Structure in Approximation Classes

(Extended abstract)

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
[2] Department of Numerical Analysis and Computing Science
Royal Institute of Technology
S-100 44 Stockholm, Sweden
E-mail: viggo@nada.kth.se

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 has been obtained in [19] where the APX-completeness of several important optimization problems is proved thus reconciling 'two distinct views of approximation classes: *syntactic* and *computational*'. In this paper we obtain new results on the structure of two important computationally-defined classes: the class NPO (that is, the class of optimization problems whose underlying decision problem is in NP), and the class APX (that is, the class of constant-factor approximable NPO problems). In particular, we give the first examples of natural APX-intermediate problems and the first examples of natural NPO-complete problems. Moreover, we state new connections between the approximability properties and the query complexity of NPO problems.

1. Introduction

In his pioneering paper on the approximation of combinatorial optimization problems [15], 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 all 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 able to come back from "good" solutions in B to "good" solutions in A. Surprisingly, the first definition of this kind of reducibility was given as long as 13 years after Johnson's paper [26] and, after that, at least seven different definitions of approximation preserving reducibility appeared in the literature (see Fig. 1). These definitions 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 [16]).

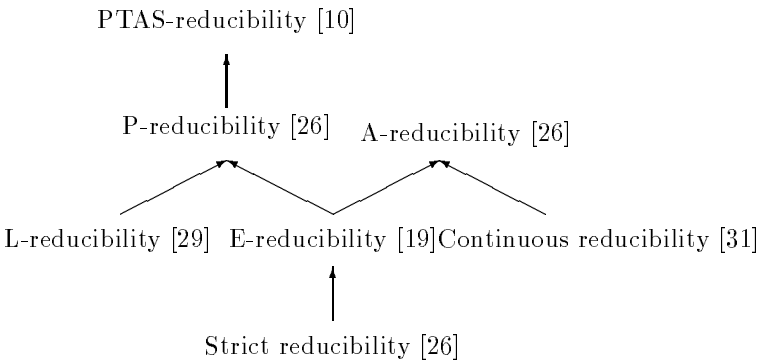

Figure 1. The taxonomy of approximation preserving reducibilities

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 whose approximability properties were well understood, such as NPO and APX: 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 [26, 9]. 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 [29, 27, 20]: 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 on the approximability of complete problems for syntactically defined classes, unless P = NP (or some other unlikely condition) [3, 23]. More recently, another step has been performed since the closure of syntactically defined classes with respect to approximation preserving reducibility has been proved to be equal to the more familiar computationally defined classes [19].

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 [20] 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 [9], a natural question arises: do natural intermediate problems exist? Observe that this question is also open in the field of decision problems even though the existence of artificial NP-intermediate problems has been already proved [22]. For example, it is known that the graph isomorphism problem cannot be NP-complete unless the polynomial-time hierarchy collapses [30], but no similar result has ever been obtained showing that the problem does not belong to P.

1.1. Summary of the Results

The first goal of this paper is to define an approximation preserving reducibility such that all reductions that have appeared in the literature still hold and such that it can be used for as many approximation classes as possible. 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 the L-reducibility is too weak and is not approximation preserving (unless $\mathrm{P} = \mathrm{NP} \cap$ *co*-NP), on the other it is too strict and does not allow to reduce problems which are known to be easy to approximate to problems which are known to be hard to approximate (unless $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$). 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 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 the maximum satisfiability problem [10, 19] and the poly-APX-completeness of the maximum clique problem [19].

The second group of results refers to the existence of natural complete problems in NPO. Indeed, both [26] and [9] 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, Minimum $0-1$ Programming, and Minimum Weighted Independent Dominating Set are NPO-complete. This result shows how making use of a natural approximation preserving reducibility is enough powerful to encompass the 'duality' problem raised in [20]. Moreover, the same result can also be obtained when restricting ourselves to the class NPO PB (that is, the class of polynomially bounded NPO problems). In particular, we prove that Maximum PB $0-1$ Programming, Minimum PB $0-1$ Programming and Minimum Independent Dominating Set, are NPO PB-complete. Indeed, this result can also be obtained as a consequence of Theorem 6(a) of [19]. However, *our proof does not make use of the PCP model*.

The third group of results refers to the existence of natural APX-intermediate problems. In particular, 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 [25] 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 thus 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 is APX-complete then the problem of answering any set of k non-adaptive queries to an NP-complete problem can 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*: as far as we know, no general technique to obtain this kind of results was previously known.

In the last group of results, we state 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 [8, 6] upper and lower bounds have been proved on the number of queries needed to approximate certain optimization problems (such as the maximum satisfiability problem and the maximum clique problem): these results dealt 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, in Sect. 4.1 we will be able to solve an open question of [6] proving that *finding the vertices of the largest clique is more difficult than merely finding the vertices of a 2-approximate clique* (that is, a clique with at least half the size of the largest clique) unless the polynomial-time hierarchy collapses. On the one hand, the results of [8, 6] show that the query complexity is a good measure of complexity to study approximability properties of optimization problems. On the other, our results show that completeness in approximation classes implies lower bounds on the query complexity. In Sect. 5 we finally 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 in approximation classes and they reconcile the approach based on standard computation models with the approach based on the computation model for approximation proposed by [7]. As a final observation, our results can be seen as an extension of some results of [19] in which general sufficient conditions for APX-completeness are proved.

Due to the lack of space, the proofs of our results are all contained in the appendix where the problems mentioned in the text are also defined.

1.2. Preliminaries

Since several introductory books on computational complexity theory [5, 12, 28] make some mention of approximation classes, we will start the following preliminaries on approximation classes with some mention of computational complexity theory.

Definition 1. *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 function classes $\mathrm{FP}^{\mathrm{NP}[f(n)]}$.

Theorem 1 ([33]). *For any function $f(n) \in O(\log n)$, if $\mathrm{P}^{\mathrm{NP}[f(n)+1]} \subseteq \mathrm{P}^{\mathrm{NP}[f(n)]}$ then the polynomial-time hierarchy collapses.*

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

Definition 2. *An* NP optimization problem *A is a fourtuple $(I, sol, m, goal)$ 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, it is decidable in polynomial time whether, for any x and for any y such that $|y| \leq p(|x|)$, $y \in sol(x)$.*
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. *$goal \in \{\max, \min\}$.*

The *class* NPO is the set of all NP optimization problems.

The goal of an NPO problem with respect to an instance x is to find an *optimum solution*, that is, a feasible solution y such that $m(x, y) = goal\{m(x, y') \ : \ y' \in sol(x)\}$.

In the following *opt* will denote the function mapping an instance x to the measure of an optimum solution. 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. NPO PB = Max PB $\cup$ Min PB where 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 3. *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*

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

The performance ratio is always a number greater than 1 and is as close to 1 as the feasible solution is close to the optimum one.

Definition 4. *Let A be an* NPO *problem and let T be an algorithm that, for any instance x of A, returns a feasible solution $T(x)$. Given an arbitrary function $r : N \to (1, \infty)$, we say that T is an* $r(n)$-approximate algorithm for *A if, for any instance x, 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 5. *Given a class of functions F, an* NPO *problem A belongs to the* class F-APX *if an $r(n)$-approximate polynomial-time algorithm T for A exists, for some function $r \in F$.*

In particular, APX, log-APX, and poly-APX will denote the classes F-APX with F equal to the set of constant functions, to the set $O(\log n)$, and to the set of polynomials, respectively.

Definition 6. *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 a polynomial-time r-approximate algorithm for A.*

Clearly, the following inclusions hold:

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

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

2. A new approximation preserving reducibility

We will justify our definition by emphasizing the disadvantages of previously known reducibilities.

2.1. The L-reducibility

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

Definition 7. *Let A and B be two* NPO *problems. A is said to be* L-reducible *to B, in symbols $A \leq_{\text{L}} B$, 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 $x \in I_A$, $opt_B(f(x)) \leq \alpha opt_A(x)$.*
4. *For any $x \in I_A$ and for any $y \in sol_B(f(x))$,*

$$|opt_A(x) - m_A(x, g(x, y))| \leq \beta |opt_B(f(x)) - m_B(f(x), y)|$$

Clearly the L-reducibility preserves membership in PTAS. However, the next result gives a strong evidence that, in general, this reducibility is not approximation preserving. It also shows that the behavior of L-reductions depends on the type (that is, maximization or minimization) of the problems involved.

Theorem 2. *The following hold:*

1. L-*reductions from minimization problems to optimization problems are approximation preserving.*
2. L-*reductions from maximization problems to optimization problems are not approximation preserving if and only if the* γ*-reducibility is different from the many-one reducibility.*

Observe that in [14] it is shown that the hypothesis of the point (2) above is somewhat intermediate between $\mathrm{P} \neq \mathrm{NP} \cap$ *co*-NP and $\mathrm{P} \neq \mathrm{NP}$. In other words, there is strong evidence that, even though the L-reducibility is suitable to prove APX-completeness results, this reducibility cannot be used to define the notion of completeness within classes beyond APX. Moreover, it cannot be used to obtain positive results, that is, the existence of approximation algorithms via reductions.

2.2. The E-reducibility

The drawbacks of the L-reducibility are mainly due to the fact the relation between the performance ratios (not necessarily linear) is obtained by putting separate linear constraints on the relations between both the optimum values and the absolute errors. The E-reducibility (for *error* reducibility), instead, imposes a linear relation directly between the performance ratios.

Definition 8. *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* $y \in sol_B(f(x))$*,*

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

Observe that, for any function r, an E-reduction maps $r(n)$-approximate solutions into $(1 + \alpha(r(n) - 1))$-approximate solutions so that it not only preserves membership in PTAS but also membership in any F-APX class where F is closed with respect to linear applications, such as 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* $A \leq_{\mathrm{E}} B$*.*

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.

The E-reduction is still somewhat too strict. Indeed, the next result shows that, unless $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$, PTAS problems exist which are not reducible to APX problems (observe that from the above proposition this fact holds for the L-reducibility as well). Intuitively, this unnatural behavior is due to the fact that an E-reduction preserves optimum values.

Proposition 2. Maximum Knapsack *is not E-reducible to any* NPO PB *problem unless* $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

2.3. The AP-reducibility

We have just observed that a drawback of the E-reducibility consists of preserving optimum solutions. This is due to the fact that the linear relation between the performance ratios is too restrictive. According to the definition of approximation preserving reducibilities given in [9], we could overcome this problem by expressing this relation by means of an implication. However, this solution 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 in letting the function g depend on the performance ratio. Indeed, in the following definition (which is a restriction of the PTAS-reducibility introduced in [10]), we also let the function f depend on this ratio because this feature will turn out to be useful in order to prove interesting characterizations of complete problems in approximation classes.

Definition 9. *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 $r > 1$, $f(x,r) \in I_B$.*
2. *For any $x \in I_A$, for any $r > 1$, and for any $y \in sol_B(f(x,r))$, $g(x,y,r) \in sol_A(x)$.*
3. *f and g are computable by two algorithms T_f and T_g, respectively, whose running time is polynomial for any fixed r.*
4. *For any $x \in I_A$, for any $r > 1$, and for any $y \in sol_B(f(x,r))$,*

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

Observe that, clearly, the AP-reducibility is a generalization of the E-reducibility. Moreover, it is easy to see that Proposition 2 does not hold for the AP-reducibility: indeed, 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 [10, 19]).

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 [9, 26, 4, 17]. One example of such a result is the following theorem.

Theorem 3 ([26, 9]). 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 and $w(v_i) = 2^{s-i}$ for $i \in [1..s]$.*

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 4. Minimum Weighted Satisfiability *and* Maximum Weighted Satisfiability *are* NPO*-complete.*

Corollary 1. *Any* Min NPO*-complete problem is* NPO*-complete and any* Max NPO*-complete problem is* NPO*-complete. The problems* Minimum $0-1$ Programming, Traveling Salesperson Problem, *and* Minimum Weighted Independent Dominating Set *are* NPO*-complete.*

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

Theorem 5. Maximum PB $0-1$ Programming *and* Minimum PB $0-1$ Programming *are* NPO PB*-complete.*

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

By using the construction of the proof of Theorem 5 together with the result that Longest Induced Path is not approximable within $|V|^{1-\varepsilon}$ for any $\varepsilon > 0$ unless P = NP [24], one can show the following new hardness results for some NPO PB-complete problems.

Theorem 6. *The following problems are not approximable within* $n^{1-\varepsilon}$ *for any* $\varepsilon > 0$ *unless* P = NP*:* Maximum Number of Satisfiable Formulas [27] (n is the number of equations), Maximum Distinguished Ones [27] (n is the number of distinguished variables), Maximum PB $0-1$ Programming (n is the number of inequalities).
The following problems are not approximable within $n^{1/2-\varepsilon}$ *for any* $\varepsilon > 0$ *unless* P = NP*:* Maximum PB $0-1$ Programming (n is the number of variables), Maximum Constrained Binary Satisfiable Linear Subsystem [1] (n is the number of variables).
The following problems are not approximable within $n^{1/3-\varepsilon}$ *for any* $\varepsilon > 0$ *unless* P = NP*:* Maximum Ones [27] (n is the number of variables), Maximum Irrelevant Binary Variables in Linear System [2] (n is the number of variables).

4. Query complexity and APX-intermediate problems

Definition 10. *Let* A *be an* NPO *problem and* r *be a function, then* $A_{r(n)}$ *is the following multi-valued partial function: given an instance* x *of* A, $A_{r(n)}(x)$ *is the set of feasible solutions* y *of* x *such that* $R(x,y) \leq r(|x|)$.

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, *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 $\bigcup_{r>1} \mathrm{P}^{A_r}$. Using techniques similar to those of [6, 8], we can prove the following result.

Proposition 3. *For any problem* A *in* APX, $\mathrm{AQH}(A) \subseteq \mathrm{QH}$.

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 $r > 1$, $R(x, T(x,r)) \leq r + k/opt(x)$ 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 as PTAS^{∞}. The following result shows that, for this class, the previous fact can be strengthened.

Proposition 4. *Let* $A \in \mathrm{PTAS}^{\infty}$. *Then, a constant* h *exists such that* $\mathrm{AQH}(A) \subseteq \mathrm{P}^{\mathrm{NP}[h]}$.

The following fact, instead, states that any language L in the query hierarchy can be decided using just one query to A_ε where A is APX-complete and ε depends on the level of the query hierarchy L belongs to.

Proposition 5. *For any* APX*-complete problem* A, $\mathrm{QH} \subseteq \mathrm{AQH}(A)$.

By combining Propositions 5 and 3, we thus have the following result that characterizes the approximation query hierarchy of the hardest problems in APX.

Theorem 7. *For any* APX*-complete problem* A, $\mathrm{AQH}(A) = \mathrm{QH}$.

Finally, as a consequence of this theorem, of Proposition 4, of Theorem 1, and of the results of [13, 18, 11] we have the following result.

Corollary 3. *If the polynomial-time hierarchy does not collapse, then* Minimum Degree Spanning Tree, Minimum Bin Packing, *and* Minimum Edge Coloring *are* APX*-intermediate.*

4.1. A remark on Maximum Clique

The following two propositions are the analogous of Propositions 3 within NPO.

Proposition 6. *For any* NPO *problem* A *and for any* $r > 1$, $\mathrm{P}^{A_r} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$.

Proposition 7. *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 Proposition 7, from the fact that $\mathrm{P}^{\mathrm{NP}[\log n]}$ is contained in $\mathrm{P}^{\mathrm{MC}_1}$ where MC stands for Maximum Clique [21], and from Theorem 1, it thus follows the next result that solves an open question posed in [6]. 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 8. *If* $\mathrm{P}^{\mathrm{MC}_1} \subseteq \mathrm{P}^{\mathrm{MC}_2}$ *then the polynomial-time hierarchy collapses, where* MC *stands for* Maximum Clique.

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 query complexity.

Definition 12. $\mathrm{NPF}^{\mathrm{NP}[q(n)]}$ *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.*[1]

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* $y \in G(t_1(x))$, $t_2(x, y) \in F(x)$.

[1] 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 $F(x)$.

We shall say that a function F is hard for $\mathrm{NPF}^{\mathrm{NP}[q(n)]}$ if, for any $G \in \mathrm{NPF}^{\mathrm{NP}[q(n)]}$, $G \leq_{\mathrm{mv}} F$.

The following definition is a constructive version of the definition of self-improvability given in [27].

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 rational $r_1, r_2 > 1$, $x' = t_1(x, r_1, r_2)$ 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.*

From [27] it follows that the equivalence with respect to the AP-reducibility preserves the self-improvability property. We are now ready to state the main result of this section.

Theorem 9. *A poly-*APX *problem A is poly-*APX*-complete if and only if it is self-improvable and A_{r_0} is $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$-hard for some $r_0 > 1$.*

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 $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$-hard and A is not poly-APX-complete.

In order to characterize APX-complete problems, we have to define the following combinatorial property.

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 $\varepsilon > 0$ and for any sequence $x_1, \ldots, x_k$ of instances of A, $x' = t_1(x_1, \ldots, x_k, \varepsilon)$ is an instance of A and, for any $y' \in A_{1+\varepsilon\beta/k}(x')$, $t_2(x_1, \ldots, x_k, y', \varepsilon) = y_1, \ldots, y_k$ where each y_i is a $(1+\varepsilon)$-approximate solution of x_i. Moreover, the running time of t_1 and t_2 is polynomial for every fixed $\varepsilon > 0$.*

Theorem 10. *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 $\mathrm{NPF}^{\mathrm{NP}[1]}$-hard.*

Note that linear additivity plays for APX problem more or less the same role of self-improvability in poly-APX. These two properties are, in a certain sense, one the opposite of the other: while the query complexity of APX-complete problems depends on the performance ratio and does not depend on the size of the instance, the query complexity of poly-APX-complete 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 also possible to establish query complexity results for log-APX-complete problem. In particular, even though we have not been able to establish a full characterization of log-APX-complete problems, we can prove the following result.

Theorem 11. *No log-*APX*-complete problem can be self-improvable unless the polynomial time-hierarchy collapses.*

It is then an interesting open question to find a characterizing combinatorial property of log-APX-complete problems. Moreover, as a consequence of the above theorem and of the results of [19], we conjecture that the minimum set cover problem is not self-improvable.

References

1. Amaldi, E., and Kann, V. (1993), "The complexity and approximability of finding maximum feasible subsystems of linear relations", Technical Report ORWP-11-93, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne. *Theoretical Comput. Sci.*, to appear.

2. Amaldi, E., and Kann, V. (1994), "On the approximability of removing the smallest number of relations from linear systems to achieve feasibility", Technical Report ORWP-6-94, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne.

3. 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.

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. 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.

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

8. 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.

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

10. 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.

11. 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.

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

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

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

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

16. 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.

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

18. 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.

19. 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.

20. 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.

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

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

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

24. Lund, C., and Yannakakis, M. (1993), "The approximation of maximum subgraph problems", *Proc. of 20th International Colloquium on Automata, Languages and Programming*, Lecture Notes in Comput. Sci. 700, Springer-Verlag, 40–51.

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

26. 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.

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

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

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

30. 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.

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

32. Valiant, L. (1976), "Relative complexity of checking and evaluating", *Information Processing Letters* **5**, 20–23.

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

Appendix

We will now give the proofs of the results presented in the paper, the definitions of the problems and some additional references.

A. Proof of the results of Section 2

Proof of Theorem 2. (1) follows from the fact, noted in [29], that if a minimization problem A L-reduces to an optimization problem B and there is a polynomial-time r-approximation algorithm for B, then there is a polynomial-time $(1+\alpha\beta(r-1))$-approximation algorithm for A.

In order to prove (2), first recall that in [36] it has been shown that the γ-reducibility is different from the many-one reducibility if and only if 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. Assume now that there exists a polynomial-time recognizable set S of satisfiable Boolean formulas for which no polynomial-time algorithm can compute a satisfying assignment for each of them. Consider the following maximization problems $A = (I_A, sol_A, m_A)$ and $B = (I_B, sol_B, m_B)$ where

1. $I_A = S$, $sol_A(x) = \{y | y \text{ is a truth assignment for } x\}$, and
2. $$m_A(x,y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 1 & \text{otherwise} \end{cases}$$

and

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

Clearly, problem B is in APX, while if A were in APX then there were 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 is L-reducible to B.

Conversely, 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 a maximization problem A is L-reducible to a maximization problem B via functions f and g and that B is r-approximable ($r > 1$). Let x be an instance of A and let y be a solution of $f(x)$ such that $opt_B(f(x))/m_B(f(x),y) \leq r$. For the sake of convenience, set $opt_A = opt_A(x)$, $m_A = m_A(x, g(x,y))$, $opt_B = opt_B(f(x))$, and $m_B = m_B(f(x), y)$. We show that $m = \max\{m_A, m_B/\alpha\}$ is such that $m \leq opt_A$ and $opt_A/m \leq 1 + \alpha\beta(r-1)$, that is, m is a non-constructive approximation of opt_A. Let $\gamma = \frac{\alpha r}{1+\alpha\beta(r-1)}$. There are two cases.

1. $opt_B \leq \gamma opt_A$. By the definition of the L-reducibility, $opt_A - m_A \leq \beta(opt_B - m_B)$. Since $opt_B \leq \gamma opt_A$ and $opt_B/m_B \leq r$, it holds that $\frac{opt_A - m_A}{opt_A} \leq \gamma\beta\frac{opt_B - m_B}{opt_B} \leq \gamma\beta(1-1/r)$. Hence, $opt_A/m \leq opt_A/m_A \leq 1 + \alpha\beta(r-1)$.

2. $opt_B > \gamma opt_A$. It holds that

$$\begin{aligned}
\frac{opt_A}{m} &\leq \frac{opt_A}{m_B/\alpha} \\
&< \frac{\alpha(opt_B/\gamma)}{m_B} \text{ (since } opt_A < opt_B/\gamma) \\
&\leq \frac{\alpha(opt_B/\gamma)}{(opt_B/r)} \text{ (since } m_B \geq opt_B/r) \\
&= \frac{\alpha r}{\gamma} \\
&= 1 + \alpha\beta(r-1).
\end{aligned}$$

Now, it is not hard to see that a satisfiable Boolean formula ϕ can be constructed, in polynomial time in the length of x, so that any satisfying assignment for ϕ encodes a solution of x whose measure is at least m. By assumption it is possible to compute in polynomial time a satisfying assignment for ϕ and thus an approximate solution for x.

Finally, if B is a minimization problem, we first L-reduce B to a maximization problem C in APX [19] and then apply the above argument. □

Proof of Proposition 1. Let T be an r-approximation 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 $y \in sol_B(f_L(x))$, $E_A(x, g_L(x,y)) \leq \alpha_L\beta_L E_B(f_L(x), y)$ where

$$E(w,z) = \frac{|opt(w) - m(w,z)|}{opt(w)}$$

denotes the relative error of the feasible solution z with respect to the instance w. If A is a minimization problem then, for any $x \in I_A$ and for any $y \in sol_B(f_L(x))$,

$$R_A(x, g_L(x,y)) = 1 + E_A(x, g_L(x,y)) \leq 1 + \alpha_L\beta_L E_B(f_L(x), y) \leq 1 + \alpha_L\beta_L(R_B(f_L(x), y) - 1).$$

Otherwise 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

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

We can thus define the E-reduction (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 $y \in sol_B(f_E(x))$,

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

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

From the above discussion it follows that this reduction is indeed an E-reduction. □

Proof of Proposition 2. From the results of [21] it follows that $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{MK}_1}$, where MK stands for MAXIMUM KNAPSACK. If MAXIMUM KNAPSACK is E-reducible to an NPO PB problem A, then $\mathrm{P}^{\mathrm{MK}_1} \subseteq \mathrm{P}^{A_1}$. It is easy to see that $\mathrm{P}^{A_1} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$ which, in turn, implies that $\mathrm{P}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}[O(\log n)]}$. □

B. Proof of the results of Section 3

Proof of Theorem 4. 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 3.

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 $\alpha_i = (z_i \equiv \overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i)$, where $z_1, \ldots, z_s$ are new variables with weights $w(z_i) = 2^i$ for $i \in [1..s]$ 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 $f(x)$. If all z-variables were false, then all v-variables would be false and the value of the objective function of x would be zero, which is not allowed.

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

This is in particular true for the optimum solution. Thus the performance ratio for MAXIMUM WEIGHTED SATISFIABILITY is

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

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 $k = 0$.

We use $2^k \cdot s$ new variables named $z_{i,b_1,\ldots,b_k}$, where $i \in [1..s]$ and $b_j \in \{0, 1\}$ for $j \in [1..k]$. Let $f_k(x) = \phi \wedge \bigwedge \alpha_{i,b_1,\ldots,b_k}$, where

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

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

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

By choosing K large enough (about 2^s) 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 $f_k(x)$, $z_{i,b_1,\ldots,b_k} = 1$, then we have $m(f_k(x), y) = w(z_{i,b_1,\ldots,b_k})$ and we know that

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

and that

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

Thus we 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 $2^{-k} \leq (r - 1)/r$, e.g. $k = \lceil \log r - \log(r - 1) \rceil$, then $R(f_k(x), y) \leq r$ implies $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y) \leq r + r2^{-k} \leq r + r - 1 = 1 + 2(r - 1)$. 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. □

Proof of Corollary 1. Theorem 4 says that MINIMUM WEIGHTED SATISFIABILITY is NPO-complete. Per definition MINIMUM WEIGHTED SATISFIABILITY can be reduced to any Min NPO-complete problem. Hence any Min NPO-complete problem is also complete for NPO. In the same way, since MAXIMUM WEIGHTED SATISFIABILITY is NPO-complete and can be reduced to any Max NPO-complete problem, any Max NPO-complete problem is NPO-complete.

Min NPO-completeness for MINIMUM $0-1$ PROGRAMMING and TRAVELING SALESPERSON PROBLEM was shown in [26], and therefore they are NPO-complete. For MINIMUM WEIGHTED INDEPENDENT DOMINATING SET we need a PTAS-reduction from MINIMUM WEIGHTED SATISFIABILITY.

Given an instance of MINIMUM WEIGHTED SATISFIABILITY, we first write the formula in conjunctive normal form. For every variable v_i we construct two nodes a_i and $\overline{a_i}$ with weights

$w(a_i) = w(\overline{a_i}) = w(v_i)$. For every clause c_j we construct a node b_j with enormous weight. We choose a weight larger that the sum of the weights of all variables, namely $\sum_{i=1}^{s} w(v_i) + 1$.

We add edges between a_i and $\overline{a_i}$ for each i. For each i and j we also add edges as follows. If the variable v_i is used positively in clause c_j we add an edge between the corresponding nodes a_i and b_j. If the variable v_i is used negatively in clause c_j we add an edge between $\overline{a_i}$ and b_j.

One solution to the independent dominating set problem is $\{a_1, \ldots, a_s\}$, and this solution has smaller objective value than any solution that contains a b-node. It is easy to see that in any solution at most one of a_i and $\overline{a_i}$ for any i is included, that the corresponding assignment ($v_i = 1$ if and only if a_i is included) satisfies the CNF formula, and that the objective values of both solutions are the same. The reduction is PTAS-preserving. □

Proof of Theorem 5. 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 [17]. Thus we just have to show that there are AP-reductions from a Min PB-complete problem to Maximum PB $0-1$ Programming and from a Max PB-complete problem to Minimum PB $0-1$ Programming. As the Min PB-complete problem we will use Minimum Independent Dominating Set and as the Max PB-complete problem we will use Longest Induced Path.

Both reductions follow the same idea. The objective function, i.e. the number of nodes in the solution (in the independent dominating set and in the induced path, respectively), is encoded by introducing an order of the nodes in the solution. The order is encoded by a squared number of $0-1$ variables in the programming problem, see Fig. 2. A solution of size 1 shall correspond to the $0-1$ programming objective value n, and a solution of size p shall correspond to an objective value of $\left\lfloor \frac{n}{p} \right\rfloor$.

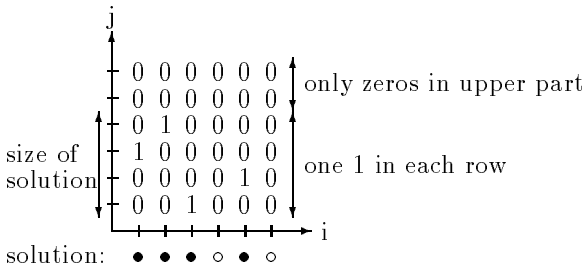

Figure 2. The idea of the reduction from Minimum Independent Dominating Set/Longest Induced Path to Maximum/Minimum PB $0-1$ Programming. The variable $x_i^j = 1$ if and only if v_i is the jth node in the solution. There is at most one 1 in each column and in each row.

The reduction from Minimum Independent Dominating Set to Maximum PB $0-1$ Programming is constructed as follows. Given an instance of Minimum Independent Dominating Set, i.e. a graph with nodes $V = \{v_1, \ldots, v_m\}$ and edges E, construct m^2 variables x_i^j, $1 \leq i, j \leq m$ and the following inequalities:

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

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

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

$$\forall (v_i, v_j) \in E \qquad \sum_{k=1}^{m} x_i^k + \sum_{k=1}^{m} x_j^k \leq 1 \qquad \text{(independence)} \tag{4}$$

$$\forall i \in [1..m] \quad \sum_{k=1}^{m} x_i^k + \sum_{\substack{j:(v_i,v_j)\in E \\ k\in[1..m]}} x_j^k \geq 1 \quad \text{(domination)} \tag{5}$$

The objective function is defined as

$$n - \sum_{p=2}^{m} \left(\left\lfloor \frac{n}{p-1} \right\rfloor - \left\lfloor \frac{n}{p} \right\rfloor \right) \sum_{i=1}^{m} x_i^p. \tag{6}$$

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 $\lfloor n/p \rfloor < j \leq \lfloor n/(p-1) \rfloor$ and $y_j = 1$ for $j \leq \lfloor n/m \rfloor$. The objective function is then $\sum_{j=1}^{n} y_j$. One can now verify that an independent dominating set of size s will exactly correspond to a solution of the $0-1$ programming problem with objective value $\lfloor \frac{n}{s} \rfloor$ and vice versa.

Suppose that the minimum independent dominating set has size M, then the performance ratio s/M for the independent dominating set problem will correspond to the performance ratio

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

for the $0-1$ programming 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.

Halldórsson has proved that, unless P = NP, Minimum Independent Dominating Set is not approximable within $n^{1-\varepsilon}$ for any $\varepsilon > 0$, where n is the sum of the number of nodes and edges in the graph [34]. Together with the reduction above this result will tell that, unless P = NP, Maximum PB $0-1$ Programming is not approximable within $q^{1-\varepsilon}$ for any $\varepsilon > 0$, where q is the number of inequalities, and is not approximable within $r^{1/2-\varepsilon}$ for any $\varepsilon > 0$, where r is the number of variables.

A similar construction can be used to reduce Longest Induced Path to Minimum PB $0-1$ Programming. Given an instance of Minimum Independent Dominating Set, i.e. a graph with nodes $V = \{v_1, \ldots, v_m\}$ and edges E, construct m^2 variables x_i^j, $1 \leq i, j \leq m$ as above and use the inequalities (1)–(3) together with the new inequalities:

$$\forall (v_i, v_j) \in E, k \in [1..m-2], l \in [k+2..m] \qquad x_i^k + x_j^l \leq 1 \qquad \text{(induced)} \tag{7}$$

$$\forall (v_i, v_j) \notin E \; \forall k \in [1..m-1] \quad x_i^k + x_j^{k+1} \leq 1 \quad \text{(path)} \tag{8}$$

Use the same objective function as (6) above. It is not hard to show that this reduction is a AP-reduction. □

The results of Theorem 6 are proved in [35].

C. Proof of the results of Section 4

Proof of Proposition 3. Assume that A is a maximization problem (the proof for minimization problems is similar), let T be an r-approximate polynomial-time algorithm for A, for some

$r > 1$, and let $L \in \mathrm{P}^{A_\rho}$ for some ρ. Two polynomial-time computable functions f and g then exist witnessing this fact. For any x, let $m = m(f(x), T(f(x)))$, so that $m \leq opt(f(x)) \leq rm$. We can then partition the interval $[m, rm]$ into $\lceil \log_\rho r \rceil$ subintervals

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

and start looking for the subinterval containing the optimum value (a similar technique has been used in [8, 6]). This can clearly be done using $\lceil \log_\rho r \rceil$ queries to an NP-complete oracle. One more query is sufficient to know whether a feasible solution y exists in that interval such that $g(x, y) = 1$. Since y is ρ-approximate, it follows that L can be decided using $\lceil \log_\rho r \rceil + 1$ queries, that is, $L \in \mathrm{QH}$. □

Proof of Proposition 4. 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 $r > 1$

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

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

$$opt(x) \leq l_r(x) \leq r \cdot opt(x).$$

Given an instance x, we can check whether $opt(x) = 0$ by means of a single query to an NP oracle, so we can restrict ourselves to instances such that $opt(x) \geq 1$. Note that, for these instances, T_2 is a $(k+2)$-approximate algorithm. Let us fix an $r > 1$, let $\varepsilon = r - 1$, $y = T_{1+\varepsilon/2}(x)$ and $m_{app} = m(x, T_2(x))$. We have to distinguish two cases.

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

$$m(x, y) \leq opt(x)(1 + \varepsilon/2) + opt(x)\varepsilon/2 = opt(x)(1 + \varepsilon) = r \cdot opt(x).$$

   That is, y is a r-approximate solution for x, and we can set $l_r(x) = m(x, y)$.
2. $m(x, y) < 2k(k+2)/\varepsilon$: in this case, $opt(x) < 2k(k+2)/\varepsilon$. Then,

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

   If $h = \lceil \log k(2k+5) \rceil + 1$, then $h - 1$ queries to NP are sufficient to find the optimum value $m^* = opt(x)$ by means of a binary search technique: in this case $l_r(x) = m^*$.

Let now L be a language in $AQH(A)$, 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 $m(x, y) \geq l_r(x)$ and $g(x, y) = 1$: that is, given $h_r(x)$, deciding whether $x \in L$ is an NP problem. Since $l_r(x)$ is computable by means of $h - 1$ queries to NP, we have that $L \in \mathrm{P}^{np[h]}$. □

In order to prove Proposition 5, we need the following technical result.

Claim. 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 be an instance of MINIMUM ORDERED BIN PACKING defined as follows (a similar construction has been used in [37] in order to prove negative results on the approximability of MINIMUM ORDERED BIN PACKING).

1. $U = \bigcup_{i=1}^{k} U_i \cup \{u_1, \ldots, u_{k-1}\}$ where the u_is are new items.
2. For any $u \in U_i$, $s(u) = s_i(u)$ and $s(u_i) = 1$ for $i = 1, \ldots, k-1$.
3. For any $i < j \leq k$, for any $u \in U_i$, and for any $u' \in U_j$, $u \preceq u_i \preceq u'$.

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 u_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 upper bounded by $4k - 1$ so that any $(1 + 1/4k)$-approximate solution is optimum.

Since MINIMUM ORDERED BIN PACKING belongs to APX and A is APX-complete, then an AP-reducibility (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 $x_1, \ldots, x_k$. □

Proof of Proposition 5. Let $L \in \mathrm{QH}$, then $L \in \mathrm{P}^{\mathrm{NP}[h]}$. It is well known that L can be reduced to the problem of answering $k = 2^{h-1}$ non-adaptive queries to NP. More formally, two 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 $I_j \in$ PARTITION, 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 from the preceding Claim 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 a r-approximate solution for x', then $t_2(g(x, y)) = 1$ if and only if $x \in L$. Thus, $L \in \mathrm{P}^{A_r}$. □

Propositions 6 and 7 can be easily proved similarly to Proposition 3 by means of the binary search technique.

D. Proof of the results of Section 5

Proof of Theorem 9. Let A be a poly-APX-complete problem. Since MAXIMUM CLIQUE is self-improvable [12] and poly-APX-complete [19], we have that A is self-improvable. It is then sufficient to prove that A_2 is hard for $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$.

Since A is poly-APX-complete, MAXIMUM CLIQUE $\leq_{\mathrm{AP}}$ A: let α be the constant of this reduction. From [7] we have that any function F in $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$ many-one reduces to MAXIMUM CLIQUE$_{1+1/\alpha}$. From the definition of AP-reducibility, we also have that MAXIMUM CLIQUE$_{1+1/\alpha} \leq_{\mathrm{mv}} A_2$ 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 $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$-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 $m \leq \log |(\phi_1, \ldots, \phi_m)|$, a possible output is a satisfying truth-assignment for ϕ_{i^*} where

$i^* = \max\{i | \phi_i \text{ is satisfiable}\}$. From [7] it follows that this function is $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$-complete.

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 $r > 1$, $t_1(x) = t_1^B(x, r)$ and $t_2(x) = t_2^B(x, r)$ are a many-one reduction from B_r to `multisat`. Moreover, since A_{r_0} is $\mathrm{NPF}^{\mathrm{NP}[\log\log n + O(1)]}$-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:

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

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. □

Recall that in [7] an extension of Theorem 1 is proved, that is, for any $q(n) \in O(\log n)$, if $\mathrm{NPF}^{\mathrm{NP}[q(n)+1]}$ is contained in $\mathrm{NPF}^{\mathrm{NP}[q(n)]}$, then the polynomial hierarchy collapses.

In the same paper, a characterization of the classes $\mathrm{NPF}^{\mathrm{NP}[q(n)]}$ is given such that any function $F \in \mathrm{NPF}^{\mathrm{NP}[k]}$ is reducible to answering with witnesses a set of 2^k non-adaptive queries to NP. Moreover, it is easy to see that any function that is reducible to answering with witnesses $2^k - 1$ non-adaptive queries to NP is contained in $\mathrm{NPF}^{\mathrm{NP}[k]}$. From the proofs of Propositions 3 and 4, it follows the following fact.

Claim. Let A be an r_A-approximable APX problem, then, for any $r > 1$, A_r is reducible to answering with witnesses a set of $\lceil \log_r r_A \rceil$ non-adaptive queries to an NP oracle. Let A be an APX-complete problem, then a constant γ exists such that, for any k, the problem of answering with witnesses a set of k non-adaptive queries to Partition is reducible to $A_{1+\gamma/k}$.

Proof of Theorem 10. Let A be an r_A-approximable APX-complete problem, let γ be the constant in the preceding Claim, then $A_{1+\gamma/2}$ is hard for $\mathrm{NPF}^{\mathrm{NP}[1]}$. Fix any $r > 1$, let $r = 1 + \varepsilon$ and let $x_1, \ldots, x_k$ be instances of A: for any $i = 1, \ldots, k$ the problem of finding a r-approximate solution y_i for x_i is reducible to the problem of answering with witnesses a set of $\lceil \log_r r_A \rceil$ parallel queries to Partition. Without loss of generality, we can assume $r < r_A$ (otherwise the reduction is trivial), and thus we have $\lceil \log_r r_A \rceil \leq 1 + (r_A - 1)/(r - 1) \leq c/\varepsilon$ for a certain constant ε. Moreover, answering kc/ε non-adaptive queries to Partition is reducible to $(1 + \gamma\varepsilon/kc)$-approximating a single instance of A, that is, A is linearly additive.

Conversely, let A be a linearly additive APX problem such that A_{r_0} is $\mathrm{NPF}^{\mathrm{NP}[1]}$-hard and let B be an r_B-approximable APX problem. Given an instance x of B, for any $r = 1 + \varepsilon > 1$ we can reduce the problem of finding an $1 + \varepsilon$-approximate solution for x to the problem of answering with witnesses c/ε queries to Partition, for a proper constant c not depending on ε. Moreover, each of these questions is reducible to A_{r_0}, since an $\mathrm{NPF}^{\mathrm{NP}[1]}$ can clearly answer with witness to an NP query. From linear additivity, it follows that r_0-approximating c/ε instance of A is reducible to $(1 + \beta\varepsilon/c)$-approximating a single instance of A. This is an AP-reduction from B to A with $\alpha = c/\beta$. □

Proof of Theorem 11. Let us consider the optimization problem Max Number of Satisfiable Formulas-log defined as follows.

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

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

Given $\log\log n$ queries to NP (of size polynomial in n) $\psi_1, \ldots, \psi_{\log\log n}$, we can construct an instance $\Phi = \phi_1, \ldots, \phi_m$ of MAX NUMBER OF SATISFIABLE FORMULAS-log where ϕ_1 is a tautology and the formulas $\phi_{2^i} = \ldots = \phi_{2^{i+1}-1}$ are satisfiable if and only if at least i clauses among $\psi_1, \ldots, \psi_{\log\log n}$ (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 a r-approximate solution for Φ we can decide how many clauses in $\psi_1, \ldots, \psi_{\log\log n}$ are satisfiable, and we can also recover witnesses for such formulas, that is, any function in $\text{NPF}^{\text{NP}[\log\log\log n - 1]}$ is MV-reducible to MNSF_r.

Let A be a self-improvable log-APX-complete problem, then, for any function $F \in \text{NPF}^{\text{NP}[\log\log\log n - 1]}$, $F \leq_{\text{mv}} \text{MNSF}_{1.5} \leq_{\text{mv}} A_{1+\alpha/2} \leq_{\text{mv}} A_{256}$ where α is the constant in the AP-reduction from MAX NUMBER OF SATISFIABLE FORMULAS-log to A. Thus, for any x instance of F, computing $F(x)$ is reducible to finding a 256-approximate solution for an instance x' of A, moreover, the size of x' is polynomial in $|x|$, that is $|x'| \leq |x|^c$ for a certain constant c. Since $A \in \mathit{log} - \text{APX}$, it is possible to find in polynomial time a $(r \log |x'|)$-approximate solution y for x'. By means of the usual binary search technique, we can find a 256-approximate solution for x' using $\lceil \log \lceil \log_{256}(r \log |x'|) \rceil \rceil \leq \log\log\log |x|^{rc} - 3$ adaptive queries to NP. Thus,

$$\text{NPF}^{\text{NP}[\log\log\log n - 1]} \subseteq \text{NPF}^{\text{NP}[\log\log\log |x|^{rc} - 3]}$$

which implies the collapse of the polynomial hierarchy. □

E. A List of NPO Problems

E.1. 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., $|V'|$.

E.2. MINIMUM INDEPENDENT DOMINATING SET

INSTANCE: Graph $G = (V, E)$.
SOLUTION: An independent dominating set for G, i.e., a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is a $v \in V'$ for which $(u, v) \in E$, and such that no two vertices in V' are joined by an edge in E.
MEASURE: Cardinality of the independent dominating set, i.e., $|V'|$.

E.3. Minimum Weighted Independent Dominating Set

Instance: Graph $G = (V, E)$ and a weight function $w : V \to N$.
Solution: An independent dominating set for G, i.e., a subset $V' \subseteq V$ such that for all $u \in V - V'$ there is a $v \in V'$ for which $(u, v) \in E$, and such that no two vertices in V' are joined by an edge in E.
Measure: The sum of the weights of the independent dominating set, i.e., $\sum_{v \in |V'|} w(v)$.

E.4. Maximum Weighted Satisfiability **and** Minimum Weighted Satisfiability

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

E.5. Maximum $0-1$ Programming **and** Minimum $0-1$ Programming

Instance: Integer $m \times n$-matrix $A \in Z^{m \cdot n}$, integer m-vector $b \in Z^m$, nonnegative integer n-vector $c \in N^n$.
Solution: A binary n-vector $x \in \{0,1\}^n$ such that $Ax \geq b$.
Measure: The scalar product of c and x, i.e., $\sum_{i=1}^{n} c_i x_i$.

E.6. Maximum PB $0-1$ Programming **and** Minimum PB $0-1$ Programming

Instance: Integer $m \times n$-matrix $A \in Z^{m \cdot n}$, integer m-vector $b \in Z^m$, nonnegative binary n-vector $c \in \{0,1\}^n$.
Solution: A binary n-vector $x \in \{0,1\}^n$ such that $Ax \geq b$.
Measure: The scalar product of c and x, i.e., $\sum_{i=1}^{n} c_i x_i$.

F. A List of APX Problems

F.1. 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 of disjoint sets, m.

F.2. 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, for any $u \in U_i$ and for any $u' \in U_j$ such that $u \preceq u'$, $i \leq j$.
Measure: The number of used bins, i.e., the number of disjoint sets, m.

F.3. MAXIMUM KNAPSACK

INSTANCE: Finite set U, for each U a size $s(u) \in Z^+$ and a value $v(u) \in Z^+$, a positive integer $B \in Z^+$.
SOLUTION: A subset $U' \subseteq U$ such that $\sum_{u \in U'} s(u) \leq B$.
MEASURE: Total weight of the chosen elements, i.e., $\sum_{u \in U'} v(u)$.

G. Additional references

34. Halldórsson, M. M. (1993), "Approximating the minimum maximal independence number", *Inform. Process. Lett.* **46**, 169–172.

35. Kann, V. (1995), "Strong lower bounds of the approximability of some NPO PB-complete maximization problems", Technical Report TRITA-NA-9501, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm.

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

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