variables, that Program 3.1 always satisfies at least $c/2$ clauses. Since no optimal solution can have value larger than c, the theorem will follow.

The result is trivially true in the case of one variable. Let us assume that it is true in the case of $n-1$ variables $(n>1)$ and let us consider the case in which we have n variables. Let v be the variable corresponding to the literal which appears in the maximum number of clauses. Let c_1 be the number of clauses in which v appears positive and c_2 be the number of clauses in which v appears negative. Without loss of generality suppose that $c_1 \geq c_2$, so that the algorithm assigns the value TRUE to v. After this assignment, at least $c-c_1-c_2$ clauses, on $n-1$ variables, must be still considered. By inductive hypothesis, Program 3.1 satisfies at least $(c-c_1-c_2)/2$ such clauses. Hence, the overall number of satisfied clauses is at least $c_1+(c-c_1-c_2)/2 \geq c/2$. QED

Within the class NPO, the class of problems that allow polynomial-time r-approximate algorithms (or, equivalently, ε-approximate algorithms) plays a very important role. In fact, the existence of a polynomial-time r-approximate algorithm for an NP-hard optimization problem shows that, despite the inherent complexity of finding the exact solution of the problem, such a solution can somehow be approached.

◀ Definition 3.9
Class APX

APX *is the class of all* NPO *problems* $\mathcal{P}$ *such that, for some* $r \geq 1$*, there exists a polynomial-time* r*-approximate algorithm for* $\mathcal{P}$.

As shown above and in the previous chapter, MAXIMUM SATISFIABILITY, MAXIMUM KNAPSACK, MAXIMUM CUT, MINIMUM BIN PACKING, MINIMUM GRAPH COLORING restricted to planar graphs, MINIMUM SCHEDULING ON IDENTICAL MACHINES, and MINIMUM VERTEX COVER are all in APX.

The definition of the class APX provides a first important notion for characterizing NPO problems with respect to their degree of approximability. For several important NPO problems, in fact, it can be shown that they do not allow any r-approximate algorithm, unless P = NP. In other words, for these problems, approximate solutions with guaranteed performance are as hard to determine as the optimal solutions. This means that, under the hypothesis that P $\neq$ NP, the class APX is strictly contained in the class NPO.

In the next subsection we will show that MINIMUM TRAVELING SALESPERSON is a problem for which determining approximate solutions with constant bounded performance ratio is computationally intractable. Other NPO problems that do not belong to APX, such as MAXIMUM CLIQUE and MAXIMUM INDEPENDENT SET, will be seen in Chap. 6, where some techniques needed to prove such results will also be provided.

at most four colors (see Bibliographical notes), the problem of deciding whether a planar graph is colorable with at most three colors is NP-complete. Hence, in this case, the gap is $1/3$ and we can hence conclude that no polynomial-time r-approximate algorithm for MINIMUM GRAPH COLORING can exist with $r < 4/3$, unless P = NP. Actually, much stronger results hold for the graph coloring problem: it has been proved that, if P $\neq$ NP, then no polynomial-time algorithm can provide an approximate solution whatsoever (that is, MINIMUM GRAPH COLORING belongs to NPO $-$ APX).

The considerations of the previous example can be extended to show that, for any NPO minimization problem $\mathcal{P}$, if there exists a constant k such that it is NP-hard to decide whether, given an instance x, $m^*(x) \leq k$, then no polynomial-time r-approximate algorithm for $\mathcal{P}$ with $r < (k+1)/k$ can exist, unless P = NP (see Exercise 3.8). Another application of the gap technique has been shown in the proof of Theorem 3.3: in that case, actually, we have seen that the constant `gap` can assume any value greater than 0. Other results which either derive bounds on the performance ratio that can be achieved for particular optimization problems or prove that a problem does not allow a polynomial-time approximation scheme can be obtained by means of a sophisticated use of the gap technique. Such results will be discussed in Chap. 6.

3.2 Polynomial-time approximation schemes

AS WE noticed before, for most practical applications, we need to approach the optimal solution of an optimization problem in a stronger sense than it is allowed by an r-approximate algorithm. Clearly, if the problem is intractable, we have to restrict ourselves to approximate solutions, but we may wish to find better and better approximation algorithms that bring us as close as possible to the optimal solution. Then, in order to obtain r-approximate algorithms with better performances, we may be also ready to pay the cost of a larger computation time, a cost that, as we may expect, will increase with the inverse of the performance ratio.

◀ Definition 3.10 *Polynomial-time approximation scheme*

Let $\mathcal{P}$ be an NPO *problem. An algorithm $\mathcal{A}$ is said to be a* polynomial-time approximation scheme (PTAS) *for $\mathcal{P}$ if, for any instance x of $\mathcal{P}$ and any rational value $r > 1$, $\mathcal{A}$ when applied to input (x, r) returns an r-approximate solution of x in time polynomial in $|x|$.*

While always being polynomial in $|x|$, the running time of a PTAS may also depend on $1/(r-1)$: the better is the approximation, the larger may be the running time. In most cases, we can indeed approach the optimal solution of a problem arbitrarily well, but at the price of a dramatic increase

Problem 3.2: Maximum Integer Knapsack

INSTANCE: Finite set X of types of items, for each $x_i \in X$, value $p_i \in Z^+$ and size $a_i \in Z^+$, positive integer b.

SOLUTION: An assignment $c : X \mapsto \mathbb{N}$ such that $\sum_{x_i \in X} a_i c(x_i) \leq b$.

MEASURE: Total value of the assignment, i.e., $\sum_{x_i \in X} p_i c(x_i)$.

$w(Y_1) \geq L \geq w(Y_2)$ and $m^*(x) \geq L$, the performance ratio of the computed solution is

$$\frac{w(Y_1)}{m^*(x)} \leq \frac{w(Y_1)}{L} \leq 1 + \frac{a_h}{2L} \leq 1 + \frac{1}{k(r)+1} \leq 1 + \frac{1}{\frac{2-r}{r-1}+1} = r.$$

Finally, we prove that the algorithm works in time $O(n \log n + n^{k(r)})$. In fact, we need time $O(n \log n)$ to sort the n items. Subsequently, the first phase of the algorithm requires time exponential in $k(r)$ in order to perform an exhaustive search for the optimal solution over the $k(r)$ heaviest items $x_1, \ldots, x_{k(r)}$ and all other steps have a smaller cost. Since $k(r)$ is $O(1/(r-1))$, the theorem follows. QED

3.2.1 The class PTAS

Let us now define the class of those NPO problems for which we can obtain an arbitrarily good approximate solution in polynomial time with respect to the size of the problem instance.

PTAS *is the class of* NPO *problems that admit a polynomial-time approximation scheme.* ◄ Definition 3.11 *Class PTAS*

The preceding result shows that MINIMUM PARTITION belongs to PTAS. Let us now see other examples of problems in PTAS. The first example will also show another application of the algorithmic technique of Program 3.3, which essentially consists of optimally solving a "subinstance" and, then, extending the obtained solution by applying a polynomial-time procedure.

Problem 3.2 models a variant of MAXIMUM KNAPSACK in which there is a set of types of items and we can take as many copies as we like of an item of a given type, provided the capacity constraint is not violated (observe that this problem is equivalent to Problem 2.8 with $d = 1$).

MAXIMUM INTEGER KNAPSACK *belongs to the class* PTAS. Theorem 3.9 ►

regardless of the truth-assignment, for each random string ρ for which the verifier will reject, at least one clause in the group of clauses constructed from A_ρ is not satisfied. Hence there will be at least $2^{r(n)}/2$ clauses that are not satisfied. The fraction of unsatisfiable clauses is therefore at least $(2^{r(n)}/2)/(q-2)2^{q+r(n)} \leq (2^{r(n)}/2)/2^{q+r(n)} = 2^{-(q+1)}$, that is a constant fraction.

QED

It is important to observe that, if we have a precise information on the number of bits queried by the verifier, the proof of the above theorem not only shows that MAXIMUM 3-SATISFIABILITY is not in PTAS but it also allows to explicitly derive a precise bound on the approximability of this problem (see Exercise 6.23).

6.4.2 The maximum clique problem

In this final section, we show non-approximability results for MAXIMUM CLIQUE. To this aim, we first show that this problem is at least as hard to approximate as MAXIMUM 3-SATISFIABILITY.

◀ Lemma 6.4

MAXIMUM CLIQUE *is not in* PTAS *unless* P=NP.

PROOF

We will show that MAXIMUM 3-SATISFIABILITY is "reducible" to MAXIMUM CLIQUE in such a way that any approximate solution for the latter problem could be transformed in polynomial time into an approximate solution for the former problem having at most the same performance ratio. From this fact and from Theorem 6.3, the lemma will follow.

Given an instance (U,C) of MAXIMUM 3-SATISFIABILITY where U is the set of variables and C is the set of clauses, we define a graph $G = (V,E)$ such that:

$$V = \{(l,c) : l \in c \text{ and } c \in C\}$$

and

$$E = \{((l_1,c_1),(l_2,c_2)) : l_1 \neq \overline{l_2} \wedge c_1 \neq c_2\}$$

where l, l_1, l_2 denote literals, i.e., variables or negation of variables in U, and c, c_1, c_2 denote clauses. For any clique V' in G, let τ be a truth assignment defined as follows: for any variable u, $\tau(u)$ is TRUE if and only if a clause c exists such that $(u,c) \in V'$. This assignment is consistent since no variable receives both the value TRUE and the value FALSE. In fact, if this happens for a variable u then two clauses c_1 and c_2 exist such that

$(u,c_1),(\overline{u},c_2) \in V'$. From the definition of E, these two nodes are not connected thus contradicting the hypothesis that V' is a clique. Moreover, τ satisfies any clause c containing a literal l such that $(l,c) \in V'$: indeed, either $l = u$ and $\tau(u) = \text{TRUE}$ or $l = \overline{u}$ and, since V' is a clique, there is no clause c' such that $(u,c') \in V'$, so that $\tau(u) = \text{FALSE}$. From the definition of E the number of these clauses is equal to $|V'|$. Since τ may satisfy some more clauses, we have that $m((U,C),\tau) \geq |V'|$.

It remains to show that the maximum number of satisfiable clauses is equal to the size of the maximum clique in G. Given a satisfiable set of clauses $C' \subseteq C$, for any truth-assignment satisfying C' and for any $c \in C'$, let l_c be any literal in c which is assigned the value TRUE. The set of nodes (l_c,c) defined in this way is clearly a clique in G.

Hence the performance ratio of τ is no worse than the performance ratio of V' and the lemma is proved. QED

By exploiting an interesting property of the MAXIMUM CLIQUE problem, known as *self-improvability*, we can now strenghten the previous non-approximability result to the following.

MAXIMUM CLIQUE *is not in* APX *unless* P=NP. Theorem 6.5 ▶

PROOF

Let us suppose that there is an algorithm $\mathcal{A}$ approximating MAXIMUM CLIQUE within a constant $\alpha > 1$ in polynomial time. Instead of applying $\mathcal{A}$ directly on an instance $G = (V,E)$ of MAXIMUM CLIQUE, we will transform G into another, larger, instance $f(G)$ of MAXIMUM CLIQUE, and apply $\mathcal{A}$ to $f(G)$. Then we will use the approximate solution $\mathcal{A}(f(G))$ to find a better approximate solution to G than we would have got by applying $\mathcal{A}$ directly to G.

As the transformation f we will use the k-th graph product of G, defined in the following way (where k is a constant to be specified later). Let the vertex set V^k of $f(G)$ be the k-th Cartesian product of V. This means that each vertex in V^k can be denoted by $(v_1,\ldots,v_k)$ where $v_i \in V$ for every $1 \leq i \leq k$. The number of vertices in V^k is $|V|^k$. There is an edge between $(v_1,\ldots,v_k)$ and $(w_1,\ldots,w_k)$ in $f(G)$ if and only if, for every $1 \leq i \leq k$, either $v_i = w_i$ or $(v_i,w_i) \in E$ (see Fig. 6.9).

Now suppose that there is a clique $C \subseteq V$ in G. Then it is easy to verify that

$$\{(v_1,\ldots,v_k) : v_i \in C \text{ for every } 1 \leq i \leq k\}$$

is a clique in $f(G)$ (for example, from the clique $\{v_1,v_2\}$ of graph G of Fig. 6.9 we obtain the clique $\{(v_1,v_1),(v_1,v_2),(v_2,v_1),(v_2,v_2)\}$ of its second product). Since this clique is of size $|C|^k$, we have shown that $m^*(f(G)) \geq (m^*(G))^k$.

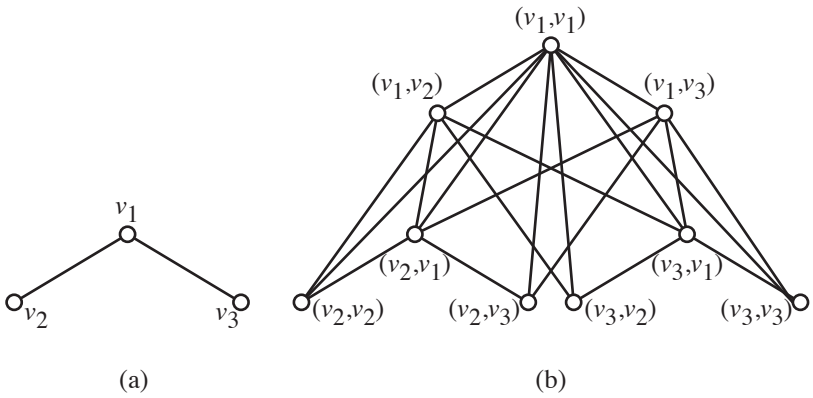

Figure 6.9
The graph product: (a) a graph G and (b) its second product

Next we suppose that there is a clique $C' \subseteq V^k$ in $f(G)$ with at least m^k vertices, for some integer m. There must then be a coordinate i between 1 and k such that, in the vertices of C' written as k-tuples, there are at least m different elements in coordinate i. Each such element corresponds to a vertex in V, and moreover these elements must form a clique in V. Thus, given a clique of size $|C'|$ in $f(G)$, we can use this procedure to find a clique C of size at least $|C'|^{1/k}$ in G. Call this procedure g.

The new approximation algorithm that we get by applying g to the clique obtained by running the approximation algorithm $\mathcal{A}$ on the constructed graph $f(G)$ has the following performance ratio:

$$\frac{m^*(G)}{m(G, g(\mathcal{A}(f(G))))} \leq \left(\frac{m^*(f(G))}{m(f(G), \mathcal{A}(f(G)))}\right)^{1/k} \leq \alpha^{1/k}.$$

That is, for any $r > 1$, if we choose $k \geq \log\alpha / \log r$, then we have a polynomial-time r-approximate algorithm for MAXIMUM CLIQUE. But Theorem 6.4 said that MAXIMUM CLIQUE is not in PTAS unless P=NP. Thus, unless P=NP, we have a contradiction.

QED

Theorems 6.3 and 6.5 are two examples of non-approximability results that can be derived from the PCP theorem. The following chapter will be devoted to the proof of this theorem while in Chap. 8 we will see other non-approximability results for large classes of NP optimization problems that do not require the explicit use of the PCP theorem but are an indirect consequence of it.

6.5 Exercises

Exercise 6.1 Prove that the language recognized by the Turing machine of Example 6.1 consists of all strings over the alphabet $\{a, b\}$ with at least