CHAPTER 17

Approximation

Many optimization and counting problems are inherently difficult and we studied descriptive classifications in chapter 14. It is natural to ask if these problems remain difficult if we relax the exactness of the solution. In this chapter we define the notions of *approximation* for optimization and counting problems and show that some of these problems are easy to approximate while some others remain hard.

We follow the descriptive point of view and give sufficient conditions for such problems to be approximable. We also prove, using PCP techniques that some problems are not approximable.

The approximation of optimization problems is studied in the first section while the approximation of counting problems is studied in the second section. In the third section, we introduce *property testing*, an approximation of verification problems.

1. Optimization problems

In chapter 14, we defined an optimization problem, but recall the definition. An optimization problem specifies a set of possible solutions which satisfy certain constraints, and a cost function which associates a numeric value to each solution. The goal is to find an optimum value, maximum or minimum, for this cost function.

DEFINITION 17.1. *An* **optimization problem** *is a triplet* $(Sol,\ val,\ goal)$ *such that:*

- *For every input* x , $Sol(x)$ *is the set of possible solutions.*
- *The function* val *associates with every possible solution an integer value.*
- *The variable* $goal = max$ *(resp.* $goal = min$*) for maximization problems (resp. minimization problems).*

An **optimum** *of the optimization problem is an element* $y^* \in Sol(x)$ *such that* $val(y^*) = Max\{\ val(y) :\ y \in\ Sol(x)\}$ *if* $goal = max$ *for maximization, or* $val(y^*) = Min\{\ val(y) :\ y\ \in Sol(x)\}$ *if* $goal = min$ *for minimization. The value of the function* val *on the optimum is noted* $opt(x)$.

The parameter *goal* simply determines if the goal is to maximize or to minimize the function val.

Example. In the case of the problem CLIQUE, the input x is a graph of size n, $Sol(x)$ is the set of cliques, $val(y) =\mid y \mid$ i.e. the cardinality of a clique y and

$goal = max$ because we look for the size of the largest clique. This problem is called polynomially bounded because $val(y)$ is smaller or equal than n.

DEFINITION 17.2. *An optimization problem is in the class* NPO *if:*

- *There is a polynomial p such as for any $y \in Sol(x)$, $\mid y \mid \leq p(\mid x \mid)$.*
- *The decision problem $y \in Sol(x)$ is decidable in polynomial time.*
- *The function val is computable in polynomial time.*

Notice that the CLIQUE problem is in the class NPO. An algorithm A for an optimization problem takes an input x and computes a value, y denoted by $A(x)$.

DEFINITION 17.3. *An algorithm approximates up to ϵ an optimization problem if it computes for any input x a solution y such that:*

$$\frac{\mid val(y) - opt(x) \mid}{Max\{opt(x), val(y)\}} \leq \epsilon$$

An optimization problem is ϵ-approximable if there exists $\epsilon < 1$ and a polynomial time algorithm which approximates it up to ϵ.

In the case of a maximization problem, the error between $A(x)$ and $opt(x)$ is smaller than $\epsilon.opt(x)$ and $A(x) \geq (1-\epsilon).opt(x)$, as indicated in the figure below.

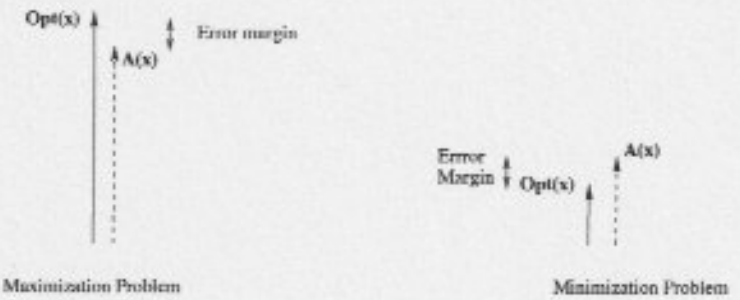

FIGURE 17.1. Approximation of a maximization or minimization problem.

The definition of an approximation can be generalized by considering ϵ dependent on n, the size of the input, for example $\epsilon(n) = \log n$.

DEFINITION 17.4. *The class* APX *is the class of* NPO *problems such that there exists ϵ for which the problem is ϵ-approximable. The class* PTAS *is the class of the problems which are ϵ-approximable for all ϵ.*

For a problem of the class PTAS, the time of computation of the algorithm A is polynomial in $\mid x \mid$, but depends on ϵ. If the time of computation depends polynomially on $\frac{1}{\epsilon}$, we call such a schema a FPTAS, or Fully Polynomial Time Approximation Schema.

1.1. Examples of approximation algorithms. Let us consider some NPO problems which show different situations with respect to the approximation:

- the VC problem, the minimum Vertex Cover, is $\frac{1}{2}$-approximable ,
- the TSP problem, or Traveling Salesman Problem, is not approximable, unless $\mathrm{P} = \mathrm{NP}$,
- the KNAPSACK problem is ϵ-approximable for any ϵ.

We also study the MAXSAT problem which plays an important role and which is approximable. The decisions problems associated with these optimization problems are all NP-complete and yet completely different from the approximation point of view. We show how logical definability illustrates these different situations.

1.1.1. *Vertex cover:* VC. The input is a graph $G = (D_n, E)$ and the problem consists in finding a set of nodes C of *minimum* size such that any edge $e \in E$ has its origin or its extremity in C.

One may try to select first the nodes of maximum degree to minimize the number of nodes spanning the edges. It is interesting to notice that this procedure does not approximate the VC problem up to a constant. Consider the following procedure which will approximate VC: take an edge $e = (u, v)$, add $\{u, v\}$ to the cover C and remove the points u, v from the graph. When all the nodes are removed, we end up with a set C which approximates the minimum cover. Formally, we can describe this procedure with the following instructions.

Approximation of a minimum cover:
1. $C := \emptyset, G_0 := G$, $E_0 := E$, $V_0 := D_n$.
2. If $e = (u, v) \in E_i$ *add* u, v *to* C *and* $V_{i+1} := V_i - \{u, v\}$, $G_{i+1} := (V_{i+1}, E_{i+1})$ *where* $E_{i+1} := E_i \restriction V_{i+1}$.

Which approximation ratio can be guaranteed? Let us show that this algorithm approximates the VC problem up to $\frac{1}{2}$. It chooses $\frac{|C|}{2}$ edges (u, v) of the graph whose extremities define the cover. Any cover C' has to cover the edges of C, in particular the edge (u, v) and has to contain at least one of the nodes u or v, in particular the minimum cover C_{min}. We conclude that:

$$|\, C_{min} \,| \geq \frac{|\, C \,|}{2}$$

$$\frac{|\, C \,| - |\, C_{min} \,|}{|\, C \,|} \leq \frac{|\, C \,| - \frac{|C|}{2}}{|\, C \,|} \leq \frac{1}{2}$$

We conclude that the cover C obtained by this algorithm approximates up to $\frac{1}{2}$ the VC problem.

1.1.2. *The traveling salesman:* TSP. Given a graph and a cost function which associates with every edge e a weight c_e, the **traveling salesman** consists in finding an hamiltonian[1] circuit of minimum cost. An important observation, known since [**Joh74**], is that the existence of an approximation algorithm for TSP implies that P = NP. Let us call ϵ-TSP the problem which consists in finding a solution ϵ-close or close up to ϵ to the TSP problem.

THEOREM 17.1. *For all* $0 \leq \epsilon \leq 1$, *if there is an algorithm which approximates* TSP *close to* ϵ, *then* P = NP.

Proof : We show how to reduce an NP-complete problem, hamiltonicity, to the approximation of TSP. To decide HAM on G_n, it is enough to build a new graph G'_n in polynomial time and to solve ϵ-TSP on G'_n. Let G'_n the complete graph with n nodes where the cost of every edge of G_n is 1 and the cost of all the other edges is $\frac{n}{1-\epsilon}$. The cost of the new edges of G'_n is greater than n.

Let us decide HAM according to the following simple principle: if the algorithm which approximates up to ϵ gives a solution of cost n, then there is an hamiltonian circuit in G_n. If the solution is of cost greater than n, at least one of the edges is of cost $\frac{n}{1-\epsilon}$ and the cost of the circuit is much larger than n. If an ϵ-TSP algorithm approaches TSP close to ϵ, the cost of the optimal solution is greater than $1-\epsilon$ times the cost of the solution, i.e. greater than $(1-\epsilon).\frac{n}{(1-\epsilon)}$, that is greater than n. It can not be an hamiltonian in G_n and there is no hamiltonian circuit in G_n. □

The previous reduction introduces a big variation between the solution and the optimal solution. The approximation algorithm would allow to decide the original NP-complete problem.

1.1.3. KNAPSACK. The input of the KNAPSACK problem is a sequence of n pairs (c_i, w_i) of integers and an integer K. One interprets the value c_i as the cost of the object i, the value w_i as the volume of the object i and K as the capacity of the knapsack. The goal is to find $S \subseteq \{1, 2, ..., n\}$ such that:

$$\sum_{i \in S} w_i \leq K$$

$$\sum_{i \subset S} c_i \; is \; maximum$$

The first condition forces the solution S to satisfy the volume constraint. The second condition is the optimization condition: one looks for a solution S which maximizes the total cost. Another useful notation consists in writing $\sum_{i=1}^{n} x_i.w_i \leq K$ for the volume constraint and $Max\{\sum_{i=1}^{n} x_i.c_i\}$ for the function to be maximized where $x_i = 1$ if $i \in S$ and $x_i = 0$ if $i \notin S$.

[1] An hamiltonian circuit goes through each node of the graph exactly once. The decision problem Hamiltonicity, or HAM, decides if there exists an hamiltonian circuit in a graph G.

An important observation is the existence of a **pseudo-polynomial** approximation algorithm. The time of the algorithm is polynomial in n and in the numerical values of the input, i.e. (c_i, w_i, K) in the case of the knapsack.

PROPOSITION 17.1. *There is an algorithm for the* KNAPSACK *problem in time* $O(n.K)$.

Proof : Let us define the binary function C such that

$$C(w, I) = J$$

if J is the largest cost $(\sum_{i \in S} c_i)$ of a set $S \subseteq \{1, 2, ..., I\}$ such that $\sum_{i \in S} w_i = w$. One can compute $C(w, I)$ for $w \leq K$ and $I \leq n$ with the equations:

$$C(w, 0) = 0$$
$$C(w, I+1) = max\{C(w, I), c_{i+1} + C(w - w_{i+1}, I)\}$$

We express the fact that $C(w, I+1)$ is defined from $C(w, I)$ by considering the maximum solution among all the solutions which include the object $I+1$ or don't. After $n.K$ computation steps, the value we are looking for is in $\{C(1, n), ..., C(K, n)\}$. □

It is important to observe that this procedure does not constitute a polynomial time algorithm for the Knapsack problem. The size of the input is $m = \sum_{i=1,...,n} \log c_i + \sum_{i=1,...,n} \log w_i + \log K$. A polynomial time algorithm must have a time complexity polynomial in $\log K$. The solution above is called a *pseudo-polynomial* algorithm, a procedure whose time complexity is polynomial in n, the number of variables, and in the numerical values. A variation of this technique will produce an approximation schema, i.e. a FPTAS.

THEOREM 17.2. *For any* ϵ, *there is an algorithm which approximates up to* ϵ *the* KNAPSACK *problem.*

Proof : Let us define the binary function:

$$W(c, I) = J$$

where J is the smallest volume $(\sum_{i \in S} w_i)$ of a set $S \subseteq \{1, 2, ..., I\}$ such that $\sum_{i \in} c_i = c$. One can compute $W(c, I)$ for $c \leq n.Max\{c_1, c_2, ..., c_n\}$ and $I \leq n$ with the equations:

$$W(c, 1) = w_1 \text{ if } c = c_1$$
$$W(0, I) = 0$$
$$W(c, I+1) = min\{W(c, I), w_{i+1} + W(c - c_{i+1}, I)\}$$

for $I \leq n$ and $c \leq n.C$, where $C = Max\{c_1, c_2, ..., c_n\}$. These equations express $W(c, I+1)$ as the minimum of the volumes of the sets which include the object $I+1$ or don't include it. One chooses the largest c such that $W(c, n) \leq K$ and the algorithm is in $O(n^2.C)$.

This algorithm is pseudo-polynomial because the time of computation is proportional to C while the size of the input is of the order $O(\log C)$. One can imagine to divide every c_i by 2^b for a b fixed in advance: this means taking into account only the most significant bits by eliminating the b least significant bits for every value c_i coded in binary. Let us define

$c'_i = 2^b.d_i$ where $d_i = \lfloor \frac{c_i}{2^b} \rfloor$ and the input $x' = (w_1, ..., w_n, K, d_1, ..., d_n)$. The solution found is the same as the solution on the input $(w_1, ..., w_n, K, c'_1, ..., c'_n)$. The previous algorithm is of complexity $O(n^2.C/2^b)$ and finds a solution S':

$$\sum_{i \in S} c_i \geq \sum_{i \in S'} c_i \geq \sum_{i \in S'} c'_i \geq \sum_{i \in S} c'_i \geq (\sum_{i \in S} c_i) - n.2^b$$

The first inequality uses the fact that S is optimal, the second inequality the fact that $c_i \geq c'_i$, the third inequality the fact that S' is optimal and the fourth the fact that $c'_i \geq c_i - 2^b$. Considering the second and the fourth expression, one deduces:

$$\sum_{i \in S} c_i - \sum_{i \in S'} c_i \leq n.2^b$$

$$\frac{\sum_{i \in S} c_i - \sum_{i \in S'} c_i}{\sum_{i \in S} c_i} \leq \frac{\sum_{i \in S} c_i - \sum_{i \in S'} c_i}{C} \leq \frac{n.2^b}{C}$$

The algorithm approaches up to ϵ for $\epsilon = \frac{n.2^b}{C}$. It is enough to choose $b = \log \frac{\epsilon.C}{n}$ and the time of computation is $O(\frac{n^2.C}{2^b}) = O(\frac{n^3}{\epsilon})$ and we obtain a fully polynomial approximation schema (FPTAS). □

1.2. Approximation of optimization problems. Recall from 14 the KT (Kolaitis-Thakur) hierarchies associated with optimization problems. In this section, we show the important result **[PY91]**: *if an optimization problem is in the class* $\mathrm{MAX}\Sigma_1$, *then it is* ϵ*-approximable*.

It is useful to show certain intermediate lemmas concerning the problem MAXGSAT, a generalization of MAXSAT, where the input bfx is a set of propositional formulas $bfx = \{\phi_1, \ldots, \phi_m\}$ on n variables. These formulas are not supposed clausal.

Example. Consider the 3 formulas below on the 4 variables p_1, p_2, p_3, p_4:

$$\phi_1 \ : \ (p_1 \wedge (p_2 \vee \neg p_3))$$

$$\phi_2 \ : \ ((\neg p_1 \wedge p_2) \vee p_3)$$

$$\phi_3 \ : \ (\neg p_1 \wedge \neg p_4)$$

Let us show that MAXGSAT is ϵ-approximable under certain conditions. Let $t \in \{0,1\}^n$ a boolean model, $Var(\phi_i)$ the set of the propositional variables of ϕ_i, let us note $t_i \in \{0,1\}^{|Var(\phi_i)|}$ a model on the variables of ϕ_i and let us define:

$$nsat(x,t) = \mid \{\phi_i \in x : t \text{ satisfies } \phi_i\} \mid$$

$$\#sat(\phi) = \mid \{t \ : \ t \text{ satisfies } \phi\} \mid$$

$$\sigma(\phi_i) = \frac{\mid \{t_i \ : \ t_i \text{ satisfies } \phi_i\} \mid}{\mid \{0,1\}^{|Var(\phi_i)|} \mid} = \#sat(\phi_i).2^{-|Var(\phi_i)|}$$

By extension, one write:

$$\#sat(x) = \mid \{t \ : \ t \text{ satisfies all the } \phi_i\} \mid$$

Consider now a random model t chosen with an uniform distribution. Let N_i be a random variable such as:

$$N_i = \begin{cases} 1 & \text{if } t \text{ satisfies } \phi_i \\ 0 & \text{otherwise} \end{cases}$$

Let $\widetilde{\mathsf{nsat}}(x)$ the expectation of the random variable $nsat(x,t)$. One can then write:

$$\widetilde{\mathsf{nsat}}(x) = \mathbb{E}(\sum_{i=1}^{m} N_i) = \sum_{i=1}^{m} \mathbb{E}(N_i) = \sum_{i=1}^{m} \sigma(\phi_i)$$

We now show how a greedy algorithm can make sure that the number of satisfied clauses is greater than $\widetilde{\mathsf{nsat}}(x)$.

LEMMA 17.1. *If we can compute the number of models satisfying every ϕ_i in time $T(x)$, we can find a model satisfying more than $\widetilde{\mathsf{nsat}}(x)$ formulas in time $O(n.m.T(x))$.*

Proof : Let us denote $\phi_{i,p_j=1}$ (resp. $\phi_{i,p_j=0}$) the formula ϕ_i where the variable p_j is replaced by the value 1 (resp. 0). Let us notice that:

$$\sigma(\phi_i) = \frac{\sigma(\phi_{i,p_j=1}) + \sigma(\phi_{i,p_j=0})}{2}$$

Consider the following procedure to choose the values of $p_1, ..., p_n$:

Decision of $p_1, ..., p_n$:
For $j := 1, ..., n$
1. $p_j = 1$ if $\sum_{i=1}^{m} \sigma(\phi_{i,p_j=1}) \geq \sum_{i=1}^{m} \sigma(\phi_{i,p_j=0})$. Otherwise $p_j = 0$.
2. Simplify the formulas ϕ_i according to the value affected to p_j.

Suppose we choose $p_1 = 1$ and let $x_1 = \{\phi_{i,p_1=1}\}$, that is the new input where the boolean value 1 replaces the variable p_1 in all the formulae ϕ. From the remark above, $\widetilde{\mathsf{nsat}}(x_1)$ is definable from the densities $\sigma(\phi_i)$:

$$\widetilde{\mathsf{nsat}}(x_1) = \sum_{\phi \in x_1} \sigma(\phi) = \sum_{\phi \in x} \sigma(\phi_{p_1=1})$$

$$\widetilde{\mathsf{nsat}}(x_1) = \frac{\sum_{\phi \in x} \sigma(\phi_{p_1=1}) + \sum_{\phi \in x} \sigma(\phi_{p_1=1})}{2} \geq \frac{\sum_{\phi \in x} \sigma(\phi_{p_1=1}) + \sum_{\phi \in x} \sigma(\phi_{p_1=0})}{2}$$

$$\widetilde{\mathsf{nsat}}(x_1) \geq \sum_{\phi \in x} \sigma(\phi) = \widetilde{\mathsf{nsat}}(x)$$

We simply expressed that $\sum_{i=1}^{m} \sigma(\phi_{i,p_1=1}) \geq \sum_{i=1}^{m} \sigma(\phi_{i,p_1=0})$. Let x_i the new set of formulas obtained from x_{i-1}, following the same procedure applied to the variable p_i. We then obtain:

$$\widetilde{\mathsf{nsat}}(x_n) \geq \widetilde{\mathsf{nsat}}(x_{n-1}), ..., \geq \widetilde{\mathsf{nsat}}(x_1) \geq \widetilde{\mathsf{nsat}}(x)$$

Let t^* a boolean model obtained by this procedure. The value $\widetilde{\widetilde{\mathsf{nsat}}}(x_n)$ is directly computable, by evaluating the number of formulas ϕ such as $\phi(t^*) = 1$. This procedure allows to find t^* in time $O(n.m.T(x))$ if we compute each density $\sigma(\phi)$ in time $O(T(x))$. □

Example. For the previous formulas ϕ_1, ϕ_2, ϕ_3, the values of the densities are:

density	$p_1 = 1$	$p_1 = 0$
$\sigma(\phi_1)$	0.75	0
$\sigma(\phi_2)$	0.5	0.75
$\sigma(\phi_3)$	0	0.5
$\sum_i \sigma(\phi_i)$	1.25	1.25

We conclude that $\widetilde{\widetilde{\mathsf{nsat}}}(x_1) = 1.25$. The procedure chooses $p_1 = 1$, but $p_1 = 0$ could also have been chosen. We obtain the new input $x_1 = \phi_1^1, \phi_2^1, \phi_3^1$:

$$\phi_1^1 \ : \ p_2 \vee \neg p_3$$

$$\phi_2^1 \ : \ p_3$$

$$\phi_3^1 \ : \ 0$$

The values of densities for the choices of p_2 are:

density	$p_2 = 1$	$p_2 = 0$
$\sigma(\phi_1^1)$	1	0.5
$\sigma(\phi_2^1)$	0.5	0.5
$\sigma(\phi_3^1)$	0	0
$\sum_i \sigma(\phi_i^1)$	1.5	1

In this case $p_2 = 1$ and we obtain $x_2 = \phi_1^2, \phi_2^2, \phi_3^2$:

$$\phi_1^2 \ : \ 1$$

$$\phi_2^2 \ : \ p_3$$

$$\phi_3^2 \ : \ 0$$

The values of the densities for the choices of p_3 are:

density	$p_3 = 1$	$p_3 = 0$
$\sigma(\phi_1^2)$	1	1
$\sigma(\phi_2^2)$	1	0
$\sigma(\phi_3^2)$	0	0
$\sum_i \sigma(\phi_i^2)$	2	1

The variable p_3 is chosen as $p_3 = 1$ and the model is $t^* = (1, 1, 1)$ which satisfies two formulas of the input x.

In two important cases, we can compute the number of models of a formula in polynomial time:

- when each formula has at most k variables,
- when each formula is a disjunction.

In each case, the previous algorithm allows us to approximate MAXGSAT. We describe the argument when the number of variables is at most k. The problem MAXGSAT is then noted $\mathsf{MAXGSAT}_k$.

THEOREM 17.3. *The problem* MAXGSAT$_k$ *is* ϵ*-approximable.*

Proof : Every formula ϕ which can be satisfied is such that $\sigma(\phi) \geq \frac{1}{2^k}$. Let $x' \subseteq x$ the subset of formulas of x which can be independently satisfied and let $m = \mid x' \mid$.
By linearity of the expectation, we obtain: $\widetilde{\mathsf{nsat}}(x) \geq m.2^{-k}$. The previous algorithm finds t^* such that

$$nsat(x, t^*) \geq \widetilde{\mathsf{nsat}}(x) \text{ and } m \leq MaxGSAT_k(x)$$

We deduce that

$$\frac{nsat(x, t^*)}{MaxGSAT_k(x)} \geq \frac{\widetilde{\mathsf{nsat}}(x)}{m} \geq 2^{-k}$$

$$\frac{MaxGSAT_k(x) - nsat(x, t^*)}{MaxGSAT_k(x)} \leq 1 - 2^{-k}$$

and the approximation condition is satisfied. □

The approximation is better when k is small. In the case of $k = 3$, the error $\epsilon = 7/8$ and $nsat(x, t^*) \geq MaxGSAT_k(x)/8$. Let us show how to generalize this procedure to a MAXΣ_0 problem.

THEOREM 17.4. *If a problem is in the class* MAXΣ_0*, it is* ϵ*-approximable.*

Proof : Let A be an optimization problem of the class MAXΣ_0 and let $\psi(\mathbf{y}, S)$ the formula Σ_0 (without quantifiers) such that for any structure U of size n:

$$MaxA(x) = Max_S \mid \{\mathbf{a} : (U, S, \mathbf{a}) \models \psi(\mathbf{a}, S)\} \mid$$

The formula ψ is built from the explicit relations of U and $S(y_{i_1}, ..., y_{i_s})$ where s is the arity of S and y_{i_j} is either a constant of the structure U or a variable among $\mathbf{y} = y_1, ..., y_p$. Let us introduce n^s propositional variables: $p_{1,...,1}$ for $S(1, ..., 1)$, $p_{1,...,2}$ for $S(1, ..., 2)$, ... $p_{1,...,n}$ for $S(1, ..., n)$, ... $p_{n,...,n}$ for $S(n, ..., n)$. For every formula $\psi(a_1, ..., a_p, S)$ let us write $\psi_{[a_1,...,a_p]}$ where $1 \leq a_j \leq n$, the propositional formula obtained by substituting every y_j by a_j, replacing the explicit relations on U by their thuth values and the instances of $S(y_{i_1}, ..., y_{i_s})$ by the corresponding propositional variables. We obtain n^p propositional formulas having each at most k_ψ variables, where k_ψ is the number of distinct occurrences of S in ψ.

To maximize on S the size of the set defined by $\psi(S, \mathbf{y})$ is equivalent to the problem $MaxGSAT_{k_\psi}$ on the input $\{\psi_{[a_1,...,a_p]} : 1 \leq a_j \leq n\}$. Every formula has at most k_ψ variables and the problem is $1 - 2^{-k_\psi}$ approximable by the previous result. □

Let us show that there is an algorithm which approximates up to ϵ the maximum cut problem (MAXCUT).

Example. The MAXCUT problem consists in finding a partition of the nodes into two sets, one containing s the other containing t such that the number of edges between the two sets is maximal. It is definable by the formula $\psi(x, y)$:

$$E(x, y) \wedge [(S(x) \wedge \neg S(y)) \vee (S(y) \wedge \neg S(x))]$$

$$MaxCUT = Max_S \mid \{(x, y) : (U, S) \models \psi(x, y, S)\} \mid$$

Let us apply the previous transformation with $k_\psi = 2$. We obtain n^2 formulas $\psi_{[a_1,a_2]}$ on the n variables $p_1, ..., p_n$ of the type:

$$E(a_1, a_2) \wedge ((S(a_1) \wedge \neg S(a_2)) \vee (S(a_2) \wedge \neg S(a_1)))$$

which can also be written as:

$$E(a_1, a_2) \wedge [(p_{a_1} \wedge \neg p_{a_2}) \vee (p_{a_2} \wedge \neg p_{a_1})]$$

If $(U, a_1, a_2) \models \neg E(a_1, a_2)$, i.e. if there is no edge between a_1 and a_2, the formula $\psi_{[a_1,a_2]}$ is false. If $(U, a_1, a_2) \models E(a_1, a_2)$, i.e. if there is an edge between a_1 and a_2, the formula $\psi_{[a_1,a_2]}$ becomes:

$$(p_{a_1} \wedge \neg p_{a_2}) \vee (p_{a_2} \wedge \neg p_{a_1})$$

Notice that the value of MAXCUT is exactly the value of the function $MaxGSAT(\mathbf{x}')$. The set $\mathbf{x}'$ of formulas to be satisfied is determined by m formulas of this type if m is the number of edges of the graph. The algorithm finds a cut with an approximation of $1 - 2^{-2} = 3/4$.

We can now obtain the main result of [**PY91**].

THEOREM 17.5. *If an optimization problem is in the class* $\mathrm{MAX}\Sigma_1$, *then it is* ϵ*-approximable.*

Proof : Let A be an optimization problem of the class $\mathrm{MAX}\Sigma_1$. There is a formula $\exists y \psi(z, y, S)$ such that:

$$MaxA(x) = Max_S \mid \{z : (U,\ S, z) \models \exists y \psi(z, y, S)\} \mid$$

Without loss of generality, let us suppose that the formula has only one free variable z and only one quantified variable y. Let us consider all the z such that $z = i$ where $1 \leq i \leq n$ and the corresponding formula $\psi_i :\ \exists y\, \psi(i, y, S)$. As in the previous construction, we generate a set x' of formulas which form the input of $\mathsf{MAXGSAT}_k$ on the same propositional variables but contrary to the previous case,

$$MaxGSAT(x') \leq A(x)$$

whereas we had an equality in the case of Σ_0 formulas. For each formula ψ_i, we consider two cases.

Either there is no value for y which makes $\exists y \psi(i, y, S)$ true on the structure $(U,\ S,\ i)$ and we dont write any propositional formula, or there are values for y and in this case we choose the first value $y = j^*$ in the lexicographical order (any other choice would also work). We obtain a formula $\psi(i, j^*, S)$ which we write with propositional variables as before.

The propositional variable $\psi_{[i,j^*]}$ is one of the formulas of the new input x'. Let k the number of different occurrences of S in $\psi(z, y, S)$. The approximation algorithm for $\mathsf{MAXGSAT}_k$ gives us a value α such that:

$$2^{-k}. \mid x' \mid \leq \alpha \leq MaxGSAT(x') \leq A(x) \leq \mid x' \mid$$

The important inequality is $A(x) \leq \mid x' \mid$. Indeed $A(x)$ is smaller than the number of clauses which can be satisfied. During the arbitrary selection of j^*, the choices made may

not be the best and $MaxGSAT(x') \leq A(x)$. The first two inequalities directly express the approximation of MAXGSAT$_k$. We conclude that:

$$\alpha \geq 2^{-k}.A(x)$$

$$\frac{A(x) - \alpha}{A(x)} \leq 1 - 2^{-k}$$

□

Example. Consider problem MAXSAT, which is MAXΣ_1 definable, on the input x with the following 3 clauses on the variables p_1, p_2, p_3, p_4:

$$(p_1 \vee p_2 \vee \neg p_3 \vee p_4)$$

$$(p_2 \vee \neg p_4)$$

$$(\neg p_1 \vee p_4)$$

The structure is $U_7 = (D_{n=7}, C, POS, NEG)$ where $1, 2, 3, 4$ are the 4 boolean variables and $5, 6, 7$ the 3 clauses. Relations are $C = \{5, 6, 7\}$,
$POS = \{(5,1), (5,2), (5,4), (6,2), (7,4)\}$ and $NEG = \{(5,3), (6,4), (7,1)\}$. The SAT problem is defined by :

$$\exists T[\forall z \exists y(C(z) \rightarrow ((POS(z,y) \wedge T(y)) \vee (NEG(z,y) \wedge \neg T(y))))]$$

The $MaxSAT(x)$ function is then defined by:
$Max_T \mid \{z \ : \ \exists y(C(z) \wedge ((POS(z,y) \wedge T(y)) \vee (NEG(z,y) \wedge \neg T(y))))\} \mid$
The formula $\psi(i, j^*)$ is: $(POS(i, j^*) \wedge T(j^*)) \vee (NEG(i, j^*) \wedge \neg T(j^*))$ if the formula can be true (for a choice of T) and its propositional representation is $\psi_{[i,j^*]}$:

$$(POS(i, j^*) \wedge p_j) \vee (NEG(i, j^*) \wedge \neg p_j)$$

Recall that p_i is the propositional variable associated with $T(i)$. For example:

$$\psi_{[5,1]} : p_1$$

$$\psi_{[5,2]} : p_2$$

$$\psi_{[7,1]} : \neg p_1$$

In this case $\mid x' \mid = 3$ and $A(x) = 3$, $MaxGSAT_2(x') = 2$ and the previous inequality is satisfied.

We can conclude with the figure which gives the position of the class APX in the KT maximization hierarchy.

1.3. Reductions and completeness. In this section, we introduce a notion of *reduction* between two optimization problems which preserves the approximation. The most classical reduction is the Lin-reduction or *linear* reduction, defined in **[PY91]**. Other more general reductions were introduced later, in particular the PTAS-reduction of **[ACP95]** which we now study.

Let us recall that if x is the input and y a solution for x, the relative error $E(x, y)$ is defined by:

$$E(x, y) = \frac{\mid opt(x) - val(y) \mid}{Max\{opt(x), val(y)\}}$$

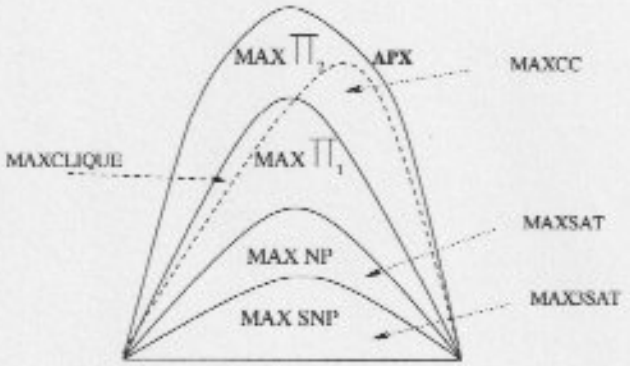

FIGURE 17.2. The position of the class APX in the KT hierarchy.

Let A and B two optimization problems with inputs x_A and x_B which have solutions $y_A \in Sol(x_A)$ and $y_B \in Sol(x_B)$ whose optima are $val(y_A)$ and $val(y_B)$. The parameter ϵ is the relative error for the problem A.

DEFINITION 17.5. *Let A and B two optimization problems in the class* NPO. *We say that A is* PTAS*-reducible to B and write $A \leq_{PTAS} B$ if there are two functions f and g such that:*

- *the function f is computable in polynomial time in $\mid x_A \mid$ and $1/\epsilon$,*
- *the function g is computable in polynomial time in $\mid x_A \mid, \mid y_B \mid$ and $1/\epsilon$, and the solution y_A is defined by the value of $g(x_A, y_B, \epsilon)$,*
- *there is a constant β smaller than 1 which depends on ϵ such that*

$$E(x_A, g(x_A, y_B, \epsilon)) \leq \beta . E(f(x_A, \epsilon), y_B) \leq \epsilon$$

The function f takes an instance of A and associates an instance of B, while the function g takes a solution of B and associates a solution of A, as the figure below indicates. The error for the problem A is smaller than the error for the problem B and can be made smaller than a given ϵ. An Lin-reduction, noted $\leq_L$, is a PTAS-reduction where the functions f and g do not depend on ϵ.
The fundamental property of an Lin-reduction is that it preserves the approximation.

LEMMA 17.2. *If $B \in PTAS$ and $A \leq_L B$, then $A \in PTAS$.*

Proof : For any ϵ, we associate $x_B = f(x_A, \epsilon)$. Because B admits a PTAS, one can bound the error on B such that $E(f(x_A), y_B) \leq \frac{\epsilon}{\beta}$. This insures that the error on A, i.e. $E(x_A, g(x_A, y_B, \epsilon)) \leq \epsilon$. All the computations are polynomially bounded and the problem A is in the class PTAS. □

DEFINITION 17.6. *An optimization problem A is $\mathrm{MAX}\Sigma_0$-complete if $A \in \mathrm{MAX}\Sigma_0$ and if any problem $B \in \mathrm{MAX}\Sigma_0$ is Lin-reducible to A.*

We say that a problem is $\mathrm{MAX}\Sigma_0$-hard if any problem $B \in \mathrm{MAX}\Sigma_0$ is Lin-reducible to A.

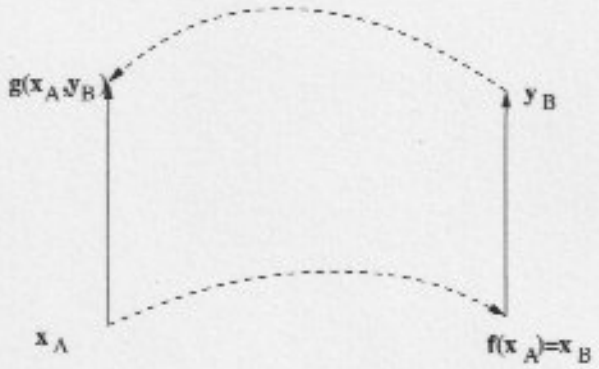

FIGURE 17.3. *Lin*-reduction between two optimization problems.

THEOREM 17.6. MAX3SAT *is* $\mathrm{MAX}\Sigma_0$*-complete.*

Proof : In the previous section, we showed (theorem 17.4) that any problem A of the class $\mathrm{MAX}\Sigma_0$ was *Lin*-reducible to the problem $\mathrm{MAXGSAT}_k$. Let us show how to reduce $\mathrm{MAXGSAT}_k$ to MAX3SAT by an *Lin*-reduction.

Let x_A be the input of the problem A, i.e. the finite structure U. Every formula ψ_j of $\mathrm{MAXGSAT}_k$ is of constant length and contains at most k variables. There is an equivalent formula φ_j which contains m clauses. One can so associate to the formulas ψ_j of $\mathrm{MAXGSAT}_k$ a set of clauses $x_{3SAT} = f(x_A)$. If an approximation algorithm for MAX3SAT approximates x_{3SAT} within a constant ratio, it finds a model $\sigma = y_{3SAT}$. This model σ defines a set $S = g(\sigma)$ which gives an estimate for the problem $\mathrm{MAXGSAT}_k$ and the problem A within a constant ratio. Hence there exists β such that:

$$E(x_A, g(y_{3SAT})) \leq \beta . E(f(x_A), y_{3SAT})$$

and the problem MAX3SAT is complete. □

There are many other optimization problems which are $\mathrm{MAX}\Sigma_0$-complete but the reductions are much more elaborate. In particular, the problem MAXCUT is $\mathrm{MAX}\Sigma_0$-complete.

An optimization problem is *polynomially bounded* if there exists a polynomial q such that the function $val(y) \leq q(\mid x \mid)$. We write APX_{PB} the class of polynomially bounded optimization problems. The classes APX and APX_{PB} have been logically characterized by **[KMSV94]**.

THEOREM 17.7. *Any problem A of the class* APX_{PB} *is Lin-reducible to* MAX3SAT. *Any problem B of the class* APX *is* PTAS*-reducible to* MAX3SAT.

1.4. Non Approximability. In this section we prove two results of non approximability, under the hypothesis $\mathrm{P} \neq \mathrm{NP}$. We use the characterization of NP in term of the probabilistic class $PCP(\log n, 1)$ defined in the previous chapter.

We show the impossibility to approximate MAXCLIQUE up to ϵ and the impossibility to approximate MAX3SAT with an approximation schema. We deduce from it the impossibility to obtain an approximation schema for any $\mathrm{MAX}\Sigma_0$-complete problem.

1.4.1. *Non Approximability of* MAXCLIQUE. Let $clique(G)$ the size of the largest clique in a graph G_n.

THEOREM 17.8. *If there is a polynomial time algorithm which approximates up to ϵ the problem* MAXCLIQUE, *then* $\mathrm{P} = \mathrm{NP}$.

Proof : Let L a language of the class NP and V a verifier of the class $\mathrm{PCP}(\log n, 1)$ for L. For each input x of L, let us construct a graph G_x and a function f such that if $x \in L$, then $clique(G_x) = f(n)$ and if $x \notin L$, then $clique(G_x) \leq f(n).(1 - 2\epsilon)$. If there were a polynomial time algorithm which would approximate up to ϵ the problem MAXCLIQUE, one could decide L in polynomial time and one would conclude that $\mathrm{P} = \mathrm{NP}$.

Let $c = O(1)$ be the number of bits tested and $r = d.\log n$ the number of random bits of the PCP protocol for L. A node of G is a tuple $\alpha = (\rho, s_1, ..., s_c)$ where ρ is a random vector ($\mid \rho \mid = r$), where $s_1, ..., s_c$ are the bits tested in the proof π when the verifier V accepts. The number of nodes is $O(n^d)$, hence polynomial.

The algorithm of the verifier computes the positions $p_1, ..., p_c$ depending on x and ρ for which the values of π at the positions $p_1, ..., p_c$ are $s_1, ..., s_c$. Two nodes α and α' are *contradictory* if there are positions p_i and p'_j such as $p_i = p'_j$ and $s_i \neq s'_j$. An edge connects two nodes of the graph if they are not contradictory. Observe that if two nodes $(\rho, s_1, ..., s_c)$ and $(\rho, s'_1, ..., s'_c)$ are connected, then $s_i = s'_i$ for $i = 1, ..., c$. The verifier will compute in both cases the same positions $p_1, ..., p_c$. One can determine in polynomial time if there is an edge between two nodes by considering all the possible ρ.

The size of the largest clique of G_x is 2^r. If there were a larger clique, two tuples $(\rho, s_1, ..., s_c)$ and $(\rho, s'_1, ..., s'_c)$ where at least one $s_i \neq s'_i$ would be connected with an edge, which is impossible because they are contradictory. Let us show that if $x \in L$ then $clique(G) = 2^r$ and that if $x \notin L$ then $clique(G) \leq 2^r / K$ for a K chosen according to ϵ.

If $x \in L$, there is a transparent proof which accepts for all the possible choices of ρ. Every node is compatible for the various ρ and we obtain a clique of size 2^r. Let us show that if $x \notin L$, the largest clique is of size $\frac{2^r}{K}$ for $K > 1$. If there were a clique of size greater than $\frac{2^r}{K}$, one could build a proof π which would answer the questions of the verifier with values $s_1, ..., s_c$ if $(\rho, s_1, ..., s_c)$ is in the clique. For a random choice of ρ, the probability that the point falls in the clique is $\frac{2^r}{K}.\frac{1}{2^r} = \frac{1}{K}$ and the probability that V accepts π would be greater than $\frac{1}{K}$, contradicting the hypothesis of PCP according to which if $x \notin L$, for all π, the probability to accept is smaller than a constant value between 0 and 1.

For any language L of the class NP, we built a graph G such as if $x \in L$, then $clique(G) = f(n) = 2^r$ and if $x \notin L$, then $clique(G) \leq f(n).(1 - 2\epsilon)$ if $K = \frac{1}{1-2\epsilon}$. If there were

a polynomial time algorithm which approximates up to ϵ the problem MAXCLIQUE, one could decide L in polynomial time and one would have $\mathrm{P} = \mathrm{NP}$. □

It is also possible to show that MAXCLIQUE can not be approximated up to n^{ϵ}, but this requires much more elaborated techniques.

1.4.2. *Non existence of a* PTAS *for* $\mathrm{MAX}\Sigma_0$*-complete problems.* We now show that the existence of a schema for the class MAXSNP implies $\mathrm{P} = \mathrm{NP}$. It is customary to conclude that such schemas dont exist, but it remains unproven.

THEOREM 17.9. *If there is an approximation schema (*PTAS*) for a* $\mathrm{MAX}\Sigma_0$*-complete problem, then* $\mathrm{P} = \mathrm{NP}$.

Proof : Let us show that the existence of such schema for MAX3SAT implies that $\mathrm{P} = \mathrm{NP}$. Let L be a language in the class NP and let us show how to reduce its decision problem to the approximation of MAX3SAT with a schema. For any x, we construct an instance S_x of MAX3SAT such that:

- S_x is satisfiable iff $x \in L$
- if $x \notin L$ then $Max3SAT(S_x) \leq \epsilon . \mid S_x \mid$

A PTAS for MAX3SAT could then decide if $x \in L$ and this would give a polynomial time algorithm for L which would imply that $\mathrm{P} = \mathrm{NP}$.

The language L has a verifier $\mathrm{PCP}(\log n, 1)$. Suppose that the content of each position of the proof π is coded by a boolean variable p_i. For any random vector α of length $r = \log n$, let us build the boolean function which expresses the test of the verifier. As a constant number of bits is tested, this function contains a constant number of boolean variables and is of constant length. This function can be expressed with a $3CNF$ formula, noted $S_{x,\alpha}$, and contains k clauses. The formula:

$$S_x = \bigwedge_{\alpha} S_{x,\alpha}$$

contains a number $N = O(n^r)$ of formulas $S_{x,\alpha}$.

If $x \in L$, there is a proof π_x such as V accepts for any random vector α and π_x satisfies S_x and $Max3SAT(S_x) = N.k$.

If $x \notin L$, then for any proof π the verifier V accepts at most $\frac{1}{4}$ of the $S_{x,\alpha}$. At least $\frac{3}{4}$ of the $S_{x,\alpha}$ are not satisfied, i.e. for each of them at least one of the k clauses is false. At least $\frac{3}{4}.N$ of the clauses are false, i.e. $\frac{3}{4.k}$ of the $N.k$ global clauses. At most $(1 - \frac{3}{4.k})$ of them are satisfied.

Hence $Max3SAT(S_x) \leq N.k.(1 - \frac{3}{4.k})$.

We found a constant interval to separate $x \in L$ from $x \notin L$ for the function $Max3SAT(S_x)$. A PTAS for MAX3SAT would allow to distinguish these two situations and to decide in polynomial time if $x \in L$. □

2. Counting problems

The non deterministic class NP admits a generalization in terms of a counting class, introduced in chapter 14. Recall that a function F of $\Sigma^* \to \mathbb{N}$ is in the class $\#P$ if there is a p-predicate R computable in polynomial time and a constant k such as for all $\mathbf{x} \in \Sigma$:

$$F(\mathbf{x}) = \mid \{\mathbf{y} : R(\mathbf{x}, \mathbf{y})\} \mid \text{ and } \mid \mathbf{y} \mid \leq \mid \mathbf{x} \mid^k$$

The counting problems play an important role in complexity theory. We show the connection between the logical definability of the counting problems and the approximation of these problems.

2.1. Approximation of counting problems. We introduce the notion of a randomized approximation schema used to approximate counting functions. Recall the definitions of chapter 14 for these functions and their descriptive hierarchies. The goal is to approximate a counting function $F : \Sigma^* \to \mathbf{N}$ with a randomized algorithm whose relative error is bounded by ϵ with a high probability, for all ϵ.

DEFINITION 17.7. *An algorithm A is a* **polynomial time randomized approximation schema**[2] *or* PRAS *for a function $F : \Sigma^* \to \mathbb{N}$ if for every ϵ and* $\mathbf{x}$,

$$\mathbb{P}rob\{A(\mathbf{x}, \epsilon) \in [(1-\epsilon).F(\mathbf{x}), (1+\epsilon).F(\mathbf{x})]\} \geq \frac{3}{4}$$

and $A(\mathbf{x}, \epsilon)$ stops in polynomial time in $\mid \mathbf{x} \mid$. The algorithm A is an FPRAS *if the time of computation is also polynomial in $1/\epsilon$.*

If the algorithm A is deterministic, one speaks of a PAS and of a FPAS. A PRAS(δ) (resp. FPRAS(δ)), is an algorithm $A(\mathbf{x}, \epsilon, \delta)$ such that:

$$\mathbb{P}rob\{A(\mathbf{x}, \epsilon, \delta) \in [(1-\epsilon).F(\mathbf{x}), (1+\epsilon).F(\mathbf{x})]\} \geq 1 - \delta$$

and whose time complexity is also polynomial in $\log(1/\delta)$. The error probability is less than δ and let us show how to amplify the probability of success from $3/4$ to $1 - \delta$ at the cost of extra computation of length polynomial in $\log(1/\delta)$.

One of the most classical results used for the analysis of probabilistic algorithms is the **Hoeffding-Chernoff's bound [Hoe63]**. It expresses the probability of deviation of the sum of independent Bernouilli variables from its mean (see theorem A 4 of **[AS92a]**).

THEOREM 17.10. *If $X_1, ..., X_n$ are n independent random variables with value 1 with probability p and 0 with probability $1 - p$ and $Y = \sum_{i=1}^{n} X_i$, then:*

$$\mathbb{P}rob[Y - np > a.n.p] < e^{-2a^2 n}$$

LEMMA 17.3. *Let F a function: $F : \Sigma^* \to \mathbb{N}$. If F admits an* PRAS *(resp.* FPRAS*), then F admits an* PRAS(δ) *(resp.* FPRAS(δ)*) algorithm* $\mathsf{A}(C, \epsilon, \delta)$ *whose error probability is δ and whose time complexity is polynomial in $\mid \mathbf{x} \mid$, $\log(1/\delta)$ (and $1/\epsilon$ for an* FPRAS*).*

---

[2] PRAS is the abbreviation of *Polynomial time Randomized Approximation Schema* and FPRAS is the abbreviation of *Fully Polynomial-time Randomized Approximation Schema*.

Proof : Let A' a PRAS (resp. FPRAS) for F and let $X_1, X_2, \ldots, X_K$ the values obtained after K independent runs of A'. Let Z the median value, i.e. the value X_i such that

$$| \{j : X_j \leq X_i\} | - | \{j : X_j \geq X_i\} | | \leq 1$$

which is the new estimated value. Each of the values X_j is in the interval $[F(\mathbf{x}) \cdot (1-\varepsilon), F(\mathbf{x}) \cdot (1+\varepsilon)]$ with probability greater than $3/4$. If at least the majority (i.e. $(K+1)/2$ where K is odd) of the X_j are in this interval, Z is also in the interval. The error probability is:

$$\mathbb{P}rob\left[| \{i : X_i \notin [F(\mathbf{x}) \cdot (1+\varepsilon), F(\mathbf{x}) \cdot (1-\varepsilon)]\} | \geq (K+1)/2 \right] \leq \delta$$

Let Y_i a random variable whose value is 1 if $X_i \notin [F(\mathbf{x}) \cdot (1-\varepsilon), F(\mathbf{x}) \cdot (1+\varepsilon)]$ and 0 otherwise. Observe that $\mathbb{P}rob[Y_i = 1] = 1/4$ and $\mathbb{P}rob[Y_i = 0] = 3/4$. However,

$$\mathbb{P}rob[| \{i : X_i \notin [F(\mathbf{x}) \cdot (1-\varepsilon), F(\mathbf{x}) \cdot (1+\varepsilon)]\} | \geq (K+1)/2] =$$

$$\mathbb{P}rob[\sum_{i=1}^{K} Y_i > (K+1)/2]$$

because K is odd and one can apply the Hoeffding-Chernoff's bound. Hence,

$$\mathbb{P}rob[\sum_{i=1}^{K} Y_i > (K+1)/2] < e^{-2a^2K} = \delta$$

where $a = \frac{K+1}{2K} - \frac{1}{4}$. One concludes from the inequality $e^{-2a^2K} \leq \delta$ that $K \geq 8 \log \frac{1}{\delta} + 1$ suffices. The algorithm which estimates the median value from random values $X_1, \ldots, X_K$ is an FPRAS.□

2.2. Approximation of #DNF. A probabilistic method consists in sampling random valuations and to test if they satisfy the property expressed by a DNF formula. If we compute the fraction of valuation which satisfy the property, we may approximate the real value with high probability.

Suppose we try to compute the size of a set $A \subseteq D$, for a fixed D. In the case of #SAT, the set D is the set of possible valuations, $| D | = 2^n$ and A is the set of valuations which satisfy the formulas. We can randomly choose N valuations and test if each one satisfies the formulas. If K valuations satisfy the formulas, a possible estimate would be $\frac{2^n \cdot K}{N}$. In the general case, this estimate is an (ϵ, δ)-estimator, i.e. an FPRAS, after an exponential number N of random selections, because K can be exponentially small. In certain cases, one can however find in polynomial time a good estimator.

An important case is when $| A | \geq \frac{|D|}{b}$ for a constant b (which can be also be polynomially bounded). The set A is of exponential size and let $X = \frac{2^n \cdot K}{N}$ the estimated value. If δ is the Chernoff's bound, the minimum number of necessary trials to bound the difference between the real value $| A |$ and the estimated value X less than ε with probability greater than $1 - \delta$ is given by the following result.

PROPOSITION 17.2. *If* $N \geq 4 \cdot b \cdot \log(\frac{2}{\delta})/\epsilon^2$, *then*

$$\#Prob[\mid A \mid \cdot(1-\varepsilon) \leq X \leq \mid A \mid \cdot(1+\varepsilon)] \geq 1-\delta$$

In this case, the sampling method is an FPRAS. Let us show how to apply this technique to the #DNF problem. In this case $D = \{0,1\}^n$ and $\mid D \mid = 2^n$ but the set A of valuations which satisfy the formulas can be exponentially small.

Let $x = \{c_1, .., c_m\}$ the input of the problem #DNF where each c_i is a conjunction. We wish to count the number of valuations σ which satisfy $\bigvee_1^m c_i$ but we need to design a clever way to do it. Let $D_1 = \{(i,\sigma) \ : \ i = 1, ..., m$ and σ satisfies $c_i\}$ and A_1 the set of (i,σ) such that c_i is the first conjunction satisfied by σ:

$$A_1 = \{(i,\sigma) \ : \ (i,\sigma) \in D_1 \ and \ if \ j < i, \ then \ (j,\sigma) \notin D_1\}$$

Each valuation in the set we try to count satisfies one c_i and the value is $\mid A_1 \mid$. Let us note C_i the set of valuations satisfying the clause c_i.

Observe that $\mid D_1 \mid = \sum_{i=1}^m \mid C_i \mid$ and that $\mid A_1 \mid \geq Max_i \mid C_i \mid$ because there are more valuations satisfying the disjunction of the c_i than valuations satisfying a particular c_i. One concludes that

$$\frac{\mid D_1 \mid}{\mid A_1 \mid} \leq \frac{\sum_{i=1}^m \mid C_i \mid}{Max_i \mid C_i \mid} \leq m$$

and one can directly apply the random sampling method.

Estimation of #DNF
X: =0; Iterate N times:
1. Choose $i \in \{1, ..., m\}$ *with probability* $\frac{|C_i|}{|D_1|}$.
2. Choose σ *uniformly among the valuations of* C_i.
3. If $(i,\sigma) \in A_1$ *(if* σ *satisfies* c_i *and does not satisfy* $c_1, ..., c_{i-1}$*) X: =X+1.*

The estimation is: $\frac{X \cdot \sum_{i=1}^m |C_i|}{N}$ and is an (ϵ,δ)-estimation of $\mid A_1 \mid$. One verifies that every element of D_1 is chosen with the same probability and this procedure is an FPRAS.

2.2.1. *Approximability of* $\#\Sigma_1$. Let us show that as in the case of optimization problems, if a counting problem is defined by an existential formula, then it has an FPRAS.

THEOREM 17.11. *If* F *is a counting function definable by a formula* Σ_1, *then* F *admits an* FPRAS.

The proof is similar to the one presented for the optimization problems and is based on a reduction to the problem #3DNF.

Proof : Let F a counting function definable by a formula $\exists y \psi(x, y, T)$ in the language associated with a class **K** with the additional relational symbol T. Suppose that:

$$F(U) = \mid \{(T, c) : (U, T, c) \models \exists y\, \psi(c, y, T)\} \mid$$

where T is unary. If U is a structure of the class **K** with n elements, one can write:

$$(U, T, c) \models \exists y\, \psi(c, y, T) \text{ iff } (U, T, c) \models \bigvee_{d=1}^{n} \psi(c, d, T)$$

Every $\psi(c, d, T)$ can be written in a 3DNF form as

$$\bigvee_{i=1}^{m} \varphi_i(c, d, T)$$

where each φ_i has at most three atoms. For each c let us count the number of relations T which satisfy $\bigvee_{d=1}^{n} \psi(c, d, T)$ and let us make the sum for $c = 1, ..., n$ (if we assume c unary).

Let us introduce the propositional variables $p_1, ..., p_n$ for the n atoms of the type $T(a_i)$.

Let ψ_c the formula 3DNF where we replaced in $\bigvee_{d=1}^{n} \psi(c, d, T)$ each atom using an explicit relation by the the truth value on the model U and every $T(a_i)$ by p_i. Let n_c the number of propositional variables which appear in ψ_c. Let $\#\mathsf{DNF}(c, \psi)$ the function which counts the number of valuations on $p_1, ..., p_n$ of the propositional formula. The number of possible T is

$$2^{n-n_c}\, \#\mathsf{DNF}(c, \psi)$$

and the total number of tuples (T, c) is:

$$F(U) = \sum_{c=1}^{n} 2^{n-n_c}\, \#\mathsf{DNF}(c, \psi)$$

Every function $\#\mathsf{DNF}(c, \psi)$ is approximable by an FPRAS and $F(U)$ admits also an FPRAS because the estimation is preserved by addition and multiplication by positive constants. □

The problem #3DNF admits a deterministic approximation schema (FPAS) [**AW85, NW94, LV91**]. It is an example of derandomization. Consequently every problem of $\#\Sigma_1$ also admits a deterministic approximation schema. The relation between the approximation classes FPRAS, FPAS and the SST hierarchy [**SST92**] is given in the figure below.

3. Approximate verification

Consider the Hamming distance between two structures of the same cardinality n. On binary words, it measures the number of positions where two words disagree and the relative distance is the distance divided by n. On ordered graphs, it measures the number of pairs where the two graphs disagree, and the relative distance is the distance divided by n^2, as we deal with binary relations. More generally, if

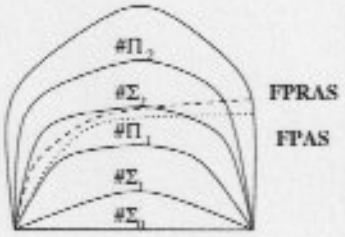

FIGURE 17.4. The approximation classes FPAS and FPRAS in the SST hierarchy for counting classes: $\#\Sigma_0, \#\Sigma_1, \#\Pi_1, \#\Sigma_2$ and $\#\Pi_2$.

$\mathcal{U}_1$ and $\mathcal{U}_2$ are two structures of the same cardinality on a class of ordered relational structures $\mathbf{K}$ with a relation R of arity k,

$$Dist(\mathcal{U}_1, \mathcal{U}_2) = \mid \{(a_1, ..., a_k) \ : \ [R(a_1, ..., a_k)]^{\mathcal{U}_1} \neq [R(a_1, ..., a_k)]^{\mathcal{U}_2}\} \mid$$

and the relative distance dist is defined by:

$$dist(\mathcal{U}_1, \mathcal{U}_2) = \frac{\mid \{(a_1, ..., a_k) \ : \ [R(a_1, ..., a_k)]^{\mathcal{U}_1} \neq [R(a_1, ..., a_k)]^{\mathcal{U}_2}\} \mid}{n^k}$$

Two structures $\mathcal{U}_1$ and $\mathcal{U}_2$ are *ϵ-close* if their relative distance is less than ϵ for $0 \leq \epsilon \leq 1$. Otherwise, they are *ϵ-far*.

Given a language L or a subclass $\mathbf{K}' \subset \mathbf{K}$, the distance from a structure U to $\mathbf{K}'$ is the *minimum* of the distances $dist(\mathcal{U}, \mathcal{U}')$ with $\mathcal{U}' \in \mathbf{K}'$ and we write $dist(\mathcal{U}, L)$ for this relative distance. Similarly, we say that $\mathcal{U}$ is *ϵ-close* to L if their relative distance is less than ϵ.

There are other measures, in particular the Edit Distance for words [**WF74**] which allows to compare two different strings and counts the number of deletions, insertions and modifications.

A *tester* is a randomized algorithm which queries the inputs on some instances, and the *query complexity* is the number of such queries.

DEFINITION 17.8. *A property* P *on a class* $\mathbf{K}$ *is ϵ-testable if there exists a polynomial time probabilistic algorithm which accepts every structure which has the property and rejects with probability 2/3 structures which are ϵ-far from the property and such that the number of queries depends on ϵ and not on n.*

One of the classical example is the kCOL property, i.e. the property that the graph can be partitioned in k subsets such that no edge connects two nodes of the same subset. In the case of 3COL, it produces a constant time algorithm which approximates an NP-complete problem. Exercise 8 details intermediate lemmas needed to show that 2COL has a tester, which can be generalized to kCOL.

On the class of finite words, [**AKNS99**] prove:

PROPOSITION 17.3. *Regular properties are ϵ-testable.*

On the class of graphs [**AFKS00**] prove:

PROPOSITION 17.4. *Σ_2 properties of graphs are ϵ-testable.*

Bibliographical notes. The study of the optimization problems developed with the notion of NP-completeness [**GJ79**]. The approximation of the optimization problems and in particular the approximation algorithm of MAXSAT is due to D. Johnson [**Joh74**]. The *Lin*-reductions were initially introduced in [**PY91**] and developed in many other articles [**ACP95, CP91**]. The logical characterization of APX was presented in [**KMSV94**]. The use of PCP to prove non approximability results appeared for the first time by [**Con91**]. The non approximability of MAXCLIQUE was proved by U. Feige and al. [**FGL+91**] and the non existence of an approximation schema by S. Arora and al. [**AS92b**].

The approximation of the class $\#\Sigma_1$ is also in [**SST92**] and the reduction of $\#\Sigma_1$ to #3DNF is due to A. Sharell [**Sha90**]. The amplification technique is presented by M. Jerrum, U. Vazirani and V. Vazirani in [**JVV86**] and the approximation of #DNF is due to R. Karp and M. Luby [**KL83**]. Other approximation techniques, based on Markov's chains with rapid mixture were elaborated by Mr Jerrum and A. Sinclair in [**Sin92**] , [**JS89**] and [**SJ89**]. The definition of an ϵ-testable property was proposed in [**RS96**] and is related to checkers introduced in [**BK89**]. Its applications to graphs properties were studied in [**GGR96**].