \left\lfloor \frac{a_i}{t} \right\rfloor \quad (33)$$

and the following holds:

$$\begin{aligned} \text{opt}(I') &\geq \sum_{i \in S^*} a'_i \geq \sum_{i \in S^*} \left( \frac{a_i}{t} - 1 \right) \\ &\geq \frac{\text{opt}(I)}{t} - |S^*| \geq \frac{\text{opt}(I)}{t} - n \implies t \text{opt}(I') \geq \text{opt}(I) - nt \end{aligned} \quad (34)$$

Note now that the largest  $a_i$ , i.e., that whose index verify:  $i_0 = \text{argmax}\{a_{\max}\}$  is feasible for  $I$ . Hence,

$$\text{opt}(I) = \sum_{i \in S^*} a_i \geq a_{\max} \quad (35)$$

Putting (32), (35) and (34) together, we get:

$$nt = a_{\max}\epsilon \leq \epsilon \text{opt}(I) \quad (36)$$

Then the following hold for the value of the solution  $S$  returned by the algorithm:

$$m(I, S) = \sum_{i \in S} a_i \stackrel{(33)}{\geq} t \sum_{i \in S} a'_i = t \text{opt}(I') \stackrel{(34)}{\geq} \text{opt}(I) - nt \stackrel{(36)}{\geq} (1 - \epsilon) \text{opt}(I)$$

To conclude, it suffices to observe that the complexity of the algorithm is “fully” polynomial, since it does not depend on  $\epsilon$ . Moreover, since it depends on the logarithm of  $c_{\max}$ , the algorithm remains polynomial even if  $c_{\max}$  is exponential with the size  $n$  of the instance. The proof of the theorem is completed. ■

Let us note that, taking nothing is feasible for  $\text{KNAPSACK}$ , producing a solution of value 0 that is the worst solution for any instance  $I$ . So, standard- and differential-approximation ratios coincide for  $\text{KNAPSACK}$ . Henceforth,  $\text{KNAPSACK}$  belongs also to **DFPTAS**.

## 5 Approximability preserving reductions

The use to transform a problem into a different, but related, problem with the aim of exploiting the information we have on the latter in order to solve the former, has been always present in mathematics. Consider, for example, how Greek mathematicians and, subsequently, Al Khuwarizmi ([14]) made use of geometrical arguments in order to solve algebraic problems.

In recent times, a particular type of transformation, called *reduction* has been introduced by logicians in computability theory ([51]). In this case, a reduction from a problem  $\Pi$  to a problem  $\Pi'$  not only specifies how the former can be solved starting from the solution of the latter but, possibly more important in such context, it allows to show that if problem  $\Pi$  is unsolvable (i.e., no algorithm for its solution may exist), so is problem  $\Pi'$ . Such development of the notion of problem transformation is of great importance because it determines a twofold application of mathematical transformations: on one side they allow to transfer positive results (solution techniques) from one problem to another and on the other side they may also be used for deriving negative (impossibility) results.

The first application of the concept of reduction in computer science and combinatorics arises in the early seventies in the seminal paper by [21], soon followed by the equally fundamental paper by [47]. Actually, both Cook's and Karp's reductions were conceived for relating decision problems from a complexity theoretic point of view. So, if we want to use reductions in solving optimization problems, we need other types of more "optimization-oriented" reductions.

Why do we need them? **NPO** hierarchy discussed in Section 2 (Figures 1 and 2) has been built in a somewhat ad-hoc and "absolute" way in the sense that problems are (constructively) classified following algorithms solving them. However, this classification does not allow comparisons between approximability properties of problems. For instance, we cannot answer or we can only very partially answer, to questions as:

- how can one compare problems with respect to their approximability properties and independently on their respective approximation levels?
- how one can compare approximability of different versions of the same problem (for example, weighted version vs. unweighted one)?
- how one can link different types of approximation for a same problem (for instance, do there exist transfers of approximability results between standard and differential approximation for a given problem)?
- how to apprehend the role of parameters in the study of approximability (for example, we have seen in Section 4.1 that the functions describing approximation ratios for MAX INDEPENDENT SET are different when dealing with  $n$  or when dealing with  $\Delta(G)$ )?
- can we transfer approximation results from a problem to another one?
- can we refine structure of the approximation classes given above by showing, for instance, that some problems are harder than some other ones within the same approximability class (completeness results)?

Researchers try to provide answers to these questions by using carefully defined reductions called *approximation preserving reductions*. Any of the existing ones imposes particular conditions on the way optimal solutions, or approximation ratios, or  $\dots$ , are transformed from one problem to another. For more details, the interested reader can be referred to [5, 8, 9, 22, 32, 44, 66, 72].

In general, given two **NPO** problems  $\Pi = (\mathcal{I}, \text{Sol}, m, \text{opt})$  and  $\Pi' = (\mathcal{I}', \text{Sol}', m', \text{opt}')$ , an approximation preserving reduction  $R$  from  $\Pi$  to  $\Pi'$  (denoted  $\Pi \leq_R \Pi'$ ) is a triple  $(f, g, c)$  of polynomially computable functions such that:

- $f$  transforms an instance  $I \in \mathcal{I}$  into an instance  $f(I) \in \mathcal{I}'$ ;
- $g$  transforms a solution  $S' \in \text{Sol}'(f(I))$  into a solution  $g(I, S') \in \text{Sol}(I)$ ;
- $c$  transforms ratio  $\rho'(f(I), S')$  into  $\rho(I, g(I, S')) = c(\rho'(f(I), S'))$ .

A basic property of an approximation preserving reduction  $\Pi \leq_R \Pi'$  is that:

- if  $\Pi'$  is approximable within ratio  $\rho'$ ,  $\Pi$  is approximable within ratio  $\rho = c(\rho')$ ;
- on the other hand, if, under a likely complexity hypothesis,  $\Pi$  is not approximable within ratio  $\rho$ , then (provided that  $c$  is invertible)  $\Pi'$  is not approximable within ratio  $\rho' = c^{-1}(\rho)$ .

Every reduction can be seen as a binary hardness-relation among problems.

Study and use of approximation preserving reductions is interesting and relevant for both theoretical computer science and operational research communities for two main reasons.

The first reason is “structural”. By means of these reductions, one refines the class of **NP**-hard problems by drawing a hierarchy of classes in the interior of **NP**-hard. This hierarchy can be seen as a sequence of strata, each stratum containing problems of “comparable approximability hardness (or easiness)”. Indeed, any stratum **C** draws the capacity of its problems to be approximable within the approximation level represented by **C** and, simultaneously, the limits to the approximability of these problems. For instance let us refer to Figure 10. Assume two strata **C'** and **C''** representing two approximation levels and suppose, w.l.o.g. that **C'** is the class **APX** and **C''** is the class **Log-APX**. Suppose also that a new problem  $\Pi$ , for which no approximation result was known, has been reduced to a problem  $\Pi' \in \mathbf{Log-APX}$  by a reduction preserving logarithmic approximation ratios. An immediate corollary is then that also  $\Pi$  belongs to **Log-APX** and, unless a stronger positive result is proved for it, does not belong to **APX**.

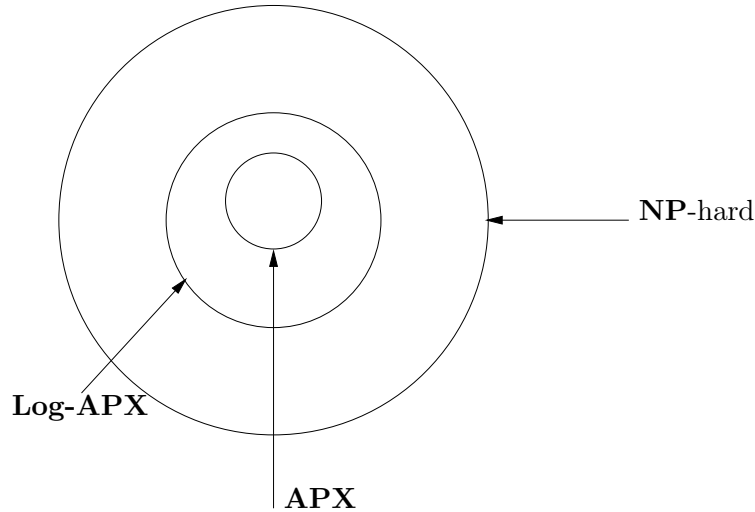


Figure 10: Designing an approximability hierarchy for **NP**-hard class.

The second reason is “operational”. Approximability preserving reductions represent a kind of alternative in the achievement of new approximation results for particular hard problems. When one tries to approximately solve a new problem  $\Pi$  (or to improve existing approximation results for it) a possible way is to operate autonomously by undertaking a thorough and “from-the-beginning” study of the approximability of  $\Pi$ . However, another way to apprehend  $\Pi$  is to put in contribution not only the structural characteristics of this problem but also the whole knowledge dealing with all the problems “similar” to  $\Pi$ . For instance, assume that there exist two approximation preserving reductions **R** from  $\Pi$  to a problem  $\Pi'$  and **Q** from a problem  $\Pi''$  to  $\Pi$  and that we know a positive approximation result for  $\Pi'$  (i.e., an algorithm achieving approximation ratio  $r'$  for it), and a negative result for  $\Pi''$  i.e, an assertion that  $\Pi''$  is not approximable within some ratio  $r''$ , unless an unlikely complexity-theoretical assumption (for example,  $\mathbf{P} = \mathbf{NP}$ )

holds. Then, according to the particular characteristics of  $R$  and  $Q$ , one derives that  $\Pi$  is approximable within ratio, say  $c'(r')$ , but it is not approximable within ratio  $c''(r'')$ , where  $c'$  and  $c''$  are positive real functions depending on  $R$  and  $Q$ , respectively (see the definition of an approximability preserving reduction given above).

There exist a lot of approximability preserving reductions devised today for several combinatorial optimization problems. Let us give two very simple examples.

**Example 1. MAX INDEPENDENT SET and MAX CLIQUE<sup>11</sup>.** It is very well-known from graph theory that an independent set in a graph  $G$  becomes a clique of the same size in  $\bar{G}$  and vice-versa. Assume now that we know an approximation algorithm  $A$  for MAX INDEPENDENT SET achieving an approximation ratio expressed as function of  $n$  (the size of the input-graph). Assume also that we want to approximately solve MAX CLIQUE in a graph  $G$ . Then, we can build  $\bar{G}$  and run  $A$  on it. Algorithm  $A$  will return an independent set of  $\bar{G}$  that becomes a clique seen in  $G$ . This clique has the same size as the independent set initially computed and since  $G$  and  $\bar{G}$  have the same size, the approximation ratio achieved for MAX INDEPENDENT SET is also achieved for MAX CLIQUE. The inverse is also true. ■

**Example 2. MIN COLORING and MIN PARTITION INTO CLIQUES<sup>12</sup>.** Using the relation between a clique and an independent set in Example 1, it can be easily seen that a coloring in  $G$  becomes a partition into cliques (of the same size) in  $\bar{G}$ . So, an approximation algorithm  $A$  for say MIN COLORING achieving an approximation ratio expressed as function of  $n$  can be used in the same way as previously to solve MIN PARTITION INTO CLIQUES (and vice-versa) with the same approximation ratio in both standard and differential paradigms. ■

For a long time, approximability-preserving reductions have been considered as a kind of “universal” tools allowing us to produce any kind of results and for any kind of approximation ratios (i.e., independently on their forms and parameters). But this is absolutely not true. In fact, reductions are not universal. Most of them cannot preserve neither every value nor every form of approximation ratio.

Let us revisit the reduction of Example 1. If, as we did there, we assume that the ratio is function of  $n$ , then preservation works. The same would hold if the ratio assumed was a constant. If, on the other hand, this ratio is a function of, say,  $\Delta(G)$ , then things become complicated, since no general relation exists between  $\Delta(G)$  and  $\Delta(\bar{G})$ . So, reduction of Example 1 does not preserve ratios functions of the maximum degree of the input graph. The same observation can be also made for Example 2.

Let us finally note that most of the reductions known are devised to preserve approximation ratios that are “better than constants”.

As a last example, let us consider the following classical reduction between MAX WEIGHTED INDEPENDENT SET<sup>13</sup> and MAX INDEPENDENT SET ([70]).

Let us consider an instance  $(G(V, E), \vec{w})$  of MAX WEIGHTED INDEPENDENT SET and suppose, in order that the reduction that follows is polynomial, that weights are polynomial with the order  $n$  of  $G$ . We transform it into an instance  $G'(V', E')$  of MAX INDEPENDENT SET as follows:

- we replace every vertex  $v_i \in V$  by an independent set  $W_i$  of  $w_i$  new vertices;

---

<sup>11</sup>Given a graph  $G(V, E)$ , MAX CLIQUE consists of determining a maximum-size subset  $V' \subseteq V$  such that  $G[V']$  is a complete graph.

<sup>12</sup>Given a graph  $G$ , MIN PARTITION INTO CLIQUES consists of determining a minimum partition of the vertex-set of  $G$  into sets each of them inducing a complete subgraph of  $G$ , i.e., a clique.

<sup>13</sup>Given a vertex-weighted graph  $G$ , the objective is to determine an independent set maximizing the sum of the weights of its vertices.



- we replace every edge  $(v_i, v_j) \in E$  by a complete bipartite graph among the vertices of the independent sets  $W_i$  et  $W_j$  in  $G'$  (see Figure 11 where the vertices  $v_i$  and  $v_j$  have respectively weights 3 and 2).

This transformation is polynomial since the resulting graph  $G'$  has  $\sum_{i=1}^n w_i$  vertices and every  $w_i$  is polynomial with  $n$ .

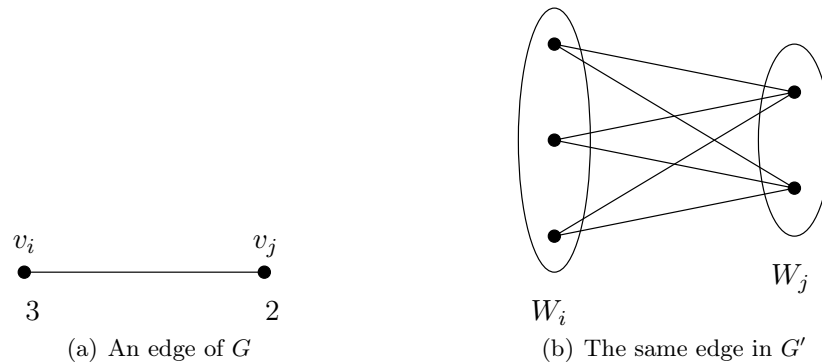


Figure 11: Transformation of an instance  $(G(V, E), \vec{w})$  of MAX WEIGHTED INDEPENDENT SET into an instance  $G'(V', E')$  of MAX INDEPENDENT SET.

Let us now consider an independent set  $S'$  of  $G'$  and w.l.o.g. let us assume it is maximal with respect to inclusion (in case it is not, we can easily add vertices until we reach a maximal independent). Then,  $S' = \cup_{j=1}^k W_{i_j}$ , i.e., there exists a  $k$  such that  $S'$  consists of  $k$  independent sets  $W_{i_j}$ ,  $j = 1, \dots, k$ , corresponding to  $k$  independent vertices  $v_{i_1}, \dots, v_{i_k} \in V$ . So,  $|S'| = \sum_{j=1}^k w_{i_j}$ .

Hence, consider an independent set  $S'$  of  $G'$ . If  $W_i$ ,  $i = 1, \dots, k$ , are the independent sets that form  $S'$  ( $|S'| = \sum_{i=1}^k w_i$ ), then an independent set  $S$  for  $G$  can be built that contains all corresponding vertices  $v_1, \dots, v_k$  of  $V$  with weights  $w_1, \dots, w_k$ , respectively. The total weight of  $S$  is then  $\sum_{i=1}^k w_i = |S'|$

Let us now suppose that we have a polynomial time approximation algorithm **A** with ratio  $r$  for MAX INDEPENDENT SET and consider an instance of  $(G(V, E), \vec{w})$  of MAX WEIGHTED INDEPENDENT SET. Then the following algorithm is a polynomial time approximation algorithm for MAX WEIGHTED INDEPENDENT SET, denoted by **WA**:

- construct  $G'$  as previously;
- run **A** on  $G'$ ; let  $S'$  be the computed independent set;
- construct an independent set  $S$  of  $G$  as explained before.

From what has been discussed just above from any solution for  $S'$  in  $G'$  we can build a solution  $S$  for  $G$  of value (total weight)  $|S'|$ . So, the same ratio achieved by **A** on  $G'$  is also guaranteed by **WA** on  $G$ .

It is easy to see that this reduction preserves constant approximation ratios. But, it is also easy to see that a ratio  $f(\Delta(G))$  (i.e., function of  $\Delta(G)$ ) for MAX INDEPENDENT SET transforms into a ratio  $O(f(\Delta(G)w_{\max}))$  (where  $w_{\max}$  is the maximum weight-value) and not into  $f(\Delta(G))$  for MAX WEIGHTED INDEPENDENT SET. Hence, the reduction does not preserve ratios functions of  $\Delta$ . The same observation immediately holds for ratios functions of the order of the input-graph.

## 6 Some words on inapproximability

Study of approximability properties of a problem includes two complementary issues: the development of approximation algorithms guaranteeing “good” approximation ratios and the achievement of inapproximability results. The goal of an inapproximability result is to provide answers to a global question, addressed this time not to a single algorithm but to a combinatorial optimization problem  $\Pi$  itself. This stake does not only consist of answering if the analysis of a specific approximation algorithm for  $\Pi$  is fine or not but, informally, if the algorithm devised is the best possible (with respect to the approximation ratio it guarantees); in other words, it provides answers to questions as: “do there exist other better algorithms for  $\Pi$ ?”. Or, more generally, “what is the best approximation ratio that a polynomial algorithm could ever guarantee for  $\Pi$ ?”. Hence, the goal is to prove that  $\Pi$  is inapproximable within some ratio  $r$  unless a very unlikely complexity hypothesis becomes true (the strongest such hypothesis is obviously  $\mathbf{P} = \mathbf{NP}$ ).

This type of results is very interesting and adds new insights to computationally hard problems. An important characteristic of complexity theory is that very frequently knowledge is enriched more by impossibility proofs than by possibility ones, even if the latter introduce some pessimism.

“When we exclusively see things from a positive point of view, very frequently we elude fine and efficient approaches, we ignore or we cannot see large avenues that lead to new areas and open new possibilities. When we try to prove the impossible we have to firstly apprehend the complete spectrum of the possible ([60], translation from Greek by the author).

There exist three fundamental techniques to prove inapproximability results: the *GAP-reductions*, the *PCP theorem* and the *approximability preserving reductions*. In what follows we shortly explain the former of these techniques, that is the older technique for proving such results. For the *PCP* theorem, the interested reader can be referred to [2] as well as to [5, 66, 72].

As we have already mentioned, an inapproximability result for an *NPO* problem  $\Pi$  consists of showing that if we had an approximation algorithm achieving some approximation ratio  $r$ , then this fact would contradict a commonly accepted complexity hypothesis (e.g.,  $\mathbf{P} \neq \mathbf{NP}$ ). How can we do this? Let us consider the following example showing that *MIN COLORING* is not approximable within standard-ratio less than  $4/3 - \epsilon$ , for any  $\epsilon > 0$ .

**Example 3.** Revisit the *NP*-completeness proof for the decision version of *MIN COLORING* given in [35]. The reduction proposed there constructs, starting from an instance  $\varphi$  of *E3SAT*<sup>14</sup>, a graph  $G$  such that if  $\varphi$  is satisfiable then  $G$  is 3-colorable (i.e., its vertices can be colored by 3 colors), otherwise  $G$  is at least 4-colorable.

Suppose now that there exists a polynomial algorithm for *MIN COLORING* guaranteeing standard-approximation ratio  $(4/3) - \epsilon$ , with  $\epsilon > 0$ . Run it on the graph  $G$  constructed from  $\varphi$ . If  $\varphi$  is not satisfiable, then this algorithm computes a coloring for  $G$  using more than 4 colors. On the other hand, if  $\varphi$  is satisfiable (hence  $G$  is 3-colorable), then the algorithm produces a coloring using at most  $3((4/3) - \epsilon) < 4$  colors, i.e., a 3-coloring. So, on the hypothesis that a polynomial algorithm for *MIN COLORING* guaranteeing approximation ratio  $(4/3) - \epsilon$  exists, one can in polynomial time decide if a formula  $\varphi$ , instance of *E3SAT*, is satisfiable or not, contradicting so the *NP*-completeness of this problem. Hence, *MIN COLORING* is not  $((4/3) - \epsilon)$ -standard-approximable, unless  $\mathbf{P} = \mathbf{NP}$ . ■

---

<sup>14</sup>Given a set of  $m$  clauses over a set of  $n$  variables, *SAT* consists of finding a model for the conjunction of these clauses, i.e., an assignment of truth values to the variables that simultaneously satisfies all the clauses; *E3SAT* is the restriction of *SAT* to clauses with exactly three literals.

The reduction of Example 3 is a typical example of a **GAP**-reduction. Via such reductions, one tries to create a gap separating *yes*-instances (i.e., the instances of the problem where answer is *yes*) of a decision problem from *no*-instances (i.e., the instances of the problem where answer is *no*).

More generally, denoting, for some decision problem  $\Pi$ , by  $\mathcal{O}_\Pi$  the set of its *yes*-instances, the basic idea of a **GAP**-reduction is the following.

Consider a decision problem  $\Pi$  that is **NP**-complete and an **NPO** problem  $\Pi'$  (suppose that  $\text{goal}(\Pi') = \min$ ). If we devise a polynomial reduction from  $\Pi$  to  $\Pi'$  such that, there exist  $c, r > 1$  for which:

- if  $I$  is a *yes*-instance of  $\Pi$ , then  $\text{opt}(I') \leq c$ ,
- if  $I$  is a *no*-instance of  $\Pi$ , then  $\text{opt}(I') > rc$ ,

then, the reduction above is a **GAP**-reduction proving that  $\Pi'$  is inapproximable within standard-ratio  $r$ .

Indeed, if  $\Pi'$  was approximable within  $r$ , then:

- $\forall I \in \mathcal{O}_\Pi$  (the set *yes*-instances),  $m(f(I), S') \leq rc$ ;
- $\forall I \in \mathcal{I}_\Pi \setminus \mathcal{O}_\Pi$  (the set *no*-instances),  $m(f(I), S') \geq \text{opt}(f(I)) > rc$ .

Consequently, we would have a separation criterion, i.e., a *gap* (see also Figure 12), checkable in polynomial time, between the *yes*- and the *no*-instances of  $\Pi$ . Such a criterion is impossible since  $\Pi$  is **NP**-complete.

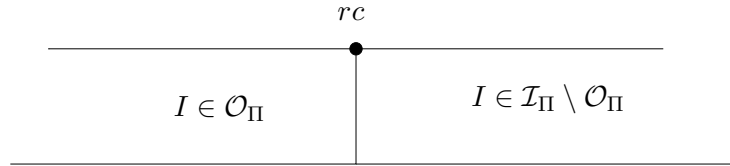


Figure 12: The gap.

How can we extend such results to other problems? Assume, for example, that a **GAP**-reduction is devised between an **NP**-complete decision problem  $\Pi$  and an **NPO** problem  $\Pi'$ , deriving that  $\Pi'$  is not approximable within ratio better than  $r$ . Suppose also that an approximability preserving reduction  $(f, g, c)$  (see Section 5) is devised from  $\Pi'$  to some other **NPO** problem  $\Pi''$ . Then  $\Pi''$  is not approximable within better than  $c^{-1}(r)$ .

We conclude this short section with another example of **GAP**-reduction settling **MIN TSP**.

**Example 4.** ([35, 68]) Let us prove that (general) **MIN TSP** is not approximable within any constant standard-approximation ratio, unless  $\mathbf{P} = \mathbf{NP}$ . Suppose a contrario, that **MIN TSP** is polynomially approximable within standard-approximation ratio  $r$  for some constant  $r$ , denote by **A** an approximation algorithm achieving this ratio and consider a graph  $G(V, E)$  of order  $n$ , instance of the **HAMILTONIAN CYCLE**<sup>15</sup> problem. Starting from  $G$  construct an instance of **MIN TSP** as follows:

- complete  $G$  in order to build a complete graph  $K_n$ ;

<sup>15</sup>Given a graph  $G$  **HAMILTONIAN CYCLE** consists of deciding if  $G$  contains a Hamiltonian tour (see footnote 8); this problem is **NP**-complete ([35]).

- for every  $e \in E(K_n)$  (the edge-set of  $K_n$ ) set:

$$d(e) = \begin{cases} 1 & \text{if } e \in E(G) \\ rn & \text{if } e \notin E(G) \end{cases}$$

Consider now the following two cases depending on the fact that  $G$  is Hamiltonian (i.e., it has a Hamiltonian cycle) or not:

1.  $G$  is Hamiltonian; then,  $\text{opt}(K_n) = n$ , since the edges of the Hamiltonian cycle of  $G$  exist in  $K_n$  and have all weights equal to 1; in this case, Algorithm A will produce a solution (tour)  $T$  for MIN TSP of value  $m(K_n, T) \leq rn$ ;
2.  $G$  is not Hamiltonian; then, obviously, no tour in  $K_n$  can only use edges weighted by 1 (if such tour existed, it would be a Hamiltonian cycle in  $G$ ) so, any tour will use at least one edge of weight  $rn$ ; hence,  $\text{opt}(K_n) \geq rn + n - 1 > rn$  and, since A will produce something worse than the optimum,  $m_A(K_n, T) \geq \text{opt}(K_n) > rn$ .

We so have a gap between Hamiltonian graphs, deriving MIN TSP-solutions of value at most  $rn$ , and non-Hamiltonian graphs implying MIN TSP-solutions of value greater than  $rn$ .

So, on the hypothesis that MIN TSP is approximable within standard-approximation ratio  $r$ , one could polynomially solve HAMILTONIAN CYCLE as follows:

- starting from an instance  $G$  of HAMILTONIAN CYCLE, construct instance  $K_n$  of MIN TSP as described;
- run A on  $K_n$  and if A returns a tour of value at most  $rn$ , answer *yes* to HAMILTONIAN CYCLE, otherwise answer *no*.

Since everything in the transformation of  $G$  into  $K_n$  is polynomial and, furthermore, Algorithm A is assumed to be polynomial, the whole algorithm for HAMILTONIAN CYCLE is also polynomial, contradicting so the fact that this latter problem is **NP**-complete. ■

Inapproximability result of Example 4 can be importantly strengthened. Indeed, observe that the only case where transformation of  $G$  into  $K_n$  is not polynomial, is when parameter  $r$  (the approximation ratio of the polynomial algorithm assumed for MIN TSP) is doubly exponential. Otherwise, even if  $r$  is exponential, say of the form  $2^{p(n)}$  for any polynomial  $p$  of  $n$ , the described transformation remains polynomial since any number  $k$  can be represented using  $O(\log k)$  bits. So, the following result finally holds and concludes the section.

**Theorem 9.** *Unless  $P = NP$ , MIN TSP cannot be approximately solved within standard-approximation ratios better than  $2^{p(n)}$  for any polynomial  $p$ .*

## 7 A quick “tour d’horizon” about completeness in approximability classes

Given a set  $\mathbf{C}$  of problems and a reduction  $R$ , it is natural to ask if there exist problems  $\Pi \in \mathbf{C}$  such that any problem  $\Pi' \in \mathbf{C}$ ,  $R$ -reduces to  $\Pi$ . Such “maximal” problems are called in complexity theory *complete problems*. Let  $\mathbf{C}$  be a class of problems and  $R$  be a reduction. A problem  $\Pi \in \mathbf{C}$  is said to be **C**-complete (under  $R$ -reduction) if for any  $\Pi' \in \mathbf{C}$ ,  $\Pi' \leq_R \Pi$ . A **C**-complete problem (under reduction  $R$ ) is then (in the sens of this reduction) a computationally hardest problem for class  $\mathbf{C}$ . For instance, in the case of **NP**-completeness, **NP**-complete problems (under Karp-reduction ([35])) are the hardest problems of **NP** since if one could polynomially solve just one of them, then one would be able to solve in polynomial time any other problem in **NP**. Let  $\mathbf{C}$  be a class of problems and  $R$  a reduction. A problem  $\Pi$  is said to be **C**-hard (under  $R$ -reduction)

if for any  $\Pi' \in \mathbf{C}$ ,  $\Pi' \leq_R \Pi$ . In other words, a problem  $\Pi$  is  $\mathbf{C}$ -complete if and only if  $\Pi \in \mathbf{C}$  and  $\Pi$  is  $\mathbf{C}$ -hard.

The above general definitions can be immediately applied to approximability classes defined in Section 3 in order to produce strong inapproximability results but also in order to create a structure for these classes. In fact, even if approximability preserving reductions mainly concern transfer of results among pairs of problems, we can use them as mentioned, in order to complete the structure of approximability classes.

Consider some approximability preserving reduction  $R$  and suppose that it preserves membership in, say,  $\mathbf{PTAS}$ , in other words, if a problem  $\Pi$   $R$ -reduces to  $\Pi'$  and if  $\Pi' \in \mathbf{PTAS}$ , then  $\Pi \in \mathbf{PTAS}$ . Consider now an approximation class that contains  $\mathbf{PTAS}$ , say  $\mathbf{APX}$  and assume that the existence of a problem  $\Pi$  that is  $\mathbf{APX}$ -complete under  $R$ -reduction has been proved. If  $\Pi$  admits a polynomial time approximation schema then, since  $R$ -reduction preserves membership in  $\mathbf{PTAS}$ , one can deduce the existence of polynomial time approximation schemata for any problem that is  $R$ -reducible to  $\Pi$ , hence, in particular, for any problem in  $\mathbf{APX}$ . In other words, by the assumptions just made, we have:

$$\Pi \in \mathbf{PTAS} \implies \mathbf{APX} = \mathbf{PTAS}$$

Since, under the hypothesis  $\mathbf{P} \neq \mathbf{NP}$ ,  $\mathbf{PTAS} \subsetneq \mathbf{APX}$ , one can conclude that, under the same hypothesis,  $\Pi \notin \mathbf{PTAS}$ .

The above schema of reasoning can be generalized for any approximation class. Let  $\mathbf{C}$  be a class of problems. We say that a reduction  $R$  preserves membership in  $\mathbf{C}$ , if for every pair of problems  $\Pi$  and  $\Pi'$ :

$$\Pi \in \mathbf{C} \text{ and } \Pi' \leq_R \Pi \implies \Pi' \in \mathbf{C}$$

We then have the following proposition.

**Proposition 1.** *Let  $\mathbf{C}$  and  $\mathbf{C}'$  be two problem-classes with  $\mathbf{C}' \subsetneq \mathbf{C}$ . If a problem  $\Pi$  is  $\mathbf{C}$ -complete under some reduction preserving membership in  $\mathbf{C}'$ , then  $\Pi \notin \mathbf{C}'$ .*

Obviously, if the strict inclusion of classes is subject to some complexity hypothesis, the conclusion  $\Pi \notin \mathbf{C}'$  is subject to the same hypothesis.

The analogy with  $\mathbf{NP}$ -completeness is immediate. The fundamental property of Karp- (or Turing-) reduction is that it preserves membership in  $\mathbf{P}$ . Application of Proposition 1 to  $\mathbf{NP}$ -completeness framework simply says that  $\mathbf{NP}$ -complete problems can never be in  $\mathbf{P}$ , unless  $\mathbf{P} = \mathbf{NP}$ .

When the problem of characterizing approximation algorithms for hard optimization problems was tackled, soon the need arose for a suitable notion of reduction that could be applied to optimization problems in order to study their approximability properties.

What is it that makes algorithms for different problems behave in the same way? Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain these results, or are they due to some structural similarity between the problems as we define them? ([46]).

The first answer to the above questions was given by [6, 7] where the notion of *structure preserving reduction* is introduced and for the first time the completeness of MAX VARIABLE-WSAT (a weighted version of MAX SAT) in the class of  $\mathbf{NPO}$  problems is proved. Still it took a few more years until suitable kinds of reductions among optimization problems were introduced by [58]. In particular, this paper presented the so-called *strict reduction* and provided the first examples of complete problems under approximation preserving reductions (MIN VARIABLE-WSAT, MIN 0-1 LINEAR PROGRAMMING and MIN TSP).

After [58] a large variety of approximation preserving reductions have appeared in the literature. The introduction of powerful approximation preserving reductions and the beginning of the structural theory of approximability of optimization problems can be traced back to the fundamental paper by [24] where the first **PTAS** preserving reductions (the *PTAS-reduction*) is introduced and the first complete problems in **APX** under such types of reductions are presented. Unfortunately the problem which is proved **APX**-complete in this paper is quite artificial, **MAX VARIABLE-WSAT- $B$** , a version of **MAX VARIABLE-WSAT** in which, given a constant  $B$ , the sum of weights of variables is contained between  $B$  and  $2B$ .

Along a different line of research, during the same years, the study of logical properties of optimization problems has led to the syntactic characterization of an important class of approximable problems, the class **Max-SNP** (see [62]) strongly based upon characterization of **NP** by [33]. Completeness in the class **Max-SNP** has been defined in terms of *L-reductions* and natural complete problems (e.g., **MAX 3SAT**, **MAX 2SAT**, **MIN VERTEX COVER** etc.) have been found. The relevance of such approach is related to the fact that it is possible to prove that **Max-SNP**-complete problems do not allow polynomial time approximation schemata, unless  $\mathbf{P} = \mathbf{NP}$  ([2]).

The two approaches have been reconciled by [49], where the closure of syntactically defined classes with respect to an approximation preserving reduction were proved equal to the more familiar computationally defined classes. As a consequence of this result any **Max-SNP**-completeness result appeared in the literature can be interpreted as an **APX**-completeness result. In this paper a new type of reduction is introduced, the *E-reduction*. This reduction is fairly powerful since it allows to prove that **MAX 3SAT** is **APX**-complete. On the other side, it remains somewhat restricted because it does not allow the transformation of **PTAS** problems (such as **MAX KNAPSACK**) into problems belonging to **APX-PB** (the class of problems in **APX** whose solution-values are bounded by a polynomial in the size of the instance) such as **MAX 3SAT**. In [49], completeness in approximability classes beyond **APX**, as **Log-APX** and **Poly-APX** has been also tackled and completeness results for subclasses of them (**Log-APX-PB** and **Poly-APX-PB**, respectively, where problems have solution-values bounded by a polynomial in the size of the instance) have been proved. Existence of natural complete problems for the whole classes **Log-APX** and **Poly-APX** has been proved in [31, 11], respectively, under *FT-reduction* and *MPTAS-reduction*.

The final answer to the problem of finding the suitable kind of reduction (powerful enough to establish completeness results both in **NPO** and **APX**) is the *AP-reduction*, introduced by [23].

A large number of other reductions among optimization problems have been introduced throughout the years. Overviews of the world of approximation preserving reductions and completeness in approximability classes can be found in [8, 9, 22, 23].

On the other hand, a structural development analogous to the one that has been carried on for the standard paradigm has been elaborated also for the differential paradigm that is much younger than the former since it has been defined at the beginning of the 90's by [27]. In [4] and in [11] the approximability classes **DAPX** and **DPTAS** are introduced, suitable approximation preserving reductions are defined and complete problems in **NPO**, **0-DAPX**, **DAPX**, **DPTAS** and **Poly-DAPX**, under such kind of reductions, are shown.

Finally, in [31], together with the existence of complete problems for **Log-APX** and **Poly-APX**, completeness in class **Exp-APX** is also proved and tools for proving completeness in classes **Log-DAPX** and **Exp-DAPX**, where no natural problems are still proved to belong to, are given.

We conclude this *tour d'horizon*, with a synthesis of the several completeness results briefly presented just above for the combinatorially defined approximability classes seen given in Section 3 (excluding so the syntactically defined class **Max-SNP**).

For the standard-approximation paradigm:

- **NPO**-complete: several versions of variable-weighted SAT ([6, 7, 58]);
- **Exp-APX**-complete: MIN TSP ([31]);
- **Poly-APX**-complete: MAX INDEPENDENT SET ([11]);
- **Log-APX**-complete: MIN SET COVER ([31]);
- **APX**-complete: MAX 3-SAT, MIN VERTEX COVER- $B^{16}$ , MAX INDEPENDENT SET- $B^{17}$ , MIN TSP with edge-weights 1 and 2 ... ([24, 23, 49], etc.);
- **PTAS**-complete: MIN PLANAR VERTEX COVER, MAX PLANAR INDEPENDENT SET ([11]).

For the differential-approximation paradigm:

- **0-DAPX**-complete: MIN INDEPENDENT DOMINATING SET, ... ([4]);
- **Exp-DAPX**-complete: no natural problem is still known to be in **Exp-DAPX**;
- **Poly-DAPX**-complete: MAX INDEPENDENT SET, MIN VERTEX COVER, MIN SET COVER, MAX CLIQUE, ... ([31]);
- **Log-DAPX**-complete: the same as for class **Exp-DAPX** holds;
- **DAPX**-complete: MIN VERTEX COVER- $B$ , MAX INDEPENDENT SET- $B$ , ... ([4]), MIN COLORING ([11]);
- **DPTAS**-complete: MIN PLANAR VERTEX COVER, MAX PLANAR INDEPENDENT SET ([11]), BIN PACKING, ... ([11]).

## 8 Further remarks

Polynomial approximation is a research area in the boundaries several research fields the main among them being combinatorial optimization and theoretical computer science. For more than thirty years, it constitutes a very active research programme that has rallied numerous researchers all over the world. Furthermore, it has inspired several new approaches in both operational research and computer science.

One of these new approaches is a dynamic computation model, called *online approximation*, where the basic hypotheses are that instance to be solved is revealed step-by-step and the algorithm supposed to solve it has to maintain a feasible solution for the part of the instance already revealed. The quality of such an algorithm is measured by means of its *competitive ratio* defined as the ratio of the value of the solution computed during instance's revealing divided by the value of the optimal solution of the whole instance (called offline solution). Extensions of online computation can deal not only with data arrival but also with data elimination.

Another approach is the so-called *reoptimization*. Here we suppose that we have an optimal solution for an instance (no matter how this solution is produced) and some new data arrive. Can we operate a fast transformation of the solution at hand in order to fit the augmented instance? Is the new solution optimal or not? If not, does it achieves some good approximation ratio?.

Notions and tools from polynomial approximation are also used in a relatively new research field that is actually in full expansion: the algorithmic game theory. The so-called *price of anarchy* is fully inspired from polynomial approximation.

---

<sup>16</sup>This is MIN VERTEX COVER in graphs with maximum degrees bounded above by a fixed constant  $B$ .

<sup>17</sup>This is MAX INDEPENDENT SET in graphs with maximum degrees bounded above by a fixed constant  $B$ .

What are the mid- and long-term perspectives of this area? Certainly, producing new operational and structural results are such perspectives. For instance, syntactic classes, as class **Max-SNP** is not yet fully studied in this paradigm. Also, optimal satisfiability problems, central problems in the standard paradigm, deserve further research in differential approximation.

But, to our opinion, major long-term perspective is to match polynomial approximation with exact computation. Indeed, another very active area of combinatorial optimization is the development of exact algorithms for **NP**-hard problems with non-trivial worst-case running times. For example, it is obvious that an exhaustive method for **MAX INDEPENDENT SET** will run in time at most  $2^n$ . But can we go faster? Such faster algorithms with improved worst-case complexity have been recently devised for numerous **NPO** problems. Polynomial approximation and exact computation (with worst case upper time-bounds) can be brought together in several ways. For instance, are we able to produce approximation ratios “forbidden” for polynomial algorithms (e.g., constant ratios for **MAX INDEPENDENT SET**, or ratios smaller than 2 for **MIN VERTEX COVER**) by exponential or super-polynomial algorithms running in times much lower than those for the exact computation for the corresponding problems? And, in a second time, can we adapt the concept of approximability preserving reductions to fit this new issue?

## References

- [1] S. Arora. Polynomial time approximation schemes for Euclidean TSP and other geometric problems. In *Proc. FOCS'96*, pages 2–11, 1996.
- [2] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and intractability of approximation problems. *J. Assoc. Comput. Mach.*, 45(3):501–555, 1998.
- [3] B. Aspvall and R. E. Stone. Khachiyan’s linear programming algorithm. *J. Algorithms*, 1:1–13, 1980.
- [4] G. Ausiello, C. Bazgan, M. Demange, and V. Th. Paschos. Completeness in differential approximation classes. *International Journal of Foundations of Computer Science*, 16(6):1267–1295, 2005.
- [5] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi. *Complexity and approximation. Combinatorial optimization problems and their approximability properties*. Springer-Verlag, Berlin, 1999.
- [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] G. Ausiello, A. D’Atri, and M. Protasi. Lattice-theoretical ordering properties for NP-complete optimization problems. *Fundamenta Informaticæ*, 4:83–94, 1981.
- [8] G. Ausiello and V. Th. Paschos. Reductions, completeness and the hardness of approximability. *European J. Oper. Res.*, 172:719–739, 2006. Invited review.
- [9] G. Ausiello and V. Th. Paschos. Reductions that preserve approximability. In T. F. Gonzalez, editor, *Handbook of approximation algorithms and metaheuristics*, chapter 15, pages 15–1–15–16. Chapman & Hall, Boca Raton, 2007.
- [10] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Ann. Discrete Appl. Math.*, 25:27–46, 1985.



- [11] C. Bazgan, B. Escoffier, and V. Th. Paschos. Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness. *Theoret. Comput. Sci.*, 339:272–292, 2005.
- [12] C. Bazgan, J. Monnot, V. Th. Paschos, and F. Serrière. On the differential approximation of MIN SET COVER. *Theoret. Comput. Sci.*, 332:497–513, 2005.
- [13] C. Bazgan and V. Th. Paschos. Differential approximation for optimal satisfiability and related problems. *European J. Oper. Res.*, 147(2):397–404, 2003.
- [14] E. T. Bell. *The Development of Mathematics*. Dover, 1992.
- [15] M. Bellare, O. Goldreich, and M. Sudan. Free bits and non-approximability — towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998.
- [16] C. Berge. *Graphs and hypergraphs*. North Holland, Amsterdam, 1973.
- [17] P. Berman and M. Karpinski. 8/7-approximation algorithm for (1,2)-tsp. In *Proc. Symposium on Discrete Algorithms, SODA'06*, pages 641–648, 2006.
- [18] R. L. Brooks. On coloring the nodes of a network. *Math. Proc. Cambridge Philos. Soc.*, 37:194–197, 1941.
- [19] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Grad. School of Industrial Administration, CMU, 1976.
- [20] V. Chvátal. A greedy-heuristic for the set covering problem. *Math. Oper. Res.*, 4:233–235, 1979.
- [21] S. A. Cook. The complexity of theorem-proving procedures. In *Proc. STOC'71*, pages 151–158, 1971.
- [22] P. Crescenzi. A short guide to approximation preserving reductions. In *Proc. Conference on Computational Complexity*, pages 262–273, 1997.
- [23] P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. *SIAM J. Comput.*, 28(5):1759–1782, 1999.
- [24] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, 93(2):241–262, 1991.
- [25] A. Croes. A method for solving traveling-salesman problems. *Oper. Res.*, 5:791–812, 1958.
- [26] M. Demange, D. de Werra, J. Monnot, and V. Th. Paschos. Time slot scheduling of compatible jobs. *J. Scheduling*, 10:111–127, 2007.
- [27] M. Demange and V. Th. Paschos. On an approximation measure founded on the links between optimization and polynomial approximation theory. *Theoret. Comput. Sci.*, 158:117–141, 1996.
- [28] M. Demange and V. Th. Paschos. Improved approximations for maximum independent set via approximation chains. *Appl. Math. Lett.*, 10(3):105–110, 1997.
- [29] M. Demange and V. Th. Paschos. Improved approximations for weighted and unweighted graph problems. *Theory of Computing Systems*, 38:763–787, 2005.

- [30] B. Escoffier and J. Monnot. Better differential approximation for symmetric tsp. Cahier du LAMSADE 261, LAMSADE, Université Paris-Dauphine, 2007.
- [31] B. Escoffier and V. Th. Paschos. Completeness in approximation classes beyond apx. *Theoret. Comput. Sci.*, 359(1–3):369–377, 2006.
- [32] B. Escoffier and V. Th. Paschos. A survey on the structure of approximation classes. Cahier du LAMSADE 267, LAMSADE, Université Paris-Dauphine, 2007. Available at <http://www.lamsade.dauphine.fr>.
- [33] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of computations*, pages 43–73. American Mathematics Society, 1974.
- [34] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proc. Conference on Computational Complexity*, pages 278–287, 1996.
- [35] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.
- [36] O. Goldsmith, D. S. Hochbaum, and G. Yu. A modified greedy heuristic for the set covering problem with improved worst case bound. *Inform. Process. Lett.*, 48:305–310, 1993.
- [37] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981.
- [38] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Inform. Process. Lett.*, 45(1):19–23, 1993.
- [39] M. M. Halldórsson. Approximations of weighted independent set and hereditary subset problems. *J. Graph Algorithms Appl.*, 4(1):1–16, 2000.
- [40] E. Halperin. Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. In *Proc. Symposium on Discrete Algorithms, SODA'00*, 2000.
- [41] R. Hassin and S. Lahav. Maximizing the number of unused colors in the vertex coloring problem. *Inform. Process. Lett.*, 52:87–90, 1994.
- [42] J. Håstad. Clique is hard to approximate within  $n^{1-\epsilon}$ . *Acta Mathematica*, 182:105–142, 1999.
- [43] D. S. Hochbaum. Efficient bounds for the stable set, vertex cover and set packing problems. *Discrete Appl. Math.*, 6:243–254, 1983.
- [44] D. S. Hochbaum, editor. *Approximation algorithms for NP-hard problems*. PWS, Boston, 1997.
- [45] O. H. Ibarra and C. E. Kim. Fast approximation algorithms for the knapsack and sum of subset problems. *J. Assoc. Comput. Mach.*, 22(4):463–468, 1975.
- [46] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.
- [47] 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.

- [48] L. G. Khachian. A polynomial algorithm for linear programming. *Dokladi Akademiy Nauk SSSR*, 244:1093–1096, 1979.
- [49] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. *SIAM J. Comput.*, 28:164–191, 1998.
- [50] S. Khot and O. Regev. Vertex cover might be hard to approximate to within  $2 - \epsilon$ . In *Proc. Annual Conference on Computational Complexity, CCC'03*, pages 379–386, 2003.
- [51] S. C. Kleene. *Introduction to Metamathematics*. Van Nostrand, Princeton, NJ, 1952.
- [52] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.
- [53] L. Lovász. Three short proofs in graph theory. *J. Combin. Theory Ser. B*, 19:269–271, 1975.
- [54] B. Monien and E. Speckenmeyer. Ramsey numbers and an approximation algorithm for the vertex cover problem. *Acta Informatica*, 22:115–123, 1985.
- [55] J. Monnot, V. Th. Paschos, and S. Toulouse. Differential approximation results for the traveling salesman problem with distances 1 and 2. *European J. Oper. Res.*, 145(3):557–568, 2002.
- [56] J. Monnot, V. Th. Paschos, and S. Toulouse. Approximation algorithms for the traveling salesman problem. *Mathematical Methods of Operations Research*, 57(1):387–405, 2003.
- [57] G. L. Nemhauser and L. E. Trotter. Vertex packings: structural properties and algorithms. *Math. Programming*, 8:232–248, 1975.
- [58] P. Orponen and H. Mannila. On approximation preserving reductions: complete problems and robust measures. Technical Report C-1987-28, Dept. of Computer Science, University of Helsinki, Finland, 1987.
- [59] C. H. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.
- [60] C. H. Papadimitriou. *Το χαμόγελο του Turing (Turing's smile)*. Νέα Σύνορα, Athens, 2000. In Greek. English title: Turing.
- [61] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Prentice Hall, New Jersey, 1981.
- [62] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. Comput. System Sci.*, 43:425–440, 1991.
- [63] V. Th. Paschos. A note on the improvement of the maximum independent set's approximation ratio. *Yugoslav Journal of Operations Research*, 5(1):21–26, 1995.
- [64] V. Th. Paschos. A survey about how optimal solutions to some covering and packing problems can be approximated. *ACM Comput. Surveys*, 29(2):171–209, 1997.
- [65] V. Th. Paschos. Polynomial approximation and graph coloring. *Computing*, 70:41–86, 2003.
- [66] V. Th. Paschos. *Complexité et approximation polynomiale*. Hermès, Paris, 2004.
- [67] R. Raz and S. Safra. A sub-constant error probability low-degree test and a sub-constant error probability PCP characterization of NP. In *Proc. STOC'97*, pages 475–484, 1997.

- [68] S. Sahni and T. Gonzalez. P-complete approximation problems. *J. Assoc. Comput. Mach.*, 23:555–565, 1976.
- [69] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, New York, 1986.
- [70] H. U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. Disc. Math.*, 3(2):294–310, 1990.
- [71] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proc. STOC'96*, pages 435–441, 1996.
- [72] V. Vazirani. *Approximation algorithms*. Springer, Berlin, 2001.