On Syntactic versus Computational Views of Approximability ¶

SANJEEV KHANNA* RAJEEV MOTWANI† MADHU SUDAN‡ UMESH VAZIRANI§
Stanford University Stanford University IBM Research U.C. Berkeley

Abstract

We attempt to reconcile the two distinct views of approximation classes: *syntactic* and *computational*. Syntactic classes such as MAX SNP permit structural results and have natural complete problems, while computational classes such as APX allow us to work with classes of problems whose approximability is well-understood. Our results provide a syntactic characterization of computational classes, and give a computational framework for syntactic classes.

We compare the syntactically defined class MAX SNP with the computationally defined class APX, and show that every problem in APX can be "placed" (i.e., has approximation preserving reduction to a problem) in MAX SNP. Our methods introduce a simple, yet general, technique for creating approximation-preserving reductions which show that any "well" approximable problem can be reduced in an approximation-preserving manner to a problem which is hard to approximate to corresponding factors. The reduction then follows easily from the recent non-approximability results for MAX SNP-hard problems. We demonstrate the generality of this technique by applying it to other classes such as RMAX(2) and MIN $F^+\Pi_2(1)$ which have the clique problem and the set cover problem, respectively, as complete problems.

The syntactic nature of MAX SNP was used by Papadimitriou and Yannakakis [23] to provide approximation algorithms for every problem in the class. We provide an alternate approach to demonstrating this result using the syntactic nature of MAX SNP. We develop a general paradigm, *non-oblivious local search*, useful for developing simple yet efficient approximation algorithms. We show that such algorithms can find good approximations for all MAX SNP problems, yielding approximation ratios comparable to the best-known for a variety of specific MAX SNP-hard problems. Non-oblivious local search provably out-performs standard local search in both the degree of approximation achieved and the efficiency of the resulting algorithms.

Keywords: approximation algorithms, complete problems, computational complexity, computational classes, polynomial reductions, local search.

AMS Subject Classification: 68Q15.

*Department of Computer Science, Stanford University, Stanford, CA 94305 (`sanjeev@theory.stanford.edu`). Supported by an OTL grant, NSF Grant CCR-9357849, and a Schlumberger Foundation Fellowship.

†Department of Computer Science, Stanford University, Stanford, CA 94305 (`rajeev@theory.stanford.edu`). Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Development Award, an OTL grant, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

‡IBM T.J. Watson Research Center, Yorktown Heights, NY 10598 (`madhu@watson.ibm.com`).

§Computer Science Division, U.C. Berkeley, CA 94720 (`vazirani@cs.berkeley.edu`). Supported by NSF Grant CCR-9310214.

¶A preliminary version of this paper appeared in [18].

1 Introduction

The approximability of NP optimization (NPO) problems has been investigated in the past via the definition of two different types of problem classes: syntactically-defined classes such as MAX SNP (the class of NPO problems expressible as bounded-arity constraint satisfaction problems) and computationally-defined classes such as APX (the class of NPO problems to which a constant factor approximation can be found in polynomial time); see Section 2 for formal definitions. The former is useful for obtaining structural results and has natural complete problems, while the latter allows us to work with classes of problems whose approximability is completely determined. We attempt to develop linkages between these two views of approximation problems and thereby obtain new insights about both types of classes. We show that a natural generalization of MAX SNP renders it identical to the class APX. This further validates Papadimitriou and Yannakakis's definition of MAX SNP as providing a structural basis to the study of approximability. As a side-effect, we resolve the open problem of identifying complete problems for MAX NP. Our techniques extend to a generic theorem that can be used to create an approximation hierarchy. We also develop a generic algorithmic paradigm which is guaranteed to provide good approximations for MAX SNP problems, and may also have other applications.

1.1 Background and Motivation

A wide variety of classes are defined based directly on the polynomial-time approximability of the problems contained within, e.g., APX (constant-factor approximable problems), PTAS (problems with polynomial-time approximation schemes), and FPTAS (problems with fully-polynomial-time approximation schemes). The advantage of working with classes defined using approximability as the criterion is that it allows us to work with problems whose approximability is well-understood. Crescenzi and Panconesi [8] have recently also been able to exhibit complete problems for such classes, particularly APX. Unfortunately such complete problems seem to be rare and artificial, and do not seem to provide insight into the more natural problems in the class. Research in this direction has to find approximation-preserving reductions from the known complete but artificial problems in such classes to the natural problems therein, with a view to understanding the approximability of the latter.

The second family of classes of NPO problems that have been studied are those defined via syntactic considerations, based on a syntactic characterization of NP due to Fagin [10]. Research in this direction, initiated by Papadimitriou and Yannakakis [23], and followed by Panconesi and Ranjan [22] and Kolaitis and Thakur [20], has led to the identification of approximation classes such as MAX SNP, RMAX(2), and MIN $\mathrm{F}^+\Pi_2(1)$. The syntactic prescription in the definition of these classes has proved very useful in the establishment of complete problems. Moreover, the recent results of Arora, Lund, Motwani, Sudan, and Szegedy [2] have established the hardness of approximating complete problems for MAX SNP to within (specific) constant factors unless $\mathrm{P} = \mathrm{NP}$. It is natural to wonder why the hardest problems in this syntactic sub-class of APX should bear any relation to all of NP.

Though the computational view allows us to precisely classify the problems based on their approximability, it does not yield structural insights into natural questions such as: Why certain problems are easier to approximate than some others? What is the canonical structure of the hardest representative problems of a given approximation class? and, so on. Furthermore, intuitively speaking, this view is too abstract to facilitate identification of, and reductions to establish, natural complete problems for a class. The syntactic view, on the other hand, is essentially a structural view. The syntactic prescription gives a natural way of identifying canonical hard problems in the class and performing approximation-preserving reductions to establish complete problems.

Attempts at trying to find a class with both the above mentioned properties, i.e., natural complete problems and capturing all problems of a specified approximability, have not been very successful. Typically the focus has been to relax the syntactic criteria to allow for a wider class of problems to be included in the class. However in all such cases it seems inevitable that these classes cannot be expressive enough to encompass all problems with a given approximability. This is because each of these syntactically defined approximation classes is strictly contained in the class NPO; the strict containment can be shown by syntactic considerations alone. As a result if we could show that any of these classes contains all of P, then we would have separated P from NP. We would expect that every class of this nature would be missing some problems from P, and this has indeed been the case with all current definitions.

We explore a different direction by studying the structure of the syntactically defined classes when we look at their closure under approximation-preserving reductions. The idea of looking at the closure of a class is implicit in the work of Papadimitriou and Yannakakis [23] who state that: *minimization problems will be "placed" in the classes through L-reductions to maximization problems*. The advantage of looking at the closure of a set is that it maintains the complete problems of the set, while managing to include all of P into the closure (for problems in P, the reduction is to simply use a polynomial time algorithm to compute an exact solution). It now becomes interesting, for example, to compare the closure of MAX SNP (denoted $\overline{\text{MAX SNP}}$) with APX. A positive resolution, i.e., $\overline{\text{MAX SNP}} = \text{APX}$, would immediately imply the non-existence of a PTAS for MAX SNP-hard problems, since it is known that PTAS is a strict subset of APX, if P $\neq$ NP. On the other hand, an unconditional negative result would be difficult to obtain, since it would imply P $\neq$ NP.

Here we resolve this question in the affirmative. The exact nature of the result obtained depends upon the precise notion of an approximation preserving reduction used to define the closure of the class MAX SNP. The strictest notion of such reductions available in the literature are the L-reductions due to Papadimitriou and Yannakakis [23]. We work with a slight extension of the reduction, which we call E-reductions. Using such reductions to define the class $\overline{\text{MAX SNP}}$ we show that this equals APX-PB, the class of all polynomially bounded NP optimization problems which are approximable to within constant factors. By using slightly looser definitions of approximation preserving reductions (and in particular the PTAS-reductions of Crescenzi et al [9]) this can be extended to include all of APX into $\overline{\text{MAX SNP}}$. We then build upon this result to identify an interesting hierarchy of such approximability classes. An interesting side-effect of our results is the positive answer to the question of Papadimitriou and Yannakakis [23] about whether MAX NP has any complete problems.

The syntactic view seems useful not only in obtaining structural complexity results but also in developing paradigms for designing efficient approximation algorithms. This was demonstrated first by Papadimitriou and Yannakakis [23] who show approximation algorithms for every problem in MAX SNP. We further exploit the syntactic nature of MAX SNP to develop another paradigm for designing good approximation algorithms for problems in that class and thereby provide an alternate computational view of it. We refer to this paradigm as *non-oblivious local search*, and it is a modification of the standard local search technique [25]. We show that every MAX SNP problem can be approximated to within constant factors by such algorithms. It turns out that the performance of non-oblivious local search is comparable to that of the best-known approximation algorithms for several interesting and representative problems in MAX SNP. An intriguing possibility is that this is not a coincidence, but rather a hint at the universality of the paradigm or some variant thereof.

Our results are related to some extent to those of Ausiello and Protasi [4]. They define a class GLO (for Guaranteed Local Optima) of NPO problems which have the property that for all locally optimum solutions, the ratio between the value of the global and the local optimum is bounded by a constant. It follows that GLO is a subset of APX, and it was shown that it is in fact a strict subset. We show that a MAX SNP problem is not contained in GLO, thereby establishing that MAX SNP is not contained in GLO. This contrasts with our notion of non-oblivious local search which is guaranteed to provide constant factor approximations for all problems in MAX SNP. In fact, our results indicate that non-oblivious local search is significantly more powerful than standard local search in that it delivers strictly better constant ratios, and also will provide constant factor approximations to problems not in GLO. Independently of our work, Alimonti [1] has used a similar local search technique for the approximation of a specific problem not contained in GLO or MAX SNP.

1.2 Summary of Results

In Section 2, we present the definitions required to state our results, and in particular the definitions of an E-reduction, APX, APX-PB, MAX SNP and $\overline{\text{MAX SNP}}$. In Section 3, we show that $\overline{\text{MAX SNP}} = $ APX-PB. A generic theorem which allows to equate the closure of syntactic classes to appropriate computational classes is outlined in Section 4; we also develop an approximation hierarchy based on this result.

The notion of non-oblivious local search and NON-OBLIVIOUS GLO is developed in Section 5. In Section 6, we illustrate the power of non-obliviousness by first showing that oblivious local search can achieve at most the performance ratio $3/2$ for MAX 2-SAT, even if it is allowed to search *exponentially* large neighborhoods; in contrast, a very simple non-oblivious local search algorithm achieves a performance ratio of $4/3$. We then establish that this paradigm yields a $2^k/(2^k-1)$ approximation to MAX *k*-SAT. In Section 7, we provide an alternate characterization of MAX SNP via a class of problems called MAX *k*-CSP. It is shown that a simple non-oblivious algorithm achieves the best-known approximation for this problem, thereby providing a *uniform* approximation for all of MAX SNP. In Section 8, we further illustrate the power of this class of algorithms by showing that it can achieve the best-known ratio for a specific MAX SNP problem and for VERTEX COVER (which is not contained in GLO). This implies that MAX SNP is not contained in GLO, and that GLO is strict subset of NON-OBLIVIOUS GLO. In Section 9, we apply it to approximating the traveling salesman problem. Finally, in Section 10, we apply this technique to improving a long-standing approximation bound for maximum independent sets in bounded-degree graphs.

2 Preliminaries and Definitions

Given an NPO problem Π and an instance $\mathcal{I}$ of Π, we use $|\mathcal{I}|$ to denote the length of $\mathcal{I}$ and $OPT(\mathcal{I})$ to denote the optimum value for this instance. For any solution S to $\mathcal{I}$, the value of the solution, denoted by $V(\mathcal{I}, S)$, is assumed to be a polynomial time computable function which takes positive integer values (see [7] for a precise definition of NPO).

Definition 1 (Error) *Given a solution S to an instance $\mathcal{I}$ of an* NPO *problem* Π, *we define its* error $\mathcal{E}(\mathcal{I}, S)$ *as*

$$\mathcal{E}(\mathcal{I}, S) = \max\left\{\frac{V(\mathcal{I}, S)}{OPT(\mathcal{I})}, \frac{OPT(\mathcal{I})}{V(\mathcal{I}, S)}\right\} - 1.$$

Notice that the above definition of error applies uniformly to the minimization and maximization problems at all levels of approximability.

Definition 2 (Performance Ratio) *An approximation algorithm A for an optimization problem* Π *has* performance ratio $\mathcal{R}(n)$ *if, given an instance $\mathcal{I}$ of* Π *with $|\mathcal{I}| = n$, the solution $A(\mathcal{I})$ satisfies*

$$\max\left\{\frac{V(\mathcal{I}, A(\mathcal{I}))}{OPT(\mathcal{I})}, \frac{OPT(\mathcal{I})}{V(\mathcal{I}, A(\mathcal{I}))}\right\} \leq \mathcal{R}(n).$$

A solution of value within a multiplicative factor r of the optimal value is referred to as an r-approximation.

The performance ratio for A is $\mathcal{R}$ if it always computes a solution with error at most $\mathcal{R} - 1$.

2.1 E-reductions

We now describe the precise approximation preserving reduction we will use in this paper. This reduction, which we call the E-reduction, is essentially the same as the L-reduction of Papadimitriou and Yannakakis [23] and differs from it in only one relatively minor aspect.

Definition 3 (E-reduction) *A problem* Π E-reduces *to a problem* Π' *(denoted* $\Pi \propto_E \Pi'$*) if there exist polynomial time computable functions* f, g *and a constant* β *such that*

- f *maps an instance* $\mathcal{I}$ *of* Π *to an instance* $\mathcal{I}'$ *of* Π' *such that* $OPT(\mathcal{I})$ *and* $OPT(\mathcal{I}')$ *are related by a polynomial factor i.e. there exists a polynomial* $p(n)$ *such that* $OPT(\mathcal{I}') \leq p(|\mathcal{I}|)OPT(\mathcal{I})$.
- g *maps solutions* S' *of* $\mathcal{I}'$ *to solutions* S *of* $\mathcal{I}$ *such that*

$$\mathcal{E}(\mathcal{I}, S) \leq \beta\mathcal{E}(\mathcal{I}', S').$$

Remark 1 *Among the many approximation preserving reductions in the literature, the* L*-reduction appears to be the strictest. The* E*-reduction appears to be slightly weaker (in that it allows polynomial scaling of the problems), but is stricter than any of the other known reductions. Since all the reductions given in this paper are* E*-reductions, they would also qualify as approximation-preserving reductions under most other definitions and in particular they fit the definitions of* F*-reductions and* P*-reductions of Crescenzi and Panconesi* [8].

Remark 2 *Having* $\Pi \propto_E \Pi'$ *implies that* Π *is as well approximable as* Π'*; in fact, an* E*-reduction is an* FPTAS*-preserving reduction. An important benefit is that this reduction can be applied uniformly at all levels of approximability. This is not the case with the other existing definitions of* FPTAS*-preserving reduction in the literature. For example, the* FPTAS*-preserving reduction (*F*-reduction) of Crescenzi and Panconesi* [8] *is much more unrestricted in scope and does not share this important property of the* E*-reduction. Note that Crescenzi and Panconesi* [8] *showed that there exists a problem* $\Pi' \in$ PTAS *such that for any problem* $\Pi \in$ APX, $\Pi \propto_F \Pi'$*. Thus, there is the undesirable situation that a problem* Π *with no* PTAS *has a* FPTAS*-preserving reduction to a problem* Π' *with a* PTAS.

Remark 3 *The* L*-reduction of Papadimitriou and Yannakakis* [23] *enforces the condition that the optima of an instance* $\mathcal{I}$ *of* Π *be linearly related to the optima of the instance* $\mathcal{I}'$ *of* Π' *to which it is mapped. This appears to be an unnatural restriction considering that the reduction itself is allowed to be an arbitrary polynomial time computation. This is the only real difference between their* L*-reduction and our* E*-reduction, and an* E*-reduction in which the linearity relation of the optimas is satisfied is an* L*-reduction. Intuitively, however, in the study of approximability the desirable attribute is simply that the errors in the corresponding solutions are closely (linearly) related. The somewhat artificial requirement of a linear relation between the optimum values precludes reductions between problems which are related to each other by some* scaling factor. *For instance, it seems desirable that two problems whose objective functions are simply related by any fixed polynomial factor should be inter-reducible under any reasonable definition of an approximation-preserving reduction. Our relaxation of the* L*-reduction constraint is motivated precisely by this consideration.*

Let $\mathcal{C}$ be any class of NPO problems. Using the notion of an E-reduction, we define hardness and completeness of problems with respect $\mathcal{C}$, as well its closure and polynomially-bounded sub-class.

Definition 4 (Hard and Complete Problems) *A problem* Π' *is said to be* $\mathcal{C}$-hard *if for all problems* $\Pi \in \mathcal{C}$, *we have* $\Pi \propto_E \Pi'$. *A* $\mathcal{C}$*-hard problem* Π *is said to be* $\mathcal{C}$-complete *if in addition* $\Pi \in \mathcal{C}$.

Definition 5 (Closure) *The* closure *of* $\mathcal{C}$*, denoted by* $\overline{\mathcal{C}}$*, is the set of all* NPO *problems* Π *such that* $\Pi \propto_E \Pi'$ *for some* $\Pi' \in \mathcal{C}$.

Remark 4 *The closure operation maintains the set of complete problems for a class.*

Definition 6 (Polynomially Bounded Subset) *The* polynomially bounded subset *of* $\mathcal{C}$*, denoted* $\mathcal{C}$*-PB, is the set of all problems* $\Pi \in \mathcal{C}$ *for which there exists a polynomial* $p(n)$ *such that for all instances* $\mathcal{I} \in \Pi$, $OPT(\mathcal{I}) \leq p(|\mathcal{I}|)$.

2.2 Computational and Syntactic Classes

We first define the basic computational class APX.

Definition 7 (APX) *An* NPO *problem* Π *is in the class* APX *if there exists a polynomial time algorithm* A *for* Π *with performance ratio bounded by some constant c.*

The class APX-PB consists of all polynomially bounded NPO problems which can be approximated within constant factors in polynomial time.

If we let F-APX denote the class of NPO problems that are approximable to within a factor F, then we obtain a *hierarchy* of approximation classes. For instance, poly-APX and log-APX are the classes of NPO problems which have polynomial time algorithms with performance ratio bounded polynomially and logarithmically, respectively, in the input length. More precise versions of these definitions are provided in Section 4.

Let us briefly review the definition of some syntactic classes.

Definition 8 (MAX SNP and MAX NP [23]) MAX SNP *is the class of* NPO *problems expressible as finding the structure* S *which maximizes the objective function*

$$V(\mathcal{I}, S) = |\{\vec{x} \mid \Phi(\mathcal{I}, S, \vec{x})\}|,$$

where $\mathcal{I} = (U; \mathcal{P})$ *denotes the input (consisting of a finite universe* U *and a finite set of bounded arity predicates* $\mathcal{P}$*),* S *is a finite structure, and* Φ *is a quantifier-free first-order formula. The class* MAX NP *is defined analogously except the objective function is*

$$V(\mathcal{I}, S) = |\{\vec{x} \mid \exists \vec{y}, \Phi(\mathcal{I}, S, \vec{x}, \vec{y})\}|.$$

A natural extension is to associate a weight with every tuple $\vec{x}$; the modified objective is to find an S which maximizes $V(\mathcal{I}, S) = \sum_{\vec{x}} w(\vec{x})\Phi(\mathcal{I}, S, \vec{x})$, where $w(\vec{x})$ denotes the weight associated with the tuple $\vec{x}$.

Example 1 (MAX k-SAT) *The* MAX *k*-SAT *problem is: given a collection of* m *clauses on* n *boolean variables where each (possibly weighted) clause is a disjunction of precisely k literals, find a truth assignment satisfying a maximum weight collection of clauses. For any fixed integer* k*,* MAX *k*-SAT *belongs to the class* MAX SNP*. The results of Papadimitriou and Yannakakis [23] can be adapted to show that for* $k \geq 2$, MAX *k*-SAT *is complete under* E*-reductions for the class* MAX SNP.

Definition 9 (RMAX(k) [22]) RMAX(k) *is the class of* NPO *problems expressible as finding a structure* S *which maximizes the objective function*

$$V(\mathcal{I}, S) = \begin{cases} |\{\vec{x} \mid S(\vec{x})\}| & \text{if } \forall \vec{y}, \Phi(\mathcal{I}, S, \vec{y}) \\ 0 & \text{otherwise} \end{cases}$$

where S *is a single predicate and* $\Phi(\mathcal{I}, S, \vec{y})$ *is a quantifier-free CNF formula in which* S *occurs at most k times in each clause and all its occurrences are negative.*

The results of Panconesi and Ranjan [22] can be adapted to show that MAX CLIQUE is complete under E-reductions for the class RMAX(2).

Definition 10 (MIN $\mathbf{F^+\Pi_2(k)}$ [20]) *MIN $\mathrm{F}^+\Pi_2(k)$ is the class of* NPO *problems expressible as finding a structure S which minimizes the objective function*

$$V(\mathcal{I}, S) = \begin{cases} |\{\vec{x} : S(\vec{x})\}| & \textit{if}\ \forall \vec{x}, \exists \vec{y}, \Phi(\mathcal{I}, S, \vec{x}, \vec{y}) \\ 0 & \textit{otherwise} \end{cases}$$

where S is a single predicate, $\Phi(\mathcal{I}, S, \vec{y})$ is a quantifier-free CNF formula in which S occurs at most k times in each clause and all its occurrences are positive.

The results of Kolaitis and Thakur [20] can be adapted to show that SET COVER is complete under E-reductions for the class MIN $\mathrm{F}^+\Pi_2(1)$.

3 MAX SNP Closure and APX-PB

In this section, we will establish the following theorem and examine its implications. The proof is based on the results of Arora et al [2] on efficient proof verifications.

Theorem 1 $\overline{\text{MAX SNP}} = \text{APX-PB}$.

Remark 5 *The seeming weakness that* $\overline{\text{MAX SNP}}$ *only captures polynomially bounded* APX *problems can be removed by using looser forms of approximation-preserving reduction in defining the closure. In particular, Crescenzi and Trevisan [9] define the notion of a* PTAS*-preserving reduction under which* $\text{APX} = \overline{\text{APX-PB}}$. *Using their result in conjunction with the above theorem, it is easily seen that* $\overline{\text{MAX SNP}} = \text{APX}$. *This weaker reduction is necessary to allow for reductions from fine-grained optimization problems to coarser (polynomially-bounded) optimization problems (cf.* [9]*).*

The following is a surprising consequence of Theorem 1.

Theorem 2 $\overline{\text{MAX NP}} = \overline{\text{MAX SNP}}$.

Papadimitriou and Yannakakis [23] (implicitly) introduced both these closure classes but did not conjecture them to be the same. It would be interesting to see if this equality can be shown independent of the result of Arora et al [2]. We also obtain the following resolution to the problem posed by Papadimitriou and Yannakakis [23] of finding complete problems for MAX NP.

Theorem 3 MAX SAT *is complete for* MAX NP.

The following sub-sections establish that $\overline{\text{MAX SNP}} \supseteq \text{APX-PB}$. The idea is to first E-reduce any minimization problem in APX-PB to a maximization problem in therein, and then E-reduce any maximization problem in APX-PB to a specific complete problem for MAX SNP, viz., MAX 3-SAT. Since an E-reduction forces the optimas of the two problems involved to be related by polynomial factors, it is easy to see that $\overline{\text{MAX SNP}} \subseteq \text{APX-PB}$. Combining these two facts, we obtain Theorem 1.

3.1 Reducing Minimization to Maximization

Observe that the fact that Π belongs to APX implies the existence of an approximation algorithm A and a constant c such that

$$\frac{OPT(\mathcal{I})}{c} \leq V(\mathcal{I}, A(\mathcal{I})) \leq c \times OPT(\mathcal{I}).$$

Henceforth, we will use $a(\mathcal{I})$ to denote $V(\mathcal{I}, A(\mathcal{I}))$. We first reduce any minimization problem $\Pi \in$ APX-PB to a maximization problem $\Pi' \in$ APX-PB, where the latter is obtained by merely modifying the objective function for Π, as follows. Let Π' have the objective function

$$V'(\mathcal{I}, S) = \max\{1, (c+1)a(\mathcal{I}) - cV(\mathcal{I}, S)\},$$

for all instances $\mathcal{I}$ and solutions S for Π. Clearly, $V'(\mathcal{I}, S)$ takes only positive values. To ensure that $V'(\mathcal{I}, S)$ is integer-valued, we can assume without loss of generality, that c is an integer (a real-valued performance ratio can always be rounded up to the next integer). It can be verified that the optimum value for any instance $\mathcal{I}$ of Π' always lies between $a(\mathcal{I})$ and $(c+1)a(\mathcal{I})$. Thus A is a $(c+1)$-approximation algorithm for Π'.

Now given a solution S' for instance $\mathcal{I}$ of Π' such that it has error δ, we want to construct a solution S for instance $\mathcal{I}$ of Π such that the error is at most $\beta\delta$ for some β. We will show this for $\beta = (c+1)$.

First consider the case when $V'(\mathcal{I}, S') = 1$ i.e. $\delta = a(\mathcal{I}) - 1$. In this case, we simply output the solution $S = A(\mathcal{I})$. If $a(\mathcal{I}) = 1$ then we are trivially done else we observe that

$$\mathcal{E}(I, S) \leq (c-1) \leq (c+1)(a(\mathcal{I}) - 1) \leq \beta\mathcal{E}'(I, S').$$

On the other hand, if $V'(\mathcal{I}, S') > 1$, we may proceed as follows. If S' is a δ-error solution to the optimum of Π', i.e.,

$$V'(\mathcal{I}, S) \geq \frac{OPT'(\mathcal{I})}{1+\delta} \geq (1-\delta)OPT'(\mathcal{I}),$$

where $OPT'(\mathcal{I})$ is the optimal value of V' for $\mathcal{I}$, we can conclude that

$$\begin{aligned}
V(\mathcal{I}, S) &= \frac{(c+1)a(\mathcal{I}) - V'(\mathcal{I}, S)}{c} \\
&\leq \frac{(c+1)a(\mathcal{I}) - OPT'(\mathcal{I}) + \delta \times OPT'(\mathcal{I})}{c} \\
&\leq \frac{c \times OPT(\mathcal{I}) + \delta \times OPT'(\mathcal{I})}{c} \\
&\leq OPT(\mathcal{I}) + (c+1)\delta\, OPT(\mathcal{I}).
\end{aligned}$$

Thus a solution s to Π' with error δ is a solution to Π with error at most $(c+1)\delta$, implying an E-reduction with $\beta = c+1$.

3.2 NP Languages and MAX 3-SAT

The following theorem, adapted from a result of Arora, Lund, Motwani, Sudan, and Szegedy [2], is critical to our E-reduction of maximization problems to MAX 3-SAT.

Theorem 4 ([2]) *Given a language $L \in$ NP and an instance $x \in \Sigma^n$, one can compute in polynomial time an instance $\mathcal{F}_x$ of* MAX 3-SAT, *with the following properties.*

1. *The formula $\mathcal{F}_x$ has m clauses, where m depends only on n.*

2. *There exists a constant $\epsilon > 0$, independent of the input x, such that $(1-\epsilon)m$ clauses of $\mathcal{F}_x$ are satisfied by some truth assignment.*

3. *If $x \in L$, then $\mathcal{F}_x$ is (completely) satisfiable.*

4. *If $x \notin L$, then no truth assignment satisfies more than $(1-\epsilon)m$ clauses of $\mathcal{F}_x$.*

5. *Given a truth assignment which satisfies more than $(1-\epsilon)m$ clauses of $\mathcal{F}_x$, a truth assignment which satisfies $\mathcal{F}_x$ completely (or, alternatively, a witness showing $x \in L$) can be constructed in polynomial time.*

Some of the properties above may not be immediately obvious from the construction given by Arora, Lund, Motwani, Sudan, and Szegedy [2]. It is easy to verify that they provide a reduction with properties (1), (3) and (4). Property (5) is obtained from the fact that all assignments which satisfy most clauses are actually close (in terms of Hamming distance) to valid codewords from a linear code, and the uniquely error-corrected codeword obtained from this "corrupted code-word" will satisfy all the clauses of $\mathcal{F}_x$.

Property (2) requires a bit more care and we provide a brief sketch of how it may be ensured. The idea is to revert back to the PCP model and redefine the proof verification game. Suppose that the original game had the properties that for $x \in L$ there exists a proof such that the verifier accepts with probability 1, and otherwise, for $x \notin L$, the verifier accepts with probability at most $1/2$. We now augment this game by adding to the proof a 0th bit which the prover uses as follows: if the bit is set to 1, then the prover "chooses" to play the old game, else he is effectively "giving up" on the game. The verifier in turn first looks at the 0th bit of the proof. If this is set, then she performs the usual verification, else she tosses an unbiased coin and accepts if and only if it turns up heads. It is clear that for $x \in L$ there exists a proof on which the verifier always accepts. Also, for $x \notin L$ no proof can cause the verifier to accept with probability greater than $1/2$. Finally, by setting the 0th bit to 0, the prover can create a proof which the verifier accepts with probability exactly $1/2$. This proof system can now be transformed into a 3-CNF formula of the desired form.

3.3 Reducing Maximization to MAX 3-SAT

We have already established that, without loss of generality, we only need to worry about maximization problems $\Pi \in$ APX-PB. Consider such a problem Π, and let A be a polynomial-time algorithm which delivers a c-approximation for Π, where c is some constant. Given any instance $\mathcal{I}$ of Π, let $p = ca(\mathcal{I})$ be the bound on the optimum value for $\mathcal{I}$ obtained by running A on input $\mathcal{I}$. Note that this may be a stronger bound than the a priori polynomial bound on the optimum value for any instance of length $|\mathcal{I}|$. An important consequence is that $p \leq c\,OPT(\mathcal{I})$.

We generate a sequence of NP decision problems $L_i = \{\mathcal{I} | \, OPT(\mathcal{I}) \geq i\}$ for $1 \leq i \leq p$. Given an instance $\mathcal{I}$, we create p formulas $\mathcal{F}_i$, for $1 \leq i \leq p$, using the reduction from Theorem 4, where ith formula is obtained from the NP language L_i.

Consider now the formula $\mathcal{F} = \bigwedge_{i=1}^{p} \mathcal{F}_i$ that has the following features.

- The number of satisfiable clauses of $\mathcal{F}$ is exactly

$$MAX = (1-\epsilon)mp + \epsilon m\,OPT(\mathcal{I}),$$

  where ϵ and m are as guaranteed by Theorem 4.

- Given an assignment which satisfies $(1-\epsilon)mp + \epsilon mj$ clauses of $\mathcal{F}$, we can construct in polynomial time a solution to $\mathcal{I}$ of value at least j. To see this, observe the following: any assignment which so many clauses must satisfy more than $(1-\epsilon)m$ clauses in at least j of the formulas $\mathcal{F}_i$. Let i be the largest index for which this happens; clearly, $i \geq j$. Furthermore, by property (5) of Theorem 4, we can now construct a truth assignment which satisfies $\mathcal{F}_i$ completely. This truth assignment can be used to obtain a solution S such that $V(\mathcal{I}, S) \geq i \geq j$.

In order to complete the proof it remains to be shown that given any truth assignment with error δ, i.e., which satisfies $MAX/(1+\delta)$ clauses of $\mathcal{F}$, we can find a solution S for $\mathcal{I}$ with error $\mathcal{E}(\mathcal{I}, S) \leq \beta\delta$ for some constant β. We show that this is possible for $\beta = (c^2 + c\epsilon)/\epsilon$. The main idea behind finding such a solution is to use the second property above to find a "good" solution to $\mathcal{I}$ using a "good" truth assignment for $\mathcal{F}$.

Suppose we are given a solution which satisfies $MAX/(1+\delta)$ clauses. Since $MAX/(1+\delta) \geq (1-\delta)\,MAX$ and $MAX = (1-\epsilon)mp + \epsilon m\,OPT(\mathcal{I})$, we can use the second feature from above to construct a solution S_1 such that

$$\begin{aligned} V(\mathcal{I}, S_1) &\geq \frac{(1-\delta)\,MAX - (1-\epsilon)mp}{\epsilon m} \\ &\geq (1-\delta)\,OPT(\mathcal{I}) - \frac{\delta}{\epsilon}p \\ &\geq \left(1 - \delta\left(1 + \frac{c}{\epsilon}\right)\right) OPT(\mathcal{I}). \end{aligned}$$

Suppose $\delta \leq (c-1)\epsilon/(c(c+\epsilon))$. Let $\delta^* = \delta(1 + c/\epsilon)$, and $\gamma = \delta^*/(1-\delta^*)$. Then it is readily seen that

$$V(\mathcal{I}, S_1) \geq \frac{OPT(\mathcal{I})}{1+\gamma}$$

and that

$$0 \leq \gamma \leq \left(\frac{c^2 + c\epsilon}{\epsilon}\right)\delta.$$

On the other hand, if $\delta > (c-1)\epsilon/(c(c+\epsilon))$, then the error in a solution S_2 obtained by running the c-approximation algorithm for Π is given by

$$c - 1 \leq \left(\frac{c^2 + c\epsilon}{\epsilon}\right)\delta.$$

Therefore, choosing $\beta = (c^2 + c\epsilon)/\epsilon$, we immediately obtain that the solution with larger value, among S_1 and S_2, has error at most $\beta\delta$. Thus, this reduction is indeed an E-reduction.

4 Generic Reductions and an Approximation Hierarchy

In this section we describe a generic technique for turning a hardness result into an approximation preserving reduction.

We start by listing the kind of constraints imposed on the hardness reduction, the approximation class and the optimization problem. We will observe at the end that these restrictions are obeyed by all known hardness results and the corresponding approximation classes.

Definition 11 (Additive Problems) *An* NPO *problem* Π *is said to be* additive *if there exists an operator* $+$ *and a polynomial time computable function* f *such that* $+$ *maps a pair of instances* $\mathcal{I}_1$ *and* $\mathcal{I}_2$ *to an instance* $\mathcal{I}_1 + \mathcal{I}_2$ *such that* $OPT(\mathcal{I}_1 + \mathcal{I}_2) = OPT(\mathcal{I}_1) + OPT(\mathcal{I}_2)$*, and* f *maps a solution* s *to* $\mathcal{I}_1 + \mathcal{I}_2$ *to a pair of solutions* s_1 *and* s_2 *to* $\mathcal{I}_1$ *and* $\mathcal{I}_2$*, respectively, such that* $V(\mathcal{I}_1 + \mathcal{I}_2, s) = V(\mathcal{I}_1, s_1) + V(\mathcal{I}_2, s_2)$.

Definition 12 (Downward Closed Family) *A family of functions $F = \{f : \mathcal{Z}^+ \rightarrow \mathcal{Z}^+\}$ is said to be* downward closed *if for all $g \in F$ and for all constants c (and in particular for all integers $c > 1$), $g'(n) \in O(g(n^c))$ implies that $g' \in F$. A function g is said to be* hard *for the family F if for all $g' \in F$, there exists a constant c such that $g'(n) \in O(g(n^c))$; the function g is said to be* complete *for F if g is hard for F and $g \in F$.*

Definition 13 (F-APX) *For a downward closed family F, the class F-*APX *consists of all polynomially bounded optimization problems approximable to within a ratio of $g(|\mathcal{I}|)$ for some function $g \in F$.*

Definition 14 (Canonical Hardness) *An* NP *maximization problem Π is said to be* canonically hard *for the class F-*APX *if there exists a transformation T mapping instances of* 3-SAT *to instances of Π, constants n_0 and c, and a gap function G which is hard for the family F, such that given an instance x of* 3-SAT *on $n \geq n_0$ variables and $N \geq n^c$, $\mathcal{I} = T(x, N)$ is an instance of Π with the following properties.*

- *If $x \in$* 3-SAT*, then $OPT(\mathcal{I}) = N$.*
- *If $x \notin$* 3-SAT*, then $OPT(\mathcal{I}) = N/G(N)$.*
- *Given a solution S to $\mathcal{I}$ with $V(\mathcal{I}, S) > N/G(N)$, a truth assignment satisfying x can be found in polynomial time.*

In the above definition, the transformation T from 3-SAT to Π is somewhat special in that one can specify the size/optimum of the reduced problem and T can produce a mapped instance of the desired size. This additional property is easily obtained for additive problems, by using sufficient number of additions till the optimum of the reduced problem is close to the target optimum, and then adding a problem of known optimum value to the reduced problem.

Canonical hardness for NP *minimization problems* is analogously defined: $OPT(\mathcal{I}) = N$ when the formula is satisfiable and $OPT(\mathcal{I}) = NG(N)$, otherwise. Given any solution with value less than $NG(N)$, one can construct a satisfying assignment in polynomial time.

4.1 The Reduction

Theorem 5 *If F is a downward closed family of functions, and an additive* NPO *problem Ω is canonically hard for the class F-*APX-PB*, then all problems in F-*APX-PB *E-reduce to Ω.*

Proof: Let Π be a polynomially bounded optimization problem in F-APX, approximable to within $c(.)$ by an algorithm A, and let Ω be a problem shown to be hard to within a factor $G(.)$ where G is hard for F. Let V and V' denote the objective functions of Π and Ω, respectively. We start with the special case where both Π and Ω are maximization problems. We describe the functions f, g and the constant β as required for an E-reduction.

Let $\mathcal{I} \in \Pi$ be an instance of size n; pick N so that $c(n)$ is $O(G(N))$. To describe our reduction, we need to specify the functions f and g. The function f is defined as follows. Let $m = V(\mathcal{I}, A(\mathcal{I}))$. For each $i \in \{1, \ldots, mc(n)\}$, let L_i denote the NP-language $\{\mathcal{I} | \, OPT(\mathcal{I}) \geq i\}$. Now for each i, we create an instance $\phi_i \in \Omega$ of size N such that if $\mathcal{I} \in L_i$ then $OPT(\phi_i)$ is N, and it is $N/G(N)$ otherwise. We define $f(\mathcal{I}) = \phi = \sum_i \phi_i$.

We now construct the function g. Given an instance $\mathcal{I} \in \Pi$ and a solution s' to $f(\mathcal{I})$, we compute a solution s to $\mathcal{I}$ in the following manner. We first use A to find a solution s_1. We also compute a second solution s_2 to $\mathcal{I}$ as follows. Let j be the largest index such that the solution s' projects down to a solution s'_j to the instance ϕ_j such that $V'(\phi_j, s'_j) > N/G(N)$. This in turn implies we can find a solution s_2 to witness $V(\mathcal{I}, s_2) \geq j$. Our solution s is the one among s_1 and s_2 that yields the larger objective function value.

We now show that the reduction is an E-reduction with $\beta = 1 + c(n)/(G(N) - 1)$.

Let $\alpha = OPT(I)/m$. Observe that

$$OPT(I') = Nm\left(\alpha + \frac{c(n)}{G(N)} - \frac{\alpha}{G(N)}\right).$$

Consider the following two cases:

Case 1 [$j \leq m$]: In this case, $V(\mathcal{I}, s) = m$. Since s is a solution to $\mathcal{I}$ of error at most $(\alpha - 1)$, it suffices to argue that the error of s' as a solution to ϕ is at least $(\alpha - 1)/\beta$. We start with the following upper bound on $V(\phi, s')$.

$$V(\phi, s') \leq Nm\left(1 + \frac{c(n)}{G(N)} - \frac{1}{G(N)}\right).$$

Thus the approximation factor achieved by s' is given by

$$\begin{aligned}
\mathcal{E}(\phi, s') &\geq \left(\frac{Nm\left(\alpha + \frac{c(n)}{G(N)} - \frac{\alpha}{G(N)}\right)}{Nm\left(1 + \frac{c(n)}{G(N)} - \frac{1}{G(N)}\right)}\right) - 1 \\
&= (\alpha - 1)\left(\frac{G(N) - 1}{G(N) + c(n) - 1}\right) \\
&= \frac{\alpha - 1}{\beta}.
\end{aligned}$$

So in this case s_1 (and hence s) is a solution to $\mathcal{I}$ with an error of at most $\beta\epsilon$, if s' is a solution to ϕ with an error of ϵ.

Case 2 [$j \geq m$]: Let $j = \gamma m$. Note that $\gamma > 1$ and that the error of s as a solution to $\mathcal{I}$ is $(\alpha - \gamma)/\gamma$. We bound the value of the solution s' to ϕ as

$$V(\phi, s') \leq Nm\left(\gamma + \frac{c(n)}{G(N)} - \frac{\gamma}{G(N)}\right),$$

and its error as

$$\begin{aligned}
\mathcal{E}(\phi, s') &= \left(\frac{\alpha + \frac{c(n)}{G(N)} - \frac{\alpha}{G(N)}}{\gamma + \frac{c(n)}{G(N)} - \frac{\gamma}{G(N)}}\right) - 1 \\
&= \left(\frac{\alpha - \gamma}{\gamma}\right)\left(\frac{1}{1 + \frac{c(n)}{\gamma(G(N)-1)}}\right) \\
&\geq \left(\frac{\alpha - \gamma}{\gamma}\right)\frac{1}{\beta}.
\end{aligned}$$

(The final inequality follows from the fact that $1 + \frac{c(n)}{\gamma(G(N)-1)} \leq 1 + \frac{c(n)}{(G(N)-1)} = \beta$.)

Thus in this case also we find that s (by virtue of s_2) is a solution to $\mathcal{I}$ of error at most $\beta\epsilon$ if s' is a solution to ϕ of of error ϵ.

We now consider the more general cases where Π and Ω are not both maximization problems. For the case where both are minimization problems, the above transformation works with one minor change. When creating ϕ_i, the NP language consists of instances $(\mathcal{I}, i)$ such that there exists s with $V(\mathcal{I}, s) \leq i$.

For the case where Π is a minimization problem and Ω is a maximization problem, we first E-reduce Π to a maximization problem Π' and then proceed as before. The reduction proceeds as follows. Since Π is a polynomially bounded optimization problem, we can compute an upper bound on the value of any solution s to an instance $\mathcal{I}$. Let m be such a bound for an instance $\mathcal{I}$. The objective function of Π' on the instance $\mathcal{I}$ is defined as $V'(\mathcal{I}, s) = \lfloor 2m^2/V(\mathcal{I}, s) \rfloor$. To begin with, it is easy to verify that $\Pi \in F$-APX implies $\Pi' \in F$-APX.

Let s be a solution to instance $\mathcal{I}$ of Π of error β. We will show that s as a solution to instance $\mathcal{I}$ of Π' has an error of at least $\beta/2$. Assume, without loss of generality, that $\beta \neq 0$. Then

$$V(\mathcal{I}, s) - OPT(\mathcal{I}) = \beta\, OPT(\mathcal{I}) \geq 1.$$

Multiplying by $2m^2/(OPT(\mathcal{I})V(\mathcal{I}, s))$, we get

$$\frac{2m^2}{OPT(\mathcal{I})} - \frac{2m^2}{V(\mathcal{I}, s)} = \beta \frac{2m^2}{V(\mathcal{I}, s)} \geq 2.$$

This implies that

$$\begin{aligned} \frac{2m^2}{OPT(\mathcal{I})} - \frac{2m^2}{V(\mathcal{I}, s)} &\geq 1 + \frac{1}{2} \times \frac{2m^2}{OPT(\mathcal{I})} - \frac{2m^2}{V(\mathcal{I}, s)} \\ &= 1 + \frac{\beta}{2} \times \frac{2m^2}{V(\mathcal{I}, s)}. \end{aligned}$$

Upon rearranging,

$$V'(\mathcal{I}, s) \leq \frac{1}{(1+\beta/2)} \left(\frac{2m^2}{OPT(\mathcal{I})} - 1 \right) \leq \frac{1}{(1+\beta/2)} \left\lfloor \frac{2m^2}{OPT(\mathcal{I})} \right\rfloor.$$

Thus the reduction from Π to Π' is an E-reduction.

Finally, the last remaining case, i.e., Π being a maximization problem and Ω being a minimization problem, is dealt with similarly: we transform Π into a minimization problem Π'. ∎

Remark 6 *This theorem appears to merge two different notions of the relative ease of approximation of optimization problems. One such notion would consider a problem Π_1 easier than Π_2 if there exists an approximation preserving reduction from Π_1 to Π_2. A different notion would regard Π_1 to be easier than Π_2 if one seems to have a better factor of approximation than the other. The above statement essentially states that these two comparisons are indeed the same. For instance, the MAX CLIQUE problem and the CHROMATIC NUMBER problem, which are both in* poly-APX*, are inter-reducible to each other. The above observation motivates the search for other interesting function classes f, for which the class f-*APX *may contain interesting optimization problems.*

4.2 Applications

The following is a consequence ofTheorem 5.

Theorem 6

a) $\overline{\text{RMAX(2)}}$ = poly-APX.

b) *If SET COVER is canonically hard to approximate to within a factor of $\Omega(\log n)$, then* log-APX = $\overline{\text{MIN F}^{+}\Pi_2(1)}$.

We briefly sketch the proof of this theorem. The hardness reduction for MAX SAT and CLIQUE are canonical [2, 11]. The classes APX-PB, poly-APX, log-APX are expressible as classes F-APX for downward closed function families. The problems MAX SAT, MAX CLIQUE and SET COVER are additive. Thus, we can now apply Theorem 5.

Remark 7 *We would like to point out that almost all known instances of hardness results seem to be shown for problems which are additive. In particular, this is true for all MAX SNP problems, MAX CLIQUE, CHROMATIC NUMBER, and SET COVER. Two cases where a hardness result does not seem to directly apply to an additive problem is that of LONGEST PATH [17] and BIN PACKING. In the former case, the closely related LONGEST S-T PATH problem is easily seen to be additive and the hardness result essentially stems from this problem. As for the case of BIN PACKING, which does not admit a PTAS, the hardness result is not of a multiplicative nature and in fact this problem can be approximated to within arbitrarily small factors, provided a small additive error term is allowed. This yields a reason why this problem will not be additive. Lastly, the most interesting optimization problems which do not seem to be additive are problems related to GRAPH BISECTION or PARTITION, and these also happen to be notable instances where no hardness of approximation results have been achieved!*

5 Local Search and MAX SNP

In this section we present a formal definition of the paradigm of non-oblivious local search. The idea of non-oblivious local search has been implicitly present in some well-known techniques such as the interior-point methods. We will formalize this idea in context of MAX SNP and illustrate its application to MAX SNP problems. Given a MAX SNP problem Π, recall that the goal is to find a structure S which maximizes the objective function: $V(\mathcal{I}, S) = \sum_{\vec{x}} \Phi(\mathcal{I}, S, \vec{x})$. In the subsequent discussion, we view S as a k-dimensional boolean vector.

5.1 Classical Local Search

We start by reviewing the standard mechanism for constructing a local search algorithm. A δ-local algorithm $\mathcal{A}$ for Π is based on a *distance function* $\mathcal{D}(S_1, S_2)$ which is the Hamming distance between two k-dimensional vectors. The *δ-neighborhood* of a structure S is given by $N(S, \delta) = \{S' \subseteq U^n \mid D(S, S') \leq \delta\}$, where U is the universe. A structure S is called *δ-optimal* if $\forall S' \in N(S, \delta)$, we have $V(\mathcal{I}, S) \geq V(\mathcal{I}, S')$. The algorithm computes a δ-optimum by performing a series of greedy improvements to an initial structure S_0, where each iteration moves from the current structure S_i to some $S_{i+1} \in N(S_i, \delta)$ of better value (if any). For constant δ, a δ-local search algorithm for a polynomially-bounded NPO problem runs in polynomial time because:

- each local change is polynomially computable, and
- the number of iterations is polynomially bounded since the value of the objective function improves monotonically by an integral amount with each iteration, and the optimum is polynomially-bounded.

5.2 Non-Oblivious Local Search

A non-oblivious local search algorithm is based on a 3-tuple $\langle S_0, \mathcal{F}, \mathcal{D}\rangle$, where S_0 is the initial solution structure which must be independent of the input, $\mathcal{F}(\mathcal{I}, S)$ is a real-valued function referred to as the *weight function*, and $\mathcal{D}$ is a real-valued *distance function* which returns the distance between two structures in some appropriately chosen metric. The weight function $\mathcal{F}$ should be such that the number of distinct values taken by $\mathcal{F}(\mathcal{I}, S)$ is polynomially bounded in the input size. Moreover, the distance function $\mathcal{D}$ should be such that given a structure S and a fixed δ, $N(S, \delta)$ can be computed in time polynomial in $|S|$. Then, as in classical local search, for constant δ, a non-oblivious δ-local algorithm terminates in time polynomial in the input size.

The classical local search paradigm, which we call *oblivious local search*, makes the natural choice for the function $\mathcal{F}(\mathcal{I}, S)$, and the distance function $\mathcal{D}$, i.e., it chooses them to be $V(\mathcal{I}, S)$ and the Hamming distance. However, as we show later, this choice does not always yield a good approximation ratio. We now formalize our notion of this more general type of local search.

Definition 15 (Non-Oblivious Local Search Algorithm) *A* non-oblivious local search *algorithm is a δ-local search algorithm whose weight function is defined to be*

$$\mathcal{F}(\mathcal{I}, S) = \sum_{\vec{x}} \sum_{i=1}^{r} p_i \Phi_i(\mathcal{I}, S, \vec{x}) ,$$

where r is a constant, Φ_i's are quantifier-free first-order formulas, and the profits p_i are real constants. The distance function $\mathcal{D}$ is an arbitrary polynomial-time computable function.

A non-oblivious local search can be implemented in polynomial time in much the same way as the oblivious local search. Note that the we are only considering polynomially-bounded weight functions and the profits p_i are fixed independent of the input size. In general, the non-oblivious weight functions do not direct the search in the direction of the actual objective function. In fact, as we will see, this is exactly the reason why they are more powerful and allow for better approximations.

We now define two classes of NPO problems.

Definition 16 (Oblivious GLO) *The class of problems* OBLIVIOUS GLO *consists of all* NPO *problems which can be approximated within constant factors by an oblivious δ-local search algorithm for some constant δ.*

Definition 17 (Non-Oblivious GLO) *The class of problems* NON-OBLIVIOUS GLO *consists of all* NPO *problems which can be approximated within constant factors by a non-oblivious δ-local search algorithm for some constant δ.*

Remark 8 *It would be perfectly reasonable to allow weight functions that are non-linear, but we stay with the above definition for the purposes of this paper. Allowing only a constant number of predicates in the weight functions enables us to prevent the encoding of arbitrarily complicated approximation algorithms. The structure S is a k-dimensional vector, and so a natural metric for the distance function $\mathcal{D}$ is the Hamming distance. In fact, classical local search is indeed based on the Hamming metric and this is useful in proving negative results for the paradigm. In contrast, the definition of non-oblivious local search allows for other distance functions, but we will use only the Hamming metric in proving positive results in the remainder of this paper. However, we have found that it is sometimes useful to modify this, for example by modifying the Hamming distance so that the complement of a vector is considered to be at distance 1 from it. Finally, it is sometimes convenient to assume that the local search makes the best possible move in the bounded neighborhood, rather than an arbitrary move which improves the weight function. We believe that this does not increase the power of non-oblivious local search.*

6 The Power of Non-Oblivious Local Search

We will show that there exists a choice of a non-oblivious weight function for MAX k-SAT such that any assignment which is 1-optimal with respect to this weight function, yields a performance ratio of $2^k/(2^k-1)$ with respect to the optimal. But first, we obtain tight bounds on the performance of oblivious local search for MAX 2-SAT, establishing that its performance is significantly weaker than the best-known result even when allowed to search exponentially large neighborhoods. We use the following notation: for any fixed truth assignment $\vec{Z}$, S_i is the set of clauses in which exactly i literals are true; and, for a set of clauses S, $W(S)$ denotes the total weight of the clauses in S.

6.1 Oblivious Local Search for MAX 2-SAT

We show a strong separation in the performance of oblivious and non-oblivious local search for MAX 2-SAT. Suppose we use a δ-local strategy with the weight function $\mathcal{F}$ being the total weight of the clauses satisfied by the assignment, i.e., $\mathcal{F} = W(S_1) + W(S_2)$. The following theorem shows that for any $\delta = o(n)$, an oblivious δ-local strategy cannot deliver a performance ratio better than $3/2$. This is rather surprising given that we are willing to allow near-exponential time for the oblivious algorithm.

Theorem 7 *The asymptotic performance ratio for an oblivious δ-local search algorithm for* MAX 2-SAT *is $3/2$ for any positive $\delta = o(n)$. This ratio is still bounded by $5/4$ when δ may take any value less than $n/2$.*

Proof: We first show the existence of an input instance for MAX 2-SAT which may elicit a relatively poor performance ratio for any δ-local algorithm provided $\delta = o(n)$. In our construction of such an input instance, we assume that $n \geq 2\delta + 1$. The input instance comprises of a disjoint union of four sets of clauses, say $\Gamma_1, \Gamma_2, \Gamma_3$ and Γ_4, defined as below:

$$\begin{aligned}
\Gamma_1 &= \bigcup_{1 \leq i < j \leq n} (z_i + \overline{z}_j), \\
\Gamma_2 &= \bigcup_{1 \leq i < j \leq n} (\overline{z}_i + z_j), \\
\Gamma_3 &= \bigcup_{0 \leq i \leq \delta} \zeta_{2i+1}, \\
\Gamma_4 &= \bigcup_{2\delta+2 \leq i \leq n} \zeta_i, \\
\zeta_i &= \bigcup_{i < j \leq n} (\overline{z}_i + \overline{z}_j).
\end{aligned}$$

Clearly, $|\Gamma_1| = |\Gamma_2| = \binom{n}{2}$, and $|\Gamma_3| + |\Gamma_4| = \binom{n}{2} - n\delta + \delta(\delta+1)$. Without loss of generality, assume that the current input assignment is $\vec{Z} = (1, 1, \ldots, 1)$. This satisfies all clauses in Γ_1 and Γ_2. But none of the clauses in Γ_3 and Γ_4 are satisfied. If we flip the assignment of values to any $k \leq \delta$ variables, it would unsatisfy precisely $k(n-k)$ clauses in $\Gamma_1 + \Gamma_2$. This is the number of clauses in $\Gamma_1 + \Gamma_2$ where a flipped variable occurs with an unflipped variable.

On the other hand, flipping the assigned values of any $k \leq \delta$ variables can satisfy at most $k(n-k)$ clauses in $\Gamma_3 + \Gamma_4$ as we next show.

Let $\Pi(n, \delta)$ denote the set of clauses on n variables given by $\bigcup_{0 \leq i \leq \delta} \zeta_{2i+1} + \bigcup_{2\delta+2 \leq i \leq n} \zeta_i$ where $2\delta + 1 \leq n$. We claim the following.

Lemma 1 *Any assignment of values to the n variables such that at most $k \leq \delta$ variables have been assigned value false, can satisfy at most $k(n-k)$ clauses in $\Pi(n, \delta)$.*

Proof: We prove by simultaneous induction on n and δ that the statement is true for any instance $\Pi(n, \delta)$ where n and δ are non-negative integers such that $2\delta + 1 \leq n$. The base case includes $n = 1$ and $n = 2$ and is trivially verified to be true for the only allowable value of δ, namely $\delta = 0$. We now assume that the statement is true for any instance $\Pi(n', \delta')$ such that $n' < n$ and $2\delta' + 1 \leq n'$. Consider now the instance

$\Pi(n, \delta)$. The statement is trivially true for $\delta = 0$. Now consider any $\delta > 0$ such that $2\delta + 1 \leq n$. Let $\{z_{j_1}, z_{j_2}, \ldots z_{j_k}\}$ be any choice of $k \leq \delta$ variables such that $j_p < j_q$ for $p < q$. Again the assertion is trivially true if $k = 0$ or $k = 1$. We assume that $k \geq 2$ from now on. If we delete all clauses containing the variables z_1 and z_2 from $\Pi(n, \delta)$, we get the instance $\Pi(n-2, \delta-1)$. We now consider three cases.

Case 1 [$j_1 \geq 3$]: In this case, we are reduced to the problem of finding an upper bound on the maximum number of clauses satisfied by setting any k variables to false in $\Pi(n-2, \delta-1)$. If $k \leq \delta - 1$, we may use the inductive hypothesis to conclude that no more than $(n-2-k)(k)$ clauses will be satisfied. Thus the assertion holds in this case. However, we may not directly use the inductive hypothesis if $k = \delta$. But in this case we observe that since by the inductive hypothesis, setting any $k-1$ variables in $\Pi(n-2, \delta-1)$ to false, satisfies at most $(n-2-(k-1))(k-1)$ clauses, assigning the value false to any set of k variables, can satisfy at most

$$(n-2-(k-1))(k-1) + \frac{1}{k-1}(n-2-(k-1))(k-1) = (n-k)k - k^2$$

clauses. Hence the assertion holds in this case also.

Case 2 [$j_1 = 2$]: In this case, z_{j_1} satisfies one clause and the remaining $k-1$ variables satisfy at most $(n-2-(k-1))(k-1)$ clauses by the inductive hypothesis on $\Pi(n-2, \delta-1)$. Adding up the two terms, we see that the assertion holds.

Case 3 [$j_1 = 1$]: We analyze this case based on whether $j_2 = 2$ or $j_2 \geq 3$. If $j_2 = 2$, then z_1 and z_2, together satisfy precisely $n-1$ clauses and the remaining $k-2$ variables, satisfy at most $(n-2-(k-2))(k-2)$ clauses using the inductive hypothesis. Thus the assertion still holds. Otherwise, z_1 satisfies precisely $n-1$ clauses and the remaining $k-1$ variables satisfy no more than $(n-1-(k-1))(k-1)$ clauses using the inductive hypothesis. Summing up the two terms, we get $(n-k)k$ as the upper bound on the total number of clauses satisfied. Thus the assertion holds in this case also.

To see that this bound is tight, simply consider the situation when the k variables set to false are $z_1, z_3, \ldots, z_{2k-1}$, for any $k \leq \delta$. The total number of clauses satisfied is given by $\sum_{i=1}^{k} |\zeta_{2i-1}| = (n-k)k$. ■

Assuming that each clause has the same weight, Lemma 1 allows us to conclude that a δ-local algorithm cannot increase the total weight of satisfied clauses with this starting assignment. An optimal assignment on the other hand can satisfy all the clauses by choosing the vector $\vec{Z} = (0, 0, \ldots, 0)$. Thus the performance ratio of a δ-local algorithm, say R_δ, is bounded as

$$\begin{aligned} R_\delta &= \frac{|\Gamma_1| + |\Gamma_2| + |\Gamma_3| + |\Gamma_4|}{|\Gamma_1| + |\Gamma_2|} \\ &\leq \frac{3\binom{n}{2} + \delta(\delta+1) - \delta n}{2\binom{n}{2}}. \end{aligned}$$

For any $\delta = o(n)$, this ratio asymptotically converges to $3/2$. We next show that this bound is tight since a 1-local algorithm achieves it. However, before we do so, we make another intriguing observation, namely, for any $\delta < n/2$, the ratio R_δ is bounded by $5/4$.

Now to see that a 1-local algorithm ensures a performance ratio of $3/2$, consider any 1-optimal assignment $\vec{Z}$ and let α_i denote the set of clauses containing the variable z_i such that no literal in any clause of α_i is satisfied by $\vec{Z}$. Similarly, let β_i denote the set of clauses containing the variable z_i such that precisely one literal is satisfied in any clause in β_i and furthermore, it is precisely the literal containing the variable z_i. If we complement the value assigned to the variable z_i, it is exactly the set of clauses in α_i which becomes satisfied and the set of clauses in β_i which is no longer satisfied. Since $\vec{Z}$ is 1-optimal, it must be the case that $W(\alpha_i) \leq W(\beta_i)$. If we sum up this inequality over all the variables, then we get the inequality $\sum_{i=1}^n W(\alpha_i) \leq \sum_{i=1}^n W(\beta_i)$. We observe that $\sum_{i=1}^n W(\alpha_i) = 2W(S_0)$ and $\sum_{i=1}^n W(\beta_i) = W(S_1)$ because each clause in S_0 gets counted twice while each clause in S_1 gets counted exactly once. Thus the fractional weight of the number of clauses not satisfied by a 1-local assignment is bounded as

$$\frac{W(S_0)}{W(S_0) + W(S_1) + W(S_2)} \leq \frac{W(S_0)}{3W(S_0) + W(S_2)} \leq \frac{W(S_0)}{3W(S_0)} = \frac{1}{3}.$$

Hence the performance ratio achieved by a 1-local algorithm is bounded from above by $3/2$. Combining this with the upper bound derived earlier, we conclude that $R_1 = 3/2$. This concludes the proof of the theorem. ■

6.2 Non-Oblivious Local Search for MAX 2-SAT

We now illustrate the power of non-oblivious local search by showing that it achieves a performance ratio of $4/3$ for MAX 2-SAT, using 1-local search with a simple non-oblivious weight function.

Theorem 8 *Non-oblivious 1-local search achieves a performance ratio of* $4/3$ *for* MAX 2-SAT.

Proof: We use the non-oblivious weight function

$$\mathcal{F}(\mathcal{I}, \vec{Z}) = \frac{3}{2}W(S_1) + 2W(S_2).$$

Consider any assignment $\vec{Z}$ which is 1-optimal with respect to this weight function. Without loss of generality, we assume that the variables have been renamed such that each unnegated literal gets assigned the value true. Let $P_{i,j}$ and $N_{i,j}$ respectively denote the total weight of clauses in S_i containing the literals z_j and $\overline{z}_j$, respectively. Since $\vec{Z}$ is a 1-optimal assignment, each variable z_j must satisfy the following equation.

$$-\frac{1}{2}P_{2,j} - \frac{3}{2}P_{1,j} + \frac{1}{2}N_{1,j} + \frac{3}{2}N_{0,j} \leq 0.$$

Summing this inequality over all the variables, and using

$$\begin{aligned} \sum_{j=1}^n P_{1,j} = \sum_{j=1}^n N_{1,j} &= W(S_1), \\ \sum_{j=1}^n P_{2,j} &= 2W(S_2), \\ \sum_{j=1}^n N_{0,j} &= 2W(S_0), \end{aligned}$$

we obtain the following inequality:

$$W(S_2) + W(S_1) \geq 3W(S_0).$$

This immediately implies that the total weight of the unsatisfied clauses at this local optimum is no more than $1/4$ times the total weight of all the clauses. Thus, this algorithm ensures a performance ratio of $4/3$. ■

Remark 9 *The same result can be achieved by using the oblivious weight function, and instead modifying the distance function so that it corresponds to distances in a hypercube augmented by edges between nodes whose addresses are complement of each other.*

6.3 Generalization to MAX k-SAT

We can also design a non-oblivious weight function for MAX k-SAT such that a 1-local strategy ensures a performance ratio of $2^k/(2^k-1)$. The weight function $\mathcal{F}$ will be of the form $\mathcal{F} = \sum_{i=0}^{k} c_i W(S_i)$ where the coefficients c_i's will be specified later.

Theorem 9 *Non-oblivious 1-local search achieves a performance ratio of* $2^k/(2^k-1)$ *for* MAX k-SAT.

Proof: Again, without loss of generality, we will assume that the variables have been renamed so that each unnegated literal is assigned true under the current truth assignment. Thus the set S_i is the set of clauses with i unnegated literals.

Let $\Delta_i = c_i - c_{i-1}$ and let $\frac{\partial \mathcal{F}}{\partial z_j}$ denote the change in the current weight when we flip the value of z_j, that is, set it to 0. It is easy to verify the following equation:

$$\frac{\partial \mathcal{F}}{\partial z_j} = -\Delta_k P_{k,j} + \sum_{i=k}^{2}(\Delta_i N_{i-1,j} - \Delta_{i-1} P_{i-1,j}) + \Delta_1 N_{0,j} \tag{1}$$

Thus when the algorithm terminates, we know that $\frac{\partial \mathcal{F}}{\partial z_j} \leq 0$, for $1 \leq j \leq n$. Summing over all values of j, and using the fact $\sum_{j=1}^{n} P_{i,j} = iW(S_i)$ and $\sum_{j=1}^{n} N_{i,j} = (k-i)W(S_i)$ we get the following inequality.

$$k\Delta_k W(S_k) + \sum_{i=k-1}^{2} (i\Delta_i - (k-i)\Delta_{i+1})W(S_i) \geq k\Delta_1 W(S_0). \tag{2}$$

We now determine the values of Δ_i's such that the coefficient of each term on the left hand side is unity. It can be verified that

$$\Delta_i = \frac{1}{(k-i+1)\binom{k}{i-1}} \sum_{j=0}^{k-i} \binom{k}{j}$$

achieves this goal. Thus the coefficient of $W(S_0)$ on the right hand side of equation (2) is $2^k - 1$. Clearly, the weight of the clauses not satisfied is bounded by $1/2^k$ times the total weight of all the clauses. It is worthwhile to note that this is regardless of the value chosen for the coefficient c_0. ∎

7 Local Search for CSP and MAX SNP

We now introduce a class of constraint satisfaction problems such that the problems in MAX SNP are exactly equivalent to the problems in this class. Furthermore, every problem in this class can be approximated to within a constant factor by a non-oblivious local search algorithm.

7.1 Constraint Satisfaction Problems

The connection between the syntactic description of optimization problems and their approximability through non-oblivious local search is made via a problem called MAX k-CSP which captures all the problems in MAX SNP as a special case.

Definition 18 (k-ary Constraint) *Let $Z = \{z_1, \ldots, z_n\}$ be a set of boolean variables. A k-ary constraint on Z is $C = (V; P)$, where V is a size k subset of Z, and $P : \{T, F\}^k \to \{T, F\}$ is a k-ary boolean predicate.*

Definition 19 (MAX k-CSP) *Given a collection $C_1, \ldots, C_m$ of weighted k-ary constraints over the variables $Z = \{z_1, \ldots, z_n\}$, the* MAX k-CSP *problem is to find a truth assignment satisfying a maximum weight sub-collection of the constraints.*

The following theorem shows that MAX k-CSP problem is a "universal" MAX SNP problem, in that it contains as special cases all problems in MAX SNP.

Theorem 10

a) *For fixed k,* MAX k-CSP $\in$ MAX SNP.

b) *Let* $\Pi \in$ MAX SNP. *Then, for some constant k, Π is a* MAX k-CSP *problem. Moreover, the* k-CSP *instance corresponding to any instance of this problem can be computed in polynomial time.*

Proof: The proof of part (b) is implicit in Theorem 1 in [23], and so we concentrate on proving part (a). Our goal is to obtain a representation of the k-CSP problem in the MAX SNP syntax:

$$\max_S |\{x \mid \Phi(\mathcal{I}, S, x)\}|.$$

The input structure is $\mathcal{I} = (Z \cup \{T, F\} \cup \text{MAX}; \{\text{ARG}, \text{EVAL}\})$, where $Z = \{z_1, \ldots, z_n\}$, MAX contains the integers $[1, \max\{k, n, m\}]$, the predicate ARG encodes the sets V_i, and the predicate EVAL encodes the predicates P_i, as described below.

- $\text{ARG}(r, s, z_t)$ is 3-ary predicate which is true if and only if the rth argument of C_s is the variable z_t, for $1 \leq r \leq k$, $1 \leq s \leq m$, and $1 \leq t \leq n$.
- $\text{EVAL}(s, v_1, \ldots, v_k)$ is a $(k+1)$-ary predicate which is true if and only if $P_s(v_1, \ldots, v_k)$ evaluates to true, for $1 \leq s \leq m$ and all $v_i \in \{T, F\}$.

The structure S is defined as $(Z; \{\text{TRUE}\})$, where TRUE is a unary predicate which denotes an assignment of truth values to the variables in Z. The vector x has $k+1$ components which will be called x_1, …, x_k and s, for convenience. The intention is that the x_i's refer to the arguments of the sth constraint.

All that remains is to specify the quantifier-free formula Φ. The basic idea is that $\Phi(\mathcal{I}, S, x)$ should evaluate to true if and only if the following two conditions are satisfied:

- the arguments of the constraint C_s are given by the variables x_1, …, x_k, in that order; and,
- the values given to these variables under the truth assignment specified by S are such that the constraint is satisfied.

The formula Φ is given by the following expression, with the two sub-formulas ensuring these two conditions.

$$\left(\bigwedge_{r=1}^{k} \text{ARG}(r, s, x_r)\right) \wedge \left(\bigvee_{v_1, \ldots, v_k \in \{T,F\}} \left(\text{EVAL}(s, v_1, \ldots, v_k) \wedge \left(\bigwedge_{r=1}^{k} v_r \Leftrightarrow \text{TRUE}(x_r)\right)\right)\right)$$

It is easy to see that the first sub-formula has the desired effect of checking that the x_r's correspond to the arguments of C_s. The second sub-formula considers all possible truth assignment to these k variables, and checks that the particular set of values assigned by the structure S will make P_s evaluate to true.

For a fixed structure S, there is exactly one choice of x per constraint that could make Φ evaluate to true, and this happens if and only if that constraint is satisfied. Thus, the value of the solution given by any particular truth assignment structure S is exactly the number of constraints that are satisfied. This shows that the MAX SNP problem always has the same value as intended in the k-CSP problem.

Finally, there are still a few things which need to be checked to ensure that this is a valid MAX SNP formulation. Notice that all the predicates are of bounded arity and the structures consist of a bounded number of such predicates, i.e., independent of the input size which is given by MAX. Further, although the length of the formula is exponential in k, it is independent of the input. ■

7.2 Non-Oblivious Local Search for MAX k-CSP

A suitable generalization of the non-oblivious local search algorithm for MAX k-SAT yields the following result.

Theorem 11 *A non-oblivious 1-local search algorithm has performance ratio* 2^k *for* MAX k-CSP.

Proof: We use an approach similar to the one used in the previous section to design a non-oblivious weight function $\mathcal{F}$ for the weighted version of the MAX k-CSP problem such that a 1-local algorithm yields 2^k performance ratio to this problem.

We consider only the constraints with at least one satisfying assignment. Each such constraint can be replaced by a monomial which is the conjunction of some k literals such that when the monomial evaluates to true the corresponding literal assignment represents a satisfying assignment for the constraint. Furthermore, each such monomial has precisely one satisfying assignment. We assign to each monomial the weight of the constraint it represents. Thus any assignment of variables which satisfies monomials of total weight W_0, also satisfies constraints in the original problem of total weight W_0.

Let S_i denote the monomials with i true literals, and assume that the weight function $\mathcal{F}$ is of the form $\sum_{i=1}^{k} c_i W(S_i)$. Thus, assuming that the variables have been renamed so that the current assignment gives value true to each variable, we know that for any variable z_j, $\frac{\partial \mathcal{F}}{\partial z_j}$ is given by equation (1). As before, using the fact that for any 1-optimal assignment, $\frac{\partial \mathcal{F}}{\partial z_j} \leq 0$ for $1 \leq j \leq n$, and summing over all values of j, we can write the following inequality.

$$k\Delta_1 W(S_0) + \sum_{i=2}^{k-1}((k-i)\Delta_{i+1} - i\Delta_i)W(S_i) \leq k\Delta_k W(S_k)\ . \tag{3}$$

We now determine the values of Δ_i's such that the coefficient of each term on the left hand side is unity. It can be verified that

$$\Delta_i = \frac{1}{i\binom{k}{i}} \sum_{j=0}^{i-1} \binom{k}{j}$$

achieves this goal. Thus the coefficient of $W(S_k)$ on the right hand side of equation (1) is $2^k - 1$. Clearly, the total weight of clauses satisfied is at least $1/2^k$ times the total weight of all the clauses with at least one satisfiable assignment. ■

We conclude the following theorem.

Theorem 12 *Every optimization problem* $\Pi \in$ MAX SNP *can be approximated to within some constant factor by a (uniform) non-oblivious 1-local search algorithm, i.e.,*

MAX SNP $\subseteq$ NON-OBLIVIOUS GLO.

For a problem expressible as k-CSP*, the performance ratio is at most* 2^k.

8 Non-Oblivious versus Oblivious GLO

In this section, we show that there exist problems for which no constant factor approximation can be obtained by any δ-local search algorithm with oblivious weight function, even when we allow δ to grow with the input size. However, a simple 1-local search algorithm using an appropriate non-oblivious weight function can ensure a constant performance ratio.

8.1 MAX 2-CSP

The first problem is an instance of MAX 2-CSP where we are given a collection of monomials such that each monomial is an "and" of precisely two literals. The objective is to find an assignment to maximize the number of monomials satisfied.

We show an instance of this problem such that for every $\delta = o(n)$, there exists an instance one of whose local optima has value that is a vanishingly small fraction of the global optimum.

The input instance consists of a disjoint union of two sets of monomials, say Γ_1 and Γ_2, defined as below:

$$\begin{aligned}\Gamma_1 &= \bigcup_{1\le i<j\le n} (\overline{z}_i \wedge \overline{z}_j),\\ \Gamma_2 &= \bigcup_{1\le i\le \delta}\ \bigcup_{i<j\le n} (z_i \wedge z_j).\end{aligned}$$

Clearly, $|\Gamma_1| = \binom{n}{2}$, and $|\Gamma_2| = n\delta - \binom{\delta+1}{2}$. Consider the truth assignment $\vec{Z} = (1,1,\ldots,1)$. It satisfies all monomials in Γ_2 but none of the monomials in Γ_1. We claim that this assignment is δ-optimal with respect to the oblivious weight function. To see this, observe that complementing the value of any $p \le \delta$ variables will unsatisfy at least $\delta p/2$ monomials in Γ_2 for any $\delta = o(n)$. On the other hand, this will satisfy precisely $\binom{p}{2}$ monomials in Γ_1. For any $p \le \delta$, we have $(\delta p)/2 \ge \binom{p}{2}$, and so Z is a δ-local optimum.

The optimal assignment on the other hand, namely $\vec{Z}_{OPT} = (0,0,\ldots,0)$, satisfies all monomials in Γ_1. Thus, for $\delta < n/2$, the performance ratio achieved by any δ-local algorithm is no more than $\binom{n}{2}/(n\delta - \binom{\delta+1}{2})$ which asymptotically diverges to infinity for any $\delta = o(n)$. We have already seen in Section 7 that a 1-local non-oblivious algorithm ensures a performance ratio of 4 for this problem. Since this problem is in MAX SNP, we obtain the following theorem.

Theorem 13 *There exist problems in* MAX SNP *such that for $\delta = o(n)$, no δ-local oblivious algorithm can approximate them to within a constant performance ratio, i.e.,*

$$\text{MAX SNP} \not\subseteq \text{OBLIVIOUS GLO}.$$

8.2 Vertex Cover

Ausiello and Protasi [4] have shown that VERTEX COVER does not belong to the class GLO and, hence, there does not exist any constant δ such that an oblivious δ-local search algorithm can compute a constant factor approximation. In fact, their example can be used to show that for any $\delta = o(n)$, the performance ratio ensured by δ-local search asymptotically diverges to infinity. However, we show that there exists a rather simple non-oblivious weight function which ensures a factor 2 approximation via a 1-local search. In fact, the algorithm simply enforces the behavior of the standard approximation algorithm which iteratively builds a vertex cover by simply including both end-points of any currently uncovered edge.

We assume that the input graph G is given as a structure $(V, \{E\})$ where V is the set of vertices and $E \subseteq V \times V$ encodes the edges of the graph. Our solution is represented by a 2-ary predicate M which is iteratively constructed so as to represent a maximal matching. Clearly, the end-points of any maximal matching constitute a valid vertex cover and such a vertex cover can be at most twice as large as any other vertex cover in the graph. Thus M is an encoding of the vertex cover computed by the algorithm.

The algorithm starts with M initialized to the empty relation and at each iteration, at most one new pair is included in it. The non-oblivious weight function used is as below:

$$\mathcal{F}(\mathcal{I}, M) = \sum_{(x,y,z)\in V^3} [\Phi_1(x,y,z) - 2\Phi_2(x,y,z) - \Phi_3(x,y,z)],$$

where

$$\begin{aligned}\Phi_1(x,y,z) &= (M(x,y) \wedge E(x,y) \wedge (x=z)),\\ \Phi_2(x,y,z) &= (M(x,y) \wedge M(x,z)),\\ \Phi_3(x,y,z) &= (M(x,y) \wedge \overline{E(x,y)}).\end{aligned}$$

Let M encode a valid matching in the graph G. We make the following observations.

- Any relation M' obtained from M by either deleting an edge from it, or including an edge which is incident on an edge of M, or including a non-existent edge, has the property that $\mathcal{F}(\mathcal{I}, M') \leq \mathcal{F}(\mathcal{I}, M)$. Thus in a 1-local search from M, we will never move to a relation M' which does not encode a valid matching of G.

- On the other hand, if a relation M' corresponds to the encoding of a matching in G which is larger than the matching encoded by M, then $\mathcal{F}(\mathcal{I}, M') > \mathcal{F}(\mathcal{I}, M)$. Thus if M does not encode a maximal matching in G, there always exist a relation in its 1-neighborhood of larger weight than itself.

These two observations, combined with the fact that we start with a valid initial matching (the empty matching), immediately allow us to conclude that any 1-optimal relation M always encodes a maximal matching in G. We have established the following.

Theorem 14 *A 1-local search algorithm using the above non-oblivious weight function achieves a performance ratio of* 2 *for the VERTEX COVER problem.*

Theorem 15 GLO *is a* strict *subset of* NON-OBLIVIOUS GLO.

As an aside, it can be seen that this algorithm has the same performance starting with an arbitrary initial solution. This is because for any relation M not encoding a matching of the input graph, deleting one of the violating members strictly increases $\mathcal{F}(\mathcal{I}, M)$.

9 The Traveling Salesman Problem

The TSP(1,2) problem is the traveling salesman problem restricted to complete graphs where all edge weights are either 1 or 2; clearly, this satisfies the triangle inequality. Papadimitriou and Yannakakis [24] showed that this problem is hard for MAX SNP. The natural weight function for TSP(1,2), that is, the weight of the tour, can be used to show that a 4-local algorithm yields a $3/2$ performance ratio. The algorithm starts with an arbitrary tour and in each iteration, it checks if there exist two disjoint edges (a,b) and (c,d) on the tour such that deleting them and replacing them with the edges (a,c) and (b,d) yields a tour of lesser cost.

Theorem 16 *A 4-local search algorithm using the oblivious weight function achieves a* $3/2$ *performance ratio for* TSP(1,2).

Proof: Let C be a 4-optimal solution and let π be a permutation such that the vertices in C occur in the order $v_{\pi_1}, v_{\pi_2}, \ldots, v_{\pi_n}$. Consider any optimal solution O. With each unit cost edge e in O, we associate a unit cost edge e' in C as follows. Let $e = (v_{\pi_i}, v_{\pi_j})$ where $i < j$. If $j = i+1$ then $e' = e$. Otherwise, consider the edges $e_1 = (v_{\pi_i}, v_{\pi_{i+1}})$ and $e_2 = (v_{\pi_j}, v_{\pi_{j+1}})$ on C. We claim either e_1 or e_2 must be of unit cost. Suppose not, then the tour C' which is obtained by simply deleting both e_1 and e_2 and inserting the edges e and $f = (v_{\pi_{i+1}}, v_{\pi_{j+1}})$ has cost at least one less than C. But C is 4-optimal and thus this is a contradiction.

Let U_O denotes the set of unit cost edges in O and let U_C be the set of unit cost edges in C which form the image of U_O under the above mapping. Since an edge $e' = (v_{\pi_i}, v_{\pi_{i+1}})$ in U_C can only be the image of unit cost edges incident on v_{π_i} in O and since O is a tour, there are at most two edges in U_O which map to e'. Thus $|U_C| \geq |U_O|/2$ and hence

$$\frac{cost(O)}{cost(C)} \geq \frac{|U_O| + 2(n - |U_O|)}{|U_O|/2 + 2(n - |U_O|/2)} \geq \frac{2}{3}.$$

■

In fact, the above bound can be shown to be tight.

10 Maximum Independent Sets in Bounded Degree Graphs

The input instance to the maximum independent set problem in bounded degree graphs, denoted MIS-B, is a graph G such that the degree of any vertex in G is bounded by a constant Δ. We present an algorithm with performance ratio $(\sqrt{8\Delta^2 + 4\Delta + 1} - 2\Delta + 1)/2$ for this problem when $\Delta \geq 10$.

Our algorithm uses two local search algorithms such that the larger of the two independent sets computed by these algorithms, gives us the above claimed performance ratio. We refer to these two algorithms as $\mathcal{A}_1$ and $\mathcal{A}_2$.

In our framework, the algorithm $\mathcal{A}_1$ can be characterized as a 3-local algorithm with the weight function simply being $|I| - 3|(I \times I) \cap E|$. Thus if we start with I initialized to empty set, it is easy to see that at each iteration, I will correspond to an independent set in G. A convenient way of looking at this algorithm is as follows. We define an $i \leftrightarrow j$ swap to be the process of deleting i vertices from S and including j vertices from the set $V - S$ to the set S. In each iteration, the algorithm $\mathcal{A}_1$ performs either a $0 \leftrightarrow j$ swap where $1 \leq j \leq 3$, or a $1 \leftrightarrow 2$ swap. A $0 \leftrightarrow j$ swap however, can be interpreted as j applications of $0 \leftrightarrow 1$ swaps. Thus the algorithm may be viewed as executing a $0 \leftrightarrow 1$ swap or a $1 \leftrightarrow 2$ swap at each iteration. The algorithm terminates when neither of these two operations is applicable.

Let I denote the 3-optimal independent set produced by the algorithm $\mathcal{A}_1$. Furthermore, let O be any optimal independent set and let $X = I \cap O$. We make the following useful observations.

- Since for no vertex in I, a $0 \leftrightarrow 1$ swap can be performed, it implies that each vertex in $V - I$ must have at least one incoming edge to I.
- Similarly, since no $1 \leftrightarrow 2$ swaps can be performed, it implies that at most $|I - X|$ vertices in $O - I$ can have precisely one edge coming into I. Thus $|O - X| - |I - X| = |O| - |I|$ vertices in $O - X$ must have at least two edges entering the set I.

A rather straightforward consequence of these two observations is the following lemma.

Lemma 2 *The algorithm $\mathcal{A}_1$ has performance ratio $(\Delta + 1)/2$ for* MIS-B.

Proof: The above two observations imply that the minimum number of edges entering I from the vertices in $O - X$ is $|I - X| + 2(|O| - |I|)$. On the other hand, the maximum number of edges coming out of the vertices in I to the vertices in $O - X$ is bounded by $|I - X|\Delta$. Thus we must have

$$|I - X|\Delta \geq |I - X| + 2(|O| - |I|) .$$

Rearranging, we get

$$\frac{|I|}{|O|} \geq \frac{2}{\Delta + 1} + \frac{|X|(\Delta - 1)}{|O|(\Delta + 1)},$$

which yields the desired result. ■

This nearly matches the approximation ratio of $\Delta/2$ due to Hochbaum [15]. It should be noted that the above result holds for a broader class of graphs, viz., k-claw free graphs. A graph is called *k-claw free* if there does not exist an independent set of size k or larger such that all the vertices in the independent set are adjacent to the same vertex. Lemma 2 applies to $(\Delta+1)$-claw free graphs.

Our next objective is to further improve this ratio by using the algorithm $\mathcal{A}_1$ in combination with the algorithm $\mathcal{A}_2$. The following lemma uses a slightly different counting argument to give an alternative bound on the approximation ratio of the algorithm $\mathcal{A}_1$ when there is a constraint on the size of the optimal solution.

Lemma 3 *For any real number $c < \Delta$, the algorithm $\mathcal{A}_1$ has performance ratio $(\Delta - c)/2$ for* MIS-B *when the optimal value itself is no more than $((\Delta - c)|V|)/(\Delta + c + 4)$.*

Proof: As noted earlier, each vertex in $V - I$ must have at least one edge coming into the set I and at least $|O| - |I|$ vertices in O must have at least two edges coming into I. Therefore, the following inequality must be satisfied:

$$|I|\Delta \geq |V| - |I| + |O| - |I| \, .$$

Thus $|I| \geq (|V| + |O|)/(\Delta + 2)$. Finally, observe that

$$\frac{|V| + |O|}{\Delta + 2} \geq \frac{2}{\Delta - c}|O|$$

whenever $|O| \leq (\Delta - c)|V|/(\Delta + c + 4)$. ■

The above lemma shows that the algorithm $\mathcal{A}_1$ yields a better approximation ratio when the size of the optimal independent set is relatively small.

The algorithm $\mathcal{A}_2$ is simply the classical greedy algorithm. This algorithm can be conveniently included in our framework if we use directed local search. If we let $N(I)$ denote the set of neighbors of the vertices in I, then the weight function is simply $|I|(\Delta+1) + |V - (I + N(I))| - |(I \times I) \cap E|(\Delta + 1)$. It is not difficult to see that starting with an empty independent set, a 1-local algorithm with directed search on above weight function simply simulates a greedy algorithm. The greedy algorithm exploits the situation when the optimal independent set is relatively large in size. It does so by using the fact that the existence of a large independent set in G ensures a large subset of vertices in G with relatively small average degree. The following two lemmas characterize the performance of the greedy algorithm.

Lemma 4 *Suppose there exists an independent set $X \subseteq V$ such that the average degree of vertices in X is bounded by α. Then for any $\alpha \geq 1$, the greedy algorithm produces an independent set of size at least $|X|/(1+\alpha)$.*

Proof: The greedy algorithm iteratively chooses a vertex of smallest degree in the remaining graph and then deletes this vertex and all its neighbors from the graph. We examine the behavior of the greedy by considering two types of iterations. First consider the iterations in which it picks a vertex outside X. Suppose in the ith such iteration, it picks a vertex in $V - X$ with exactly k_i neighbors in the set X in the remaining graph. Since each one of these k_i vertices must also have at least k_i edges incident on them, we loose at least k_i^2 edges incident on X. Suppose only p such iterations occur and let $\sum_{i=1}^{p} k_i = x$. We observe that $\sum_{i=1}^{p} k_i^2 \leq \alpha|X|$. Secondly, we consider the iterations when the greedy selects a vertex in X. Then we do not loose any other vertices in X because X is an independent set. Thus the total size of the independent set constructed by the greedy algorithm is at least $p + q$ where $q = |X| - x$.

By the Cauchy-Schwartz inequality, $\sum_{i=1}^{p} k_i^2 \geq x^2/p$. Therefore, we have $(1+\alpha)|X| \geq x^2/p + x$. Rearranging, we obtain that

$$p \geq \frac{x^2}{(1+\alpha)|X| - x} \geq \frac{x^2}{(1+\alpha)|X|} \geq \frac{|X|}{1+\alpha} + \frac{q^2}{(1+\alpha)|X|} - \frac{2q}{1+\alpha}.$$

Thus

$$p + q \geq \frac{|X|}{1+\alpha} + \frac{q^2}{(1+\alpha)|X|} - \frac{2q}{1+\alpha} + q.$$

But $2q/(1+\alpha) \leq q$ for $\alpha \geq 1$, and the result follows. ■

Lemma 5 *For $\Delta \geq 10$ and any non-negative real number $c \leq 3\Delta - \sqrt{8\Delta^2 + 4\Delta + 1} - 1$, the algorithm $\mathcal{A}_2$ has performance ratio $(\Delta - c)/2$ for* MIS-B *when the optimal value itself is at least $((\Delta - c)|V|)/(\Delta + c + 4)$.*

Proof: Observe that the average degree of vertices in O is bounded by $(|V - O|\Delta/|O|)$ and thus using the fact that $|O| \geq (\Delta - c)|V|/(\Delta + c + 4)$, we know that the algorithm $\mathcal{A}_2$ computes an independent set of size at least $|O|/(1+\alpha)$ where $\alpha = (4\Delta + 2\Delta c)/(\Delta - c)$, and $\alpha \geq 1$ for $c \geq 0$. Hence it is sufficient to determine the range of values c can take such that the following inequality is satisfied:

$$\frac{|O|}{1+\alpha} \geq \left(\frac{2}{\Delta - c}\right) |O|.$$

Substituting the bound on the value of α and rearranging the terms of the equation, yields the following quadratic equation :

$$c^2 - (6\Delta - 2)c + \Delta^2 - 10\Delta \geq 0 .$$

Since c must be strictly bounded by Δ, the above quadratic equation is satisfied for any choice of $c \leq 3\Delta - \sqrt{8\Delta^2 + 4\Delta + 1} - 1$ if $\Delta \geq 10$. ■

Combining the results of Lemmas 3 and 5 and choosing the largest allowable value for c, we get the following result.

Theorem 17 *An approximation algorithm which simply outputs the larger of the two independent sets computed by the algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, has performance ratio $(\sqrt{8\Delta^2 + 4\Delta + 1} - 2\Delta + 1)/2$ for* MIS-B.

The performance ratio claimed above is essentially $\Delta/2.414$. This improves upon the long-standing approximation ratio of $\Delta/2$ due to Hochbaum [15], when $\Delta \geq 10$. However, very recently, there has been a flurry of new results for this problem. Berman and Furer [6] have given an algorithm with performance ratio $(\Delta + 3)/5 + \epsilon$ when Δ is even, and $(\Delta + 3.25)/5 + \epsilon$ for odd Δ, where $\epsilon > 0$ is a fixed constant. Halldorsson and Radhakrishnan [14] have shown that algorithm $\mathcal{A}_1$ when run on k-clique free graphs, yields an independent set of size at least $2n/(\Delta + k)$. They combine this algorithm with a clique-removal based scheme to achieve a performance ratio of $\Delta/6(1 + o(1))$.

In conclusion, note that Khanna, Motwani and Vishwanathan [19] have recently shown that a semi-definite programming technique can be used to obtain a $(\Delta \log\log\Delta)/(\log\Delta)$-approximation algorithm for this problem.

Acknowledgements

Many thanks to Phokion Kolaitis for his helpful comments and suggestions and to Giorgio Ausiello and Pierluigi Crescenzi for guiding us through the intricacies of approximation preserving reductions and the available literature on it. Thanks also to the anonymous referees for their detailed comments and corrections in a previous draft.

References

[1] P. Alimonti. New Local Search Approximation Techniques for Maximum Generalized Satisfiability Problems. In *Proceedings of the 2nd Italian Conference on Algorithms and Complexity* (1994), pp. 40–53.

[2] S. Arora, C. Lund, R. Motwani, M. Sudan and M. Szegedy. Proof Verification and Hardness of Approximation Problems. In *Proceedings of the 33rd Annual IEEE Symposium on Foundations of Computer Science* (1992), pp. 14–23.

[3] G. Ausiello, P. Crescenzi and M. Protasi. Approximate Solution of NP Optimization Problems. *Theoretical Computer Science*, 150 (1995), pp. 1–55.

[4] G. Ausiello and M. Protasi. Local Search, Reducibility, and Approximability of NP Optimization Problems. *Inform. Process. Lett.*, 54 (1995), pp. 73-79.

[5] M. Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and Applications to Approximation. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing* (1993), pp. 294–304.

[6] P. Berman and M. Furer. Approximating Maximum Independent Set in Bounded Degree Graphs. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (1993), pp. 365–371.

[7] D.P. Bovet and P. Crescenzi. **Introduction to the Theory of Complexity.** Prentice-Hall, New York (1993).

[8] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, vol. 93 (1991), pp. 241–262.

[9] P. Crescenzi and L. Trevisan. On approximation scheme preserving reducibility and its applications. *14th Conference on Foundations of Software Technology and Theoretical Computer Science*, Lecture Notes in Computer Science, no. 880, pp 330–341, 1994.

[10] R. Fagin. Generalized First-Order Spectra and Polynomial-time Recognizable Sets. In Richard Karp (ed.), **Complexity of Computer Computations**, AMS, 1974.

[11] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Approximating clique is almost NP-complete. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science* (1991), pp. 2–12.

[12] Michael R. Garey and David S. Johnson. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. W. H. Freeman, 1979.

[13] Michel X. Goemans and David P. Williamson. .878-Approximation Algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the Twenty-Sixth ACM Symposium on Theory of Computing* (1994), pp. 422-431.

[14] M. M. Halldorsson and J. Radhakrishnan. Improved Approximations of Independent Sets in Bounded-Degree Graphs. *Nordic Journal of Computing*, v. 1, pp. 475–492, 1994.

[15] Dorit S. Hochbaum. Efficient bounds for the stable set, vertex cover, and set packing problems. *Discrete Applied Mathematics*, vol. 6 (1982), pp. 243–254.

[16] Viggo Kann. **On the Approximability of NP-complete Optimization Problems**. Ph.D. Thesis, Department of Numerical Analysis and Computing Science. Royal Institute of Technology, Stockholm, Sweden (1992).

[17] D. Karger, R. Motwani and G.D.S. Ramkumar. On approximating the longest path in a graph. In *Proceedings of the Third Workshop on Algorithms and Data Structures* (1993), pp. 421–432.

[18] S. Khanna, R. Motwani, M. Sudan, and U.V. Vazirani. On Syntactic versus Computational Views of Approximability. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science* (1994), pp. 819–830.

[19] S. Khanna, R. Motwani, and S. Vishwanathan. Approximating MAX SNP Problems via Semi-Definite Programming. *In preparation*, 1996.

[20] P. G. Kolaitis and M. N. Thakur. Approximation Properties of NP Minimization Classes. *J. Comput. System Sci.*, 50 (1995), pp. 391–411.

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

[22] A. Panconesi and D. Ranjan. Quantifiers and Approximation. Theoretical Computer Science, 107 (1993), pp. 145–163.

[23] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation, and Complexity Classes. *Journal of Computer and System Sciences*, vol. 43 (1991), pp. 425–440.

[24] C. H. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, vol. 18 (1993), pp. 1–11.

[25] M. Yannakakis. The analysis of local search problems and their heuristics. In *Proceedings of the 7th Annual Symposium of Theoretical Aspects of Computer Science* (1990), pp. 298–311.