Submitted to
SIAM Journal on Computing

Structure in Approximation Classes*

P. Crescenzi[1], V. Kann[2], R. Silvestri[1], and L. Trevisan[1]

[1] Dipartimento di Scienze dell'Informazione
Università degli Studi di Roma "La Sapienza"
Via Salaria 113, 00198 Rome, Italy
E-mail: {piluc,silver,trevisan}@dsi.uniroma1.it
Phone: +39-6-4991 83 14
Fax: +39-6-854 18 42
[2] Department of Numerical Analysis and Computing Science
Royal Institute of Technology
S-100 44 Stockholm, Sweden
E-mail: viggo@nada.kth.se
Phone: +46-8-790 62 92
Fax: +46-8-790 09 30

May 8, 1996

Summary. The study of the approximability properties of NP-hard optimization problems has recently made great advances mainly due to the results obtained in the field of proof checking. The last important breakthrough shows the APX-completeness of several important optimization problems is proved and thus reconciles 'two distinct views of approximation classes: *syntactic* and *computational*'. In this paper we obtain new results on the structure of several computationally-defined approximation classes. In particular, after defining a new approximation preserving reducibility to be used for as many approximation classes as possible, we give the first examples of natural NPO-complete problems and the first examples of natural APX-intermediate problems. Moreover, we state new connections between the approximability properties and the query complexity of NPO problems.

Key words. Complexity classes, reducibilities, approximation algorithms
AMS subject classifications. 03D30, 68Q15, 68Q20

1. Introduction

In his pioneering paper on the approximation of combinatorial optimization problems [20], David Johnson formally introduced the notion of approximable problem, proposed approximation algorithms for several problems, and suggested a possible classification of optimization problems on grounds of their approximability properties. Since then it was clear that, even though the decision versions of most NP-hard optimization problems are many-one polynomial-time reducible to each other, they do not share the same approximability properties. The main reason of this fact is that many-one reductions not always preserve the objective function and, even if this happens, they rarely preserve the quality of the solutions. It is then clear that a stronger kind of reducibility has to be used. Indeed, an approximation preserving reduction not only has to map instances of a problem A to instances of a problem B, but it also has to be

* An extended abstract of this paper has been presented at the *1st Annual International Computing and Combinatorics Conference.*

able to come back from "good" solutions for B to "good" solutions for A. Surprisingly, the first definition of this kind of reducibility [33] was given as long as 13 years after Johnson's paper and, after that, at least seven different approximation preserving reducibilities appeared in the literature (see Fig. 1). These reducibilities are identical with respect to the overall scheme but differ essentially in the way they preserve approximability: they range from the Strict reducibility in which the error cannot increase to the PTAS-reducibility in which there are basically no restrictions (see also Chapter 3 of [23]).

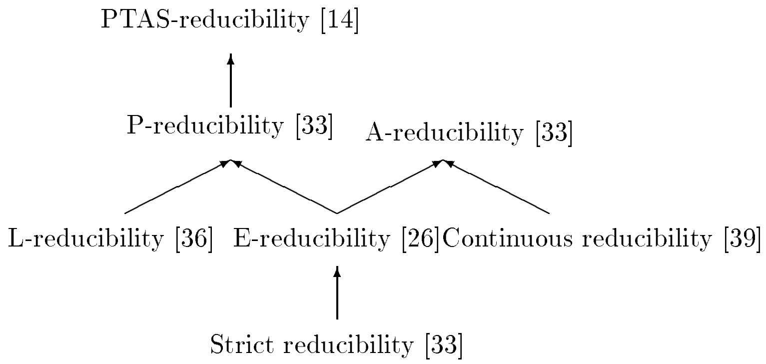

Figure 1. The taxonomy of approximation preserving reducibilities

By means of these reducibilities, several notions of completeness in approximation classes have been introduced and, basically, two different approaches were followed. On the one hand, the attention was focused on computationally defined classes of problems, such as NPO (i.e., the class of optimization problems whose underlying decision problem is in NP) and APX (i.e., the class of constant-factor approximable NPO problems): along this line of research, however, almost all completeness results dealt either with artificial optimization problems or with problems for which lower bounds on the quality of the approximation were easily obtainable [12, 33]. On the other hand, researchers focused on the logical definability of optimization problems and introduced several syntactically defined classes for which natural completeness results were obtained [27, 34, 36]: unfortunately, the approximability properties of the problems in these latter classes were not related to standard complexity-theoretic conjectures. A first step towards the reconciling of these two approaches consisted of proving lower bounds (modulo $\mathrm{P} \neq \mathrm{NP}$ or some other likely condition) on the approximability of complete problems for syntactically defined classes [1, 31]. More recently, another step has been performed since the closure of syntactically defined classes with respect to an approximation preserving reducibility has been proved to be equal to the more familiar computationally defined classes [26].

In spite of this important achievement, beyond APX we are still forced to distinguish between maximization and minimization problems as long as we are interested in completeness proofs. Indeed, a result of [27] states that it is not possible to rewrite every NP maximization problem as an NP minimization problem unless NP=co-NP. A natural question is thus whether this duality extends to approximation preserving reductions.

Finally, even though the existence of "intermediate" artificial problems, that is, problems for which lower bounds on their approximation are not obtainable by completeness results was proved in [12], a natural question arises: do natural intermediate problems exist? Observe that this question is also open in the field of decision problems: for example, it is known that the graph isomorphism problem cannot be NP-complete unless the polynomial-time hierarchy collapses [38], but no result has ever been obtained giving evidence that the problem does not belong to P.

The first goal of this paper is to define an approximation preserving reducibility that can be used for as many approximation classes as possible and such that all reductions that have appeared in the literature still hold. In spite of the fact that the L-reducibility has been the most widely used so far, we will give strong evidence that it cannot be used to obtain completeness results in "computationally defined" classes such as APX, log-APX (that is, the class of problems approximable within a logarithmic factor), and poly-APX (that is, the class of problems approximable within a polynomial factor). Indeed, on the one hand in [14] it has been shown that the L-reducibility is too strict and does not allow to reduce some problems which are known to be easy to approximate to problems which are known to be hard to approximate. On the other hand in this paper we show that it is too weak and is not approximation preserving (unless $\mathrm{P} = \mathrm{NP} \cap co\text{-}\mathrm{NP}$). The weakness of the L-reducibility is, essentially, shared by all reducibilities of Fig. 1 but the Strict reducibility and the E-reducibility, while the strictness of the L-reducibility is shared by all of them (unless $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$) but the PTAS-reducibility. The reducibility we propose is *a combination of the E-reducibility and of the PTAS-reducibility* and, as far as we know, it is the strictest reducibility that allows to obtain all approximation completeness results that have appeared in the literature, such as, for example, the APX-completeness of Maximum Satisfiability [14, 26] and the poly-APX-completeness of Maximum Clique [26].

The second group of results refers to the existence of natural complete problems for NPO. Indeed, both [33] and [12] provide examples of natural complete problems for the class of minimization and maximization NP problems, respectively. In Sect. 3 we will show *the existence of both maximization and minimization* NPO*-complete natural problems*. In particular, we prove that Maximum $0-1$ Programming and Minimum $0-1$ Programming are NPO-complete. This result shows that making use of a natural approximation preserving reducibility is enough powerful to encompass the "duality" problem raised in [27] (indeed, in [26] it was shown that this duality does not arise in APX, log-APX, poly-APX, and other subclasses of NPO). Moreover, the same result can also be obtained when restricting ourselves to the class NPO PB (i.e., the class of polynomially bounded NPO problems). In particular, we prove that Maximum PB $0-1$ Programming and Minimum PB $0-1$ Programming are NPO PB-complete.

The third group of results refers to the existence of natural APX-intermediate problems. In Sect. 4, we will prove that Minimum Bin Packing (and other natural NPO problems) cannot be APX-complete unless the polynomial-time hierarchy collapses. Since it is well-known [32] that this problem belongs to APX and that it does not belong to PTAS (that is, the class of NPO problems with polynomial-time approximation schemes) unless P=NP, our result yields *the first example of a natural* APX*-intermediate problem* (under a natural complexity-theoretic conjecture). Roughly speaking, the proof of our result is structured into two main steps. In the first step, we show that if Minimum Bin Packing were APX-complete then the problem of answering any set of k non-adaptive queries to an NP-complete problem could be reduced to the problem of approximating an instance of Minimum Bin Packing within a ratio depending on k. In the second step, we show that the problem of approximating an instance of Minimum Bin Packing within a given performance ratio can be solved in polynomial-time by means of a constant number of non-adaptive queries to an NP-complete problem. These two steps will imply the collapse of the query hierarchy which in turn implies the collapse of the polynomial-time hierarchy. As a side effect of our proof, we will show that *if a problem is* APX*-complete, then it does not admit an asymptotic approximation scheme.*

The previous results are consequences of new connections between the approximability properties and the query complexity of NP-hard optimization problems. In several recent papers the notion of query complexity (that is, the number of queries to an NP oracle needed to solve a given problem) has been shown to be a very useful tool for understanding the complexity of approximation problems. In [7, 9] upper and lower bounds have been proved on the number of queries needed to approximate certain optimization problems (such as MAXIMUM SATISFIABILITY and MAXIMUM CLIQUE): these results deal with the complexity of approximating the value of the optimum solution and not with the complexity of computing approximate solutions. In this paper, instead, the complexity of "constructive" approximation will be addressed by considering the languages that can be recognized by polynomial-time machines which have a function oracle that solves the approximation problem. In particular, after proving the existence of natural APX-intermediate problems, in Sect. 4.1 we will be able to solve an open question of [7] proving that *finding the vertices of the largest clique is more difficult than merely finding the vertices of a 2-approximate clique* unless the polynomial-time hierarchy collapses.

The results of [7, 9] show that the query complexity is a good measure to study approximability properties of optimization problems. The last group of our results show that completeness in approximation classes implies lower bounds on the query complexity. Indeed, in Sect. 5 we show that the two approaches are basically equivalent by giving *sufficient and necessary conditions for approximation completeness in terms of query-complexity hardness and combinatorial properties*. The importance of these results is twofold: they give new insights into the structure of complete problems for approximation classes and they reconcile the approach based on standard computation models with the approach based on the computation model for approximation proposed in [8]. As a final observation, our results can be seen as extensions of a result of [26] in which general sufficient (but not necessary) conditions for APX-completeness are proved.

1.1. Preliminaries

We assume the reader to be familiar with the basic concepts of computational complexity theory. For the definitions of most of the complexity classes used in the paper we refer the reader to one of the books on the subject (see, for example, [2, 5, 16, 35]).

We now give some standard definitions in the field of optimization and approximation theory.

Definition 1. *An* NP optimization problem A *is a fourtuple* $(I, sol, m, type)$ *such that*

1. *I is the set of the* instances *of A and it is recognizable in polynomial time.*
2. *Given an instance x of I, $sol(x)$ denotes the set of* feasible solutions *of x. These solutions are short, that is, a polynomial p exists such that, for any $y \in sol(x)$, $|y| \leq p(|x|)$. Moreover, for any x and for any y with $|y| \leq p(|x|)$, it is decidable in polynomial time whether $y \in sol(x)$.*
3. *Given an instance x and a feasible solution y of x, $m(x, y)$ denotes the positive integer* measure *of y (often also called the value of y). The function m is computable in polynomial time and is also called the* objective *function.*
4. *$type \in \{\max, \min\}$.*

The goal of an NP optimization problem with respect to an instance x is to find an *optimum solution*, that is, a feasible solution y such that $m(x, y) = type\{m(x, y') \ : \ y' \in sol(x)\}$. In the following opt will denote the function mapping an instance x to the measure of an optimum solution.

The *class* NPO is the set of all NP optimization problems. Max NPO is the set of maximization NPO problems and Min NPO is the set of minimization NPO problems.

An NPO problem is said to be *polynomially bounded* if a polynomial q exists such that, for any instance x and for any solution y of x, $m(x, y) \leq q(|x|)$. The *class* NPO PB is the set of all polynomially bounded NPO problems. Max PB is the set of all maximization problems in NPO PB and Min PB is the set of all minimization problems in NPO PB.

Definition 2. *Let A be an* NPO *problem. Given an instance x and a feasible solution y of x, we define the* performance ratio of y with respect to x *as*

$$R(x, y) = \max\left\{\frac{m(x, y)}{\mathsf{opt}(x)}, \frac{\mathsf{opt}(x)}{m(x, y)}\right\}$$

and the relative error of y with respect to x *as*

$$E(x, y) = \frac{|\mathsf{opt}(x) - m(x, y)|}{\mathsf{opt}(x)}.$$

The performance ratio (respectively, relative error) is always a number greater than or equal to 1 (respectively, 0) and is as close to 1 (respectively, 0) as the value of the feasible solution is close to the optimum value. It is easy to see that, for any instance x and for any feasible solution y of x,

$$E(x, y) \leq R(x, y) - 1.$$

Definition 3. *Let A be an* NPO *problem and let T be an algorithm that, for any instance x of A such that $sol(x) \neq \emptyset$, returns a feasible solution $T(x)$ in polynomial time. Given an arbitrary function $r : N \to [1, \infty)$, we say that T is an $r(n)$*-approximate algorithm for A *if the performance ratio of the feasible solution $T(x)$ with respect to x verifies the following inequality:*

$$R(x, T(x)) \leq r(|x|).$$

Definition 4. *Given a class of functions F, an* NPO *problem A belongs to the* class F-APX *if an $r(n)$-approximate algorithm T for A exists, for some function $r \in F$.*

In particular, APX, log-APX, poly-APX, and exp-APX will denote the classes F-APX with F equal to the set $O(1)$, to the set $O(\log n)$, to the set $O(n^{O(1)})$, and to the set $O(2^{n^{O(1)}})$, respectively. One could object that there is no difference between NPO and exp-APX since the polynomial bound on the computation time of the objective function implies that any NPO problem is $h2^{n^k}$-approximable for some h and k. This is not true, since NPO problems exist for which it is even hard to find a feasible solution. We will see examples of such problems in Sect. 3 (e.g. Maximum Weighted Satisfiability).

Definition 5. *An* NPO *problem A belongs to the* class PTAS *if an algorithm T exists such that, for any fixed rational $r > 1$, $T(\cdot, r)$ is an r-approximate algorithm for A.*

Clearly, the following inclusions hold:

$$\text{PTAS} \subseteq \text{APX} \subseteq \text{log-APX} \subseteq \text{poly-APX} \subseteq \text{exp-APX} \subseteq \text{NPO}.$$

It is also easy to see that these inclusions are strict if and only if $\text{P} \neq \text{NP}$.

1.2. A list of NPO problems

We here define the NP optimization problems that will be used in the paper. For a much larger list of NPO problems we refer to [11].

Maximum Clique

Instance: Graph $G = (V, E)$.
Solution: A clique in G, i.e. a subset $V' \subseteq V$ such that every two vertices in V' are joined by an edge in E.
Measure: Cardinality of the clique, i.e., $|V'|$.

Maximum Weighted Satisfiability **and** Minimum Weighted Satisfiability

Instance: Set of variables X, boolean quantifier-free first-order formula ϕ over the variables in X, and a weight function $w : X \to N$.
Solution: Truth assignment that satisfies ϕ.
Measure: The sum of the weights of the true variables.

Maximum PB $0-1$ Programming **and** Minimum PB $0-1$ Programming

Instance: Integer $m \times n$-matrix A, integer m-vector b, binary n-vector c.
Solution: A binary n-vector x such that $Ax \geq b$.
Measure: $1 + \sum_{i=1}^{n} c_i x_i$.

Maximum Satisfiability

Instance: Set of variables X and Boolean CNF formula ϕ over the variables in X.
Solution: Truth assignment to the variables in X.
Measure: The number of satisfied clauses.

Minimum Bin Packing

Instance: Finite set U of items, and a size $s(u) \in Q \cap (0, 1]$ for each $u \in U$.
Solution: A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$ such that the sum of the sizes of the items in each U_i is at most 1.
Measure: The number of used bins, i.e., the number m of disjoint sets.

Minimum Ordered Bin Packing

Instance: Finite set U of items, a size $s(u) \in Q \cap (0, 1]$ for each $u \in U$, and a partial order $\preceq$ on U.
Solution: A partition of U into disjoint sets $U_1, U_2, \ldots, U_m$ such that the sum of the sizes of the items in each U_i is at most 1 and if $u \in U_i$ and $u' \in U_j$ with $u \preceq u'$, then $i \leq j$.
Measure: The number of used bins, i.e., the number m of disjoint sets.

MINIMUM DEGREE SPANNING TREE

INSTANCE: Graph $G = (V, E)$.
SOLUTION: A spanning tree for G.
MEASURE: The maximum degree of the spanning tree.

MINIMUM EDGE COLORING

INSTANCE: Graph $G = (V, E)$.
SOLUTION: A coloring of E, i.e., a partition of E into disjoint sets $E_1, E_2, \ldots, E_k$ such that, for $1 \leq i \leq k$, no two edges in E_i share a common endpoint in G.
MEASURE: Cardinality of the coloring, i.e., the number k of disjoint sets.

2. A new approximation preserving reducibility

The goal of this section is to define a new approximation preserving reducibility that can be used for as many approximation classes as possible and such that all reductions that have appeared in the literature still hold. We will justify the definition of this new reducibility by emphasizing the disadvantages of previously known ones. In the following, we will assume that, for any reducibility, an instance x such that $sol(x) \neq \emptyset$ is mapped into an instance x' such that $sol(x') \neq \emptyset$.

2.1. The L-reducibility

The first reducibility we shall consider is the L-reducibility (for *linear* reducibility) [36] which is often most practical to use in order to show that a problem is at least as hard to approximate as another.

Definition 6. *Let A and B be two* NPO *problems. A is said to be* L-reducible *to B, in symbols $A \leq_{\mathrm{L}} B$, if two functions f and g and two positive constants α and β exist such that:*

1. *For any $x \in I_A$, $f(x) \in I_B$ is computable in polynomial time.*
2. *For any $x \in I_A$ and for any $y \in sol_B(f(x))$, $g(x, y) \in sol_A(x)$ is computable in polynomial time.*
3. *For any $x \in I_A$, $\mathsf{opt}_B(f(x)) \leq \alpha \mathsf{opt}_A(x)$.*
4. *For any $x \in I_A$ and for any $y \in sol_B(f(x))$,*

$$|\mathsf{opt}_A(x) - m_A(x, g(x, y))| \leq \beta |\mathsf{opt}_B(f(x)) - m_B(f(x), y)|$$

The fourtuple (f, g, α, β) is said to be an L-reduction from A to B.

Clearly, the L-reducibility preserves membership in PTAS. Indeed, if (f, g, α, β) is an L-reduction from A to B then, for any $x \in I_A$ and for any $y \in sol_B(f(x))$, we have that

$$E_A(x, g(x, y)) \leq \alpha\beta E_B(f(x), y),$$

so that if $B \in$ PTAS then $A \in$ PTAS [36]. The above inequality also implies that if A is a minimization problem and an r-approximate algorithm for B exists, then a $(1 + \alpha\beta(r - 1))$-approximate algorithm for A exists. In other words, L-reductions from minimization problems to optimization problems preserve membership in APX. The next result gives a strong evidence that, in general, this is not true whenever the starting problem is a maximization one.

Theorem 1. *The following statements are equivalent:*

1. *Two problems* $A \in$ Max NPO *and* $B \in$ Min NPO *exist such that* $A \notin$ APX, $B \in$ APX, *and* $A \leq_{\mathrm{L}} B$.
2. *Two* Max NPO *problems* A *and* B *exist such that* $A \notin$ APX, $B \in$ APX, *and* $A \leq_{\mathrm{L}} B$.
3. *A polynomial-time recognizable set of satisfiable Boolean formulas exists for which no polynomial-time algorithm can compute a satisfying assignment for each of them.*

Proof. (1) $\Rightarrow$ (2). In this case, it suffices to L-reduce B to a maximization problem C in APX [26].

(2) $\Rightarrow$ (3). Assume that for any polynomial-time recognizable set of satisfiable Boolean formulas there is a polynomial-time algorithm computing a satisfying assignment for each formula in the set. Suppose that (f, g, α, β) is an L-reduction from a maximization problem A to a maximization problem B and that B is r-approximable for some $r > 1$. Let x be an instance of A and let y be a solution of $f(x)$ such that $\mathsf{opt}_B(f(x))/m_B(f(x), y) \leq r$. For the sake of convenience, let $\mathsf{opt}_A = \mathsf{opt}_A(x)$, $m_A = m_A(x, g(x, y))$, $\mathsf{opt}_B = \mathsf{opt}_B(f(x))$, and $m_B = m_B(f(x), y)$. Let also $m_x = \max\{m_A, m_B/\alpha\}$. Since $m_A \leq \mathsf{opt}_A$ and $m_B/\alpha \leq \mathsf{opt}_B/\alpha \leq \mathsf{opt}_A$, we have that $m_x \leq \mathsf{opt}_A$. We now show that $\mathsf{opt}_A/m_x \leq 1 + \alpha\beta(r-1)$, that is, m_x is a non-constructive approximation of opt_A. Let $\gamma = \frac{\alpha r}{1+\alpha\beta(r-1)}$. There are two cases.

1. $\mathsf{opt}_B \leq \gamma \mathsf{opt}_A$. By the definition of the L-reducibility, $\mathsf{opt}_A - m_A \leq \beta(\mathsf{opt}_B - m_B)$. Since $\mathsf{opt}_B \leq \gamma \mathsf{opt}_A$ and $\mathsf{opt}_B/m_B \leq r$, we have that
$$\frac{\mathsf{opt}_A - m_A}{\mathsf{opt}_A} \leq \gamma\beta \frac{\mathsf{opt}_B - m_B}{\mathsf{opt}_B} \leq \gamma\beta(1 - 1/r).$$
Hence,
$$\frac{\mathsf{opt}_A}{m_x} \leq \frac{\mathsf{opt}_A}{m_A} \leq \frac{1}{1 - \gamma\beta\frac{r-1}{r}} = 1 + \alpha\beta(r-1)$$
where the last equality is due to the definition of γ.
2. $\mathsf{opt}_B > \gamma \mathsf{opt}_A$. It holds that
$$\begin{aligned} \frac{\mathsf{opt}_A}{m_x} &\leq \frac{\mathsf{opt}_A}{m_B/\alpha} \\ &< \frac{\alpha(\mathsf{opt}_B/\gamma)}{m_B} \text{ (since } \mathsf{opt}_A < \mathsf{opt}_B/\gamma) \\ &\leq \frac{\alpha(\mathsf{opt}_B/\gamma)}{(\mathsf{opt}_B/r)} \text{ (since } m_B \geq \mathsf{opt}_B/r) \\ &= \frac{\alpha r}{\gamma} \\ &= 1 + \alpha\beta(r-1). \end{aligned}$$

Let us now consider the following non-deterministic polynomial-time algorithm.

begin {input: $x \in I_A$}
  compute m_x by using the r-approximate algorithm for B and the L-reduction from A to B;
  guess $y \in sol_A(x)$;
  if $m_A(x, y) \geq m_x$ **then** accept **else** reject;
end;

By applying Cook's reduction [10] to the above algorithm, it easily follows that, for any $x \in I_A$, a satisfiable Boolean formula ϕ_x can be constructed in polynomial time in the length of x so that any satisfying assignment for ϕ_x encodes a solution of x whose measure is at least m_x. Moreover, the set $\{\phi_x : x \in I_A\}$ is recognizable in polynomial time. By assumption, it is then possible to compute in polynomial time a satisfying assignment for ϕ_x and thus an approximate solution for x.

(3) $\Rightarrow$ (1). Assume that a polynomial-time recognizable set S of satisfiable Boolean formulas exists for which no polynomial-time algorithm can compute a satisfying assignment for each of them. Consider the following two NPO problems $A = (I_A, sol_A, m_A, \max)$ and $B = (I_B, sol_B, m_B, \min)$ where $I_A = I_B = S$, $sol_A(x) = sol_B(x) = \{y : y$ is a truth assignment to the variables of $x\}$,

$$m_A(x,y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 1 & \text{otherwise,} \end{cases}$$

and

$$m_B(x,y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 2|x| & \text{otherwise.} \end{cases}$$

Clearly, problem B is in APX, while if A is in APX then there is a polynomial-time algorithm that computes a satisfying assignment for each formula in S, contradicting the assumption. Moreover, it is easy to see that A L-reduces to B via $f \equiv \lambda x.x$, $g \equiv \lambda x \lambda y.y$, $\alpha = 1$, and $\beta = 1$. □

Observe that in [30] it is shown that the third statement of the above theorem holds if and only if the γ-reducibility is different from the many-one reducibility. Moreover, in [19] it is shown that the latter hypothesis is somewhat intermediate between P $\neq$ NP $\cap$ *co*-NP and P $\neq$ NP. In other words, there is strong evidence that, even though the L-reducibility is suitable for proving completeness results within classes contained in APX (such as Max SNP [36]), this reducibility cannot be used to define the notion of completeness for classes beyond APX. Moreover, it cannot be blindly used to obtain positive results, that is, to prove the existence of approximation algorithms via reductions. Finally, it is possible to L-reduce the maximization problem B defined in the last part of the proof of the previous theorem to MAXIMUM 3-SATISFIABILITY: this implies that the closure of Max SNP with respect to the L-reducibility is not included in APX, contrary to what is commonly believed (e.g. see [35], page 314).

2.2. The E-reducibility

The drawbacks of the L-reducibility are mainly due to the fact that the relation between the performance ratios is set by two separate linear constraints on both the optimum values and the absolute errors. The E-reducibility (for *error* reducibility) [26], instead, imposes a linear relation directly between the performance ratios.

Definition 7. *Let A and B be two* NPO *problems. A is said to be* E-reducible *to B, in symbols $A \leq_{\mathrm{E}} B$, if two functions f and g and a positive constant α exist such that:*

1. *For any $x \in I_A$, $f(x) \in I_B$ is computable in polynomial time.*

2. For any $x \in I_A$ and for any $y \in sol_B(f(x))$, $g(x,y) \in sol_A(x)$ is computable in polynomial time.
3. For any $x \in I_A$ and for any $y \in sol_B(f(x))$,

$$R_A(x, g(x,y)) \leq 1 + \alpha(R_B(f(x), y) - 1).$$

The triple (f, g, α) is said to be an E-reduction from A to B.

Observe that, for any function r, an E-reduction maps $r(n)$-approximate solutions into $(1+\alpha(r(n^h)-1))$-approximate solutions where h is a constant depending only on the reduction. Hence, the E-reducibility not only preserves membership in PTAS but also membership in exp-APX, poly-APX, log-APX, and APX. As a consequence of this observation and of the results of the previous section, we have that NPO problems should exist which are L-reducible to each other but not E-reducible. However, the following result shows that within the class APX the E-reducibility is just a generalization of the L-reducibility.

Proposition 1. *For any two* NPO *problems A and B, if $A \leq_{\mathrm{L}} B$ and $A \in$* APX*, then $A \leq_{\mathrm{E}} B$.*

Proof. Let T be an r-approximate algorithm for A with r constant and let $(f_L, g_L, \alpha_L, \beta_L)$ be an L-reduction from A to B. Then, for any $x \in I_A$ and for any $y \in sol_B(f_L(x))$, $E_A(x, g_L(x,y)) \leq \alpha_L \beta_L E_B(f_L(x), y)$. If A is a minimization problem then, for any $x \in I_A$ and for any $y \in sol_B(f_L(x))$,

$$\begin{aligned} R_A(x, g_L(x,y)) &= 1 + E_A(x, g_L(x,y)) \\ &\leq 1 + \alpha_L \beta_L E_B(f_L(x), y) \\ &\leq 1 + \alpha_L \beta_L (R_B(f_L(x), y) - 1), \end{aligned}$$

and thus $(f_L, g_L, \alpha_L\beta_L)$ is an E-reduction from A to B. Otherwise (that is, A is a maximization problem) we distinguish the following two cases.

1. $E_B(f_L(x), y) \leq \frac{1}{2\alpha_L\beta_L}$: in this case we have that

$$\begin{aligned} R_A(x, g_L(x,y)) - 1 &= \frac{E_A(x, g_L(x,y))}{1 - E_A(x, g_L(x,y))} \\ &\leq \frac{\alpha_L\beta_L E_B(f_L(x), y)}{1 - \alpha_L\beta_L E_B(f_L(x), y)} \\ &\leq 2\alpha_L\beta_L(R_B(f_L(x), y) - 1). \end{aligned}$$

2. $E_B(f_L(x), y) > \frac{1}{2\alpha_L\beta_L}$: in this case we have that $R_B(f_L(x), y) - 1 \geq \frac{1}{2\alpha_L\beta_L}$ so that

$$R_A(x, T(x)) - 1 \leq r - 1 \leq 2\alpha_L\beta_L(r-1)(R_B(f_L(x), y) - 1)$$

where the first inequality is due to the fact that T is an r-approximation algorithm for A.

We can thus define a triple (f_E, g_E, α_E) as follows:

1. For any $x \in I_A$, $f_E(x) = f_L(x)$.
2. For any $x \in I_A$ and for any $y \in sol_B(f_E(x))$,

$$g_E(x,y) = \begin{cases} g_L(x,y) & \text{if } m_A(x, g_L(x,y)) \geq m_A(x, T(x)), \\ T(x) & \text{otherwise.} \end{cases}$$

3. $\alpha_E = \max\{2\alpha_L\beta_L, 2\alpha_L\beta_L(r-1)\}$.

From the above discussion it follows that (f_E, g_E, α_E) is an E-reduction from A to B. □

Clearly, the converse of the above result does not hold since no problem in NPO − NPO PB can be L-reduced to a problem in NPO PB while any problem in PO can be E-reduced to any NPO problem. Moreover, in [26] it is shown that MAXIMUM 3-SATISFIABILITY is (NPO PB ∩ APX)-complete with respect to the E-reducibility. This result is not obtainable by means of the L-reducibility: indeed, it is easy to prove that MINIMUM BIN PACKING is not L-reducible to MAXIMUM 3-SATISFIABILITY unless P = NP (see, for example, [6]).

The E-reducibility is still somewhat too strict. Indeed, in [14] it has been shown that natural PTAS problems exist, such as MAXIMUM KNAPSACK, which are not E-reducible to polynomially bounded APX problems, such as MAXIMUM 3-SATISFIABILITY (unless a logarithmic number of queries to an NP oracle is as powerful as a polynomial number of queries).

2.3. The AP-reducibility

The above mentioned drawback of the E-reducibility is mainly due to the fact that an E-reduction preserves optimum values (see [14]). Indeed, the linear relation between the performance ratios seems to be too restrictive. According to the definition of approximation preserving reducibilities given in [12], we could overcome this problem by expressing this relation by means of an implication. However, this is not sufficient: intuitively, since the function g does not know which approximation is required, it must still map optimum solutions into optimum solutions. The final step thus consists of letting the functions f and g depend on the performance ratio[1]. This implies that different constraints have to be put on the computation time of f and g: on the one hand, we still want to preserve membership in PTAS, on the other we want the reduction to be efficient even when poor performance ratios are required. These constraints are formally imposed in the following definition of *approximation preserving* reducibility (which is a restriction of the PTAS-reducibility introduced in [14]).

Definition 8. *Let A and B be two* NPO *problems. A is said to be* AP-reducible *to B, in symbols $A \leq_{\mathrm{AP}} B$, if two functions f and g and a positive constant α exist such that:*

1. *For any $x \in I_A$ and for any $r > 1$, $f(x,r) \in I_B$ is computable in time $t_f(|x|, r)$.*
2. *For any $x \in I_A$, for any $r > 1$, and for any $y \in sol_B(f(x,r))$, $g(x,y,r) \in sol_A(x)$ is computable in time $t_g(|x|, |y|, r)$.*
3. *For any fixed r, both $t_f(\cdot, r)$ and $t_g(\cdot,\cdot,r)$ are bounded by a polynomial.*
4. *For any fixed n, both $t_f(n,\cdot)$ and $t_g(n,n,\cdot)$ are non-increasing functions.*
5. *For any $x \in I_A$, for any $r > 1$, and for any $y \in sol_B(f(x,r))$,*

$$R_B(f(x,r),y) \leq r \textit{ implies } R_A(x, g(x,y,r)) \leq 1 + \alpha(r-1).$$

The triple (f, g, α) is said to be an AP-reduction from A to B.

According to the above definition, functions like $2^{1/(r-1)}n^h$ or $n^{1/(r-1)}$ are admissible bounds on the computation time of f and g, while this is not true for functions like n^r or 2^n.

[1] We also let the function f depend on the performance ratio because this feature will turn out to be useful in order to prove interesting characterizations of complete problems for approximation classes.

Observe that, clearly, the AP-reducibility is a generalization of the E-reducibility. Moreover, it is easy to see that, contrary to the E-reducibility, any PTAS problem is AP-reducible to any NPO problem.

As far as we know, this reducibility is the strictest one appearing in the literature that allows to obtain natural APX-completeness results (for instance, the APX-completeness of Maximum Satisfiability [14, 26]).

3. NPO-complete problems

We will in this section prove that there are natural problems that are complete for the classes NPO and NPO PB. Previously, completeness results have been obtained just for Max NPO, Min NPO, Max PB, and Min PB [12, 33, 4, 24]. One example of such a result is the following theorem.

Theorem 2. Minimum Weighted Satisfiability *is* Min NPO*-complete and* Maximum Weighted Satisfiability *is* Max NPO*-complete, even if only a subset* $\{v_1, \ldots, v_s\}$ *of the variables has nonzero weight* $w(v_i) = 2^{s-i}$ *and any truth assignment satisfying the instance gives the value true to at least one* v_i.

We will construct AP-reductions from maximization problems to minimization problems and vice versa. Using these reductions we will show that a problem that is Max NPO-complete or Min NPO-complete in fact is complete for the whole of NPO, and that a problem that is Max PB-complete or Min PB-complete is complete for the whole of NPO PB.

Theorem 3. Minimum Weighted Satisfiability *and* Maximum Weighted Satisfiability *are* NPO*-complete.*

Proof. In order to establish the NPO-completeness of Minimum Weighted Satisfiability we just have to show that there is an AP-reduction from a Max NPO-complete problem to Minimum Weighted Satisfiability. As the Max NPO-complete problem we will use the restricted version of Maximum Weighted Satisfiability from Theorem 2.

Let x be an instance of Maximum Weighted Satisfiability, i.e. a formula ϕ over variables $v_1, \ldots, v_s$ with weights $w(v_i) = 2^{s-i}$ and some variables with weight zero. We will first give a simple reduction that preserves the approximability within the factor 2, and then adjust it to obtain an AP-reduction.

Let $f(x)$ be the formula $\phi \wedge \alpha_1 \wedge \cdots \wedge \alpha_s$ where α_i is the conjunctive normal form of $(z_i \equiv (\overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i))$, where $z_1, \ldots, z_s$ are new variables with weights $w(z_i) = 2^i$ and where all other variables (even the v-variables) have zero weight. If y is a satisfying assignment of $f(x)$, let $g(x, y)$ be the restriction of the assignment to the variables that occur in ϕ. This assignment clearly satisfies ϕ.

Note that exactly one of the z-variables is true in any satisfying assignment of $f(x)$. Indeed, if all z-variables were false, then all v-variables would be false and ϕ would not be satisfied. On the other hand, if both z_i and z_j were true with $j > i$, then v_i would be both true and false which is a contradiction. Hence,

$$\begin{aligned} m(f(x), y) = 2^i \quad &\Leftrightarrow \quad z_i = 1 \\ &\Leftrightarrow \quad v_1 = v_2 = \cdots = v_{i-1} = 0, v_i = 1 \\ &\Leftrightarrow \quad 2^{s-i} \leq m(x, g(x, y)) < 2 \cdot 2^{s-i} \end{aligned}$$

$$\Leftrightarrow \quad \frac{2^s}{m(f(x),y)} \leq m(x,g(x,y)) < 2\frac{2^s}{m(f(x),y)}.$$

In particular this holds for the optimum solution. Thus the performance ratio for Maximum Weighted Satisfiability is

$$R(x,g(x,y)) = \frac{\mathsf{opt}(x)}{m(x,g(x,y))} < \frac{2\dfrac{2^s}{\mathsf{opt}(f(x))}}{\dfrac{2^s}{m(f(x),y)}} = 2\frac{m(f(x),y)}{\mathsf{opt}(f(x))} = 2R(f(x),y),$$

which means that the reduction preserves the approximability within 2.

Let us now extend the construction in order to obtain $R(x,g(x,y)) \leq (1+2^{-k})R(f_k(x),y)$ for every nonnegative integer k. The reduction described above corresponds to $k=0$.

For any $i \in \{1,\ldots,s\}$ and for any $(b_1,\ldots,b_{k(i)}) \in \{0,1\}^{k(i)}$ where $k(i) = \min\{s-i,k\}$, we have a variable $z_{i,b_1,\ldots,b_{k(i)}}$. Let

$$f_k(x) = \phi \wedge \bigwedge_{\substack{i\in\{1,\ldots,s\} \\ (b_1,\ldots,b_{k(i)})\in\{0,1\}^{k(i)}}} \alpha_{i,b_1,\ldots,b_{k(i)}},$$

where $\alpha_{i,b_1,\ldots,b_{k(i)}}$ is the conjuctive normal form of

$$\left(z_{i,b_1,\ldots,b_{k(i)}} \equiv \left(\overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i \wedge (v_{i+1}=b_1) \wedge \cdots \wedge (v_{i+k(i)} = b_{k(i)})\right)\right).$$

Define $g(x,y)$ as above. Finally, define

$$w(z_{i,b_1,\ldots,b_{k(i)}}) = \left\lceil \frac{K\cdot 2^s}{w(v_i)+\sum_{j=1}^{k(i)} b_j w(v_{i+j})}\right\rceil = \left\lceil \frac{K\cdot 2^i}{1+\sum_{j=1}^{k(i)} b_j 2^{-j}}\right\rceil$$

(by choosing K greater than 2^k we can disregard the effect of the ceiling operation in the following computations).

As in the previous reduction exactly one of the z-variables is true in any satisfying assignment of $f_k(x)$. If, in a solution y of $f_k(x)$, $z_{i,b_1,\ldots,b_{k(i)}} = 1$, then we have $m(f_k(x),y) = w(z_{i,b_1,\ldots,b_{k(i)}})$ and we know that

$$m(x,g(x,y)) \geq w(v_i) + \sum_{j=1}^{k(i)} b_j w(v_{i+j}) = 2^{s-i}(1+\sum_{j=1}^{k(i)} b_j 2^{-j}).$$

On the other hand, if $k(i) = s-i$ then $m(x,g(x,y)) = 2^{s-i}(1+\sum_{j=1}^{k(i)} b_j 2^{-j})$, otherwise

$$m(x,g(x,y)) \leq w(v_i) + \sum_{j=1}^{k} b_j w(v_{i+j}) + \sum_{j=k+i+1}^{s} w(v_j) < 2^{s-i}(1+\sum_{j=1}^{k} b_j 2^{-j})(1+2^{-k}).$$

In both cases, we thus get

$$\frac{K\cdot 2^s}{m(f_k(x),y)} \leq m(x,g(x,y)) < \frac{K\cdot 2^s}{m(f_k(x),y)}(1+2^{-k})$$

and therefore $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y)$. Given any $r > 1$, if we choose k such that $2^{-k} \le (r-1)/r$, e.g. $k = \lceil \log r - \log(r-1) \rceil$, then $R(f_k(x), y) \le r$ implies $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y) \le r + r2^{-k} \le r + r - 1 = 1 + 2(r-1)$. This is obviously an AP-reduction with $\alpha = 2$.

A very similar proof can be used to show that MAXIMUM WEIGHTED SATISFIABILITY is NPO-complete. □

Corollary 1. *Any* Min NPO*-complete problem is* NPO*-complete and any* Max NPO*-complete problem is* NPO*-complete.*

As an application of the above corollary, we have that the MINIMUM $0-1$ PROGRAMMING problem is NPO-complete.

We can also show that there are natural complete problems for the class of polynomially bounded NPO problems.

Theorem 4. MAXIMUM PB $0-1$ PROGRAMMING *and* MINIMUM PB $0-1$ PROGRAMMING *are* NPO PB*-complete.*

Proof. MAXIMUM PB $0-1$ PROGRAMMING is known to be Max PB-complete [4] and MINIMUM PB $0-1$ PROGRAMMING is known to be Min PB-complete [24]. Thus we just have to show that there are AP-reductions from MINIMUM PB $0-1$ PROGRAMMING to MAXIMUM PB $0-1$ PROGRAMMING and from MAXIMUM PB $0-1$ PROGRAMMING to MINIMUM PB $0-1$ PROGRAMMING.

Both reductions use exactly the same construction. Given a satisfying variable assignment, we define the *one-variables* to be the variables occurring in the objective function that have the value one. The objective value is the number of one-variables plus 1.

The objective value of a solution is encoded by introducing an order of the one-variables. The order is encoded by a squared number of $0-1$ variables, see Fig. 2. The idea is to invert the objective values, so that a solution without one-variables corresponds to an objective value of n of the constructed problem, and, in general, a solution with p one-variables corresponds to an objective value of $\left\lfloor \frac{n}{p+1} \right\rfloor$.

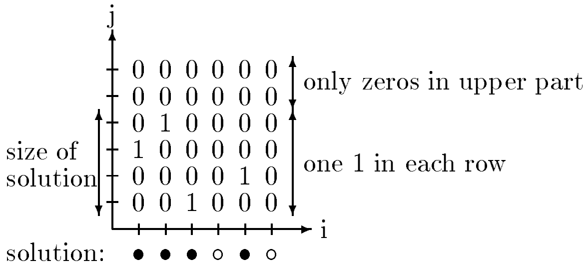

Figure 2. The idea of the reduction from MINIMUM/MAXIMUM PB $0-1$ PROGRAMMING to MAXIMUM/MINIMUM PB $0-1$ PROGRAMMING. The variable $x_i^j = 1$ if and only if v_i is the jth one-variable in the solution. There is at most one 1 in each column and in each row.

The reductions are constructed as follows. Given an instance of MINIMUM PB $0-1$ PROGRAMMING or MAXIMUM PB $0-1$ PROGRAMMING, i.e. an objective function $1 + \sum_{k=1}^{m} v_k$ and some inequalities over variables $V = \{v_1, \ldots, v_m\} \cup U$, construct m^2 variables x_i^j, $1 \le i, j \le m$, and the following inequalities:

$$\forall i \in [1..m] \qquad \sum_{j=1}^{m} x_i^j \leq 1 \qquad \text{(at most one 1 in each column)} \tag{1}$$

$$\forall j \in [1..m] \qquad \sum_{i=1}^{m} x_i^j \leq 1 \qquad \text{(at most one 1 in each row)} \tag{2}$$

$$\forall j \in [1..m-1] \quad \sum_{i=1}^{m} x_i^j - \sum_{i=1}^{m} x_i^{j+1} \geq 0 \quad \text{(only zeros in upper part)} \tag{3}$$

Besides these inequalities we include all inequalities from the original problem, but we substitute each variable v_i with the sum $\sum_{k=1}^{m} x_i^k$. The variables in U (that do not occur in the objective function) are left intact.

The objective function is defined as

$$n - \sum_{p=1}^{m} \left(\left\lfloor \frac{n}{p} \right\rfloor - \left\lfloor \frac{n}{p+1} \right\rfloor \right) \sum_{i=1}^{m} x_i^p. \tag{4}$$

In order to express the objective function with only binary coefficients we have to introduce n new variables $y_1, \ldots, y_n$ where $y_j = 1 - \sum_{i=1}^{m} x_i^p$ for $\lfloor n/(p+1) \rfloor < j \leq \lfloor n/p \rfloor$ and $y_j = 1$ for $j \leq \lfloor n/(m+1) \rfloor$. The objective function then is $\sum_{j=1}^{n} y_j$. One can now verify that a solution of the original problem instance with s one-variables (i.e. with an objective value of $s+1$) will exactly correspond to a solution of the constructed problem instance with objective value $\lfloor n/(s+1) \rfloor$ and vice versa.

Suppose that the optimum solution to the original problem instance has M one-variables, then the performance ratio $(s+1)/(M+1)$ will correspond to the performance ratio

$$\frac{\left\lfloor \dfrac{n}{M+1} \right\rfloor}{\left\lfloor \dfrac{n}{s+1} \right\rfloor} = \frac{s+1}{M+1} \left(1 \pm \frac{m}{n} \right)$$

for the constructed problem, where $\frac{m}{n}$ is the relative error due to the floor operation. By choosing n large enough the relative error can be made arbitrarily small. Thus it is easy to see that the reduction is an AP-reduction. $\square$

Corollary 2. *Any* Min PB-*complete problem is* NPO PB-*complete and any* Max PB-*complete problem is* NPO PB-*complete.*

4. Query complexity and APX-intermediate problems

The existence of APX-intermediate problems (that is, problems in APX which are not APX-complete) has already been shown in [12] where an artificial such problem is obtained by diagonalization techniques similar to those developed to prove the existence of NP-intermediate problems [29]. In this section, we prove that "natural" APX-intermediate problems exist: for instance, we will show that Minimum Bin Packing is APX-intermediate. In order to prove this result, we will establish new connections between the approximability properties and the query complexity of NP-hard optimization problems. To this aim, let us first recall the following definition.

Definition 9. *A language L belongs to the* class $\mathrm{P}^{\mathrm{NP}[f(n)]}$ *if it is decidable by a polynomial-time oracle Turing machine which asks at most $f(n)$ queries to an* NP-*complete oracle, where n is the input size. The* class QH *is equal to the union* $\bigcup_{k>1} \mathrm{P}^{\mathrm{NP}[k]}$.

Similarly, we can define the class of functions $\mathrm{FP}^{\mathrm{NP}[f(n)]}$ [28]. The following result has been proved in [21, 22].

Theorem 5. *If a constant k exists such that*

$$\mathrm{QH} = \mathrm{P}^{\mathrm{NP}[k]},$$

then the polynomial-time hierarchy collapses.

The query-complexity of the "non-constructive" approximation of several NP-hard optimization problems has been studied by using hardness results with respect to classes of functions $\mathrm{FP}^{\mathrm{NP}[\cdot]}$ [7, 9]. However, this approach cannot be applied to analyze the complexity of "constructing" approximate solutions. To overcome this limitation, we use a novel approach that basically consists of considering how helpful is an approximation algorithm for a given optimization problem to solve decision problems.

Definition 10. *Given an* NPO *problem A and a rational $r \geq 1$, A_r is a multi-valued partial function that, given an instance x of A, returns the set of feasible solutions y of x such that $R(x, y) \leq r$.*

Definition 11. *Given an* NPO *problem A and a rational $r \geq 1$, a language L belongs to P^{A_r} if two polynomial-time computable functions f and g exist such that, for any x, $f(x)$ is an instance of A with $sol(f(x)) \neq \emptyset$, and, for any $y \in A_r(f(x))$, $g(x, y) = 1$ if and only if $x \in L$. The* class $\mathrm{AQH}(A)$ *is equal to the union $\bigcup_{r>1} \mathrm{P}^{A_r}$.*

The following result states that an approximation problem does not help more than a constant number of queries to an NP-complete problem. It is worth observing that, in general, an approximate solution, even though not very helpful, requires more than a logarithmic number of queries to be computed [8].

Proposition 2. *For any problem A in* APX, $\mathrm{AQH}(A) \subseteq \mathrm{QH}$.

Proof. Assume that A is a maximization problem (the proof for minimization problems is similar). Let T be an r-approximate algorithm for A, for some $r > 1$, and let $L \in \mathrm{P}^{A_\rho}$ for some $\rho > 1$. Two polynomial-time computable functions f and g then exist witnessing this latter fact. For any x, let $m = m(f(x), T(f(x)))$, so that $m \leq \mathsf{opt}(f(x)) \leq rm$. We can then partition the interval $[m, rm]$ into $\lfloor \log_\rho r \rfloor + 1$ subintervals

$$[m, \rho m), [\rho m, \rho^2 m), \ldots, [\rho^{\lfloor \log_\rho r \rfloor - 1} m, \rho^{\lfloor \log_\rho r \rfloor} m], [\rho^{\lfloor \log_\rho r \rfloor} m, rm],$$

and start looking for the subinterval containing the optimum value (a similar technique has been used in [7, 9]). This can clearly be done using $\lfloor \log_\rho r \rfloor + 1$ queries to an NP-complete oracle. One more query is sufficient to know whether a feasible solution y exists whose value lies in that interval and such that $g(x, y) = 1$. Since y is ρ-approximate, it follows that L can be decided using $\lfloor \log_\rho r \rfloor + 2$ queries, that is, $L \in \mathrm{QH}$. □

Recall that an NPO problem admits an *asymptotic polynomial-time approximation scheme* if an algorithm T exists such that, for any x and for any $r > 1$, $R(x, T(x, r)) \leq r + k/\mathsf{opt}(x)$ with k constant and the time complexity of $T(x, r)$ is polynomial with respect to $|x|$. The class of problems that admit an asymptotic polynomial-time approximation scheme is usually denoted by PTAS^∞. The following result shows that, for this class, the previous fact can be strengthened.

Proposition 3. *Let $A \in \mathrm{PTAS}^\infty$. Then a constant h exists such that* $\mathrm{AQH}(A) \subseteq \mathrm{P}^{\mathrm{NP}[h]}$.

Proof. Let A be a minimization problem in PTAS^∞ (the proof for maximization problem is very similar). By definition, a constant k and an algorithm T exist such that, for any instance x and for any rational $r > 1$,

$$m(x, T(x, r)) \leq r \cdot \mathsf{opt}(x) + k.$$

We will now prove that a constant h exists such that, for any $r > 1$, a function $l_r \in \mathrm{FP}^{\mathrm{NP}[h-1]}$ exists such that, for any instance x of the problem A,

$$\mathsf{opt}(x) \leq l_r(x) \leq r \cdot \mathsf{opt}(x).$$

Intuitively, functions l_r form a non-constructive approximation scheme that is computable by a constant number of queries to an NP-complete oracle. Given an instance x, we can check whether $sol(x) = \emptyset$ by means of a single query to an NP oracle, so that we can restrict ourselves to instances such that $sol(x) \neq \emptyset$ (and thus $\mathsf{opt}(x) \geq 1$). Note that, for these instances, $T(\cdot, 2)$ is a $(k+2)$-approximate algorithm for A. Let us fix an $r > 1$, let $\varepsilon = r - 1$, $y = T(x, 1 + \varepsilon/2)$ and $a = m(x, T(x, 2))$. We have to distinguish two cases.

1. $a \geq 2k(k+2)/\varepsilon$: in this case, $\mathsf{opt}(x) \geq 2k/\varepsilon$, that is, $\mathsf{opt}(x)\varepsilon/2 \geq k$. Then

$$\begin{aligned} m(x, y) &\leq \mathsf{opt}(x)(1 + \varepsilon/2) + k \\ &\leq \mathsf{opt}(x)(1 + \varepsilon/2) + \mathsf{opt}(x)\varepsilon/2 \\ &= \mathsf{opt}(x)(1 + \varepsilon) = r\mathsf{opt}(x), \end{aligned}$$

   that is, y is an r-approximate solution for x, and we can set $l_r(x) = m(x, y)$ (in this case l_r has been computed by only one query).
2. $a < 2k(k+2)/\varepsilon$: in this case, $\mathsf{opt}(x) < 2k(k+2)/\varepsilon$. Then,

$$\mathsf{opt}(x) \leq m(x, y) \leq \mathsf{opt}(x) + \mathsf{opt}(x)\varepsilon/2 + k < \mathsf{opt}(x) + k(k+2) + k.$$

   Clearly, $\lceil \log k(k+3) \rceil$ queries to NP are sufficient to find the optimum value $\mathsf{opt}(x)$ by means of a binary search technique: in this case $l_r(x) = \mathsf{opt}(x)$ has been computed by $\lceil \log k(k+3) \rceil + 1$ queries.

Let now L be a language in $\mathrm{AQH}(A)$, then $L \in \mathrm{P}^{A_r}$ for some $r > 1$. Let f and g be the functions witnessing that $L \in \mathrm{P}^{A_r}$. Observe that, for any x, $x \in L$ if and only if a solution y for $f(x)$ exists such that $m(f(x), y) \geq l_r(f(x))$ and $g(f(x), y) = 1$: that is, given $l_r(f(x))$, deciding whether $x \in L$ is an NP problem. Since $l_r(f(x))$ is computable by means of at most $\lceil \log k(k+3) \rceil + 1$ queries to NP, we have that $L \in \mathrm{P}^{\mathrm{NP}[h]}$ where $h = \lceil \log k(k+3) \rceil + 2$. □

The next proposition, instead, states that any language L in the query hierarchy can be decided using just one query to A_r where A is APX-complete and r depends on the level of the query hierarchy L belongs to. In order to prove this proposition, we need the following technical result[2].

[2] Recall that the NP-complete problem PARTITION is defined as follows: given a set U of items and a size function $s : U \to Q \cap (0, 1]$, does there exists a subset $U' \subseteq U$ such that

$$\sum_{u \in U'} s(u) = \sum_{u \notin U'} s(u)?$$

Lemma 1. *For any* APX*-complete problem A and for any k, two polynomial-time computable functions f and g and a constant r exist such that, for any k-tuple $(x_1, \ldots, x_k)$ of instances of* Partition*, $x = f(x_1, \ldots, x_k)$ is an instance of A and if y is a solution of x whose performance ratio is smaller than r then $g(x, y) = (b_1, \ldots, b_k)$ where $b_i \in \{0, 1\}$ and $b_i = 1$ if and only if x_i is a yes-instance.*

Proof. Let $x_i = (U_i, s_i)$ be an instance of Partition for $i = 1, \ldots, k$. Without loss of generality, we can assume that the U_is are pairwise disjoint and that, for any i, $\sum_{u \in U_i} s_i(u) = 2$. Let $w = (U, s, \preceq)$ be an instance of Minimum Ordered Bin Packing defined as follows (a similar construction has been used in [37]).

1. $U = \bigcup_{i=1}^{k} U_i \cup \{v_1, \ldots, v_{k-1}\}$ where the v_is are new items.
2. For any $u \in U_i$, $s(u) = s_i(u)$ and $s(v_i) = 1$ for $i = 1, \ldots, k-1$.
3. For any $i < j \leq k$, for any $u \in U_i$, and for any $u' \in U_j$, $u \preceq v_i \preceq u'$.

Any solution of w must be formed by a sequence of packings of $U_1, \ldots, U_k$ such that, for any i, the bins used for U_i are separated by the bins used for U_{i+1} by means of one bin which is completely filled by v_i. In particular, the packings of the U_is in any optimum solution must use either two or three bins: two bins are used if and only if x_i is a yes-instance. The optimum measure thus is at most $4k - 1$ so that any $(1 + 1/(4k))$-approximate solution is an optimum solution.

Since Minimum Ordered Bin Packing belongs to APX [41] and A is APX-complete, then an AP-reduction (f_1, g_1, α) exists from Minimum Ordered Bin Packing to A. We can then define $x = f(x_1, \ldots, x_k) = f_1(w, 1+1/(4\alpha k))$ and $r = 1+1/(4\alpha k)$. For any r-approximate solution y of x, the fourth property of the AP-reducibility implies that $z = g_1(x, y, 1+1/(4\alpha k))$ is a $(1 + 1/(4k))$-approximate solution of w and thus an optimum solution of w. From z, we can easily derive the right answers to the k queries $x_1, \ldots, x_k$. □

We are now able to prove the following result.

Proposition 4. *For any* APX*-complete problem A,* QH $\subseteq$ AQH(A).

Proof. Let $L \in$ QH, then $L \in \mathrm{P}^{\mathrm{NP}[h]}$ for some h. It is well known (see, for instance, [3]) that L can be reduced to the problem of answering $k = 2^{h-1}$ non-adaptive queries to NP. More formally, two polynomial-time computable functions t_1 and t_2 exist such that, for any x, $t_1(x) = (x_1, \ldots, x_k)$, where $x_1, \ldots, x_k$ are k instances of the Partition problem, and for any $(b_1, \ldots, b_k) \in \{0, 1\}^k$, $t_2(x, b_1, \ldots, b_k) \in \{0, 1\}$. Moreover, if, for any j, $b_j = 1$ if and only if x_j is a yes-instance, then $t_2(x, b_1, \ldots, b_k) = 1$ if and only if $x \in L$.

Let now f, g and r be the two functions and the constant of Lemma 1 applied to problem A and constant k. For any x, $x' = f(t_1(x))$ is an instance of A such that if y is an r-approximate solution for x', then $t_2(g(x', y)) = 1$ if and only if $x \in L$. Thus, $L \in \mathrm{P}^{A_r}$. □

By combining Propositions 2 and 4, we thus have the following theorem that characterizes the approximation query hierarchy of the hardest problems in APX.

Theorem 6. *For any* APX*-complete problem A,* AQH(A) = QH.

Finally, we have the following result that states the existence of natural intermediate problems in APX.

Theorem 7. *If the polynomial-time hierarchy does not collapse, then* Minimum Bin Packing, Minimum Degree Spanning Tree, *and* Minimum Edge Coloring *are* APX*-intermediate.*

Proof. From Proposition 3 and from the fact that MINIMUM BIN PACKING is in PTAS^{∞} [25], it follows that AQH(MINIMUM BIN PACKING) $\subseteq \mathrm{P}^{\mathrm{NP}[h]}$ for a given h. If MINIMUM BIN PACKING is APX-complete, then from Proposition 4 it follows that $\mathrm{QH} \subseteq \mathrm{P}^{\mathrm{NP}[h]}$. From Theorem 5 we thus have the collapse of the polynomial-time hierarchy. The proofs for MINIMUM DEGREE SPANNING TREE and MINIMUM EDGE COLORING are identical and use the results of [18, 15]. □

Observe that the previous result does not seem to be obtainable by using the hypothesis $\mathrm{P} \neq \mathrm{NP}$, as shown by the following theorem.

Theorem 8. *If* $\mathrm{NP} = co\text{-}\mathrm{NP}$ *then* MINIMUM BIN PACKING *is* APX-*complete.*

Proof. Assume $\mathrm{NP} = co\text{-}\mathrm{NP}$, we will present an AP reduction from MAXIMUM SATISFIABILITY to MINIMUM BIN PACKING. Since $\mathrm{NP} = co\text{-}\mathrm{NP}$ a nondeterministic polynomial time Turing machine M exists that, given in input an instance ϕ of MAXIMUM SATISFIABILITY, has an accepting computation and all accepting computations halt with an optimum solution for ϕ written on the tape. Indeed, M guesses an integer k, an assigment τ such that $m(\phi, \tau) = k$ and a proof of the fact that $\mathsf{opt}(\phi) \leq k$. From the proof of Cook's theorem it follows that, given ϕ, we can find in polynomial time a formula ϕ' such that ϕ' is satisfiable and that given any satisfying assignment for ϕ' we can find in polynomial time an optimum solution for ϕ. By combining this construction with the NP-completeness proof of the MINIMUM BIN PACKING problem, we obtain two polynomial-time computable functions t_1 and t_2 such that, for any instance ϕ of MAXIMUM SATISFIABILITY, $t_1(\phi) = x_\phi$ is an instance of MINIMUM BIN PACKING such that $\mathsf{opt}(x_\phi) = 2$ and, for any optimum solution y of x_ϕ, $t_2(x_\phi, y)$ is an optimum solution of ϕ. Observe that, by construction, an r-approximate solution for x_ϕ is indeed an optimum solution provided that $r < 3/2$. Let T be a 4/3-approximate algorithm for MAXIMUM SATISFIABILITY [42, 17]. The reduction from MAXIMUM SATISFIABILITY to MINIMUM BIN PACKING is defined as follows: $f(\phi, r) = t_1(\phi)$;

$$g(\phi, y, r) = \begin{cases} T(\phi) & \text{if } r \geq 4/3, \\ t_2(t_1(\phi), y) & \text{otherwise.} \end{cases}$$

It is immediate to verify that the above is an AP-reduction with $\alpha = 1$. □

Finally, note that the above result can be extended to any APX problem which is NP-hard to approximate within a given performance ratio.

4.1. A remark on MAXIMUM CLIQUE

The following lemma is the analogoue of Proposition 2 within NPO PB and can be proved similarly by binary search techniques.

Lemma 2. *For any* NPO PB *problem A and for any $r > 1$,* $\mathrm{P}^{A_r} \subseteq \mathrm{P}^{\mathrm{NP}[\log\log n + O(1)]}$.

From this lemma, from the fact that $\mathrm{P}^{\mathrm{NP}[\log n]}$ is contained in $\mathrm{P}^{\mathrm{MC}_1}$ where MC stands for MAXIMUM CLIQUE [28], and from the fact that if a constant k exists such that

$$\mathrm{P}^{\mathrm{NP}[\log\log n + k]} = \mathrm{P}^{\mathrm{NP}[\log n]},$$

then the polynomial-time hierarchy collapses [40], it follows the next result that solves an open question posed in [7]. Informally, this result states that it is not possible to reduce the problem of finding a maximum clique to the problem of finding a 2-approximate clique (unless the polynomial-time hierarchy collapses).

Theorem 9. *If* $\mathrm{P}^{\mathrm{MC}_1} \subseteq \mathrm{P}^{\mathrm{MC}_2}$ *then the polynomial-time hierarchy collapses.*

5. Query complexity and completeness in approximation classes

In this final section, we shall give a full characterization of problems complete for poly-APX and for APX, respectively, in terms of hardness of the corresponding approximation problems with respect to classes of partial multi-valued functions and in terms of suitably defined combinatorial properties.

The classes of functions we will refer to have been introduced in [8] as follows.

Definition 12. $\mathrm{FNP}^{\mathrm{NP}[q(n)]}$ *is the class of partial multi-valued functions computable by nondeterministic polynomial-time Turing machines which ask at most* $q(n)$ *queries to an* NP *oracle in the* entire *computation tree.*[3]

In order to talk about hardness with respect to these classes we will use the following reducibility which is an extension of both metric reducibility [28] and one-query reducibility [13] and has been introduced in [8].

Definition 13. *Let* F *and* G *be two partial multi-valued functions. We say that* F *many-one reduces to* G *(in symbols,* $F \leq_{\mathrm{mv}} G$*) if two polynomial-time algorithms* t_1 *and* t_2 *exist such that, for any* x *in the domain of* F*,* $t_1(x)$ *is in the domain of* G *and, for any* $y \in G(t_1(x))$*,* $t_2(x, y) \in F(x)$*.*

The combinatorial property used to characterize poly-APX-complete problems is the well-known self-improvability (see, for instance, [34]).

Definition 14. *A problem* A *is* self-improvable *if two algorithms* t_1 *and* t_2 *exist such that, for any instance* x *of* A *and for any two rationals* $r_1, r_2 > 1$*,* $x' = t_1(x, r_1, r_2)$ *is an instance of* A *and, for any* $y' \in A_{r_2}(x')$*,* $y = t_2(x, y', r_1, r_2) \in A_{r_1}(x)$*. Moreover, for any fixed* r_1 *and* r_2*, the running time of* t_1 *and* t_2 *is polynomial.*

We are now ready to state the first result of this section.

Theorem 10. *A poly-*APX *problem* A *is poly-*APX*-complete if and only if it is self-improvable and* A_{r_0} *is* $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$*-hard for some* $r_0 > 1$*.*

Proof. Let A be a poly-APX-complete problem. Since Maximum Clique is self-improvable [16] and poly-APX-complete [26] and since the equivalence with respect to the AP-reducibility preserves the self-improvability property (see [34]), we have that A is self-improvable. It is then sufficient to prove that A_2 is hard for $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$.

From the poly-APX-completeness of A we have that Maximum Clique $\leq_{\mathrm{AP}} A$: let α be the constant of this reduction. From Theorem 12 of [8] we have that any function F in $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$ many-one reduces to Maximum $\mathrm{Clique}_{1+\alpha}$. From the definition of AP-reducibility, we also have that Maximum $\mathrm{Clique}_{1+\alpha} \leq_{\mathrm{mv}} A_2$ so that F many-one reduces to A_2.

Conversely, let A be a poly-APX self-improvable problem such that, for some r_0, A_{r_0} is $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$-hard. We will show that, for any problem B in poly-APX, B is AP-reducible to A. To this aim, we introduce the following partial multi-valued function `multisat`: given in input a sequence $(\phi_1, \ldots, \phi_m)$ of instances of the satisfiability problem with $m \leq \log |(\phi_1, \ldots, \phi_m)|$ and such that, for any i, if ϕ_{i+1} is satisfiable then ϕ_i is satisfiable, a possible output is a satisfying truth-assignment for ϕ_{i^*} where $i^* = \max\{i : \phi_i \text{ is satisfiable}\}$. From the proof of Theorem 12 of [8] it follows that this function is $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$-complete.

[3] We say that a multi-valued partial function F is computable by a nondeterministic Turing machine N if, for any x in the domain of F, an halting computation path of $N(x)$ exists and any halting computation path of $N(x)$ outputs a value of $F(x)$.

By making use of techniques similar to those of the proof of Proposition 2, it is easy to see that, since B is in poly-APX, two algorithms t_1^B and t_2^B exist such that, for any fixed $r > 1$, $t_1^B(\cdot, r)$ and $t_2^B(\cdot, \cdot, r)$ form a many-one reduction from B_r to `multisat`. Moreover, since A_{r_0} is $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$-hard, then a many-one reduction (t_1^M, t_2^M) exists from `multisat` to A_{r_0}. Finally, let t_1^A and t_2^A be the functions witnessing the self-improvability of A.

The AP-reduction from B to A can then be derived as follows:

$$\begin{array}{ccccccc}
x, r & \xrightarrow{\quad t_1^B(x,r) \quad} & x' & \xrightarrow{\quad t_1^M(x') \quad} & x'' & \xrightarrow{\quad t_1^A(x'',r_0,r) \quad} & x''' \\
 & & & & & & \downarrow \\
y & \xleftarrow{\quad t_2^B(x,y',r) \quad} & y' & \xleftarrow{\quad t_2^M(x',y'') \quad} & y'' & \xleftarrow{\quad t_2^A(x'',y''',r_0,r) \quad} & y'''
\end{array}$$

It is easy to see that if y''' is an r-approximate solution for the instance x''' of A, then y is an r-approximate solution of the instance x of B. That is, B is AP-reducible to A with $\alpha = 1$. □

The above theorem cannot be proved without the dependency of both f and g on r in the definition of AP-reducibility. Indeed, it is possible to prove that if only g has this property then, unless the polynomial-time hierarchy collapses, a self-improvable problem A exists such that A_2 is $\mathrm{FNP}^{\mathrm{NP}[\log\log n + O(1)]}$-hard and A is not poly-APX-complete.

In order to characterize APX-complete problems, we have to define a different combinatorial property. Intuitively, this property states that it is possible to merge several instances into one instance in an approximation preserving fashion.

Definition 15. *An* NPO *problem A is* linearly additive *if a constant β and two algorithms t_1 and t_2 exist such that, for any rational $r > 1$ and for any sequence $x_1, \ldots, x_k$ of instances of A, $x' = t_1(x_1, \ldots, x_k, r)$ is an instance of A and, for any $y' \in A_{1+(r-1)\beta/k}(x')$, $t_2(x_1, \ldots, x_k, y', r) = y_1, \ldots, y_k$ where each y_i is an r-approximate solution of x_i. Moreover, the running time of t_1 and t_2 is polynomial for every fixed r.*

Theorem 11. *An* APX *problem A is* APX*-complete if and only if it is linearly additive and a constant r_0 exists such that A_{r_0} is* $\mathrm{FNP}^{\mathrm{NP}[1]}$*-hard.*

Proof. Let A be an r_A-approximable APX-complete problem. From the proof of Proposition 4 a constant r_0 exists such that A_{r_0} is hard for $\mathrm{FNP}^{\mathrm{NP}[1]}$. In order to prove the linear additivity, fix any $r > 1$ and let $x_1, \ldots, x_k$ be instances of A. Without loss of generality, we can assume $r < r_A$ (otherwise the k instances can be r-approximated by using the r_A-approximate algorithm). For any $i = 1, \ldots, k$ the problem of finding an r-approximate solution y_i for x_i is reducible to the problem of constructively solving a set of $\lceil \log_r r_A \rceil$ instances of Partition. Observe that $\lceil \log_r r_A \rceil \leq c/(r-1)$ for a certain constant c depending on r_A. Moreover, we claim that a constant γ exists such that constructively solving $kc/(r-1)$ instances of Partition is reducible to $(1 + \gamma(r-1)/kc)$-approximating a single instance of A (indeed, this can be shown along the lines of the proof of Proposition 4). That is, A is linearly additive with $\beta = \gamma/c$.

Conversely, let A be a linearly additive APX problem such that A_{r_0} is $\mathrm{FNP}^{\mathrm{NP}[1]}$-hard for some r_0 and let B be an r_B-approximable problem. Given an instance x of B, for any $r > 1$ we can reduce the problem of finding an r-approximate solution for x to the problem of constructively solving $c/(r-1)$ instances of Partition, for a proper constant c not depending on r. Each of these questions is reducible to A_{r_0}, since any NP problem can be constructively solved by an $\mathrm{FNP}^{\mathrm{NP}[1]}$ function. From linear additivity, it follows that r_0-approximating $c/(r-1)$ instances of A is reducible to $(1 + \beta(r_0 - 1)(r-1)/c)$-approximating a single instance of A. This is an AP-reduction from B to A with $\alpha = c/(\beta(r_0 - 1))$. □

Note that linear additivity plays for APX more or less the same role of self-improvability for poly-APX. These two properties are, in a certain sense, one the opposite of the other: while the usefulness of APX-complete approximation problems to solve decision problems depends on the performance ratio and does not depend on the size of the instance, the usefulness of poly-APX-complete approximation problems depends on the size of the instance and does not depend on the performance ratio. Indeed, it is possible to prove that no APX-complete problem can be self-improvable (unless P = NP) and that no poly-APX-complete problem can be linearly additive (unless the polynomial-time hierarchy collapses).

It is now an interesting question to find a characterizing combinatorial property of log-APX-complete problems. Indeed, we have not been able to establish this characterization: at present, we can only state that it cannot be based on the self-improvability property as shown by the following result.

Theorem 12. *No log-APX-complete problem can be self-improvable unless the polynomial-time hierarchy collapses.*

Proof. Let us consider the optimization problem MAX NUMBER OF SATISFIABLE FORMULAS (in short, MNSF) defined as follows.

INSTANCE: Set of m boolean formulas $\phi_1, \ldots, \phi_m$ in 3CNF, such that ϕ_1 is a tautology and $m \leq \log n$, where n is the size of the input instance.
SOLUTION: Truth assignment τ to the variables of $\phi_1, \ldots, \phi_m$.
MEASURE: The number of satisfied formulas, i.e., $|\{i : \phi_i \text{ is satisfied by } \tau\}|$.

Clearly, MNSF is in log-APX, since the measure of any assignment τ is at least 1, and the optimum value is always smaller than $\log n$, where n is the size of the input. We will show that, for any $r < 2$, MNSF_r is hard for $\text{FNP}^{\text{NP}[\log\log\log n-1]}$.

Given $\log\log n$ queries to an NP-complete language (of size polynomial in n) $x_1, \ldots, x_{\log\log n}$, we can construct an instance $\Phi = \phi_1, \ldots, \phi_m$ of MNSF where ϕ_1 is a tautology and, for $i \geq 1$, the formulas $\phi_{2^i} = \ldots = \phi_{2^{i+1}-1}$ are satisfiable if and only if at least i instances among $x_1, \ldots, x_{\log\log n}$ are yes-instances (these formulas can be easily constructed using the standard proof of Cook's theorem). Note that $m = 2^{\log\log n+1} - 1$ and, by adding dummy clauses to some formulas, we can achieve the bound $m \leq \log|\phi_1, \ldots, \phi_m|$. Moreover, from an r-approximate solution for Φ we can decide how many instances in $x_1, \ldots, x_{\log\log n}$ are yes-instances, and we can also recover solutions for such instances. That is, any function in $\text{FNP}^{\text{NP}[\log\log\log n-1]}$ is many-one reducible to MNSF_r.

Let A be a self-improvable log-APX-complete problem. Then, for any function $F \in \text{FNP}^{\text{NP}[\log\log\log n-1]}$, $F \leq_{\text{mv}} \text{MNSF}_{1.5} \leq_{\text{mv}} A_{1+\alpha/2} \leq_{\text{mv}} A_{2^{16}}$ where α is the constant in the AP-reduction from MNSF to A and where the last reduction is due to the self-improvability of A. Thus, for any x, computing $F(x)$ is reducible to finding a 2^{16}-approximate solution for an instance x' with $|x'| \leq |x|^c$ for a certain constant c. Since $A \in$ log-APX, it is possible to find in polynomial time a $(k \log|x'|)$-approximate solution y for x' where k is a constant. From y, by means of binary search techniques, we can find a 2^{16}-approximate solution for x' using $\lceil \log\lceil\log_{2^{16}}(k\log|x'|)\rceil\rceil \leq \log\lceil\log\log|x|^{kc}\rceil - 3 \leq \log\log\log|x| - 2$ adaptive queries to NP where the last inequality surely holds for sufficiently large $|x|$. Thus,

$$\text{FNP}^{\text{NP}[\log\log\log n-1]} \subseteq \text{FNP}^{\text{NP}[\log\log\log n-2]}$$

which implies the collapse of the polynomial-time hierarchy [40]. □

As a consequence of the above theorem and of the results of [26], we conjecture that the minimum set cover problem is not self-improvable.

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. (1992), "Proof verification and hardness of approximation problems", *Proc. of 33rd Ann. IEEE Symp. on Foundations of Comput. Sci.*, IEEE Computer Society, 14–23.

2. Balcázar, J.L., Díaz, J., and Gabarró, J. (1988), *Structural complexity I.* Springer-Verlag.

3. Beigel, R. (1991), "Bounded queries to SAT and the Boolean hierarchy", *Theoretical Computer Science* **84**, 199–223.

4. Berman, P., and Schnitger, G. (1992), "On the complexity of approximating the independent set problem", *Inform. and Comput.* **96**, 77–94.

5. Bovet, D.P., and Crescenzi, P. (1993), *Introduction to the theory of complexity.* Prentice Hall.

6. Cai, L. (1994), Ph.D. Dissertation, Department of Computer Science, Texas A&M University.

7. Chang, R. (1994), "On the query complexity of clique size and maximum satisfiability", *Proc. 9th Ann. Structure in Complexity Theory Conf.*, IEEE Computer Society, 31–42.

8. Chang, R. (1994), "A machine model for NP-approximation problems and the revenge of the Boolean hierarchy", *EATCS Bulletin* **54**, 166–182.

9. Chang, R., Gasarch, W.I., and Lund, C. (1994), "On bounded queries and approximation", Technical Report TR CS-94-05, Department of Computer Science, University of Maryland Baltimore County.

10. Cook, S.A. (1971), "The complexity of theorem proving procedures", *Proc. of ACM Symp. on Theory of Comp.*, 151–158.

11. Crescenzi, P., and Kann, V. (1995), "A compendium of NP optimization problems", Technical Report SI/RR-95/02, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza". The list is updated continuously. The latest version is available by anonymous ftp from `nada.kth.se` as `Theory/Viggo-Kann/compendium.ps.Z`

12. Crescenzi, P., and Panconesi, A. (1991), "Completeness in approximation classes", *Inform. and Comput.* **93**, 241–262.

13. Crescenzi P., and Silvestri, R. (1990), "Relative Complexity of Evaluating the Optimum Cost and Constructing the Optimum for Maximization Problems", *Information Processing Letters* **33**, 221–226

14. Crescenzi, P., and Trevisan, L. (1994), "On approximation scheme preserving reducibility and its applications", *Proc. 14th FSTTCS*, Lecture Notes in Comput. Sci. 880, Springer-Verlag, 330–341.

15. Fürer, M., and Raghavachari, B. (1992), "Approximating the minimum degree spanning tree to within one from the optimal degree", *Proc. Third Ann. ACM-SIAM Symp. on Discrete Algorithms*, ACM-SIAM, 317–324.

16. Garey, M.R., and Johnson, D.S. (1979) *Computers and intractability: a guide to the theory of NP-completeness.* Freeman, 1979.

17. Goemans, M.X., and Williamson, D.P. (1994), "New 3/4-approximation algorithms for the maximum satisfiability problem", *SIAM J. Discrete Mathematics* **7**, 656–666.

18. Holyer, I. (1981), "The NP-completeness of edge-coloring", *SIAM J. Computing* **10**, 718–720.

19. Impagliazzo, R., and Naor, M. (1988), "Decision trees and downward closures", *Proc. 3rd Structure in Complexity Theory Conference*, IEEE Computer Society, 29–38.

20. Johnson, D.S. (1974), "Approximation algorithms for combinatorial problems", *J. Comput. System Sci.* **9**, 256–278.

21. Kadin, J. (1988), "The polynomial time hierarchy collapses if the Boolean hierarchy collapses", *SIAM J. Computing* **17**, 1263–1282.

22. Kadin, J. (1990), "ERRATUM: The Polynomial Time Hierarchy Collapses if the Boolean Hierarchy Collapses", *SIAM J. Computing* **20**, 404.

23. Kann, V. (1992), *On the approximability of NP-complete optimization problems*, PhD thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm.

24. Kann, V. (1994), "Polynomially bounded minimization problems that are hard to approximate", *Nordic J. Computing* **1**, 317–331.

25. Karmarkar, N., and Karp, R. M. (1982), "An efficient approximation scheme for the one-dimensional bin packing problem", *Proc. of 23rd Ann. IEEE Symp. on Foundations of Comput. Sci.*, IEEE Computer Society, 312–320.

26. Khanna, S., Motwani, R., Sudan, M., and Vazirani, U. (1994), "On syntactic versus computational views of approximability", *Proc. of 35th Ann. IEEE Symp. on Foundations of Comput. Sci.*, IEEE Computer Society, 819–830.

27. Kolaitis, P. G., and Thakur, M. N. (1991), "Approximation properties of NP minimization classes", *Proc. Sixth Ann. Structure in Complexity Theory Conf.*, IEEE Computer Society, 353–366.

28. Krentel, M.W. (1988), "The complexity of optimization problems", *J. Comput. System Sci.* **36**, 490–509.

29. Ladner, R.E. (1975), "On the structure of polynomial-time reducibility", *J. ACM* **22**, 155–171.

30. Long, T.J. (1981), "On γ-reducibility versus polynomial time many-one reducibility", *Theoretical Computer Science* **14**, 91–101.

31. Lund, C., and Yannakakis, M. (1994), "On the hardness of approximating minimization problems", *J. ACM* **41**, 960–981.

32. Motwani, R. (1992), "Lecture notes on approximation algorithms", Technical Report STAN-CS-92-1435, Department of Computer Science, Stanford University, 1992.

33. Orponen, P., and Mannila, H. (1987), "On approximation preserving reductions: Complete problems and robust measures", Technical Report C-1987-28, Department of Computer Science, University of Helsinki.

34. Panconesi, A., and Ranjan, D. (1993), "Quantifiers and approximation", *Theoretical Computer Science* **107**, 145–163.

35. Papadimitriou, C.H. (1993), *Computational complexity.* Addison-Wesley.

36. Papadimitriou, C. H., and Yannakakis, M. (1991), "Optimization, approximation, and complexity classes", *J. Comput. System Sci.* **43**, 425–440.

37. Queyranne, M. (1985), "Bounds for assembly line balancing heuristics", *Operations Research* **33**, 1353–1359.

38. Schöning, U. (1986), "Graph isomorphism is in the low hierarchy", *Proc. 4th Ann. Symp. on Theoretical Aspects of Comput. Sci.*, Lecture Notes in Comput. Sci. 247, Springer-Verlag, 114–124.

39. Simon, H.U. (1989), "Continuous reductions among combinatorial optimization problems", *Acta Informatica* **26**, 771–785.

40. Wagner, K. (1988), "Bounded query computations", *Proc. 3rd Ann. Structure in Complexity Theory Conf.*, IEEE Computer Society, 260–277.

41. Wee, T.S. and Magazine M.J. (1982), "Assembly line balancing as generalized bin packing", *Operations Research Letters* **1**, 56–58.

42. Yannakakis, M. (1994), "On the approximation of maximum satisfiability", *J. Algorithms* **17**, 475–502.