Towards a Syntactic Characterization of PTAS

Sanjeev Khanna* Rajeev Motwani†

Abstract

The class PTAS is defined to consist of all NP optimization problems that permit polynomial-time approximation schemes. This paper explores the possibility that a core of PTAS may be characterized through syntactic classes endowed with restrictions on the structure of the input instances. Recent work in approximability of NP-hard problems has led to the identification of a syntactic class called MAX SNP as the core of APX, the class of constant-factor approximable NP optimization problems. This has enhanced our understanding of these classes from both an algorithmic and a complexity-theoretic point of view. Our work is motivated by the hope that a similar understanding can be attained for PTAS.

We argue that while the core of APX is the purely syntactic class MAX SNP, in the case of PTAS we must identify the core in terms of syntactic prescriptions for the problem definition augmented with structural restrictions on the input instances. Specifically, we propose such a unified framework based on syntactic classes restricted to instances exhibiting a planar structure. A variety of known and some new results follow directly from this framework, thereby lending credence to our hypothesis that there exists some common structural underpinnings for problems in PTAS.

1 Introduction

This paper is concerned with the question: *Is there a syntactic characterization of the class* PTAS*?* Recent work in approximability of NP optimization (NPO) problems has derived considerable insight from such considerations, based on a purely syntactic characterization of NP due to Fagin [8]. Research in this direction, initiated by Papadimitriou and Yannakakis [23], followed by Panconesi and Ranjan [21], and Kolaitis and Thakur [16], has led to the identification of approximation classes[1] such as MAX SNP, MAX NP, RMAX(2), and MIN $F^{+}\Pi_2(1)$. The syntactic prescription of these classes has led to at least partial resolution of questions such as: *Why are certain problems easier to approximate than others? What is the canonical structure of the hardest representative problems of an approximation class?* The syntactic prescription gives a natural way of identifying canonical hard problems in the class and performing approximation-preserving reductions to identify complete problems. Khanna, Motwani, Sudan, and Vazirani [15] introduced the notion of the closure of syntactic classes under approximation-preserving reductions, leading to a characterization of APX as the *closure* of MAX SNP, using some aspects of the results of Arora, Lund, Motwani, Sudan, and Szegedy [2] which established that MAX SNP is not contained in PTAS.

In this context, it is interesting to attempt a similar characterization of other approximability classes as the closure[2] of a syntactic class. We focus on the class PTAS. The starting point for our investigation is the remarkable paper by Baker [3] which presents PTAS's for a surprisingly large number of NP-hard graph optimization problems *when the input graph is planar.* It is our contention that the defining characteristic of PTAS problems is a "restriction on input structure" for some syntactic classes. Our work provides evidence towards this claim, raising the possibility that this input restriction may well be a notion related to planarity.

This may appear surprising at first considering that several well-known PTAS problems do not exhibit a graph structure, leave alone a planar graph structure, e.g., knapsack and multiprocessor scheduling. However, we will demonstrate that these problems are in fact special cases of generic problems arising from a planar input-restriction of gener-

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

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

[1] Readers not familiar with these problem classes may refer to Appendix A for their definitions.

[2] It seems essential to use closure since syntactic classes cannot be expressive enough to encompass P, leave alone all problems approximable within a given ratio. This is because purely syntactic considerations imply that such classes are strictly contained in NPO, therefore showing that they subsume P would separate P from NP. We expect that every such class would exclude some problems in P, and this has indeed been the case with all known definitions.

alizations of classes such as MAX SNP. It appears that most problems in PTAS fit directly into our framework. Moreover, our framework trivially implies membership in PTAS for problems such as PLANAR MAX SAT that were not known to be in PTAS earlier.

We identify a family of syntactic classes with input restrictions that subsume planar-restricted versions of classes such as MAX SNP, RMAX(2), and MIN $F^+\Pi_2(1)$. We then demonstrate that these classes are contained in PTAS using a collection of techniques that considerably generalize (and simplify) the idea used by Baker [3]. Our techniques are based on a separator theorem for outerplanar graphs due to Bodlaender [4] that strengthens the planar separator theorem of Lipton and Tarjan [18]. While almost every known problem in PTAS is easily seen to belong to our syntactic classes, there are a few problems, namely bin packing, planar graph TSP and some optimization problems on dense instances, that do not fit into our framework. We point out that, strictly speaking, many of these problems do not belong to PTAS, and hence would not serve as counterexamples to the possibility that some extension of our framework (via closure) is a characterization of PTAS.

Before describing our results in greater detail, we briefly review some earlier results concerning the role of planarity in PTAS. Lipton and Tarjan [19] were the first to present a PTAS for an NP-hard optimization problem on planar graphs, namely, maximum independent sets (MIS) in planar graphs. Subsequently, Chiba, Nishizeki, and Saito [7] extended this to finding a maximum induced subgraph satisfying a hereditary property determined by connected components.

The essential idea in results of this nature is that a recursive application of the planar separator theorem allows one to delete $o(n)$ vertices from a given planar graph G, so as to create a graph G' such that every component of G' has size $O(\log n)$. Then, it is possible to compute optimal solutions in each component in polynomial time, and the union of these optimal solutions is guaranteed to be a solution to the original problem. Using the fact that any such property must have a solution of size $\Omega(n)$, we can conclude that the resulting solution is a $(1-\epsilon)$ approximation, for any fixed $\epsilon > 0$ and sufficiently large n.

Hunt *et al* [13] applied Baker's technique to satisfiability problems where each clause consists of an application of a bounded-arity function from a given set of boolean functions. They also showed that for certain MAX SNP-complete problems, such as bounded-degree MIS, Baker's results can be extended to δ-near-planar graphs, i.e., graphs embeddable in the plane with at most δn edge crossings. They also show that Baker's framework can be suitably adapted to yield parallel approximation schemes.

Recently, Grigni, Koustsoupias, and Papadimitriou [12] gave a PTAS for TSP on unweighted planar graphs, resolving a longstanding question. Their algorithm combines a new separator theorem with a careful decomposition to create such an approximation scheme.

The rest of the paper is organized as follows. Section 2 gives an overview of the results and discusses some applications and directions for extending our results. Section 3 reviews basic concepts related to outerplanar graphs and tree decompositions. Finally, in Sections 4, 5, 6, and 7, we sketch the PTAS results for our proposed classes.

2 Summary of Results and the Implications

We begin by defining a new family of syntactic subclasses of NPO that subsume known syntactic classes MAX SNP, MAX NP, RMAX and MIN $F^+\Pi_2$ [16, 21, 23] (see Appendix A for definitions). Let $x_1, \ldots, x_n$ be a set of boolean variables. A *positive literal* is a variable x_i, and a *negative literal* is a negated variable $\overline{x}_i$. We use the abbreviation FOF to denote quantifier-free first-order formulas over these literals. We associate a weight with each FOF and with each variable; depending on the problem at hand, we may refer to the weights as *profits* or *costs*. The weight of a collection $\mathcal{C}$ of FOF's for a given truth assignment T is the net weight of the satisfied formulas, and the weight of T is the net weight of the variables assigned TRUE.

Definition 1 (Positive and Negative Minterms)
A minterm *is any conjunction of literals. A minterm is* positive *if it has only positive literals, and is* negative *if it has only negative literals.*

The following are the new *syntactic* classes of optimization problems. The class MPSAT captures constraint satisfaction problems where the objective is to compute solutions satisfying as many constraints as possible. It contains the usual satisfiability problems such as MAX 3-SAT. The other two classes, TMAX and TMIN, capture problems where a feasible solution must satisfy given constraints, and the objective is to find extremal feasible solutions. For instance, the MIS problem requires a solution containing an extremal number of vertices that satisfy constraints ruling out the inclusion of both end-points of an edge. We restrict our attention to monadic structures. The latter two classes complement each other, and are essentially dual to the class MPSAT. Naturally, we do not consider the minimization counterpart of MPSAT (where the goal is to *minimize* the number of unsatisfied constraints) since it contains problems that are NP-hard for small optimum values *even for planar input-restriction* [17].

Definition 2 (MPSAT)
The class MPSAT *consists of all* NPO *problems expressible as: given a collection $\mathcal{C}$ of* FOF*s over n variables such that each formula $\Phi \in \mathcal{C}$ is a disjunction of $O(n^{O(1)})$ minterms, find a truth assignment T of weight at most W maximizing the total weight of the* FOF*s in $\mathcal{C}$ that are satisfied.*

Definition 3 (TMAX)
The class TMAX *consists of* NPO *problems expressible as: given a collection $\mathcal{C}$ of* FOF*s over n variables such that each formula $\Phi \in \mathcal{C}$ is a disjunction of $O(n^{O(1)})$* negative *minterms, find a maximum weight truth assignment T that satisfies each* FOF *in $\mathcal{C}$.*

Definition 4 (TMIN)
The class TMIN *consists of* NPO *problems expressible as: given a collection $\mathcal{C}$ of* FOF*s over n variables such that each formula $\Phi \in \mathcal{C}$ is a disjunction of $O(n^{O(1)})$* positive *minterms, find a minimum weight truth assignment T that satisfies each* FOF *in $\mathcal{C}$.*

Observe that these classes are fairly strong generalizations of existing syntactic classes, in that they allow each constraint to depend on an unbounded number of variables and also allow each minterm to be of unbounded size. In

Section 7, we further generalize these definitions to problems over multi-valued domains allowing Min-Max objective functions, thereby obtaining the class MINMAX. This class captures problems such as multiprocessor scheduling.

We now endow these definitions with a restriction on the structure of the input formulas. This restriction is formulated in terms of incidence graphs. The classes of interest to us are those where the instances are restricted to having planar incidence graphs.

Definition 5 (Incidence Graph) *Let $\mathcal{I}$ be an instance of a problem* Π *in* MPSAT, TMAX, TMIN, *or* MINMAX. *Then, the* incidence graph *of* $\mathcal{I}$, *denoted* $G(\mathcal{I}) = (V, E)$, *is defined as follows:*

- *V has a v-vertex for each variable, and an f-vertex for each* FOF*;*
- *for each* FOF $\Phi \in \mathcal{S}$, *and each variable x_i in* Φ, *there is an edge between the v-vertex for x_i and the f-vertex for* Φ.

Definition 6 (Planar Input-Restructions) *Let $\mathcal{S}$ be any syntactic optimization class. The class* PLANAR $\mathcal{S}$ *is the class* $\mathcal{S}$ *restricted to instances with planar incidence graphs.*

2.1 Main Results and Implications

Our main results are summarized in the following proposition and brief proof sketches are given in the following sections. A similar result can be obtained for the class k-OUTERPLANAR MINMAX defined in Section 7; we omit the detailed exposition of this and other extensions.

Proposition 1 *Each of* PLANAR MPSAT, PLANAR TMAX, *and* PLANAR TMIN *is contained in* PTAS.

Some representative applications of this proposition are listed below; a more complete list will appear in the final version of the paper.

- PLANAR MAX SAT belongs to PLANAR MPSAT and hence has a PTAS. This is a new result. We believe that no PTAS was known earlier for any class of satisfiability problems involving constraints with an *unbounded* number of variables. The most general previous result, due to Hunt *et al* [13], was for formulas of bounded-arity drawn from a fixed family.
- The *knapsack* problem is in PLANAR MPSAT. A variable encodes the placement of an item in the knapsack, and a formula encodes the profit obtained by placing an item in the knapsack. The incidence graph is a matching and hence planar. This explains the PTAS [25] for knapsack.
- The *weighted MIS* problem in planar graphs is in PLANAR TMAX. Vertices are encoded by variables, and for each edge $e = (x, y)$ there is a formula $\Phi_e = (\overline{x} + \overline{y})$. For a planar graph G, the resulting incidence graph is also planar. This explains the PTAS [3] for planar weighted MIS.
- The *minimum weighted vertex cover (WVC)* problem in planar graphs is in PLANAR TMIN. Vertices are encoded by variables, and for each edge $e = (x, y)$ there is a formula $\Phi_e = (x + y)$. For a planar input graphs, the incidence graph is also planar. This explains the PTAS [3] for planar WVC.
- *Multiprocessor scheduling* on m machines, for any fixed m, is in OUTERPLANAR MINMAX with an empty set of constraints. This explains the PTAS [11] for multiprocessor scheduling.
- Consider the problem of finding the maximum induced subgraph satisfying a hereditary property Π determined by components [7]. For planar graphs, this can be placed in the class PLANAR TMAX by an approximation-preserving reduction such as the E-reduction [15]. The idea is to first apply planar separator theorem to delete $o(n)$ vertices such that the largest component size is only $O(1)$. This explains the PTAS [7] for such induced subgraph problems.

We can show that almost every known problem in PTAS can be placed in our framework in a like manner. There are three notable exceptions: bin packing [26, 14], some "dense" NP-hard problems [1], and planar-graph TSP [12]. We present one possible explanation for this incompleteness in the first two cases.

Consider first the classical bin packing problem. Vega and Lueker [26] show that there is a bin packing algorithm that, for any $\epsilon > 0$, achieves a packing into at most $(1+\epsilon)\mathrm{OPT}+1$ bins in polynomial time; in fact, Karmakar and Karp [14] give an algorithm that achieves a bound of $\mathrm{OPT}+o(\mathrm{OPT})$, implying a *fully polynomial* approximation scheme (FPTAS). While this may appear to be a PTAS, observe that via a reduction from the partition problem [9] it is easy to show that bin packing cannot be approximated within a ratio better than 1.5 unless P = NP. The apparent contradiction is resolved by noting the crucial role played by the additive error of 1. Their scheme is in fact an *asymptotic* PTAS [20]. The asymptotic approximation ratio of an algorithm A is defined as

$$R_A^\infty = \inf \left\{ \begin{array}{l} r \geq 1 : \text{for some} N \in Z^+,\ R_A(I) \leq r \text{ for all} \\ I \text{ such that } OPT(I) \geq N \end{array} \right\}$$

where R_A is the performance ratio for an instance I [9]. The point is that while bin packing has an asymptotic PTAS, it does not belong to PTAS and hence is ineligible for consideration in our framework. In fact, Karmakar and Karp [14] give an algorithm that achieves a bound of $\mathrm{OPT}+o(\mathrm{OPT})$, implying a *fully polynomial* approximation scheme (FPTAS). But bin packing is *strongly* NP-complete and hence cannot have an FPTAS [9]. This apparent contradiction can also be resolved by noting that the additive error could be relatively large unless one assumes that the optimum value is some suitably fast growing function of the input size, i.e., the Karmakar and Karp result is actually an *asymptotic* FPTAS [20].

Consider now the recent results due to Arora, Karger, and Karpinski [1] where they provide a unified scheme to obtain PTAS for dense instances of certain NP-hard optimization problems where, informally speaking, "dense" means that the optimal solution value is within a constant factor of the maximum possible value of a solution. For instance, they show that MAX CUT has a PTAS if the input instance contains $\Omega(n^2)$ edges. Specifically, for any $\epsilon > 0$, they obtain in polynomial time a solution of size $\mathrm{OPT} - \epsilon n^2$. Consequently, restricting to problem instances such that the optimum value is $\Omega(n^2)$, such a result translates into a PTAS scheme for MAX CUT.

In other words, we contend that these approximation schemes are possible because of the re-interpretation of the additive error term as a relative error term when the optimum value is appropriately constrained. These observations should however point to the delicacy of the arguments that may be required to make further progress on the questions raised in our work.

However, there remain a couple of notable exceptions. Arora et al [1] provide a PTAS for dense graph bisection even when the optimum value is small; clearly, this remains unexplained by the preceding discussion. Similarly, we are also unable to capture the planar-graph TSP problem [12] in our framework in any obvious manner. It is possible that this is only a technical difficulty, since the PTAS for that problem uses techniques that bear some similarity to the ones used by Baker and in this paper. In any case, our contention here is not that this framework exhaustively captures all problems in PTAS, but merely that it provides a unified framework for explaining known membership in PTAS, and for obtaining new results of this type. Such a unification raises the possibility that a generalization of this framework, or some variant thereof, may yield a syntactic characterization under closure for PTAS.

2.2 Possibility of Extension of Classes

In this section, we discuss the potential for extending our framework to more general classes that retain membership in PTAS. There are two natural directions for generalizing the syntactic classes wherein our results apply. One possibility is to relax some constraints in the logical definition of the optimization classes. So, for instance, it may seem natural to relax the restriction of $n^{O(1)}$ minterms in the class MPSAT by considering CNF representations of the formulas (or oracle representations, for that matter). But this definition is immediately general enough, even with an input restriction to *trees*, to capture 3-SAT as a special case, and hence would not permit a PTAS. Similarly, one could considering relaxing the constraint enforcing strictly negative minterms in the class TMAX and strictly positive minterms in the class TMIN. But once again, relaxing this immediately rules out the possibility of PTAS in a strong sense since then we can encode PLANAR 3-SAT. However, our results for MPSAT, TMAX, and TMIN generalize to non-boolean domains, i.e., they hold even when the variables take on a polynomially-bounded number of distinct values. Furthermore, they hold even when the notion of constraints is relaxed to functions over non-boolean domains modulo some restrictions. Our techniques can also be used to obtain PTAS for certain classes of integer linear programs. We omit a formal exposition of these extensions.

The other direction for extending our results is by relaxing the planarity restriction. In the final version, we will describe an extension our results to δ-near-planar graphs provided that the optimum value is proportional to the number of vertices in the incidence graph. (This is similar to the results of Hunt et al [13] for specific MAX SNP problems which clearly satisfy the restriction on the optimum value.) The key idea is to start with a near-planar embedding and replace each edge crossing with a Lichtenstein gadget [17] that introduces auxiliary formulas. In other words, problems on such graphs have an approximation-preserving reduction to planar graphs. An interesting direction for future work is: are there more general classes of graphs to which our results can be extended? One possibility is to consider families of graphs with excluded minors.

2.3 Overview of Proof Techniques

All known PTAS results for planar graph problems employ the following paradigm: find a separator of the planar graph G such that the problem is either optimally or approximately solvable on the components resulting from the deletion of the separator; show that the separator contributes only a small fraction of the optimal solution, obtain a solution by combining the solutions on the components. Some earlier results [19, 7] were based on finding small separators that yielded small components. Baker [3] observed that many NP-hard optimization problems can be solved optimally in polynomial time on $O(1)$-*outerplanar graphs*, and that separators can be found that yield $O(1)$-outerplanar components containing sufficient information about the optimal solution to permit a PTAS.

Our approach extends Baker's paradigm by computing either an exact solution or a PTAS for generic syntactic problems on outerplanar graphs, and then extending this to planar graphs. While Baker's paradigm employs dynamic programming on $O(1)$-outerplanar explicitly based on a planar embedding, our approach, on the other hand, is to use a more structured and uniform representation of outerplanar graphs in terms of a *tree-decomposition* [4]. In our abstraction, dynamic programming combined with rounding techniques can be used to solve optimally or approximately the proposed syntactic problems when the incidence graphs are restricted to being $O(1)$-outerplanar. The syntactic structure of the problem is then used to extend the solution to the original planar graph by either using a second-level of dynamic programming, or a pigeon-hole argument similar to Baker's.

3 Outerplanar Graphs and Tree Decompositions

Our constructions of PTAS for the planar versions of these complexity classes depends on the strengthening of planar separator theorem by Bodlaender [4] for $O(1)$-outerplanar graphs.

Definition 7 (p-Outerplanar Graph)
A 1-outerplanar *graph, or simply an* outerplanar *graph, is a planar graph that has an embedding on the plane with all vertices appearing on the outer face; a p-outerplanar graph is a planar graph that has an embedding on the plane in which deleting all vertices on the outer face yields a $(p-1)$-outerplanar graph.*

Observe that the outer face of a 1-outerplanar graph need not be a simple cycle; for example, a tree is outerplanar and two cycles meeting at a single vertex is also outerplanar. We describe a useful structural property of outerplanar graphs based on the notion of treewidth due to Robertson and Seymour [24].

Definition 8 (Tree Decomposition)
A tree decomposition of a graph $G = (V, E)$ is a 2-tuple $\langle \{X(i) \mid i \in I\}, T = (I, F)\rangle$ such that:

- *for each $v \in V$, there exists an $i \in I$ with $v \in X(i)$;*
- *for all $(v, w) \in E$, there exists an $i \in I$ with $v, w \in X(i)$;*
- *and, for all $v \in V$, the set $\{i \in I \mid v \in X(i)\}$ induces a subtree.*

Definition 9 (Width and Treewidth)
The width *of a tree-decomposition $\langle\{X(i) \mid i \in I\}, T = (I, F)\rangle$ of a graph G is $\max\{|X(i)| \mid i \in I\} - 1$. The* treewidth *of a graph is the minimum width over all tree-decompositions.*

For any node Y in the decomposition tree, define the graph $G[Y]$ as the subgraph of the graph G induced by the set of vertices $V[Y]$ consisting of the vertices stored at all the nodes in the subtree rooted at Y. Thus, for the root X, we have $V[X] = V$ and $G[X] = G$.

Proposition 2 *A tree decomposition of a graph G may be viewed as a separator tree. In particular, at the root X with children L and R,*

1. *$V[L] \cap V[R] \subseteq X$, and*
2. *there are no edges from $V[L] \setminus X$ to $V[R] \setminus X$ in G.*

Further, the same property holds recursively at each node in the tree.

Lemma 1 ([4]) *A p-outerplanar graph has treewidth at most $3p - 1$.*

Definition 10 (Nice Tree Decomposition)
A tree decomposition $\langle\{X(i) \mid i \in I\}, T = (I, F)\rangle$ is nice *if one can choose a root r in the tree such that:*

- *T is a binary tree,*
- *if i is a leaf, then $|X(i)| = 1$* (START node)*;*
- *if i has two children j and k, then $X(i) = X(j) = X(k)$* (JOIN node)*;*
- *and, if i has one child j, then either*
  - *there exists a vertex v with $X(i) = X(j) - \{v\}$* (FORGET node)*, or*
  - *there exists a vertex v with $X(i) = X(j) \cup \{v\}$* (INTRODUCE node)*.*

We use the following known results: there is always a nice tree decomposition of optimal width [4]; and, there is a linear time algorithm to a construct tree-decomposition of graphs with $O(1)$ treewidth [5].

4 Planar MPSAT

Let w_v denote the weight associated with the variable v and p_f denote the profit associated with the formula f; furthermore, let W denote the weight threshold and let P denote the sum of all the profits.

We begin by sketching a *pseudo-polynomial* time algorithm for the case of $O(1)$-outerplanar graphs. The first step is to compute a nice tree decomposition T for a given $O(1)$-outerplanar graph G; recall that it has treewidth $O(1)$. We need some definitions.

Definition 11 (Tuple)
A tuple over a set of variables $x_1, ..., x_p$ and a set of formulas $f_1, \ldots, f_q$ is a vector of the form

$$\langle v_1, \ldots, v_p, m_1, \ldots, m_q\rangle,$$

where each $v_i \in \{0, 1\}$ and each m_i is either a minterm that occurs in formula f_i or is NULL.

Definition 12 (Consistent Tuple)
A tuple

$$\vec{t} = \langle v_1, \ldots, v_p, m_1, \ldots, m_q\rangle$$

over a set of variables $x_1, \ldots, x_p$ and formulas $f_1, \ldots, f_q$ is called a consistent tuple *if whenever a variable x_i occurs in a minterm m_j in t, then $v_i = 1$ if x_i occurs as a positive literal in m_j, and $v_i = 0$ otherwise.*

In our algorithms, we maintain the following 2-dimensional structure at each node in the tree T.

Definition 13 (VLIST)
For a node X in the decomposition tree, $\text{VLIST}[\vec{t}, p]$ *denotes the minimum weight needed to achieve a profit of p in the* MPSAT *subproblem represented by the subgraph $G[X]$ when the variables and formulas at node X are set in accordance with the tuple $\vec{t}$.*

Definition 14 (EXT)
Given a tuple $\vec{t}$ and a variable or formula x, the extension set $\text{EXT}(\vec{t}, x)$ consists of all tuples that are extensions of $\vec{t}$ with each possible value that can be assigned to x.

Definition 15 (PROJ)
Given a tuple $\vec{t}$ and a variable or formula x, the projection $\text{PROJ}(\vec{t}, x)$ is the tuple obtained from $\vec{t}$ by dropping the component corresponding to x.

Definition 16 (WT **and** PT)
*Given a tuple $\vec{t}$, $\text{WT}(\vec{t})$ denotes the weight of variables set to true in $\vec{t}$, and $\text{PT}(\vec{t})$ denotes the total profit associated with formulas appearing in $\vec{t}$ with non-*NULL *minterms.*

We now show that the VLIST array can be efficiently computed at the root of a nice treewidth decomposition tree using a bottom-up approach. The entries in the VLIST array at each node are initialized to $+\infty$. At the leaves, the array can be trivially computed in polynomial time. To compute the array at an intermediate node such that it has already been computed for its children, we analyze three cases:

- at a JOIN node X with children L and R, we compute

$$\text{VLIST}_X[\vec{t}, p] = \min_{0 \le i \le p} \left\{ \begin{array}{l} \text{VLIST}_L[\vec{t}, i + \text{PT}(\vec{t})] \ + \\ \text{VLIST}_R[\vec{t}, p - i] - \text{WT}(\vec{t}) \end{array} \right\},$$

- at a FORGET node X with a child L such that $X = L \setminus \{x\}$, we compute

$$\text{VLIST}_X[\vec{t}, p] = \min_{\vec{t'} \in \text{EXT}(\vec{t}, x)} \text{VLIST}_L[\vec{t'}, p],$$

- finally, at an INTRODUCE node X with a child L such that $X = L \cup \{x\}$, we consider two subcases.

  If x corresponds to a variable v then

$$\text{VLIST}_X[\vec{t}, p] = \begin{cases} \text{VLIST}_L[\text{PROJ}(\vec{t}, x), p] \\ \quad \text{if } \vec{t} \text{ is consistent and } v \text{ is false in } \vec{t} \\ \text{VLIST}_L[\text{PROJ}(\vec{t}, x), p] + w_v \\ \quad \text{if } \vec{t} \text{ is consistent and } v \text{ is true in } \vec{t} \\ +\infty \qquad \text{otherwise} \end{cases}$$

On the other hand, if x corresponds to a formula f then

$$\text{VLIST}_X[\vec{t},p] = \begin{cases} \text{VLIST}_L[\text{PROJ}(\vec{t},x), p-p_f] \\ \quad \text{if } \vec{t} \text{ is consistent and contains a} \\ \quad \text{non-NULL minterm for } f \\ \text{VLIST}_L[\text{PROJ}(\vec{t},x), p] \\ \quad \text{if } \vec{t} \text{ is consistent and contains a} \\ \quad \text{NULL minterm for } f \\ +\infty \qquad\qquad\qquad \text{otherwise} \end{cases}$$

In all cases, the computation of VLIST takes polynomial time, and we perform this computation exactly once at each node. Hence, the VLIST computation at the root R_T of T takes only polynomial time. Finally, observe that by definition of VLIST, the largest value occurring in VLIST(R_T) is the desired optimal solution. The proof of correctness of this computation, as well as a bound of $O(n^{O(1)}P^2)$ on the running time, is omitted. By using a standard technique of rounding the lower order bits, we can obtain a PTAS from this pseudo-polynomial time algorithm. The details are omitted; see for example [22]. We obtain the following results.

Lemma 2 *The class* $O(1)$-OUTERPLANAR MPSAT *is contained in* PTAS.

Corollary 1 $O(1)$-OUTERPLANAR MPSAT *is in* P *when the formulas have uniform profits.*

Thus satisfiability problems such as MAX 3-SAT become polynomial-time solvable on this subclass of planar graphs. Consider now the following theorem.

Theorem 1 *The class* PLANAR MPSAT *is in* PTAS.

Proof Sketch: This is established by combining Baker-style arguments with a dynamic programming scheme. Consider the input graph G and assume that it is t-outerplanar for $t \le n+m$, where n is the number of variables and m is the number of formulas in the collection $\mathcal{C}$. Divide the vertices into t levels, called $L_1,\ldots,L_t$, such that L_t corresponds to the outer face and each level L_i is the outer face obtained by removing the levels $L_t,\ldots,L_{i+1}$.

Fix any optimal truth assignment, and let s_i denote the *weight of the clauses satisfied* in level L_i by this assignment. Partition the levels $L_1,\ldots,L_t$ into $p+1$ groups, $S_0,\ldots,S_p$, where group S_r is the union of the levels L_i whose index i is congruent to $3r$, $3r+1$, or $3r+2$ modulo p', where $p' = 3(p+1)$. By pigeon-holing, there exists a group S_j such that

$$\sum_{L_i \in S_j} s_i \le \frac{\text{OPT}}{p+1}.$$

This special group S_j may be identified by trying all possible choices of j, and picking the one which yields the best solution. Having fixed the choice of S_j, delete all vertices in the levels whose index is congruent to $3j+1$ modulo p', thereby separating the graph into a collection of disjoint $(p'-1)$-outerplanar graphs, say $G_1, G_2, \ldots, G_t$, such that the total optimal value on this collection is at least $(1-(3/p)) \times \text{OPT}$.

For a given weight threshold W, we know how to obtain a pseudo-polynomial time algorithm for any G_i. But we need to find an "optimal partitioning" of the total weight W into sub-thresholds $W_1, W_2, \ldots, W_t$, where W_i is the weight threshold allocated to G_i. To this end, we begin by performing the VLIST computation for each G_i. Once this two-dimensional array is computed, we can extract from it a one-dimensional profit array $A_i[0..P]$ for each G_i, such that $A_i[p]$ denotes the minimum weight needed to achieve profit p in the instance encoded by G_i.

Next, we create a complete binary tree whose leaves correspond to $G_1, G_2, \ldots, G_t$ and an intermediate node corresponds to the union of the graphs represented by its children. Thus, the root corresponds to $\bigcup_{i=1}^{t} G_i$. This can now be used to iteratively compute the profit arrays associated with any non-leaf vertex. Consider a non-leaf vertex v such that we already have the profit arrays for its two children u and w. The array at v can be computed as follows:

$$\forall i \in [0..P]\ \forall j \in [0..i]\ A^v[i] \leftarrow \min\{A^v[i], A^u[j] + A^w[i-j]\}.$$

We omit the formal proof of correctness of this computation and of the total running time bound as a polynomial in n and P. Thus, we have a pseudo-polynomial time approximation scheme for the constrained optimization problem. The final step is to use the standard lower-order bit rounding technique to obtain a PTAS. We omit the details (refer to Papadimitriou [22]). □

5 Planar TMAX

We sketch the PTAS for PLANAR TMAX. As before, w_v is the weight of variable v; since all constraints must be satisfied, we do not associate any profits with the formulas. We begin with the following lemma.

Lemma 3 *The class* $O(1)$-OUTERPLANAR TMAX *is contained in* P.

In the proof sketch for this lemma, we will define VLIST as a one-dimensional array.

Definition 17 (VLIST)
For a node X in the decomposition tree, VLIST$[\vec{t}]$ *denotes the maximum weight achievable in the* TMAX *subproblem represented by the subgraph $G[X]$ when the variables and formulas at node X are set in accordance with the tuple $\vec{t}$.*

The entries in the VLIST array at each node are initialized to $-\infty$. In the rest of this section, we will assume that a tuple *does not* contain a NULL term for any formula, i.e., we consider only those tuples that satisfy all their formulas.

We briefly sketch the computation of the VLIST array at the root of a nice treewidth decomposition tree using a bottom-up approach. At the leaves, the array can be trivially computed in polynomial time. To compute the array at an intermediate node where it has already been computed for its children, we consider three cases:

- at a JOIN node X with children L and R, we compute

$$\text{VLIST}_X[\vec{t}] = \text{VLIST}_L[\vec{t}] - \text{WT}(\vec{t}),$$

- at a FORGET node X with a child L such that $X = L \setminus \{x\}$, we compute

$$\text{VLIST}_X[\vec{t}] = \max_{\vec{t'} \in \text{EXT}(\vec{t},x)} \text{VLIST}_L[\vec{t'}],$$

- finally, at an INTRODUCE node X with a child L such that $X = L \cup \{x\}$, we consider two subcases.

  If x corresponds to a variable v then

$$\text{VLIST}_X[\vec{t}] = \begin{cases} \text{VLIST}_L[\text{PROJ}(\vec{t}, x)] \\ \quad \text{if } \vec{t} \text{ is consistent and } v \text{ is false in } \vec{t} \\ \text{VLIST}_L[\text{PROJ}(\vec{t}, x)] + w_v \\ \quad \text{if } \vec{t} \text{ is consistent and } v \text{ is true in } \vec{t} \\ -\infty \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

  On the other hand, if x corresponds to a formula f then

$$\text{VLIST}_X[\vec{t}] = \begin{cases} \text{VLIST}_L[\text{PROJ}(\vec{t}, x)] & \text{if } \vec{t} \text{ is consistent} \\ -\infty & \text{otherwise} \end{cases}$$

We omit the proof of correctness of this computation and the running time bound of $O(n^{O(1)})$. A careful look at this computation reveals that we did not really use the negativity constraint on the minterms and thus we can conclude a stronger result.

Corollary 2 *The class $O(1)$ OUTERPLANAR TMAX without negativity constraints on the minterms is in* P.

We now sketch the extension to planar graphs leading to the following theorem.

Theorem 2 *The class* PLANAR TMAX *is contained in* PTAS.

Proof Sketch: The idea is to remove a collection of layers whose deletion leaves an instance with optimal value at least $(1-O(1/p))\times$OPT and which is a collection of p-outerplanar graphs, say $G_1, \ldots, G_t$. Fix any optimal assignment and let s_i denote the *weight of the variables* set to true in level L_i. Partition the levels $L_1, \ldots, L_t$ into $p+1$ groups, $S_0, \ldots, S_p$, where group S_r is the union of the levels L_i whose index i is congruent to $2r$ or $2r+1$ modulo p', where $p' = 2(p+1)$. By pigeon-holing, there exists a group S_j such that

$$\sum_{L_i \in S_j} s_i \leq \frac{\text{OPT}}{p+1}.$$

This special group S_j may be identified by trying all possible choices of j, and picking the one that yields the best solution. Having fixed the choice of S_j, delete all *variable vertices* in the levels represented in S_j. Compute the optimal solutions in each of the resulting $(2p+1)$-outerplanar graphs (recall that the formula vertices do not have any edges between them). Since these graphs do not share any vertices, we can compute their optimal solutions independently and combine these solutions without any conflict. The combined solution is finally extended by setting all the variables in the special group S_j to FALSE and the remaining variables according to the value assigned by the above solutions. It is easy to see that due to the *negative minterm property*, this scheme of composing the optimal solution results in a solution satisfying all the constraints. And finally, the weight of the solution is at least $(1 - 1/(p+1))$OPT. □

6 Planar TMIN

An algorithm similar to the one in Section 5 can be used to establish that $O(1)$-OUTERPLANAR TMIN is in P and also that $O(1)$-OUTERPLANAR TMIN without positivity constraints on the minterms lies in P. However, the scheme needed to create a PTAS for planar graph is slightly different; it is a variant of the scheme used by Baker [3] for the minimum vertex cover problem. The idea is to consider groups of p consecutive layers such that the last two layers of any group are identical to the first two layers of the next group. If we compute solutions for the p-outerplanar graphs defined by these layers and combine them by setting a variable to TRUE whenever it is set to TRUE in any solution and FALSE otherwise, it is easy to see that such a solution must indeed satisfy all the constraints because each constraint vertex along with its constituent variable vertices must be completely contained in one of the groups. Now, if we fix any optimal assignment, there exists a way of defining a shift of these layers such that the total weight of OPT in the overlapping layers is only $O(\text{OPT}/p)$. Hence, we have a PTAS.

7 $O(1)$-Outerplanar MINMAX

In this section we generalize our syntactic classes to capture problems over multi-valued domains where the objective is to have a "balanced solution" satisfying a given set of constraints. However, our results hold only for $O(1)$-outerplanar graphs – a class general enough to capture problems such as multiprocessor scheduling as a special case.

Definition 18 (k-ary term)
A k-ary term *over a set of n variables is a constraint of the form $\langle x_{i_1} = s_1, \ldots, x_{i_p} = s_p \rangle$ where $s_i \in [0..k]$ and k is a constant. An* assignment satisfies *the term if and only if its projection on the variables in the term coincides with the specified values.*

Definition 19 (k-ary formula)
A k-ary formula *over a set of n variables is a set of k-ary terms such that the formula is* satisfied *by an assignment if and only if it satisfies at least one term in the formula.*

Definition 20 (Variable Class)
Given an assignment s, a variable class i *denotes the set of variables set to value i in the assignment s.*

We define a new class as follows.

Definition 21 (MINMAX)
The class MINMAX *consists of* NPO *problems expressible as: given a closed extendible collection $\mathcal{C}$ of* FOF*s over n variables such that each formula $\Phi \in \mathcal{C}$ has $O(n^{O(1)})$ k-ary terms, find an assignment which minimizes the maximum-weight variable class.*

We briefly sketch the ideas. As we did before, for $O(1)$-outerplanar graphs we perform a bottom-up VLIST computation on a nice tree-decomposition. The VLIST array is a 2-dimensional array – the array entry $\text{VLIST}[\vec{t}, \langle w_0, w_1, \ldots, w_k \rangle]$ stores a YES/NO answer to the question of whether there is a feasible solution which respects the constraint set and the variable assignment specified by the tuple, and partitions the variable weights into $\langle w_0, w_1, \ldots, w_k \rangle$; where $0 \leq w_i \leq W$

and $W = \sum_{i=1}^{n} \mathrm{WT}(x_i)$. This computation can be performed in time $O((nW)^{O(1)})$, since k is a constant. At the root, we simply choose the optimal feasible solution over all tuples. Hence, we have a pseudo-polynomial time algorithm. But observe that OPT $\geq W/(k+1)$ and therefore, once again using the lower order bit rounding techniques, we can convert this into a PTAS result.

Acknowledgements

We are grateful to Sanjeev Arora and Phokion Kolaitis for several motivating and useful discussions throughout the course of this work. In particular, we thank Sanjeev Arora for his collaboration in the preliminary stages. We are also grateful to Chandra Chekuri, Christos Papadimitriou, and Mihalis Yannakakis for helpful comments and suggestions.

References

[1] S. Arora, D. Karger, and M. Karpinski. Polynomial Time Approximation Schemes for Dense Instances of NP-hard Problems. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing* (1995).

[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] B.S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41:153–180 (1994).

[4] H. L. Bodlaender. Some Classes of Graphs with Bounded Treewidth. *Bulletin of the European Association for Theoretical Computer Science* (1988), pp. 116–126.

[5] H. L. Bodlaender. A Linear Time Algorithm for Finding Tree-decompositions of Small Treewidth. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing* (1993), pp. 226–234.

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

[7] N. Chiba, T. Nishizeki, and N. Saito. Applications of the Planar Separator Theorem. *Journal of Information Processing*, 4:203–207 (1981).

[8] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In Richard Karp (ed.), *Complexity of Computation*, AMS (1974).

[9] M.R. Garey and D.S. Johnson. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. W. H. Freeman (1979).

[10] N. Garg, H. Saran, and V. Vazirani. Finding separator cuts in planar graphs within twice the optimal. In *Proceedings of the 35th Annual ACM Symposium on Foundations of Computer Science* (1994), pp. 14–23.

[11] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical Journal*, 45:1563–1581 (1966).

[12] M. Grigni, E. Koustsoupias, and C. Papadimitriou. An Approximation Scheme for Planar Graph TSP. In *Proceedings of the 36th Annual ACM Symposium on Foundations of Computer Science* (1995), pp. 640–645.

[13] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R. E. Stearns. Approximation Schemes using L-reductions. In *Proceedings of the Conference on Foundations of Software Technology and Theoretical Computer Science* (1994), pp. 342-353.

[14] N. Karmakar and R.M. Karp. An Efficient Approximation Scheme For The One-Dimensional Bin Packing Problem. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science* (1982), pp. 312–320.

[15] 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.

[16] P.G. Kolaitis and M.N. Thakur. Approximation Properties of NP Minimization Classes. In *Proceedings of the Sixth Annual Structure in Complexity Theory Conference* (1991), pp. 353–366.

[17] D. Lichtenstein. Planar Formulae and their Uses. *SIAM Journal on Computing*, 11:329–343 (1980).

[18] R.J. Lipton and R.E. Tarjan. A separator theorem for planar graphs. *SIAM Journal of Applied Mathematics*, 36:177–189 (1979).

[19] R.J. Lipton and R.E. Tarjan. Applications of Planar Separator Theorem. *SIAM Journal on Computing*, 9:615–627 (1980).

[20] R. Motwani. Lecture Notes on Approximation Algorithms. Report No. STAN-CS-92-1435, Department of Computer Science, Stanford University (1992).

[21] A. Panconesi and D. Ranjan. Quantifiers and Approximation. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing* (1990), pp. 446–456.

[22] C.H. Papadimitriou. **Computational Complexity.** Addison-Wesley (1994).

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

[24] N. Robertson and P. D. Seymour. Graph Minors. II. Algorithmic Aspects of Treewidth. *Journal of Algorithms*, 7:309–322 (1986).

[25] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 22:115–124 (1975).

[26] W. Fernandez de la Vega and G.S. Lueker. Bin Packing can be solved within $1+\epsilon$ in Linear Time, *Combinatorica*, 1:349–355 (1981).

Appendix

A Definitions of Classes of Optimization Problems

Given an NPO problem Π and an instance $\mathcal{I}$ of Π, we use $|\mathcal{I}|$ to denote the length of $\mathcal{I}$ and $\mathrm{OPT}(\mathcal{I})$ 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 [6] for a precise definition of NPO).

Definition 22 (PTAS)
An NPO *problem* Π *is in the class* PTAS *if for any fixed* $\epsilon > 0$*, there exists a polynomial time computable function* A *mapping instances of* Π *to solutions, such that for all instances* $\mathcal{I}$ *of* Π*,*

$$\frac{V(I, A(\mathcal{I}))}{1+\epsilon} \leq \mathrm{OPT}(\mathcal{I}) \leq (1+\epsilon)V(I, A(\mathcal{I})).$$

Definition 23 (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*

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

Definition 24 (RMAX(k) [21])
RMAX(k) *is the class of* NPO *problems expressible as finding a structure* S *which maximizes the objective function*

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

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.*

Definition 25 (MIN $\mathbf{F}^+\Pi_2(k)$ [16])
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.*