A STRUCTURAL OVERVIEW OF NP OPTIMIZATION PROBLEMS

by

Danilo Bruschi
Deborah Joseph
Paul Young

Computer Sciences Technical Report #861

July 1989

A STRUCTURAL OVERVIEW OF NP OPTIMIZATION PROBLEMS [†]

DANILO BRUSCHI – Università degli Studi di Milano and University of Wisconsin

DEBORAH JOSEPH – University of Wisconsin

PAUL YOUNG – University of Washington and University of Wisconsin

1. INTRODUCTION.

Although *NP* complete problems are typically stated as decision problems there are frequently underlying optimization problems that are of more interest. For example, in the *traveling salesperson* problem finding a minimum length tour is generally of more interest than knowing whether there is a tour of length less than k. However, early in the study of *NP* completeness it was recognized that computing the optimal solution of an *NP* optimization problem can be done in polynomial time only if there is a polynomial time solution for the corresponding decision problem. For this reason, and because nondeterministic functional computations are more difficult to formalize, the complexity theory for these problems has been built primarily on the study of the associated decision problems. Nevertheless, as the theory of *NP* complete sets and polynomial time reductions developed, it was also recognized that even the polynomial computable isomorphisms between known *NP* complete sets are not strong enough to preserve all of the underlying structure of the optimization versions of these problems. Furthermore, it was recognized early on (e.g., [GaJo 79], [GoSa 76], [Jo 74], and [PaMo 81]) that despite the fact that all known naturally occurring *NP* complete decision problems are polynomially isomorphic, ([BeHa 77], [JoYo 85]), some *NP* optimization problems have very good polynomially computable approximations while it is impossible for others to have good polynomially computable approximations unless $P = NP$.

[†] In this paper, we survey structural work on calculating optimal solutions for *NP* complete problems. This overview is not intended to be an exhaustive description of the area, nor even of all of the work in those papers which we discuss. Instead, it is an attempt to provide a general introduction to the area. As a consequence we have freely unified notations and simplified statements of results. We hope that this survey will provide a *guide* to some of the more interesting ideas and problems in the area, but readers wishing definitive statements of the results we discuss will need to consult journal versions of the papers we survey.

This work was presented as an invited paper at the Second International Symposium on Optimal Algorithms, and will appear in Springer-Verlag Lecture Notes in Computer Science as one of the selected papers from that symposium. The work was supported in part by the Ministero della Pubblica Istruzione, through "Progetto 40%: Algoritmi e Strutture di Calcolo," the National Science Foundation under grant DCR-8402375, and by the Wisconsin Alumni Research Foundation under a Brittingham Visiting Professorship.

Following the early work of Johnson, ([Jo 74]), on classifying approximation properties of NP complete optimization problems, in the late 1970's, Ausiello, D'Atri, and Protasi, and independently Paz and Moran moved beyond the study of approximation algorithms for specific NP complete problems and began the systematic study of the structure of NP optimization problems and their reductions. They addressed the question of what types of classifications can be made among NP optimization problems and the question of how the structural and combinatorial properties and reductions of NP problems relate to their ability to be polynomially approximated.

Their work was followed by work of Leggett and Moore, which considered the location of NP optimization problems within the polynomial time hierarchy. Among other results Leggett and Moore showed that NP optimization problems that arise from strongly complete NP decision problems are proper $\Delta_2^P (= P^{NP})$ problems. That is, these problems are not in $NP \cup coNP$ unless $NP = coNP$.

More recently, Papadimitriou and Yannakakis showed that certain languages associated with NP optimization problems are complete for complexity classes thought to lie just above NP. For example, they showed that the language $\{\langle G, k\rangle :$ the maximum clique in G is of size $k\}$ is complete for $D^P (=$ all languages that can be expressed as the intersection of a language in NP and a language in $coNP$), while the language $\{G : G$ has a unique optimal traveling salesperson tour$\}$ is complete for Δ_2^P.

Following the work of Papadimitriou and Yannakakis, Krentel and Wagner began the task of developing a complexity theory for NP optimization problems based on traditional notions of complexity classes. Krentel, drawing from natural problems such as *clique* and *traveling salesperson,* defined a complexity class of *optimization functions* which he called $OptP$. To handle the problem that polynomial many-one reductions are not strong enough to preserve structural properties of optimization problems, Krentel defined the notion of a *polynomial time metric reduction*, which is a polynomial time truth-table reduction between functions that is allowed only one oracle computation. Using these reductions, Krentel showed that the optimization function for the *traveling salesperson* problem is complete for $OptP$, while the optimization function for the *clique* problem is complete for a weaker complexity class. He also showed that the optimization function for the *bin packing* problem is contained in a complexity class which is still weaker than the class related to the *clique* problem. Thus, by defining natural *functional* complexity classes for NP optimization problems, Krentel was able to use completeness for these classes as a way of distinguishing between optimization problems of varying difficulties.

Wagner's results have followed a similar vein using set classes rather than functional classes, but, in addition, he has developed close connections between the Boolean hierarchy over *NP* and sets related to *NP* optimization problems. Noting that all such *NP* optimization problems lie in classes at or below the functional analog of P^{NP}, and in fact lie in the *extended Boolean hierarchy* over *NP*, Wagner has classified problems by locating them in this Boolean hierarchy. His work carefully details both the types of functions that one might try to optimize for *NP* decision problems, (that is, the various ways that an optimization problem might be created from a decision problem), and the classes of the *Boolean hierarchy* for which the resulting problems are complete.

At the same time that the work on classification of optimization problems was taking place, additional research was being done on polynomial approximations. Papadimitriou and Yannakakis defined, using Fagin's logical characterization of *NP*, the classes *MAX NP* and *MAX SNP*, which characterize many *NP* optimization problems that are polynomially approximable. Similarly, Johnson, Papadimitriou, and Yannakakis defined the class *PLS* of functions that map instances of optimization problems to feasible solutions which witness local optima, and Krentel showed that the "local optima" function for *weighted satisfiability* is complete for *PLS.* On a more applied side, recent work in [HoKe 82], [JAMS 89], and [KGV 83] has provided unified techniques for polynomially approximating a wide variety of *NP* complete problems.

In the sections which follow we attempt to give an overview of the diverse work on *NP* optimization. In doing so, we trace the development of concepts and definitions that originated with Ausiello, D'Atri and Protasi, Moran and Paz, Leggett and Moore through the more recent work of Johnson, Papadimitriou, Yannakakis, Krentel and Wagner. Where appropriate we raise open questions.

2. NP OPTIMIZATION PROBLEMS.

In general, to specify an *NP* problem B, one identifies a polynomial time testable predicate P and a polynomial p such that $P(x,y) \Rightarrow |y| \leq p(|x|)$, and defines B by

$$B(x) \Longleftrightarrow (\exists y)P(x,y).$$

Those elements y such that $P(x,y)$ are called *feasible solutions for* x, and the relation $|y| \leq p(|x|)$ guarantees that the sizes of feasible solutions are bounded by a polynomial in the size of x.

Without loss of generality, one can usually think of y as an accepting computation of a nondeterministic Turing machine on input x, although for our purposes it is probably better to think more directly of y as a potential solution to a question about x. For example, y might be a *clique* in a graph x, or a complete *tour* of the graph. It is also natural to have a polynomially computable *measure* or *valuation* function $m(x,y)$ of the goodness of y as a solution to the question about x, and to be interested only in feasible solutions which are sufficiently good. For example, in the case of *clique*[1] if $m(x,y)$ is the number of vertices in the clique y, then the clique problem, $CLIQUE$ is specified by the special form,

$$CLIQUE(x,k) \Longleftrightarrow (\exists y)[P(x,y) \ \& \ m(x,y) \geq k],$$

where $P(x,y)$ asserts that y is a clique of the graph x. Similarly, the *traveling salesperson* problem, is specified by

$$TSP(x,k) \Longleftrightarrow (\exists y)[P(x,y) \ \& \ m(x,y) \leq k],$$

where $P(x,y)$ asserts that y is a tour of the graph x and $m(x,y)$ is a function giving the total length of the tour y.

In the setting of this paper an NP complete problem C that is specified in either of these forms is called an NP *optimization* problem, and we sometimes refer to the class of all such problems as the class NPO. In practice, for NP optimization problems one is often more interested in calculating some variation of the function

$$opt_C(x) =_{def} Max\{m(x,y) : P_C(x,y)\}, \quad \text{or}$$
$$opt_C(x) =_{def} Min\{m(x,y) : P_C(x,y)\}.$$

Proving that the underlying problem is NP complete shows that the problem of computing opt_C is probably hard, but it does not show *how* hard computing opt_C must be, nor does it answer the question of whether we can efficiently find approximate solutions for opt_C.

Of course, by simply taking $m(x,y)$ to be the characteristic function, $\chi_{P_C}(x,y)$, of P_C we can make the problem of solving the underlying NP complete problem fairly directly equivalent to calculating the function $opt_{P_C}(x)$. However, for more interesting cases, for example where $m(x,y)$ measures the size of a clique, or perhaps where $m(x,y)$ measures the weight of a tour of a graph, the reader will see that solving the underlying NP decision problem can be trivially reduced to calculating $opt_{P_C}(x)$, but it is not evident that the problem of calculating $opt_{P_C}(x)$ is so trivially reducible to solving the underlying NP decision problem.

[1] An informal description of some of the decision problems and the functional variants of the optimization problems considered in this paper is given in the Appendix. For a description of the remaining decision problems, we refer the reader to [GaJo 79].

Much work on optimization problems has focused, not on directly calculating the function opt_C, or even the more natural problem of finding a feasible solution y which optimizes the value of $m(x,y)$, but instead on using the already developed machinery for classifying set recognition problems to classify set recognition problems related to the optimization problem.

For example, in [LeMo 81], Leggett and Moore considered the difficulty, for an *NP* complete optimization problem C, of the related set

$$OPT(C) =_{def} \{\langle x,k\rangle : \; k = opt_C(x)\} \qquad (*)$$

which is the problem, given x and k, of deciding whether k is the optimum value of any feasible solution of x. It is easily seen that for any *NP* complete optimization problem C, $OPT(C)$ is ***always*** in the class $\Delta_2^P =_{def} P^{NP} \subseteq \Sigma_2^P \cap \Pi_2^P$ of the polynomial time hierarchy. Leggett and Moore called a set, S, a ***proper*** Δ_2^P set if it is in Δ_2^P and if

$$S \in NP \cup coNP \implies NP = coNP.$$

With this definition, they were able to show that for many natural *NP* optimization problems C, the corresponding optimization problem $OPT(C)$ is a proper Δ_2^P set and is therefore very unlikely to fall in either *NP* or in *coNP*.

The basic tool Leggett and Moore used is the notion of a Σ-preserving reduction, $\leq_\Sigma^P$, which is a polynomial time reduction that preserves membership in the Σ_k^P levels of the polynomial time hierarchy. Examples of Σ-preserving reductions include polynomial time many-one reducibility, $\leq_m^P$, and polynomial time conjunctive truth-table reducibility, $\leq_c^P$. Their basic theorem is that if C is any *NP* optimization problem for which there exists some *NP* complete problem B satisfying $B \leq_\Sigma^P OPT(C)$, then $OPT(C)$ must be a proper Δ_2^P set.

This result can easily be applied to all *NP* complete optimization problems which are ***polynomially bounded***; i.e., to problems for which there is some polynomial p such that for each instance x of C, $opt_C(x) \leq p(|x|)$. It is not hard to see that for any polynomially bounded *NP* complete optimization problem C, $C \leq_c^P OPT(C)$, and thus $OPT(C)$ is a proper Δ_2^P set. It then follows trivially from this that the optimization problems for all ***strongly*** *NP* complete sets[2] are proper Δ_2^P sets. Since many natural

[2] The ***strongly NP complete*** problems as defined by Gary and Johnson ([GaJo 79]) are just those problems of the form $(\exists y)P(x,y)$, which remain *NP* complete when they are restricted to $(\exists y)[P(x,y) \& m(x,y) \leq p(|x|)]$ for some polynomial p. I.e., they are just those which remain *NP* complete when their feasible solutions are restricted to include just those which keep the problem ***polynomially bounded*** as defined by Leggett and Moore.

NP complete optimization problems are strongly *NP* complete, or even polynomially bounded, it turns out that many natural *NP* complete sets automatically give rise to optimization problems which are proper Δ_2^P sets.[3]

Examples of *NP* complete sets for which the general Leggett and Moore technique applies include the polynomially bounded problems CLIQUE, CHROMATIC NUMBER, and SAT, as well as KNAPSACK which is neither polynomially bounded nor strongly NP-complete.

Although the problems Leggett and Moore investigated are proper Δ_2^P sets, Leggett and Moore do not use the full power of Δ_2^P. Specifically, to be sure that k is the size of an optimal solution for an instance x one need only verify that there is some solution of x of size at least k and that there are no solutions of x of size greater than k. From this it is clear that all of the Leggett and Moore sets $OPT(C)$ can be defined as the intersection of a set in *NP* and a set in *coNP*.

In [PaYa 84] Papadimitriou and Yannakakis formally defined D^P to be the class of sets which can be written as the intersection of a set in *NP* and a set in *coNP*, and they showed that MAX CLIQUE $=_{def}$ OPT(CLIQUE) is $\leq_m^P$ complete for D^P. Other problems they showed to be complete for D^P include SAT-UNSAT and problems like CLIQUE FACETS, which is the problem, given a graph G and an inequality, of deciding whether the inequality is a facet of the clique polytope of G?

In [PaWo 85] and in [CaMe 87] the additional polynomially bounded *NP* complete optimization problems MINIMAL UNSATISFIABILITY and (for all k) MINIMAL-k-UNCOLORABILITY were shown to be $\leq_m^P$ complete for D^P.

Prompted by the Papadimitriou and Yannakakis definition of D^P, a number of people began investigating the *Boolean hierarchy* over *NP*, ([CGHHSWW 88], [CGHHSWW 89]), and over other complexity classes as well, ([BBJSY 89]). Boolean hierarchies are typically formed by taking at the base level a class of sets which is closed under union and intersection but not complement, and then forming the $k+1^{st}$ level of the

[3] Leggett and Moore called polynomially bounded problems *essentially unary,* but the terminology *polynomially bounded* has since become more standard. *Traveling salesperson* is a natural example of a problem which is strongly *NP* complete but whose unrestricted version is not polynomially bounded.

Leggett and Moore go on to show that if one starts, not with optimization problems for *NP*, but rather with optimization problems in the Σ_k^P level of the polynomial hierarchy, then corresponding results to those for Δ_2^P will show that various optimization sets are proper Δ_{k+1}^P sets. These results are very similar to the results discussed here for Δ_2^P, but are beyond the scope of this brief survey.

Boolean hierarchy by iterated closure under k operations involving union, intersection, and complement. If done properly, D^P is just $NP[2]$, the second level of the Boolean hierarchy over NP.

Following the development of the Boolean hierarchy over NP and the proof by Papadimitriou and Yannakakis that MAX CLIQUE is complete for D^P, Wagner was interested in investigating when different variations of optimization problems are complete for the k^{th} level of the Boolean hierarchy over NP. To do this, he defined a number of interesting variations of optimization problems. For our purposes, the most interesting of these is the following generalization of (*):

$$OPT_k(C) =_{def} \{\langle x, a_1, \ldots, a_k\rangle : opt_C(x) \in \{a_1, \ldots, a_k\}\ \}. \qquad (**)$$

Note that $OPT_1(C) = OPT(C)$. Wagner observed that if C is any NP optimization problem, then the corresponding problem $OPT_k(C)$ is in the $2k^{th}$ level, $NP[2k]$, of the Boolean hierarchy over NP.[4] He defined a valuation function, m to be *polynomially invertible* if the set, $Sol(x,k)$, of feasible solutions of size k,

$$Sol(x,k) =_{def} \{y : P_C(x,y) \ \&\ m(x,y) = k\},$$

is completely enumerable (from x and k) in polynomial time. He then observed that for every NP optimization problem, C, with an invertible valuation function, the optimization set $OPT_k(C)$ is in $coNP$, and hence these sets are very *unlikely* to be complete for $NP[2k]$. Wagner also showed that the one specific problem that he considered with an invertible valuation function, MAX SAT ASSIGN, is *complete* for $coNP$.

Wagner next considered nine common NP optimization problems for which the valuation function is not polynomially invertible, including seven which are polynomially bounded in the sense of Leggett and Moore's definition and two which are neither polynomially bounded nor polynomially invertible. (Of these latter two, TRAVELING SALESPERSON is strongly NP complete, but the other, SUM OF SUBSETS is *not* strongly NP complete.) Wagner gave detailed reductions showing that for each of these nine optimization problems, C, the corresponding optimization problem $OPT_k(C)$ is complete for $NP[2k]$.

[4] At first glance, one might expect it to be easier to answer whether $opt_{P_C}(x) \in \{a_1, a_2\}$ than to answer whether $opt_{P_C}(x) = a_1$, but note that it seems harder to answer that $opt_{P_C}(x)$ is *neither* a_1 *nor* a_2 than to answer that $opt_{P_C}(x)$ is *not* a_1, and this is what makes deciding $OPT_2(C)$ (presumably) harder than deciding $OPT_1(C)$.

Recall that Leggett and Moore's basic theorem gives very *general* conditions on NP complete optimization problems which force their associated optimization sets $OPT(C)$ to be proper Δ_2^P sets. Given the many examples of optimization problems that are complete for D^P, and given Wagner's nine examples of optimization problems that are complete for $NP[2k]$, it is natural to ask whether there are similar *general* conditions which force the NP complete optimization problems $OPT_k(C)$ to be *complete* for $NP[2k]$. For example, it is tempting to conjecture that for every NP complete optimization problem which is polynomially bounded, the set $OPT_k(C)$ is complete for $NP[2k]$. This seems an interesting problem even for the special case $k = 1$, where $NP[2] = D^P$.

Question 1. *Is every NP complete optimization problem which also is polynomially bounded, or even strongly NP complete, $\leq_m^P$ complete for D^P?*

In addition to the variations of optimization problems considered by Wagner, Papadimitriou investigated another type of optimization problem, optimization problems which have a *unique* solution. In [Pa 84] he proved that the problem of determining whether the optimum solution for a *traveling salesperson* problem is *unique*, the problem of determining whether the optimum solution for an *integer program* is *unique*, and the problem of determining whether the optimum solution for the *knapsack* problem is *unique* are each $\leq_m^P$ complete for Δ_2^P. This leads us to ask:

Question 2. *What type of conditions, such as those of Leggett and Moore, must one place on an NP complete optimization problem C in order to guarantee that $UNIQUEOPT(C) =_{def} \{x : \exists! y[m(x,y) = opt_C(x)]\}$ is complete for Δ_2^P?*

So far we have indirectly discussed how difficult it is to compute the *optimization function*

$$opt_C(x) =_{def} Max\{m(x,y) : P_C(x,y)\}$$

by instead examining the difficulty of the associated set

$$OPT(C) =_{def} \{\langle x, k\rangle : k = opt_C(x)\}.$$

Motivated partly by a desire to better understand which NP complete optimization problems can be approximated in polynomial time, Krentel, ([Kr 88]), initiated a direct study of the *function* class:[5]

[5] At this point notation becomes overbearing and one quickly realizes that the characters "P" and "Opt" are being greatly over used. In the literature the situation is commonly worse. Wagner and Wechsung, ([WaWe 86]) carefully provide unique names for all sets and functions of interest to the study of NP optimization. But our belief is that their naming does not solve

$$\mathbf{OptP} =_{def} \{opt_C : C \text{ is an } NP \text{ optimization problem}\}.$$

To study differences within this class, he also introduced, for any "smooth" function, z, the subclasses:

$$\mathbf{OptP}[z(n)] =_{def} \{f : f \in \mathbf{OptP} \ \& \ |f(x)| \le z(|x|)\}.$$

To study these functional classes, Krentel needed some suitable notion of a polynomial time reduction from one *function,* f, to another function, g. In the context of his work, the notion of a *one-truth-table reduction* turned out to be appropriate.[6]

Definition. *Let f and g be functions. We define $f \le_{1tt}^P g$ if there exist polynomially computable functions t_1 and t_2 such that $f(x) = t_2(x, g(t_1(x)))$.*

These definitions enabled Krentel to separate various optimization problems by directly separating the *functional* versions of the problems. For example, he showed that for the *NP* complete optimization problems MAX WEIGHTED CLAUSES, MIN TRAVELING SALESPERSON, MAX SAT ASSIGN, 0-1 INTEGER PROGRAMMING, and MAX KNAPSACK the corresponding functions opt_C are all $\le_{1tt}^P$ complete for *OptP.* He also showed that for the following *NP* complete optimization problems MAX SAT, MAX CLIQUE, MIN CHROMATIC-NUMBER, and LENGTH OF LONGEST CYCLE the corresponding functions opt_C are $\le_{1tt}^P$ complete for $\mathbf{OptP}[log(n)]$. Most, but not all, of the proofs of these results come by straightforward variations of standard reductions used to prove that the underlying *NP* complete sets are complete problems for *NP*.

Krentel also observed that most (but not all) of the $\le_{1tt}^P$ reductions used in the preceding proofs can be taken to be *linear reductions,* i.e., reductions in which the outermost component t_2 of the reduction separates nicely into *linear components,* t_3 and t_4. I.e.,

$$f(x) \ =_{def} \ t_2(x, g(t_1(x))) \ = \ t_3(x) * g(t_1(x)) + t_4(x).$$

This enabled Krentel to directly relate completeness for the functional classes to completeness for the more standard set classes. For example:

the problem of providing *intuitive* names for the common functions and classes. (Surely this is an open problem which should be solved before someone introduces yet another class!)

[6] Krentel actually called these *metric* reductions, but since the reductions are basically polynomial time with one free *oracle computation* of the function g, they correspond directly to the classical notion of a polynomial time one-truth-table reduction.

Theorem.

- *If f is complete for* **OptP** *under linear reductions, then $L_{2,f} =_{def} \{\langle x, k_1, k_2\rangle : f(x) = k_1 \pmod{k_2}\}$ is $\leq_m^P$ complete for Δ_2^P.*
- *If f is complete for* **OptP**[2] *under linear reductions, then $L_{3,f} =_{def} \{\langle x, k\rangle : f(x) = k\}$ is $\leq_m^P$ complete for D^P.*
- *If f is complete for* **OptP**[1] *under linear reductions, then $L_{4,f} =_{def} \{\langle x, k\rangle : f(x) \geq k\}$ is $\leq_m^P$ complete for NP.*

For the same functions z used to define $\mathbf{OptP}[z(n)]$, Krentel also establishes a close relationship between

$$\mathbf{FP}^{SAT}[z(n)] =_{def} \{f : \ f(x) \text{ is computable in polynomial time given } z(|x|) \text{ queries to an oracle for } SAT\}$$

and $\mathbf{OptP}[z(n)]$. In particular, he proved, for suitable "smooth" f and g :

Theorem. *Suppose that for all n, $f(n) < g(n)$ and that $f(n) < (1 - \epsilon) * log(n)$ for some constant $\epsilon > 0$. Then*

$$\mathbf{FP}^{SAT}[f(n)] = \mathbf{FP}^{SAT}[g(n)] \quad \Longrightarrow \quad P = NP.$$

This result implies that it is unlikely that certain **OptP** complete functions can be approximated well by a polynomially computable function. For example, notice that if $opt_C(x)$ is an **OptP** function and if in polynomial time we can approximate it within an additive quantity $a(|x|)$, then we can use binary search to compute $opt_C(x)$ with $log(a(|x|))$ queries to a SAT oracle. This gives particularly intriguing information about MIN BIN PACKING. On the face of it, based on the size of its output, the optimization function opt_{BIN} for MIN BIN PACKING belongs to $\mathbf{OptP}[log(n)]$, and hence to $\mathbf{FP}^{SAT}[log(n)]$. But Karmakar and Karp, ([KaKa 82]) have given a polynomial time algorithm which approximates the optimal solution to MIN BIN PACKING to within an additive constant of at most $O(log^2(n))$. As pointed out in [Kr 88], this implies that $opt_{BIN} \in \mathbf{FP}^{SAT}[O(loglog(n))]$ and that problems such as opt_{CLIQUE} cannot be reduced to opt_{BIN} under linear polynomial time reductions unless $P = NP$.

Krentel goes on to prove the following more general result about lower bounds for polynomial approximations of **OptP** complete functions:

Theorem. *Suppose $P \neq NP$. Suppose also that opt_C is* **OptP**$[f(n)]$ *complete, where $f \in O(log(n))$ is smooth. Then there exists an $\epsilon > 0$ such that any polynomial time approximation algorithm A for opt_C must have $|A(x) - opt_C(x)| \geq (1/2)^{2^{f(|x|^\epsilon)}}$ infinitely often.*

As we proceed, we shall see that to obtain such a nice result on the difficulty of *approximating* optimization problems, it is not surprising that Krentel was forced to use some form of *linear*, as opposed to just $\leq^P_{1tt}$, reductions.

Given that Karmakar and Karp's result places opt_{BIN} in $\mathbf{FP}^{SAT}[O(loglog(n))]$, it is natural to conjecture that opt_{BIN} is *complete* for $\mathbf{FP}^{SAT}[O(loglog(n))]$; but the best that is known is that opt_{BIN} is hard for $\mathbf{FP}^{SAT}[1]$. In light of these considerations Krentel raised the following question:

Question 3.

- *Which function classes $\mathbf{FP}^{SAT}[z(n)]$ contains opt_{BIN}?*
- *Are there natural complete problems for subclasses of FP^{SAT} other than $FP^{SAT}[O(log(n))]$ and $FP^{SAT}[n^{O(1)}]$?*

3. APPROXIMATING SOLUTIONS TO OPTIMIZATION PROBLEMS.

In spite of the fact that finding optimal solutions of NP complete optimization problems is at least NP hard, it was recognized early on that some, but not all, such functions have "approximate" solutions which can be calculated in polynomial time. This phenomenon stimulated two lines of research on optimization problems. In one direction it stimulated research to find faster and better approximation algorithms, and in another direction it led to attempts to give structural properties common to all polynomially approximable optimization problems.

Among the techniques that have wide applicability, *neighborhood* or *local search* is perhaps the most used. An interesting problem is thus the extent to which this technique can be applied. For example: Is it possible to apply a local search algorithm to obtain approximate values for each function contained in **OptP** ? This question was first investigated by Johnson, Papadimitriou, and Yannakakis ([JPY 85]), and subsequently by Krentel ([Kr 89]). Their main results are briefly discussed in Section 3.1.

General questions related to the study of combinatorial or structural properties common to polynomially approximable optimization problems have been investigated by many people and we discuss a variety of these results in Section 3.2. Section 3.3 contains a brief discussion of various restricted reducibilities which have been used in structural studies of NP optimization problems.

3.1. The Difficulty of Local Search.

Johnson, Papadimitriou and Yannakakis have noted that in practice local search is one of the most successful techniques used to compute polynomial time approximations for optimization functions. The applicability of this technique is related to the existence of a *neighborhood structure* which specifies for each feasible solution y a neighboring set of feasible solutions whose measure is "close" to the measure of y.

Given an optimization problem with a neighborhood structure, a local search algorithm operates as follows: Starting from a randomly chosen feasible solution, until no better neighboring solution exists it repeatedly replaces the current solution by a neighboring solution with the best measure. Once this has been done the algorithm has identified a "local optimum." Typically, th algorithm is repeated a certain number of times with different initial solutions and the locally optimal solution that is found is chosen.[7]

In [JPY 85], Johnson, Papadimitriou, and Yannakakis began an investigation of the class of problems which can be approximated through local search techniques. First they introduced a complexity class of functions called *PLS*. Specifically *PLS*, for *Polynomial Local Search*, is the class of functions that map instances of optimization problems with a given neighborhood structure to local optima. The *PLS* function is computed from an optimization problem and a neighborhood structure which must satisfy the following requirements:

- Given an instance of the problem, we must be able to produce in polynomial time some solution.
- Given an instance and a solution, we must be able in polynomial time to compute the measure of the solution.
- Given an instance and a solution, we must be able in polynomial time to determine whether that solution is locally optimal and if not to generate a neighboring solution of improved measure.

[7] Obviously, there are different variations of this technique. The algorithm that we have described is probably the most simple. "Simulated annealing" is another popular method, ([KGV 83]). But see [JAMS 89] for a critical evaluation. In the remainder of this paper, when we refer to a local search technique we implicitly intend any reasonable (polynomial time) local search technique.

The class PLS turns out to lie somewhere between FP and FNP (i.e. the functional analogues of P and NP). Johnson, Papadimitriou, and Yannakakis note that if PLS contains some NP-hard function, then $NP = coNP$; thus it seems unlikely that $PLS = FNP$. On the other hand $PLS = P$ implies the existence of a general method for finding local optima, and to date no such method is known. So the questions $FP =? PLS =? FNP$ have no obvious answers.

To gain insight into this new complexity class Johnson, Papadimitriou, and Yannakakis defined a reduction between the optimization problems which underlie PLS functions. Intuitively, these reductions not only must map instances of a problem A to instances of a problem B, but must also map local optima of A to local optima of B. More precisely, we say that a problem A in PLS is reducible to another problem B if there are polynomially computable functions f and g such that:

- f maps instances of A to instances of B,
- g maps $\langle$*solutions, instances*$\rangle$ pairs for instances in the range of f back to solutions of A,
- for all instances x of A, if s is a local optimum for the instance $f(x)$ of B, then $g(s, f(x))$ is a local optimum for x.

With this notion of reducibility, Johnson, Papadimitriou, and Yannakakis introduce the corresponding notion of PLS completeness and gave several examples of functional problems which are PLS complete. (The most natural of these is the problem, FLIP, which given a circuit produces a binary *input* whose *output* cannot be increased by flipping a single bit of the input. Feasible solutions which differ in only bit are then in the same *neighborhood,* so that in polynomial time one can generate and evaluate all feasible solutions which lie in the neighborhood of a given candidate solution.) These authors conjectured that for each PLS complete problem the problem of verifying local-optimality is itself $LOGSPACE$-complete for P. However, this conjecture is very unlikely to be true since Krentel, ([Kr 89]), has proven that the problem MAX WEIGHTED CLAUSES is PLS complete, but that the corresponding verification problem is in $LOGSPACE$.

3.2. Polynomially Approximable NP Complete Optimization Problems.

One of the first people to study the problem of finding approximate solutions to NP optimization problems was Johnson, who in his seminal paper, ([Jo 74]), provided a series of results on the approximability of problems such as MAX CLIQUE, MAX SAT, and MIN CHROMATIC NUMBER and gave what are now the standard definitions of polynomial time approximability.

Definition. *Let C be an NP maximization problem. Let f be any polynomially computable function mapping instances x of C to feasible solutions of x. Let $approx(x) =_{def} m(x, f(x))$. We say that approx is a polynomial time approximation algorithm for $opt_C(x)$ if the ratio $\frac{opt_C(x)}{approx(x)}$ is bounded by some constant ϵ greater than one.*[8]

Note that the ratio $\frac{opt_C(x)}{approx(x)}$ always lies between one and $(+)$infinity. Thus to say that a *maximization* problem C has a polynomial time approximation algorithm merely says that the solution $approx(x)$ always comes within a multiplicative factor of $opt_C(x)$. Notice also that this notion of approximability differs from that used by Krentel to state his results about approximability. In fact, ϵ is a multiplicative factor here while Krentel's results require an additive factor.[9]

In order to preserve the fact that the ratio $\frac{opt_C(x)}{approx(x)}$ is in the range $[1, +\infty)$, for *minimization* problems Johnson reverses the ratio and requires that the ratio $\frac{approx(x)}{opt_C(x)}$ is bounded by some constant ϵ greater than one.

Of course, it is even better if we can approximate the optimal solution to within any desired accuracy. In this case, the function f which produces the approximating feasible solution will also have to be a function of some ϵ which gives the desired accuracy:

[8] Where necessary, we shall assume without further discussion that $opt_C(x)$ is bounded away from zero.

[9] Another difference is that Krentel's approximation algorithm A need not be computed through an intermediate function f which actually finds feasible solutions $f(x)$ such that $A(x) = m(x, f(x))$. In practice of course, if one is building approximation algorithms, one wants to find, not just an approximation to the *value* of the optimal solution, but also a *feasible solution* whose value gives a good approximation to the value of the optimal solution. Paz and Moran, ([PaMo 81]), call approximation algorithms which work by actually finding feasible solutions *constructive* approximation algorithms, using the unmodified term *approximation algorithms* for those which may merely find vaules which approximate the value of the optimal solution. They also prove a number of results relating the two concepts. However their terminology is not currently standard, and most authors require in their definitions, as we do, that all approximation algorithms be constructive in the sense that they work by actually finding feasible solutions. Thus in the remainder of this survey, we assume that approximation algorithms work by finding feasible solutions. Nevertheless many results, particularly those involving lower bounds, will hold for approximation algorithms which are not constructive in the sense of Paz and Moran, and the reader interested in such results will have to carefully consult the relevant literature.

Definition. *Let C be an NP optimization problem.*

- *We say that opt_C has a polynomial time approximation scheme if there exists some function $approx(x,\epsilon)$ such that for every ϵ greater than zero $approx(x,\epsilon)$ is polynomially computable as a function of $|x|$ and for every instance x of C, $\frac{opt_C(x)}{approx(x,\epsilon)} \leq 1+\epsilon$ if C is a maximization problem, (or such that $\frac{approx(x,\epsilon)}{opt_C(x)} \leq 1+\epsilon$ if C is a minimization problem).*
- *We say that opt_C has a full polynomial time approximation scheme if the time complexity of $approx(x,\epsilon)$ can be bounded by a polynomial in both $|x|$ and $1/\epsilon$.*

It is well-known that there are approximation algorithms for the optimization versions of NP complete problems such as TRAVELING SALESPERSON (with triangle inequality), VERTEX COVER, and SAT. It is also known that unless $P = NP$ these problems do not have full polynomial time approximation schemes.[10] This leaves as an important open question whether such problems have polynomial time approximation schemes.

An early study characterizing which NP complete optimization problems are approximable and which are fully approximable was given by Paz and Moran, ([PaMo 81]). They began by observing that in many optimization problems, once one *fixes* k, the set

$$\{x : opt_C(x) \leq k\} \qquad (***)$$

is recognizable in polynomial time. For example, for any fixed k it is possible to decide in polynomial time whether a given graph has a clique of size at least k. Paz and Moran called problems for which (***) is always solvable in polynomial time *simple,* and they called them *p-simple* if there is a *uniform* polynomial q such that (***) is always recognizable in time $q(|x|, k)$. Using the notions of simplicity together with multiplicative and polynomial bounds on the optimal solutions, they then went on to completely characterize those NP complete optimization problems which are polynomially approximable and those which have fully polynomial time approximation schemes:

[10] The formulation of these quite general results comes from the work of many researchers. Some of the key early contributions were made in [Jo 74], [GoSa 76], [IbKi 75], [Sa 76], [Ni 75], [GaJo 79]. However the reader should refer to [GaJo 79] or to [WaWe 86] for an exhaustive list.

Theorem. *An NP complete maximization problem, C, is polynomially approximable if and only if*
(1) C is simple, and
(2) there is a constant Q_0 such that for each instance x and each integer $h > 0$,

$$0 \leq opt_C(x)/h - b(x,h) \leq Q_0,$$

where b is a function mapping $\langle instance, nonzero\ \ integer \rangle$ pairs to integers whose time complexity is bounded by a polynomial in $|x|$ and $opt_C(x)/h$.

Theorem. *An NP complete maximization problem, C, has a fully polynomial time approximation scheme if and only if*
(1) C is p-simple, and
(2) there is a polynomial $q(n)$, such that for each instance x and for each integer $h > 0$:

$$0 \leq opt_C(x)/h - b(x,h) \leq q(|x|),$$

where b is a function mapping $\langle instance, nonzero\ \ integer \rangle$ pairs to integers whose time complexity is bounded by a polynomial in $|x|$ and $opt_C(x)/h$.

Exactly analogous theorems hold for *NP* complete *minimization* problems, but with the inequalities appropriately reversed.

While Paz and Moran's results give necessary and sufficient conditions for optimization problems to possess approximate solutions, the characterizations are not completely satisfying since the characterizations themselves are stated in terms of approximating solutions. Clearly what one would like are *intrinsic* characterizations of optimization problems which are themselves sufficient to guarantee the existence of approximating solutions.

One such *intrinsic* characterization was recently given by Papadimitriou and Yannakakis, ([PaYa 88]), who introduced yet another pair of complexity classes for optimization problems: *MAX NP* and *MAX SNP*. These classes give characterizations which are sufficient to guarantee the existence of polynomial time approximation algorithms.

To formalize the definitions of *MAX SNP* and *MAX NP*, Papadimitriou and Yannakakis used Fagin's logical characterization of *NP*, ([Fa 74]). Recall that Fagin showed that every predicate in *NP* can be expressed in the form $\exists S \phi(G, S)$ where G and S are structures and ϕ is a first order formula. For instance we can express SAT as:

$$\exists T \forall c \exists y \, [P(c, y) \,\&\, y \in T \ \vee \ N(c, y) \,\&\, y \notin T],$$

where $P(c, y)$ is *true* if the variable y occurs positively in the clause c and $N(c, y)$ is *true* if the variable y occurs negatively in the clause c. In general the first order predicate ϕ can always be expressed in the form $\forall \overline{x} \exists \overline{y} \, \psi(\overline{x}, \overline{y}, G, S)$, where ψ is quantifier free. Furthermore, in many cases with a little work ϕ can be expressed in the restricted form $\forall \overline{x} \, \psi(\overline{x}, G, S)$.

The class *MAX NP* is then defined to be the following class of maximization problems:

$$\text{If } \ \exists S \forall \overline{x} \exists \overline{y} \, \psi(\overline{x}, \overline{y}, G, S) \in NP, \quad \text{then } \ Max_S |\{\overline{x} : \exists \overline{y} \, \psi(\overline{x}, \overline{y}, G, S)\}| \in MAX \ NP.$$

Similarly, the class *MAX SNP* [11] is defined to be the following class of maximization problems:

$$\text{If } \ \exists S \forall \overline{x} \, \psi(\overline{x}, G, S) \in NP, \quad \text{then } \ Max_S |\{\overline{x} : \psi(\overline{x}, G, S)\}| \in MAX \ SNP.$$

Obviously, MAX SAT is in *MAX NP*, and Papadimitriou and Yannakakis showed that MAX 3SAT is in *MAX SNP*. To investigate these classes, they defined the notion of an *L-reduction* from an optimization problem A to an optimization problem B (which are both assumed to be maximization problems) to be a polynomial transformation f such that there are two constants $\alpha, \beta > 0$, and for each instance x of A:

- the optima of x and $f(x)$, $opt_A(x)$ and $opt_B(f(x))$ respectively, satisfy $opt_B(f(x)) \ \leq \ \alpha * opt_A(x)$ and
- for any feasible solution y of $f(x)$, we can find in polynomial time a feasible solution y' of x satisfying $\ opt_A(x) - m(x, y') \ \leq \ \beta * [opt_B(f(x)) - m(f(x), y)]$.

Using this notion they prove that every problem in *MAX SNP* has a polynomial approximation algorithm, and they state that the same is true for *MAX NP* . They also give a dozen examples of problems which are complete for *MAX SNP* with respect to L-reductions, of which the most natural are perhaps MAX 3SAT, MAX 2SAT, and MAX CUT.

Their work naturally suggests the following general problems:

[11] The name *SNP* comes from the fact that the formulas $\exists S \forall \overline{x} \, \psi(\overline{x}, G, S)$, without the first order existential quantifier, are called "strict" Σ_1^1 formulas.

Question 4. *MAX NP and MAX SNP contain problems which are clearly NP optimization problems, but it is surely not true that every problem which we think of as an NP optimization problem is in MAX NP. Can one characterize in some more complexity theoretic fashion (other than the definition) those NP optimization problems that are in MAX NP? Can one give intrinsic complexity theoretic characterizations of other classes of NP complete optimization problems which always have some form of polynomial approximations?*

Question 5. *As an alternative to the second order characterization of Fagin, Hodgson and Kent, ([HoKe 82]), have given a first order characterization of NP by bounding the domain of existential and universal quantifiers. Can this characterization be used either to give an alternative definition of MAX NP and MAX SNP, or to define other classes of NP complete optimization problems which always have some form of polynomial approximations? If so, is this first-order approach more natural, and how do its "max classes" compare with those of Papadimitriou and Yannakakis?*

3.3. Reductions for NP Optimization Problems.

The fact that all naturally occurring examples of *NP* complete problems are polynomial time isomorphic but not all are approximable tells us that the standard reductions of complexity theory are inadequate for studying relations among optimization problems. For example, we earlier saw that Krentel needed to introduce the notion of a "linear" reduction to study such relations, and we have just seen that Papadimitriou and Yannakakis needed to introduce "L-reductions" to study the relations between classes of optimization problems with similar approximation properties.

Earlier studies of relations among optimization problems also introduced similar restrictions on the reductions. For example, Paz and Moran, ([PM 81]), called the reductions used in their studies of approximations of *NP* optimization problems *polynomial time measure preserving reductions* and *polynomial time ratio preserving reductions.* (The *measure preserving* reductions are merely reductions among *NP* optimization problems which always carry instances of the reduced problem which have maximal solutions of size k to instances of another problem which have maximal solutions of size exactly $Q_0 * k$ where Q_0 is some multiplicative constant. The *ratio preserving* reductions are similar, but the size of the optimal solution is only preserved to within a multiplicative *interval.*)

Paz and Moran used these reductions to show that a variety of *NP* complete problems can be interreduced via reductions that preserve approximability properties. Thus they were able to transfer known approximability results (or nonapproximability results) from one *NP* complete optimization problem to another. Among other examples, they used this technique in reducing MAX SAT to MAX CLIQUE, MIN CLIQUE COVER to MIN CHROMATIC NUMBER, and MIN CHROMATIC NUMBER to MIN CLIQUE COVER.

In this context, it is interesting to note that Paz and Moran were the first to use such restricted reductions to obtain *complete* versions of optimization problems. For example they showed that, with respect to polynomial time measure preserving reductions, MAX WEIGHTED VARIABLE is complete for the class of *NP* optimization problems. Thus, if MAX WEIGHTED VARIABLE is approximable, so is *every NP* optimization problem.

More recently, Crescenzi and Panconesi ([CrPa 88]), building on earlier work by Orponen and Mannila ([OrMa 87]), have developed a theory of complete problems not only for NPO, the class of *NP* optimization problems, but more importantly, for APX, the class of problems which admit polynomial approximations, for $PTAS$, the class of problems which admit polynomial time approximation schemes, and for $FPAS$, the class of problems which are fully polynomially approximable. Obviously,

$$FPAS \subseteq PTAS \subseteq APX \subseteq NPO.$$

For each of these classes except $FPAS$, extending a definition due to Orponen and Mannila, Crescenzi and Panconesi defined an appropriate natural form of reduction[12] and then use these reductions to define *complete* sets for $PTAS$, APX, and NPO. The reductions are not only reasonably natural, but most important, just as any complete problem for *NP* falling into *P* implies that $NP = P$, if any of the complete problems for any of these three largest classes falls into the immediately smaller class, then the whole class collapses to the immediately smaller class. *Thus proving that a problem is complete for a class is strong evidence that it cannot be approximated by a stronger form of polynomial approximation.*

[12] The first two of the Crescenzi and Panconesi reductions are credited to Orponen and Mannila, ([OrMa 87]), and are very similar to the L-reductions of [PaYa 88].

Orponen and Mannila show that TRAVELING SALESPERSON and $0-1$ INTEGER PROGRAMMING are NPO complete. Following this, using techniques similar to those used by Paz and Moran for showing that MAX WEIGHTED VARIABLE is complete for NPO under measure preserving reductions, Crescenzi and Panconesi showed that BOUNDED MAX WEIGHTED VARIABLE is complete for APX, and that LINEAR BOUNDED MAX WEIGHTED VARIABLE (which they call LINEAR BSAT) is $PTAS$ complete under the reductions they use. They also show that NPO contains various problems which are not complete for NPO and which are not in APX unless $P = NP$.

Question 6, ([CrPa 88]). *Are there natural incomplete problems for NPO or natural sets which are complete for $PTAS$?*

Question 7. *The reductions used by Orponen and Mannila and those used by Paz and Moran are different. What is the relationship between the reductions and between the complete sets for NPO under these reductions? Almost every paper we have discussed introduces one or more new restricted reductions to study approximation problems. Although many ideas for the restrictions appear similar, it is not clear what the actual relations are among the different restricted reductions used by different authors. It would be interesting to know what the relationship between these various restrictions really is and whether all are necessary to obtain the results which we have surveyed. Is there some clear notion of what a "correct" reduction should be? Of the results surveyed here, the work of Crescenzi and Panconesi provides the strongest justification for the "correctness" of the Orponen-Mannila type reducibilities, (and, because of the similarities of the reductions, thus also for the L-reductions of Papadimitriou and Yannakakis). These reducibilities seem to give appropriately fine distinctions without begin overly cumbersome.*

While discussing restricted forms of reducibilities which preserve approximability properties, we should mention the early work of Ausiello, D'Atri and Protasi. In ([ADP 80]) they obtained information about the approximability of NP optimization problems by giving a very detailed examination of the combinatorial and internal structures both of the problems being reduced and of the reductions. In this paper they introduced the notion of the *structure* of an optimization problem, the notion of a *convex* optimization problem, and the notion of a *structure preserving reduction.* Informally, the structure of an instance x of an optimization problem is a list which contains the number of feasible solutions for x associated with each admissible value of the measure m. An optimization problem is said to be *convex* if for each instance x there is at least one feasible solution y such that $m(x,y) = h$ for every h such that $worst(x) \leq h \leq opt_C(x)$.

Using much more restricted reductions than the restricted reductions we have already discussed, Ausiello, D'Atri and Protasi specify reductions by pairs of polynomially computable functions $\langle f_1, f_2 \rangle$ such that an optimization problem A is reducible to an optimization problem B if each instance x, of A, is mapped by f_1 to an instance of B, f_2 carries the *measures* of feasible solutions of x to the *measures* of feasible solutions of $f(x)$. A polynomially structure preserving reduction is such a reduction which maps an optimization problem A to an optimization problem B in such a way that each instance of A is mapped to an instance of B with *exactly* the same structure. They showed that parsimonious reductions which satisfy very strong linearity conditions are always structural preserving.[13]

Ausiello, *et. al.*, useed these reducibilities to examine relationships among a variety of *NP* complete optimization problems. While from our current perspective the conditions on their structure preserving reductions may seem too restrictive to still be of general interest, Ausiello *et. al.* were among the first to give conditions under which combinatorial problems have similar approximability properties:[14]

Theorem. *Let A and B be two convex NP optimization problems which are both maximization or minimization problems. If there are reductions $\langle f_1, f_2 \rangle$ from A to B, and $\langle g_1, g_2 \rangle$ from B to A such that:*

- *both are structure preserving,*
- *both are strictly monotone,*
- *both are essentially linear:*

$$f_2(x,k) = a(x) + k \ , \quad g_2(y,h) = b(y) + h \ , \quad \text{and} \quad a(x) \geq -b(f_1(x)),$$

[13] P. Orponen has pointed out to us that, rather interestingly, motivated partially by Simon's results on "parsimonious" reductions, Lynch and Lipton, ([LyLi 78]), also investigated "structure preserving reductions" which are very similar in spirit to those introduced by Ausiello, *et.al.* However, unlike Ausiello, *et.al.*, Lynch and Lipton do not discuss the application of structure preserving reductions to approximating solutions for optimization problems.

[14] [ADP 80] gives a more general theorem than state here, but we have adopted the following version because it gives a good intuition while avoiding heavier notation.

The measure used here to evaluate the quality of an approximation algorithm is different from the standard one (namely the ratio $\frac{opt_C(x)}{approx(x)}$, or its reciprocal) introduced by Johnson. In particular, the measure used by Ausiello, D'Atri and Protasi is $\frac{opt_C(x) - approx(x)}{opt_C(x) - worst(x)}$, where worst($x$) is the value of the worst feasible solution for x. This measure is not only intuitively very appealing, but it has the nice property that it is symmetric with respect to maximization and minimization; a property that does not hold for the standard measure. Since the function *worst* can usually be taken to be zero for maximization problems, these measures are usually equivalent for maximization problems. But they are *not* equivalent for minimalization problems. Although this measure seems never to have been subsequently used, [ADP 80; p 145] contains a useful discussion of why the authors believe this measure should be preferred for minimization problems.

then B has a polynomial approximation algorithm if and only if A has a polynomial approximation algorithm.

Ausiello, D'Atri and Protasi then went on to group a variety of well-known *NP* complete optimization problems into distinct equivalence classes using this theorem, and they prove that some of the classes are distinct under these reductions.

4. BIBLIOGRAPHY.

[ADP 80] G. Ausiello, A. D'Atri and M. Protasi, "Structure preserving reductions among convex optimization problems," *J Comput and System Sci*, **21** (1980), 136–153; (first appeared in *Proc 4th ICALP*, Lecture Notes in Computer Science, **52** (1977), 45-60.)

[BeHa 77] L. Berman and J. Hartmanis, "On isomorphism and density of *NP* and other complete sets," *SIAM J Comput,* **1** (1977), 305-322.

[BBJSY 89] A. Bertoni, D. Bruschi, D. Joseph, M. Sitharam and P. Young, "Generalized Boolean hierarchies and Boolean hierarchies over *RP*," *Univ Wisconsin CS Dept Tech Report*, **#809**, 1–40; (short abstract to appear in *FCT '89 Proceedings.*)

[CGHHSWW 89] J. Cai, T. Gundermann, J. Hartmanis, L. Hemachandra, V. Sewelson, K. Wagner, and G. Wechsung, "The Boolean hierarchy II: applications," *SIAM J Comput,* **7** (1989), 95–111.

[CrPa 88] P. Crescenzi and A. Panconesi, "Completeness in approximation classes," to appear in *FCT '89 Proceedings,* (1989), 16 pages.

[Fa 74] R. Fagin, "Generalized first-order spectra, and polynomial-time recognizable sets," *Complexity and Computations*, R. Karp (ed.), AMS, (1974).

[GaJo 79] M. Garey and D. Johnson, "Computers and Intractability: A Guide to the Theory of *NP*-Completeness," Freeman, 1979.

[GoSa 76] T. Gonzales and S. Sahni, "P-complete approximation problems," *J ACM*, **23** (1976), 555–565.

[HoKe 82] B. Hodgson and C. Kent, "An arithmetical characterization of *NP* ," *Theor Comput Sci*, **21** (1982), 255–267.

[HoSh 86] D. Hochbaum and D. Shmoys, "A unified approach to approximation algorithms for bottleneck problems," *J ACM*, **33** (1986), 533–550.

[IbKi 75] O. Ibarra and C. Kim, "Fast approximation algorithms for the knapsack and sum of subsets problems," *J ACM*, **22** (1975), 463–468.

[JAMS 89] D. Johnson, C. Aragon, L. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation," *Operations Research*, to appear.

[Jo 74] D. Johnson, "Approximation algorithms for combinatorial problems," *J Comput and System Sci*, **9** (1974), 256–278.

[JPY 88] D. Johnson, C. Papadimitriou and M. Yannakakis, "How easy is local search?," *J Comput and System Sci*, **37** (1988), 79–100; (first appeared in *Proc 26th IEEE Symp Foundations of Comput Sci*, (1985), 39–42.)

[JoYo 85] D. Joseph and P. Young, "Some remarks on witness functions for polynomial reducibilities in *NP*," *Theor Comput Sci*, **39** (1985), 225-237.

[KaKa 82] N. Karmakar and R. Karp, "An efficient approximation scheme for the one-dimensional bin-packing problem," *IEEE 23rd Symp Foundations of Comput Sci*, (1982), 312–320.

[KGV 83] S. Kirkpatrick, C. Gelat, and M. Vecchi, "Optimization by simulated annealing," *Science*, **220** (1983), 671-680.

[Kr 88] M. Krentel, "The complexity of optimization problem," *J Comput and System Sci*, **36** (1988), 490–509; (first appeared in *Proc 18th ACM Symp Theory on Comput*, (1986), 69–76.)

[Kr 89] M. Krentel, "On finding locally optimal solutions," *Proc Structure in Complexity Conference*, (1989), 132-137.

[LeMo 81] E. Leggett and J. Moore, "Optimization problems and the polynomial time hierarchy," *Theor Comput Sci*, **15** (1981), 279–289.

[LyLi 78] N. Lynch and R. Lipton "On structure preserving reductions," *Siam J Comput*, **7** (1978), 119-126.

[OrMa 87] P. Orponen and H. Mannila, "On approximation preserving reductions: complete problems and robust measures," Tech Report, University of Helsinki, 1987.

[Ni 75] R. Nigmatullin, "Complexity of the approximate solution of combinatorial problems," *Dokl Akad Nauk*, SSSR **224**, (1975) 289–292 (in Russian), English translation in *Soviet Math Dokl*, **16**, 1199–1203.

[Pa 84] C. Papadimitriou, "On the complexity of unique solutions," *J ACM*, **31** (1984), 392–400.

[PaMo 81] A. Paz and S. Moran, "Non deterministic polynomial optimization problems and their approximation," *Theor Comput Sci*, **15** (1981), 251–277; (first appeared in *Proc 4th ICALP*, Lecture Notes in Computer Science, **52** (1977), 45-60.)

[PaWo 85] C. Papadimitriou and D. Wolfe, "The complexity of facets resolved," *J Comput and System Sci*, **37** (1988), 2–13; (first appeared in *Proc IEEE 26th Symp on Foundations of Comput Sci*, (1985), 74–78).

[PaYa 84] C. Papadimitriou and M. Yannakakis, "The complexity of facets (and some facets of complexity)," *J Computer and System Sciences*, **28** (1984), 244–259, (first appeared in *Proc 14th ACM Symp Theory of Comput*, (1982), 255–260.)

[PaYa 88] C. Papadimitriou and M. Yannakakis, "Optimization, approximation, and complexity classes," *Proc 20th ACM Symp Theory Comput*, (1988), 229–234.

[Sa 76] S. Sahni, "Algorithms for scheduling independent tasks," *J ACM*, **23** (1976), 116–127.

[Wa 86] K. Wagner, "More complicated questions about maxima and minima, and some closure properties of *NP*," *Theor Comput Sci*, **51** (1987), 53–80; (first appeared in *Proc 13th ICALP,* Lecture Notes Computer Science, **226** (1986), 434–443.)

[Wa 88] K. Wagner, "Bounded query computations," *Proc Structure in Complexity Conference,* (1988), 260–277.

[WaWe 85] K. Wagner and G. Wechsung, "Computational complexity," D. Reidel Publishing, 1986.

APPENDIX

Decision Problems

MINIMAL-k-UNCOLORABILITY

Instance: a graph G.

Question: Is G uncolorable with k colors, but by removing any node from G the resulting graph is k colorable?

MINIMAL UNSATISFIABILITY

Instance: a Boolean formula ϕ in conjunctive normal form with at most three literals per clause and at most two occurrences of each literal.

Question: Is ϕ unsatisfiable, but removing any clause renders it satisfiable?

SAT-UNSAT

Instance: two Boolean formulas ϕ and ψ.

Question: Is ϕ satisfiable and ψ is unsatisfiable?

SUM OF SUBSETS

Instance: a finite set S of positive integers and a positive integer b.

Question: Is there a set $A \subseteq S$ such that the sum of the elements is equal to b?

Functional Version of Optimization Problems

BOUNDED MAX WEIGHTED VARIABLES

(Same as MAX WEIGHTED VARIABLES except the input includes a constant W satisfying $\sum w_i \leq 2 * W$, the weight W is attached to any *unsatisfying* assignment, and the output is the maximum weight over *all* assignments, whether satisfying or unsatisfying.)

INTEGER PROGRAMMING

Instance: integer matrix A and integer vectors B and C.

Output: the maximum value of $C^T x$ over all vectors x of integers subject to the linear constraint $Ax \leq B$.

FLIP

Instance: a circuit with n inputs and n outputs.

Output: an input whose output (when viewed as a binary integer) cannot be reduced by flipping any single bit of the input.

LINEAR BOUNDED MAX WEIGHTED VARIABLES

(Same as BOUNDED MAX WEIGHTED VARIABLES except that the constant W satisfies $\sum w_i \leq (1 + \frac{1}{n-1}) * W$.)

LONGEST CYCLE

Instance: graph G.

Output: the length of the longest cycle in G.

MAX CLIQUE

Instance: graph G.

Output: the size of the largest clique in G.

MAX CUT

Instance: graph G with integer weights on the edges.

Output: the maximum k obtainable by partitioning the graph G into two subgraphs G_1 and G_2 and then summing the weights of the edges that have one endpoint in G_1 and the other endpoint in G_2.

MAX KNAPSACK

Instance: integers $x_1, \ldots, x_n, N$.

Output: the largest value of $\sum_{i \in S} x_i$ for $S \subseteq 1, \ldots, n$, which is less than N.

MAX SAT

Instance: Boolean formula in conjunctive normal form.

Output: the maximum number of simultaneously satisfiable clauses.

MAX SAT ASSIGN

Instance: Boolean formula $\phi(x_1, \ldots, x_n)$.

Output: the lexicographic maximum assignment which satisfies ϕ, or 0 if ϕ is not satisfiable.

MAX WEIGHTED CLAUSES

Instance: Boolean formula in conjunctive normal form with weights on the clauses.

Output: the maximum weight of any satisfying assignment, where the weight of an assignment is the sum of weights on the true clauses.

MAX WEIGHTED VARIABLES

Instance: Boolean formula in conjunctive normal form with nonnegative weights, w_i, on the variables, x_i.

Output: the maximum weight of any satisfying assignment, where the weight of an assignment is the sum of weights on the true variables; output negative if there is no satisfying assignment.

MIN BIN PACKING

Instance: finite set of items U, an integer "size" for each $u \in U$ and a positive integer bin capacity B.

Output: the minimum k such that there exists a partition of U into disjoint sets $U_1, \ldots, U_k$ and the sum of the size of the items in each U_i is less than or equal to B.

MIN CLIQUE COVER

Instance: a graph G.

Output: the minimum number of disjoint cliques in G whose union is G.

MIN TRAVELING SALESPERSON

Instance: graph G with integers weights on the edges.

Output: the length of the shortest traveling salesperson tour in G.

MIN VERTEX COVER

Instance: a graph G.

Output: the size of the minimum cover of G.