JOURNAL OF COMPUTER AND SYSTEM SCIENCES **43**, 425–440 (1991)

Optimization, Approximation, and Complexity Classes

CHRISTOS H. PAPADIMITRIOU

University of California, San Diego, California 92093

AND

MIHALIS YANNAKAKIS

AT & T Bell Laboratories, 600 Mountain Avenue, Murray Hill, New Jersey 079074

Received November 20, 1988; revised April 1, 1990

We define a natural variant of NP, *MAX NP*, and also a subclass called *MAX SNP*. These are classes of optimization problems, and in fact contain several natural, well-studied ones. We show that problems in these classes can be approximated with some bounded error. Furthermore, we show that a number of common optimization problems are complete for *MAX SNP* under a kind of careful transformation (called *L-reduction*) that preserves approximability. It follows that such a complete problem has a polynomial-time approximation scheme iff the whole class does. These results may help explain the lack of progress on the approximability of a host of optimization problems. © 1991 Academic Press, Inc.

1. INTRODUCTION AND DEFINITIONS

Optimization has provided much of the motivation for the development of NP-completeness (and the great majority of the problems discussed in [Ka, GJ]). However, the formal details and style of NP-completeness are somewhat awkward for optimization problems. In order to be considered in this context, an optimization problem must first give up its character, and become a language problem, via the introduction of a bound on its cost function. Furthermore, although all NP-complete optimization problems are certainly interreducible, such reductions are usually via non-optimization problems such as SAT, and important features, such as the value of the cost function and approximability, are rarely preserved.

An important, related front is the development of *approximation* algorithms for NP-complete optimization problems (and the application of NP-completeness to establish its impossibility, modulo $P \neq \text{NP}$). For many problems (e.g., the traveling salesman problem (TSP) with triangle inequality, node cover, max cut, and maximum satisfiability) there are known simple approximation algorithms with bounded error ratios (respectively, $\frac{3}{2}$, 2, $\frac{1}{2}$, and $\frac{3}{4}$). We do not know how to achieve better ratios, or, even more ambitiously, to devise a polynomial-time approximation

0022-0000/91 $3.00

scheme for these problems (a PTAS is a family of algorithms, one for each $\varepsilon > 0$, which are polynomial time, and achieve an approximation ratio of $1+\varepsilon$ ($1-\varepsilon$ for maximization problems)). The major open problem in this area is whether these and certain other problems have a PTAS. This and the more general problem of treating approximability of optimization problems in a unified way have been the object of intense research effort since more than a decade ago (see e.g., [ADP1, ADP2, AMP, Kr, PM]).

On the other hand, there is no clue on how to use reductions in order to exclude the existence of PTAS for these problems (assuming $P \neq \mathrm{NP}$). Intuitively, the difficulty lies in that our main technique for showing negative approximability results is the creation of a *gap* in the cost function, i.e., the creation of an instance in which the optimum has cost c iff the Boolean formula was satisfiable (assuming a reduction from SAT), and otherwise it has cost at least $c(1+g)$ for some gap $g > 0$. For example, in graph coloring, the original proof of NP-completeness of 3-colorability [S] establishes a gap of $\frac{1}{3}$, which can be "amplified" to 1 [GJ1]; this is one of the very few nontrivial results of this sort. Unfortunately, for certain problems such gaps are provably impossible (see [PS] for the TSP with triangle inequality). Furthermore, most problem reductions do not create or preserve such gaps. There would appear to be a last resort, namely to *create* such a gap in the generic reduction [C]. Unfortunately, this also seems doubtful. The intuitive reason is that computation is an inherently unstable, non-robust mathematical object, in the sense that it can be turned from non-accepting by changes that would be insignificant in any reasonable metric—say, by flipping a single state to accepting.

In complexity, when we are faced with such a situation, in which a family of problems cannot be identified with the known complexity classes (P and NP in our case), one suspects that a new *complexity class* may be manifesting itself. Defining this class in terms of computation however, presents us with the same problems of instability alluded to in the previous paragraph: There seems to be no clear notion of "approximately correct computation." If we move the focus of approximability to, for example, the number of correct bits in the input (so that the machine accepts), then there seems to be no generic reduction that preserves approximability. Variants of the same difficulty frustrate all other attempts along the "computational" approach.

We overcome this difficulty by defining our classes as extensions of Fagin's syntactic definition of NP[F]—the only one that does not involve computation.

Recall that NP consists of all predicates on structures G which can be expressed in the form $\exists S \phi(G, S)$, where S is a structure (and thus $\exists S$ is a second-order quantifier) and ϕ is first order. In fact, it is quite easy to check that ϕ can be assumed to be of the form $\forall \bar{x}\, \exists \bar{y}\, \psi(\bar{x}, \bar{y}, G, S)$, where ψ is quantifier free [Ko]. For example, SAT (with unboundedly long clauses) can be written as $\exists T\, \forall c\, \exists x\, [(P(c,x)\ \&\ x \in T) \vee (N(c,x)\ \&\ x \notin T)]$. Here P and N encode this instance: $P(c,x)$ means that variable x appears positively in clause c; $N(c,x)$ negatively. T is the set of *true* variables.

We can write 3SAT with one less alternation of quantifiers by assuming that the input consists of four relations C_0, C_1, C_2, C_3, where C_j contains all clauses with j negative literals: $(x_1, x_2, x_3) \in C_j$ means that there is a clause with $x_1, ..., x_j$ appearing negatively, and $x_{j+1}, ..., x_3$ positively. 3SAT then is: $\exists T \forall (x_1, x_2, x_3) [((x_1, x_2, x_3) \in C_0 \rightarrow x_1 \in T \vee x_2 \in T \vee x_3 \in T) \,\&\, \cdots \,\&\, ((x_1, x_2, x_3) \in C_3 \rightarrow x_1 \notin T \vee x_2 \notin T \vee x_3 \notin T)]$. Problems such as 3SAT, which can be expressed as $\exists S \forall \bar{x} \psi(\bar{x}, G, S)$, with ψ quantifier-free, form a subclass of NP called *strict* NP or SNP (they were called "strict $\sum_1^1$" in [KV]).

For each predicate $\Pi \in$ NP, where Π is a predicate of the form $\exists S \forall \bar{x} \exists \bar{y} \psi(\bar{x}, \bar{y}, G, S)$, we can define max$\Pi$ (the maximization version of Π) as

$$\max_S |\{\bar{x} : \exists \bar{y} \psi(\bar{x}, \bar{y}, G, S)\}|.$$

That is, instead of insisting that S be such that it satisfies $\exists \bar{y} \psi$ for all $\bar{x}$, we wish to find the S that maximizes the number of $\bar{x}$'s that do. MAX NP is the class of all such maximization problems. Similarly, MAX SNP is the class of optimization versions of all problems $\Pi \in$ SNP (no $\exists \bar{y}$ quantifier). We do not insist that the first-order part ϕ be in prenex form, but allow it to be a conjunction of such formulas.

Although it is by no means automatically true that every optimization problem in NP is also in MAX SNP or MAX NP, we show in the next section that several interesting ones are. What is more important, we exhibit that several of these are complete (for MAX SNP), under a transformation (defined next) called *L-reduction* (for *linear* reduction). This reduction is a very restricted form of transformation, expressly defined for treating approximability issues.

Let Π and Π' be two optimization (maximization or minimization) problems. We say that Π *L-reduces* to Π' if there are two polynomial-time algorithms f, g, and constants $\alpha, \beta > 0$ such that for each instance I of Π:

(a) Algorithm f produces an instance $I' = f(I)$ of Π', such that the optima of I and I', $OPT(I)$ and $OPT(I')$, respectively, satisfy $OPT(I') \leqslant \alpha OPT(I)$

(b) Given any solution of I' with cost c', algorithm g produces a solution of I with cost c such that $|c - OPT(I)| \leqslant \beta\, |c' - OPT(I')|$.

The constant β will usually be 1. The following two propositions follow easily from the definition.

PROPOSITION 1. *L-reductions compose.*

PROPOSITION 2. *If Π L-reduces to Π' (where each of Π and Π' is either a maximization or minimization problem), and there is a polynomial-time approximation algorithm for Π' with worst-case error ε, then there is a polynomial-time approximation algorithm for Π with worst-case error $\alpha\beta\varepsilon$.*

We defined our classes as classes of maximization problems. Minimization problems will be "placed" in the classes through L-reductions to maximization problems. The justification for this is that whatever conclusion about approximability we can draw for the class as a whole (approximability within some bounded error, or any bounded error), is valid for the minimization problem as well, by Proposition 2.

2. Complete Problems

We mainly concentrate on MAX SNP. At the end of the section we talk briefly about MAX NP. We next introduce several optimization problems.

1. MAX SAT: Given a set of clauses, find the truth assignment that satisfies the most. As noted above, the problem is in MAX NP. It can be approximated within $\frac{1}{2}$ by repeatedly choosing the truth value for each variable that satisfies the most yet unsatisfied clauses. (See [J] for a more sophisticated algorithm.) We consider also the following special cases and generalizations.

2. MAX 3SAT: The clauses have up to three literals. It is in MAX SNP. A special case which is very useful for L-reductions is when the number of occurrences of each variable is bounded by a constant B; we call this problem MAX 3SAT-B.

3. MAX 2SAT: Up to two literals. It is in MAX SNP (and is NP-complete in the ordinary sense).

4. MAX GSAT(B): Each clause is the disjunction of conjunctions, with each conjunction containing up to B literals. Thus, MAX GSAT(1) is the same as MAX SAT. The problem is in MAX NP.

5. INDEPENDENT SET-B: Given a graph with degrees of nodes bounded by B, fnd the largest independent set. This problem (known to be NP-complete) is in MAX SNP: Represent a graph with degree B by a $(B+1)$-ary relation A encoding the adjacency lists of the nodes. There is a tuple $(u, v_1, ..., v_B)$ for every node u listing its neighbors $v_1, ..., v_B$. (If a node has less than B neighbors, just repeat one of them). The Maximum Independent Set problem is modeled by the max version of the following predicate: $\exists I\, [\forall (u, v_1, ..., v_B) \in A \;\; (u \in I \,\&\, v_1 \notin I \,\&\, \cdots \,\&\, v_B \notin I)]$. (Note we have allowed the quantifier to run over tuples of the input structure. This is convenient for representing problems; formally, it does not make a difference, as the membership in A can be pushed through the parentheses.) Obviously, for any independent set I of the graph, the quantifier-free part $\psi = (u \in I \,\&\, v_1 \notin I \,\&\, \cdots \,\&\, v_B \notin I)$ of the predicate is satisfied exactly for those tuples of A that correspond to nodes u of I. Conversely, for any choice of I in the predicate, the set I' of nodes u whose corresponding tuple satisfies ψ is an independent set.

With the degree bounded by B, there is a trivial approximation algorithm with ratio $1/B$. For general graphs, it is not known if any fixed ratio ε can be achieved. Garey and Johnson showed (more than 10 years ago) that either all ε are achievable (i.e., there is a PTAS) or none [GJ1], but we still do not know the answer.

In hypergraphs, a set of nodes is independent if it does not contain any hyperedge. The rank of a hypergraph is the size of the largest hyperedge (thus, a graph has rank 2). The independent set problem for hypergraphs of bounded degree and rank is also in MAX SNP; if the rank is arbitrary (but the degree bounded) the problem is in MAX NP.

6. NODE COVER-B: Given a graph with degrees bounded by B, find the smallest node cover. Since B is a constant, the size of the smallest node cover is bounded below by a constant times the size of the largest independent set of the graph. Thus, NODE COVER-B can be L-reduced to INDEPENDENT SET-B, and hence is in MAX SNP.

7. DOMINATING SET-B: Given a graph with degrees bounded by B, find the smallest dominating set (a set of nodes which is adjacent to all other nodes). It is placed in MAX SNP by an L-reduction to the maximization problem modeled by the following predicate (the complement of a dominating set): $\exists S\, \forall(u, v_1, ..., v_B) \in A\ [u \notin S \mathbin{\&} (v_1 \in S \vee \cdots \vee v_B \in S)]$. Clearly, if S is any dominating set of the graph, the quantifier-free part $\psi = [u \notin S \mathbin{\&} (v_1 \in S \vee \cdots \vee v_B \in S)]$ is satisfied for the tuples of A that correspond to nodes u in $V - S$, the complement of S. Conversely, for any choice of S in the predicate, let S' be the set of nodes u whose corresponding A tuple does not satisfy ψ. Observing that $S \subseteq S'$, it is easy to see that S' is a dominating set.

The minimization problem is approximable with ratio $\log B$ [J]. This problem is L-equivalent to a restriction of MINIMUM SET COVER, where the size of the sets is bounded and every element appears in a bounded number of sets (a problem in MAX SNP). The case where the sets are bounded but an element may appear in an arbitrary number of sets can also be approximated with ratio $\log B$. For the general case, it is an old open problem to determine whether it is approximable within any fixed ratio.

8. MAX NOT-ALL-EQUAL 3SAT: The maximization version of the corresponding NP-complete problem. It is in MAX SNP.

9. MAX CUT: Given a (undirected) graph G, partition the nodes into two sets S and $\bar{S}$ to maximize the number of edges that go from S to $\bar{S}$. It is in MAX SNP (for general graphs): The predicate is $\exists S \subseteq V\ [\forall(x, y) \in E\ \{(x \in S \mathbin{\&} y \notin S) \vee (x \notin S \mathbin{\&} y \in S)\}]$. There are two simple algorithms that approximate this problem within $\frac{1}{2}$. One way is to process the nodes in arbitrary order, assigning each node to the set (S or $\bar{S}$) to which it has currently the fewest edges. A second way is to find a locally optimum solution: a partition in which moving any node from one set to the other does not increase the cut.

10. MAX DIRECTED CUT: The problem for directed graphs. It is also in MAX SNP with predicate $\exists S \subseteq V\ [\forall(x, y) \in E(x \in S \mathbin{\&} y \notin S)]$. The heuristics of the max cut do not work here, but the problem can be approximated within $\frac{1}{4}$ (see below).

11. MAX k-COLORABLE SUBGRAPH: Given a graph, find the largest subgraph (maximum number of edges) that can be colored with k colors, where k is a fixed number. For $k=2$ this is the same as MAX CUT. It is easy to see that this problem is in MAX SNP for any (fixed) k. It can be approximated with ratio $(k-1)/k$ with methods similar to MAX CUT [V]. The problem for hypergraphs of bounded rank is also in MAX SNP; for general hypergraphs it is in MAX NP. (In a hypergraph, a coloring is legal if no hyperedge is monochromatic.)

Many of the problems in MAX SNP that we saw could be approximated within some fixed ratio. This was not a coincidence.

THEOREM 1. *Every problem in* MAX SNP *can be approximated within some fixed ratio.*

Proof. Consider a problem in MAX SNP with predicate $\exists S \forall \bar{x} \phi(\bar{x}, G, S)$. (In general, we have allowed the first-order part to be a conjunction of prenex formulas; the proof for this case is similar.) ϕ is a Boolean formula with leaves of the form $S(\bar{z})$ and $G(\bar{w})$, where $\bar{z}, \bar{w}$ are projections of $\bar{x}$ and G is the input structure. Since the predicate is fixed, independent of the input G, there is a polynomial number of possible values for $\bar{x}$. For each value of $\bar{x}$, ϕ is a Boolean formula with variables of the form $S(\bar{z})$—since G is given. We may ignore values of $\bar{x}$ for which ϕ is unsatisfiable (for every S). Then we can write the first-order part as a conjunction $\phi_1 \& \cdots \& \phi_m$, with one conjunct for each value of $\bar{x}$. Every ϕ_i has a constant number, say at most k, of variables $S(\bar{z})$. Let f_i be the fraction of assignments that satisfy ϕ_i; $f_i \geqslant 2^{-k}$. A random assignment to the variables satisfies each particular ϕ_i with probability f_i; thus, it satisfies on the average $\sum f_i \geqslant m2^{-k}$ conjuncts.

Such an assignment can also be found deterministically. The algorithm is a modification of an algorithm of Johnson [J] (for approximating MAX SAT with at least k literals per clause). Assign a truth value to each variable y (atomic formula of the form $S(\bar{z})$) in arbitrary order as follows: Let $p_i(y)$ (respectively, $n_i(y)$) be the fraction of truth assignments for the remaining variables which satisfy ϕ_i with y true (resp. false). Note that we can compute these quantities for each ϕ_i in constant time, since ϕ_i has fixed size. If $\sum p_i(y) \geqslant \sum n_i(y)$ then set y to true, otherwise to false, and update the ϕ_i's and the p_i's and n_i's for the remaining variables.

It is easy to show that the sum of the fractions of satisfying assignments for the conjuncts does not decrease throughout the execution of the algorithm. To see this, suppose that there were l remaining variables before choosing the truth value of y. The number of truth assignments (for these variables) that satisfied ϕ_i was $2^l[p_i(y)+n_i(y)]$. Suppose that variable y was set to true (the other case is similar). After setting y, the fraction of assignments to the remaining variables which satisfy ϕ_i is $2^l p_i(y)/2^{l-1} = 2p_i(y)$. Thus, the sum of these fractions after assigning y is $\Sigma\, 2p_i(y) \geqslant \Sigma[p_i(y)+n_i(y)]$. At the end, this fraction is 1 for a satisfied conjunct, and 0 for an unsatisfied. Thus, the truth assignment satisfies at least Σf_i conjuncts. ∎

We say that a problem Π in MAX SNP is *MAX SNP-complete* if for any other problem Π' in MAX SNP there is an L-reduction from Π' to Π. In the problems of the following theorem, B represents a fixed constant; in each case its value can be computed from the reductions.

THEOREM 2. *The following problems are* MAX SNP-*complete.*

(a) MAX 3SAT
(b) MAX 3SAT-B
(c) INDEPENDENT SET-B
(d) NODE COVER-B
(e) DOMINATING SET-B
(f) MAX 2SAT
(g) MAX NOT-ALL-EQUAL 3SAT
(h) MAX CUT
(i) MAX DIRECTED CUT
(j) MAX k-COLORABLE SUBGRAPH (all $k \geqslant 2$)

Proof. (a) Consider a problem in MAX SNP with predicate $\exists S\ \forall \bar{x} \phi(\bar{x}, G, S)$. As in the proof of Theorem 1, we enumerate all the possible values of $\bar{x}$, ignore those values for which ϕ is unsatisfiable, and write the first-order part as a conjunction $\phi_1 \,\&\, \cdots \,\&\, \phi_m$, with one conjunct for each value of $\bar{x}$. For each ϕ_i, we introduce auxiliary variables and construct a set C_i of clauses with at most three literals each, as follows. Viewing ϕ_i as a circuit (of bounded size), introduce one new variable g for every gate g of ϕ_i. If g is a NOT gate with input a, then we include clauses $g \vee a$ and $\bar{g} \vee \bar{a}$; if g is an AND gate with inputs a, b, we include cluases $a \vee \bar{g}$, $b \vee \bar{g}$ and $g \vee \bar{a} \vee \bar{b}$; if g is an OR gate with inputs a and b, we include clauses $\bar{a} \vee g$, $\bar{b} \vee g$, and $\bar{g} \vee a \vee b$. Finally, if h is the output gate, we include a clause h. The set of clauses C_i has the property that any truth assignment τ to the variables of ϕ_i (the inputs to the circuit) can be extended to the auxiliary variables so that it satisfies all the clauses, except possibly for the clause h corresponding to the output gate (if τ does not satisfy ϕ_i).

Our instance of MAX 3SAT contains all these sets of clauses $C_1, \ldots, C_m$, for all values of $\bar{x}$. If we have a conjunction of prenex formulas, we just take the union of all clauses produced. It remains to show that this transformation f is an L-reduction. Since the ϕ_i's are of bounded size, so are the C_i's, and the total number of clauses is at most $k_1 m$ for some constant k_1. If there are $OPT(I)$ values of $\bar{x}$ satisfying ϕ in the optimal solution for the instance I of the MAX SNP problem, then the optimal assignment for the MAX 3SAT instance $f(I)$ satisfies $OPT(f(I)) = \sum_i (|C_i| - 1) + OPT(I)$ clauses. From the proof of Theorem 1, $OPT(I)$ is at least a constant fraction of m; $OPT(I) \geqslant m/k_2$. It follows that f satisfies condition (a) in the definition of L-reduction, with the constant $\alpha = (k_1 - 1) k_2 + 1$. Condition (b) is satisfied with $\beta = 1$, and the transformation is an L-reduction.

(b) The ordinary reductions that lead to bounded number of occurrences fail here. The usual reduction that shows the NP-completeness of 3SAT restricted to instances with three occurences per variable goes as follows. Replace each variable x appearing m times in the clauses (positively or negatively) by m new variables $x_1, ..., x_m$, add clauses $\bar{x}_1 \vee x_2$, $\bar{x}_2 \vee x_3, ..., \bar{x}_m \vee x_1$, and use x_i for the ith occurence of x in the original clauses. The new clauses ensure that in any satisfying assignment, the new variables $x_1, ..., x_m$ that replaced x get the same truth value. This is a valid reduction for deciding whether the set of clauses is satisfiable, but not for maximizing the number of satisfied clauses: it may be more profitable to sacrifice some of the new clauses in order to satisfy more of the original clauses. For example, if the first 10 occurences of x are positive and the rest are negative, we can satisfy all the original clauses containing x by sacrificing only one new clause $\bar{x}_{10} \vee x_{11}$.

We use *expanders* in order to ensure L-reducibility. A graph on n nodes is a c-expander (where $c > 0$ is a constant) if every subset S of at most $n/2$ nodes is adjacent to at least $c\,|S|$ nodes outside S. For every n, one can construct in polynomial time a bounded degree (even cubic) expander, with some expansion rate c (see for example [A]). For our purposes, we need a graph F_m with the following properties: (1) it has bounded degree, (2) it has $O(m)$ nodes of which m are "distinguished", and (3) for every partition of the nodes into two sets, the number of edges in the cut is at least min $(|S_1|, |S_2|)$, where S_1 and S_2 are the sets of distinguished nodes in the two sides. We can obtain such a graph F_m as follows: take m disjoint full binary trees with (at least) $1/c$ leaves each, and connect their leaves in a cubic c-expander, the distinguished nodes are the roots of the full binary trees. Note that the degree does not depend on the expansion rate: it is 4 (and with more care, it can be reduced to 3). Clearly, F_m satisfies properties (1) and (2). To see that it satisfies (3), consider a partition of the nodes into two sets Q_1 and Q_2. Let R_1, respectively R_2, be the sets of distinguished nodes whose corresponding binary trees are entirely contained in Q_1, resp. Q_2, and let R_3 be the rest of the distinguished nodes. The cut contains at least $|R_3|$ edges from the binary trees, and $\min(|R_1|, |R_2|)$ edges from the expander. Clearly, $|R_3| + \min(|R_1|, |R_2|) \geqslant \min(|S_1|, |S_2|)$.

We replace each variable x appearing m times by $O(m)$ new variable $x_1, x_2, ...$ corresponding to the nodes of the graph F_m with the above properties. For every edge (i, j) of F_m, we have clauses $\bar{x}_i \vee x_j$ and $\bar{x}_j \vee x_i$, i.e., $x_i \equiv x_j$. We use the variables corresponding to the distinguished nodes of F_m in place of x in the original clauses. It follows from property (3) that it does not pay to falsify the auxiliary clauses in order to satisfy the original ones: if a truth assignment does not give the same truth value to all the x_i's, then changing the value of the variables that do not agree with the majority of the distinguished ones does not decrease the number of clauses satisfied. Thus, the transformation is an L-reduction, where the constant α depends on the expansion rate, and the constant β is 1.

(c, d) The ordinary reductions (from MAX 3SAT-B) now work: Construct a graph with one node for every occurrence of every literal. There is an edge connecting any two occurrences of complementary literals, and also, an edge connecting literal occurrences from the same clause (thus, there is a triangle for every clause with 3 literals, and an edge for a clause with 2 literals). The size of the maximum independent set in the graph is equal to the maximum number of clauses that can be satisfied. If every variable occurs at most k times in the clauses, then the degree is at most $k+1$.

(e) From NODE COVER-B: Given a graph G, add a path of length 2 parallel to every edge, and let H be the resulting graph. It is easy to see that a set S of original nodes (from G) is a node cover of G if and only if it is a dominating set of H. Clearly, we do not gain anything by including any new nodes in the dominating set of H. Thus, the transformation is an L-reduction with $\alpha = \beta = 1$.

(f) From INDEPENDENT SET-B: We have a variable for each node. The clauses are $\bar{x} \vee \bar{y}$ for every edge (x, y), and x for every node. It is easy to see that an optimal assignment will satisfy all the clauses corresponding to the edges. Therefore, the nodes x whose variable is true form an independent set. Thus, the maximum number of clauses that can be satisfied is equal to the size of the maximum independent set plus the number of edges. Since the graph has bounded degree, the number of edges is linear in the number of nodes, and thus, the transformation is an L-reduction. Note that MAX 2SAT-B is also complete.

(g) From MAX 2SAT: Let I be an instance of MAX 2SAT with m clauses, of which $OPT(I)$ can be satisfied. Construct an instance $f(I)$ of MAX NOT-ALL-EQUAL 3SAT as follows. Add a new variable z. For each clause $a \vee b$ of I (a and b are literals), we have one NAE (not-all equal) clause (a, b, z). Similarly, for a clause of I with only one literal a, we have a NAE clause (a, z). Clearly, the number of NAE clauses satisfied by a truth assignment is not affected if we flip the truth value of all the variables. Thus, we may assume without loss of generality that the new variable z is false. A truth assignment with z false satisfies a NAE clause iff it satisfies the corresponding clause of 2SAT. Thus, $OPT(f(I)) = OPT(I)$. The transformation is an L-reduction with $\alpha = \beta = 1$.

(h) Let I be an instance of MAX NAE 3SAT with m clauses of which $OPT(I)$ can be satisfied. First, we construct a multigraph G as follows. There is one node for every literal x, $\bar{x}$ of I. For every clause of I with three literals, a, b, c, we include edges forming a triangle (a, b, c) on the corresponding nodes (other clauses may cause additional edges among these nodes), and for every clause of I with two literals a, b we include two parallel edges (a, b). In addition, for every variable x, we connect nodes x and $\bar{x}$ with $2k$ parallel edges, where k is the number of times that x or $\bar{x}$ occurs in the clauses.

Given a truth assignment τ that satisfies p clauses of I, we form a partition of G by placing on one side all nodes that correspond to false literals, and on the other side the nodes corresponding to true literals. It is easy to see that every NAE clause that is satisfied contributes two edges to the cut, while an unsatisfied clause does not contribute any edges. If the total number of literal occurrences is l ($l \leqslant 3m$), then the cut contains $2l+2p$ edges. Conversely, given a partition of G with cut c, we may assume without loss of generality that nodes corresponding to complemented literals are on different sides; for, if x and $\bar{x}$ are on the same side, moving one of them to the other side does not decrease the cut. Set the literals corresponding to nodes that are on one side of the partition to true, and those corresponding to nodes on the other side to false. Then $c = 2l + 2p$, where p is the number of satisfied clauses. Since $l \leqslant 3m$ and $OPT(I) \geqslant m/2$, the transformation is an L-reduction. The constant β is 2 in this case.

To show the completeness of MAX CUT on simple graphs (no parallel edges), just replace every edge of G with a path of length 3, and let H be the resulting graph. A partition of G can be extended to a partition of H so that the cut of H contains three edges for every edge for every edge in the cut of G, and two edges for every edge of G that is not in the cut. Thus, the optimal cut of H is $CUT(H) = 2e + CUT(G)$, where e is the number of edges of G. Using expanders, one can show that the restriction of the MAX CUT problem to bounded degree graphs is also MAX SNP-complete.

(i) MAX CUT is a special case if we replace an undirected edge by two oppositely directed edges.

(j) From MAX CUT: Let G be a graph with n nodes and e edges. We reduce MAX CUT first to the MAX k-COLORABLE SUBGRAPH problem for multigraphs, and then to simple graphs. Construct from G a multigraph H as follows. Add $k-2$ new nodes connected in a complete graph with every edge having multiplicity e. Connect every node v of G to each of the $k-2$ new nodes with an edge of multiplicity $d(v)$, equal to the degree of v in G. Note that the number e' of edges of H (including multiplicities) is linear in the number e of edges of G; i.e., $e' = c \cdot e$ for some constant c. Let $COLOR(H)$ be the maximum number of edges in a k-colorable subgraph of H, and let $CUT(G)$ be the maximum cut of G. We claim that $COLOR(H) = CUT(G) + (c-1)\,e$.

Given a partition of G into two sets, i.e., a 2-coloring of G, we can extend it to H by giving the remaining $k-2$ colors to the new nodes. The only edges that are not legally colored are the edges of G that are not in the cut. Conversely, given a k-coloring of H, we can assume that the $k-2$ new nodes have distinct colors, because otherwise we loose e edges, which is certainly suboptimal. We may assume also that every node of G has one of the two remaining colors, because of the edges connecting them to the new nodes. Thus, the k-coloring of H induces a partition (2-coloring) of G, and the only edges that are not legally colored are the edges of G that are not in the cut.

We can construct a simple graph H' from H as follows. Let F be a complete graph on $k+2$ nodes with three edges (a_1, b_1), (a_2, b_2), (a_1, a_2) missing, where a_1, b_1, a_2, b_2 are distinct nodes. Note that F can be k-colored, and in every k-coloring of F, nodes a_1 and a_2 must have different colors. Replace every multiple edge (u_1, u_2) of H of multiplicity m, by m copies of F where a_1 and a_2 are identified with u_1 and u_2. It is easy to see that H' has the same properties as H: its number of edges is linear in e, and the maximum k-colorable subgraph contains all the edges except for those that are not in the maximum cut of G. ∎

Some other problems can be shown *hard* for MAX SNP. For example, the traveling salesman problem with all distances 1 or 2, TSP(1, 2). It is a special case of the TSP with triangle inequality. There is an approximation algorithm for this problem that has ratio $\frac{7}{6}$ [PY].

MAX NP

Theorem 1 can be shown for MAX NP also. To see this, consider a problem in MAX NP with predicate $\Pi = \exists S\ \forall \bar{x}\ \exists \bar{y} \psi(\bar{x}, \bar{y}, G, S)$. As in the proof of Theorem 1, enumerate all the possible values of $\bar{x}$, and for each value $\bar{x}_i$, examine if there is a choice of $\bar{y}$ for which ψ can be satisfied (for some choice of S). If there is no such $\bar{y}$, then ignore this value of $\bar{x}$; otherwise, pick arbitrarily any value of $\bar{y}$, say $\bar{y}_j$, for which ψ can be satisfied, substitute it in ψ, and let $\phi_i = \psi(\bar{x}_i, \bar{y}_j, G, S)$. Since $\bar{x}_i, \bar{y}_j$, and the input G are now fixed, ϕ_i is a formula with a constant number, say at most k, of variables of the form $S(\bar{z})$. As in the proof of Theorem 1, we can find an S that satisfies a constant fraction 2^{-k} of the ϕ_i's.

Thus, only problems that are approximable within some constant ratio can be in MAX NP. We can show also by similar techniques, that every problem in MAX NP can be L-reduced to GSAT(B) for some B. However, we have not been able to show this for a particular value of B (unlike the case of MAX SNP were three or even two literals per clause suffice for completeness). Thus, MAX NP has no known complete problem.

Independently, Berman and Schnitger had been studying issues and problems related to the ones in this paper. They also showed the equivalence of MAX 2SAT and NODE COVER-B and found linear reductions from MAX 2SAT to some other problems (which are therefore, MAX SNP-hard) [BS]. Also, they showed that if MAX 2SAT does not have a randomized PTAS then the independent set problem (for general graphs) is not approximable within a factor of $O(n^c)$.

3. Extensions

Our definition of MAX SNP follows closely the syntactic restrictions of Fagin's Theorem, which inspired it. We may be able to encompass a richer class of problems by a definition that deviates from the first-order style of MAX SNP. We describe two such directions below.

Problems with Weights

For many of the problems we discussed, there are natural weighted versions: for example, MAX CUT where the edges have weights (or capacities), INDEPENDENT SET (Node Cover, etc) with weights on the nodes, MAX SAT with weights on the clauses. We model these problems as follows. In the weighted version of a MAX SNP problem with predicate $\exists S \forall \bar{x}\, \phi(\bar{x}, G, S)$, there is a positive number (weight) associated with every tuple in the range of the universal quantifier(s). The problem is to find an S that maximizes the total weight of the $\bar{x}$'s for which ϕ is true. Theorem 1 holds also for problems with weights; that is, every weighted MAX SNP problem can be approximated within some constant ratio. The algorithm is essentially the same as in the unweighted case. Using the notation from the proof of Theorem 1, the only difference is that we choose the truth value of a variable y (atomic formula of the form $S(\bar{z})$) by comparing the weighted sums $\sum w_i p_i(y)$ and $\sum w_i n_i(y)$, where w_i is the weight of the ith value of $\bar{x}$.

Also, the weighted versions of all the problems listed in Theorem 2 are complete. The reduction for MAX 3SAT given there works also in the weighted case with the obvious modification: the clauses of C_i that replace a conjunct ϕ_i are given weight w_i equal to the weight of the ith value of $\bar{x}$.

The reduction for MAX 3SAT-B is more complicated. Let I be an instance of WEIGHTED MAX 3SAT with p clauses with weights $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_p$. Let l be the smallest index such that $w_1 > p^2 w_{l+1}$; if no such index exists, let $l = p$. Partition the clauses of I into two groups, where the first group contains the first l clauses, and the second group contains the rest. Replace the occurrences of each variable x in the second group by new variables $x'_1, x'_2, \ldots$. We distinguish two cases: (1) $w_l > p w_{l+1}$, and (2) $w_l \leqslant p w_{l+1}$. In the first case, we add clauses $x'_1 \equiv x'_2$, $x'_2 \equiv x'_3, \ldots$, all of weight $p w_{l+1}$; if x appears also in a clause of the first group, then we add a clause $x \equiv x'_1$ of weight w_l and include it in the first group for further processing. In the second case, we add clauses $x = x'_1$, $x = x'_2, \ldots$ of weight w_l each, and include them in the first group. Observe that in both cases, the total weight of the clauses increases only by a constant factor. Also, note that it is suboptimal for a truth assignment to violate any of these new clauses.

We transform now the clauses of the first group. Replace every clause in the first group that has weight $w_i \geqslant 2w_l$ by $k_i = [w_i/w_l]$ identical clauses, each of weight w_i/k_i; note that $w_i \leqslant p^2 w_l$, and thus, $k_i \leqslant p^2$ (the number of clauses is polynomial), and the weight of all the clauses in the first group is now within a factor of 2 of w_l. Suppose that a (original) variable x occurs now m times in the clauses of the first group. Replace x by m new variables $x_1, \ldots, x_m$ connected in a cubic c-expander, whose edges (i, j) correspond to clauses $x_i \equiv x_j$ with weight $2w_l/c$. It is easy to see that the final set of clauses has total weight linear in the total weight of the original clauses. Because of the expander, all the variables x_i that replaced the occurrences of x in the clauses of the first group will have the same value in an optimal truth assignment. As we noted above, an optimal truth assignment will also satisfy the auxiliary clauses of the form $x \equiv x'_i$ or $x'_i \equiv x'_j$, and thus, the variables x'_i will also have the same value.

Once we have shown the completeness of WEIGHTED MAX 3SAT-B, we can use the same reductions as in Theorem 2 (with obvious modifications in each case) to show that the weighted versions of the other problems are also complete for the class of MAX SNP problems with weights.

Permutation Problems

We can also extend MAX SNP by restricting structure S (see the definition of MAX SNP) to be a permutation. That is, we define a new class of optimization problems, in which we are asked to find the permutation π of the elements of the universe, which maximizes $|\{\bar{x}: \phi(G, \pi, \bar{x})\}|$. Here ϕ is a quantifier-tree first-order sentence with atoms of the form "$G(\cdot)$" and "$\pi(x) < \pi(y)$." Let us call this class of optimization problems MAX SNP[π]. It contains several interesting problems, such as:

MAXIMUM SUBDAG: Given a directed graph $G=(V, E)$, find an acyclic subgraph $G'=(V, E')$, with E' as large as possible. In this case, ϕ is the sentence $\pi(x) < \pi(y) \wedge G(x, y)$.

MAXIMUM SUBDAG can be approximated within a factor of two: Try a permutation and its reverse, and choose the one in which the most edges go forward. In some sense, this problem is the complement of feedback arc set (since the optima of the two problems are complements with respect to E). Similarly for the complement of OPTIMAL LINEAR ARRANGEMENT [*GJ*]:

MAXIMAL LINEAR ARRANGEMENT: Given a graph $G=(V, E)$, find the permutation π of V that maximizes $\sum_{e \in E} |span_\pi(e)|$ where $span_\pi([u, v]) = \{w: \pi(u) < \pi(w) < \pi(v) \text{ or } \pi(v) < \pi(w) < \pi(u)\}$.

In this problem, ϕ is $\pi(u) < \pi(w) \wedge \pi(w) < \pi(v) \wedge G(u, v)$.

In fact, it is easy to see that MAX SNP[π] is in some sense larger than MAX SNP, in that MAX 3SAT can be expressed as a problem in MAX SNP[π]: The universe now consists of both variables and negations, and $\pi(x) < \pi(\bar{x})$ means that x is TRUE.

A result similar to Theorem 1 can be shown for MAX SNP(π): Every problem in MAX SNP[π] can be approximated within some ε (the argument now uses the fact that a constant fraction of all permutations satisfy each clause). Furthermore, the class has at least one natural complete problem:

THEOREM 3. MAXIMUM SUBDAG *is complete for* MAX SNP[π].

Proof. Consider any problem in MAX SNP[π], defined by a formula ϕ. First, we reduce to the following problem MAX 3SAT[π]: We are given a set of clauses; each clause is the disjunction of at most three atoms, where every atom is a comparison of the form $\pi(x) < \pi(y)$. The problem is to find a permutation π of the elements that satisfies the maximum number of clauses. The reduction to MAX 3SAT[π] is similar to the one in Theorem 2(a). The only difference is that, instead of introducing one variable g for every gate g, we introduce two new elements g_1 and g_2, and use $\pi(g_1) < \pi(g_2)$ in place of the literal g, and $\pi(g_2) < \pi(g_1)$ in place of the literal $\bar{g}$.

Let I now be an instance of MAX 3SAT[π] with m clauses, of which $OPT(I)$ can be satisfied, and let U be the universe of elements. We construct a directed graph

$H = (V, E)$. V contains all elements of the universe U, and, for each clause of the form $[\pi(x_1) < \pi(x_2) \vee \pi(y_1) < \pi(y_2) \vee \pi(z_1) < \pi(z_2)]$ we have six new nodes x'_1, x'_2, y'_1, y'_2, z'_1, z'_2, connected as in Fig. 1. (Note: Some of the elements x_1 through z_2 may coincide; in this case the primed versions are distinct nodes.) If the clause has two atoms or one, then the corresponding nodes of the figure are omitted. Note that there are three cycles in the figure, where each one of them contains two of the three arcs $x'_1 \to x'_2$, $y'_1 \to y'_2$, $z'_1 \to z'_2$, and two more distinct arcs. Therefore, we must delete at least two arcs from the gadget of the figure to obtain an acyclic subgraph.

Let $DAG(H)$ be the maximum number of arcs in an acyclic subgraph of H. We claim that $DAG(H) = OPT(I) + r$, where r is the number of arcs of H that are not of the form $a'_1 \to a'_2$. Consider first a permutation π of the elements of U. If π satisfies a clause, say $\pi(x_1) < \pi(x_2)$, then keep the corresponding arc $x'_1 \to x'_2$ and remove the other two arcs $y'_1 \to y'_2$ and $z'_1 \to z'_2$; it is easy to see that after removing these two arcs, there is no path through the gadget between any two unprimed nodes (elements of U), except for a path from x_1 to x_2, which is consistent with $\pi(x_1) < \pi(x_2)$. If π does not satisfy the clause, then we delete all three arcs corresponding to the literals of the clause. Thus, if π satisfies p clauses, then the acyclic subgraph obtained by deleting the indicated arcs has $r + p$ arcs.

Conversely, consider an acyclic subgraph of H, pick a topological ordering of the nodes, and let π be the permutation induced on U. If a clause is not satisfied by π, then the subgraph must be missing at least three arcs from the gadget of the figure: an arc must be missing from the path $x_1 \to x'_1 \to x'_2 \to x_2$, and from the analogous paths for y and z; note that these paths are edge-disjoint. Thus, the subgraph has at most $r + p$ arcs, where p is the number of clauses satisfied by π.

Since the number of arcs of the graph is linear in the number of clauses, it follows that the transformation is an L-reduction. ▌

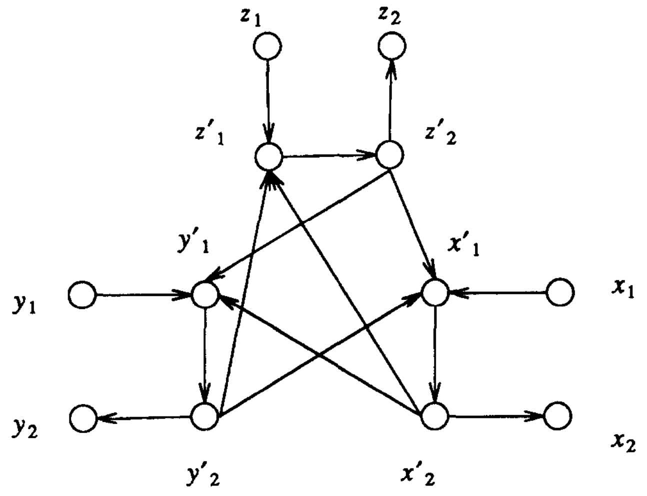

FIGURE 1

4. Discussion

The importance of a complexity class draws from the importance of the computational phenomenon that it captures, as well as that of the problems it contains (especially the complete ones). MAX NP and MAX SNP consitute the first successful attempt to capture in a complexity class the intricacies of approximability. Admittedly, only a fraction of the many interesting (in this respect) optimization problems were shown to fall in MAX NP. This, no doubt, reflects the ancient difficulties of preserving the essence of the cost through reductions (in our terminology, the intricacy of designing L-reductions). It would be interesting to add more problems to MAX NP, or demonstrate formally that certain problems are not in it. This is not impossible, in view of the syntactic definition and the restricted nature of L-reductions.

These complexity classes put in a common framework problems for which it was usually very easy to get some fixed approximation bound, but has proved extremely difficult to go beyond it, and either find a PTAS or show a gap (that no PTAS exists). They helped reveal the common roots of the difficulty in understanding the limits of approximability for 3SAT, MAX CUT, NODE COVER (even in bounded degree graphs), INDEPENDENT SET (in bounded degree graphs), and, partly, the TSP. Once more, we have decreased the number of open questions in the field—without, alas, increasing much the number of answers!

References

[A] M. Ajtai, Recursive construction for 3-regular expanders, *in* "Proc. 28th Annual IEEE Symp. on Foundations of Computer Science," pp. 295–304, 1987.

[ADP1] G. Ausiello, A. D'Atri, and M. Protasi, On the structure of combinatorial problems and structure preserving reductions, *in* "Proc. 4th Intl. Coll. on Automata, Languages and Programming," pp. 45–57, 1977.

[ADP2] G. Ausiello, A. D'Atri, and M. Protasi, Structure preserving reductions among convex optimization problems, *J. Comput. System Sci.* **21** (1980), 136–153.

[AMP] G. Ausiello, A. Marchetti-Spaccamela, and M. Protasi, Toward a unified approach for the classification of NP-complete optimization problems, *Theoretical Computer Science* **12** (1980), 83–96.

[BS] P. Berman and G. Schnitger, On the complexity of approximating the independent set problem, *in* "Proc. 6th Ann. Symp. on Theoretical Aspects of Computer Science," 1989.

[C] S. A. Cook, The complexity of Theorem Proving Procedures, *in* "Proc. 3rd Annual ACM Symp. on Theory of Computing," pp. 151–158, 1971.

[F] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets, *in* "Complexity of Computations" (R. Karp, Ed.), Amer. Math. Soc., 1974.

[GJ1] M. R. Garey and D. S Johnson, The complexity of near optimal graph coloring, *J. Assoc. Comput. Math.* **23** (1976), 43–49.

[GJ] M. R. Garey and D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness," Freeman, San Francisco, 1979.

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

[Ka] R. M. Karp, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations" (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–103, Plenum, New York, 1972.

[Ko] P. Kolaitis, private communication.

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

[KV] P. Kolaitis and M. Vardi, The decision problem for the probabilities of higher-order properties, *in* "Proc. 19th Annual ACM Symp. on Theory of Computing," pp. 425–435, 1987.

[OM] P. Orponen and H. Mannila, "On Approximation Preserving Reductions: Complete Problems and Robust Measures," Technical Report, Univ. of Helsinki 1987.

[PS] C. H. Papadimitriou and K. Steiglitz, Some example of difficult traveling saleman problems, *Oper. Res.* **26**, (1978), 434–443.

[PY] C. H. Papadimitriou and M. Yannakakis, "The traveling Salesman Problem with Distances One and Two," manuscript, 1989.

[PM] A. Paz, and S. Moran, Non-deterministic polynomial optimization problems and their approximation, *Theoret. Comput. Sci.* **15** (1981), 251–277.

[S] L. J. Stockmeyer, Planar 3-colorability is NP-complete, *SIGACT News* 5, 3 (1973), 19–25.

[V] P. Vitanyi, How well can a graph be *n*-colored?, *Discrete Math.* (1981).