LA-UR 96-1466

CONF-961004--3

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: **COMPLEXITY AND EFFICIENT APPROXIMABILITY OF TWO DIMENSIONAL PERIODICALY SPECIFIED PROBLEMS**

AUTHOR(S): M. V. Marathe, Harry B. Hunt, R. E. Stearns

SUBMITTED TO: 37th IEEE Symposium on Foundations of Computer Science
Bulington, Vermont
~~November~~ oct 1996

MASTER

Los Alamos National Laboratory
Los Alamos New Mexico 87545

Complexity and Efficient Approximability of Two Dimensional Periodically Specified Problems

MADHAV V. MARATHE[1] HARRY B. HUNT III [2] RICHARD E. STEARNS[2]

Abstract

We consider the two dimensional periodic specifications: a method to specify succinctly objects with highly regular repetitive structure. These specifications arise naturally when processing engineering designs including VLSI designs. These specifications can specify objects whose sizes are *exponentially larger* than the sizes of the specification themselves. Consequently solving a periodically specified problem by explicitly expanding the instance is prohibitively expensive in terms of computational resources. This leads us to investigate the complexity and efficient approximability of solving graph theoretic and combinatorial problems when instances are specified using two dimensional periodic specifications. We prove the following results:

1. Several classical NP-hard optimization problems become NEXPTIME-hard, when instances are specified using two dimensional periodic specifications.
2. In contrast, several of these NEXPTIME-hard problems have polynomial time approximation algorithms with guaranteed worst case performance.

Two properties of our results are:

1. For the first time, efficient approximation algorithms and schemes are developed for **natural** NEXPTIME-complete problems. (Of course it is easy to devise efficient approximation algorithms for arbitrarily "hard" artificial problems.)
2. Our results are the first polynomial time approximation algorithms with good performance guarantees for "hard" problems specified using any of the kinds of periodic specifications considered here.

[1]Current Address: Los Alamos National Laboratory P.O. Box 1663, MS K990 Los Alamos NM 87545. Email: madhav@c3.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

[2]Department of Computer Science, University at Albany - SUNY, Albany, NY 12222. Email addresses of authors: {hunt,res}@cs.albany.edu. Supported by NSF Grants CCR 90-06396 and CCR94-06611.

1 Introduction and Motivation

Many practical applications of graph theory and combinatorial optimization in CAD systems, mechanical engineering, VLSI design, transportation networks and software engineering involve processing large (but regular) objects constructed systematically from smaller and more manageable components. Consequently, the graphs that abstract the structure and operation of the underlying circuits (designs) also have a regular structure and are defined in a systematic manner using smaller graphs. Such methods for describing large and regular objects by small descriptions are referred to as *succinct specifications.* In this paper, we focus our attention on a class of succinct specifications called the *periodic specifications* [CM91, HW95, IS87, KO91, KS88, Or82a, Wa93]. Using periodic specifications, large objects are described as repetitive connections of a basic module. The modules are repeated in one, two or higher dimensions. Two dimensional periodic specifications arise naturally in the study of regular systolic arrays and VLSI signal processing arrays [HW94, CS81, IS86, IS87, IS88], discrete dynamical systems such as the cellular automata [Su95, Wo84], parallel programming [HLW92, KMW67], etc. For example, in the design of Field Programmable Gate Arrays (FPGA's), the problem of compaction and routing can be modeled as a shortest path problem in two dimensional periodically specified graphs [Br95]. Similarly the problem of mapping uniform recursive or iterative programs on a 2-dimensional mesh connected parallel computer is modeled as solving systems of periodically specified systems of equations (aka. uniform recurrence equations) [HLW92, KMW67]. In digital signal processing periodic specifications are used to design bit parallel FIR filter [CS81]. Finally, two dimensional periodic specifications can also be easily seen as a way of representing the dynamic changes in the configuration of finite one dimensional cellular automata over time (i.e the second dimension represents time) [Su95, Wo84]. Using this representation the configuration reachability problem for a finite one dimensional cellular automata is simply the circuit value problem for periodically specified circuits.

Typically, the periodic specifications studied in the literature are generalizations of standard specifications used to describe objects. In general, periodic specifications can describe objects that are exponentially larger than size of the specifications themselves. For example, the family of all rectangular grids can be specified using periodic specifications of logarithmic size. As pointed out in [LW93, CS81, IS86], rectangular grids are an important class of graphs occurring in engineering designs. Consequently, solving any periodically specified problem by first *explicitly expanding* the design explicitly can be prohibitively costly in computational resources. For example, even a linear time algorithm for the minimum spanning tree problem applied to the expanded object can take time that is exponential in the size of the periodic specification. The situation is even worse since the designs obtained by expanding the periodic specifications are frequently too large to fit into the main memory of a computer [Le86, HLW92]. As a result, even the most efficient algorithms incur a large number of page faults while processing graphs (or circuits) obtained by expanding the periodic specifications [Le86, HLW92]. Hence, algorithms designed for solving problems for graphs or circuits represented in a standard fashion are often impractical for periodically specified inputs.

Thus it is of interest to investigate if the regular structure of the underlying objects can be exploited to design efficient algorithms that take time/space that is a *polynomial function of the periodic specification rather than the expanded object.* Motivated by the practical importance of processing such engineering designs, we investigate the complexity (measured in terms of the input specification) of solving classical graph theoretic and combinatorial problems specified using two dimensional periodic specifications.

2 Summary of Results and Significance

Here we study the complexity and efficient approximability of classical graph theoretic and combinatorial problems specified using the following kinds of periodic specifications:

1. The 2-dimensional finite periodic narrow specifications of Wanke [Wa93], (referred as 2-F(B,B)PN-specifications),
2. The 2-dimensional finite periodic narrow specifications with explicit boundary conditions (referred as 2-F(B,B)PN(BC)-specifications)
3. The 2-dimensional finite toroidal periodic narrow specifications of Höfting and Wanke [HW94], (referred as 2-F(B,B)PTN-specifications) and

Let Π be a problem. We use the term *standard specification* to refer to the method or methods commonly used it the literature to specify an instance of Π. Letting α denote any of the three specifications 2-F(B,B)PN, 2-F(B,B)PN(BC) or 2F(B,B)PTN we use the notation α-Π to refer to problem Π specified by any of the four specifications. For example, 2-F(B,B)PN-3SAT denotes the problem 3SAT when instances are specified using 2-F(B,B)PN-specifications and 2-F(B,B)PTN-3SAT denotes the problem 3SAT when instances are specified using 2-F(B,B)PTN-specifications. We also refer to any of the specifications mentioned above as *succinct specifications.* For the rest of the paper we let Π be one of the following problems

MAX-3SAT, MAX-SAT(S), MIN VERTEX COVER, MAX INDEPENDENT SET, MIN DOMINATING SET, MIN EDGE DOMINATING SET, MAX H-MATCHING and MAX CUT

2.1 Approximation Algorithms for Succinctly Specified Problems

Here and in [MH+95], we prove that a number of natural graph and satisfiability problems, are NEXPTIME-complete, when instances are specified by one of the periodic specifications considered here. We also prove that most of these problems remain NEXPTIME-complete, even when restricted to planar instances. Here, given these hardness results, we investigate the existence of polynomial time approximation algorithms for these problems. We present a *simple* yet *general* approach for developing efficient approximation algorithms and/or schemes for a number of natural optimization problems, when instances are specified using one of the specifications α. The following theorem summarizes our first main result.

Theorem 2.1 For each fixed $l \geq 1$, and for each of the problems Π stated above,[3] the problem α-Π, has a polynomial time approximation algorithm with running time $O(RT_\Pi(l^2 \cdot |G|))$ with performance guarantee[4] $(\frac{l+1}{l})^2 \cdot FBEST_\Pi$. Here $|G|$ denotes the size of the specification, $FBEST_\Pi$ denotes the best known performance guarantee of an algorithm for the problem Π for non-succinctly specified instances and $RT_\Pi(n)$ denotes the running time of the algorithm with input size n which guarantees a performance of $FBEST_\Pi$ for the problem Π.

As an example, using recent results in [GW94], we get that for all $\epsilon > 0$, the problems 2-F(B,B)PN-, 2-F(B,B)PN(BC)-, 2F(B,B)PTN- and have polynomial time approximation algorithms that output solutions within a factor of $(1 + \epsilon) \cdot 1.137$ of an optimum solution. As a corollary of Theorem 2.1 and the recent nonapproximability results of [AL+92], we get the following corollary.

[3] In fact we can show that the theorem holds for most problems α-Π such that Π is in syntactic MAX SNP.

[4] For the sake of uniformity we assume that the performance guarantee is ≥ 1.

Theorem 2.2 1. For all the problems Π as stated above, the problems α-Π have polynomial time approximation schemes if and only if P = NP.

2. For all the problems Π as stated above the problems α-Π have polynomial time approximation schemes (PTAS) when restricted to planar or toroidal instances.

The approximation algorithms find approximate solutions that intuitively have a large number of "gaps" in them. We show that this phenomenon is *inherent.* Specifically, we define the "contiguous versions" of the optimization problems Π. (Roughly speaking such problems require maximum cardinality solution that is contiguous.) We prove that finding any approximate solution for the contiguous version of the problems Π are NEXPTIME-hard. (See Section 6.)

2.2 Extension to Graph Families

In many instances engineering designs can represent a whole family of designs instead of a single design. The large number of variants of a car is an example of such a family. The infinite family of carry ripple adders of numbers with arbitrary lengths is another example. Today the underlying circuits or graphs of such families is generated with a program called the *cell generator.* As pointed out in [LW93], *"the effective analysis of cell generators is one of the difficult and unsolved problems in CAD."* The question one is interested in solving for such design (graph) families is typically of the form: Do all the graphs produced by the cell generator have a certain property. For example, we might wish to check if each graph produced has a maximum independent set of size at least k. The results presented above can be extended to solve precisely such problems. Specifically, the approximation results presented can be extended to approximate lower/ upper bounds with similar performance guarantees on the best solutions to several optimization problems when the periodic specifications are extended to define families of graphs rather than a single graph. For this we first extend the notion of periodic specifications to define finite number of graphs by allowing a finite number of distinct substitutions. This is done along the lines of definitions for context free graph grammars studied in Lengauer and Wanke [LW93]. Our work is motivated by the remark made by Lengauer and Wanke [LW93], stating that

> *"there is another respect in which our methods fall short of what is needed in practice. They cannot process certain families of graphs that are popular in a wide range of applications such as family of all square grids."*

As noted earlier periodic specifications can represent families of all square grids. Thus our results represent a step towards efficient processing of such regular families of objects.

2.3 Applications

We briefly mention two applications of the ideas developed in the previous Subsection. The details will appear in the complete version of the paper.

1. We can show the our ideas in this paper can be applied to obtain polynomial time approximation algorithms and approximation schemes for NEXPTIME-hard problems specified using a subset of the hierarchical input language (HILL) studied by Bentley et al. [BOW83].

2. We consider two optimization versions of the domino or tiling decision problems studied in [Ha85, Ha86, vEB83]. For these problems we obtain both polynomial time approximations as well as non-approximability results analogous to those in Section 2.1. (For details see Section 10 in the Appendix.)

2.4 Significance of results

1. Previously no hardness results were known for any of the problems Π considered in this paper, when instances are specified using two dimensional periodic specifications.

2. Our results are the first polynomial time approximation algorithms and approximation schemes in the literature for problems specified using the specifications 2(a), 2(b) or 2(c) and also for optimization versions of "hard" domino problems. Our approximation algorithms are based on extensions of the ideas in [MH+94]. The approximation algorithms given in this paper have three desirable features: (i) they are conceptually simple, (ii) they apply to large classes of problems Π, and (iii) they apply to problems specified using any of the periodic specifications considered here.

3. To the best of our knowledge, the efficient approximability of **natural** NEXPTIME-hard problems has not been studied previously. Thus our results provide the first natural problems for which there is a proven exponential (and possibly doubly exponential) gap [5] between the time complexities of finding exact and approximate solutions. Note that it is easy to construct examples of problems, whose decision versions are NEXPTIME-hard, that have polynomial time approximation algorithms with good performance guarantees. [6] Only recently has there been work on the efficient approximability of PSPACE-hard problems. (See [AC94, Co95, CF+93, CF+94, MH+94] and the references therein.)

4. Here and in [MH+95], our NEXPTIME-hardness show that the very regular structure of problems specified periodically by the specifications considered here do **not** suffice to make problems easy. However, the efficient approximation algorithms and schemes developed here show the following:

   > The very regular structures of problem instances specified by the periodic specifications considered here suffice to make approximating many basic optimization problems easy.

   In contrast, approximating many of the optimization problems considered here, when instances are specified using the *small circuit specifications* of [PY86,LB89] is NEXPTIME-hard, by extensions of the arguments in [Ar94]. Thus our results also high-light one important difference between multiple-dimension finite periodic specifications and small circuit specifications of the same objects.

5. Polynomial time approximation algorithms for two general families of problems when instances are specified periodically: namely, optimization versions of generalized CNF satisfiability problems SAT(S) studied by Schaefer [Sc78] (denoted by MAX SAT(S) and matching problems (MAX-H-MATCHING) for any fixed subgraph H. These results are significant for the following reasons.

   (a) Several natural combinatorial optimization problems can be readily expressed as MAX SAT(S) problems, for an appropriately chosen finite set of finite arity Boolean relations

[5] Previous non-approximability results show that many optimization problems are NP-hard or PSPACE-hard to approximate beyond a certain factor. The results presented here show a **provable** exponential gap between approximation and decision problems.

[6] Any MAX-3SAT formula obtained by a reduction from Turing machine computation for an NEXPTIME-complete language can trivially be approximated within ratios arbitrarily close to 1. (by assigning to the variables values that correspond to the all 0 tableau, virtually all clauses are satisfied.) But it is important to note that a similar comment holds for NP-hard problems also.

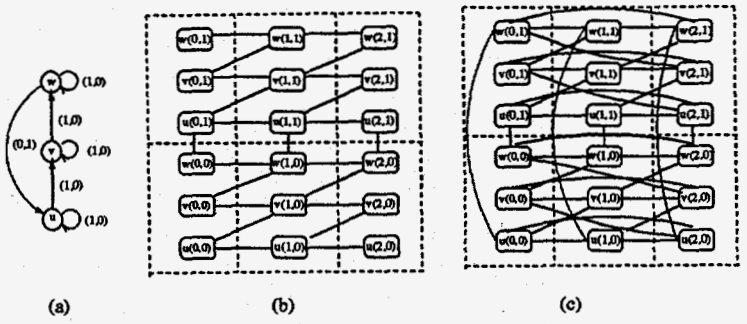

Figure 1: (a)The static graph with 2-dimensional integer vectors associated with each edge. (b) The graph $G^{2,1}$ specified by $\Gamma = (G, 10, 01)$.
(c) The toroidal periodic graph $G^{2,1}$ specified by the same specification. Observe the wrap around edges that are present. Note that we can specify that edges wrap around in only a some of the dimensions. ■

S. It is easily seen that all of the particular MAX SNP-complete problems including the ones studied in [PY91] can be so expressed.

(b) The results for MAX SAT(S) serve as a framework similar to the syntactic class MAX SNP for NP-optimization problems. To see this note that in [HMS94], we showed that every problem in MAX SNP is "equivalent" to a subproblem of some problem MAX SAT(S). The proof can be extended so as to apply to periodically specified problems with "certain" constraints on the relation types allowed in the specification of MAX SNP problems. Thus the results here are an attempt to characterize periodically specified problems that have polynomial time approximation algorithms using a logical/algebraic framework.

Due to lack of space, the remainder of the paper consists of preliminary definitions and selected proof sketches.

3 Preliminary Definitions

In order to understand the results in this paper, it is useful to understand the concept of periodically specified instances. In what follows, we formally define 2-F(B,B)PN-specifications. A brief description of other specifications can be found in relevant sections. We also refer the reader to [HW94, HW95, CM91, Wa93] for formal definitions of the periodic specifications considered in this paper. For the rest of the paper, we use $\mathbf{Z}$ and $\mathbf{N}$ to denote integers and natural numbers respectively.

2-F(B,B)PN-Specified Graphs

Given a labeled finite directed graph $G(V, E)$ (called the *static graph*) such that each edge (u, v) has an associated 2-dimensional integer vector (l, b), the **2-dimensional infinite periodic graph** $G^{\infty}(V', E')$ is defined as follows: $V' = \{v(i,j) \mid v \in V$ and $\quad i, j$ are integers $\}$ $E' = \{(u(i,j), v(i+l, v+b) \mid (l,b)$ is the label associated with $(u,v) \in E \quad \}$. For a non-negative integer vector (m, n), the **2-dimensional finite periodically specified graph** $G^{m,n}$ is the subgraph of G^{∞} induced by the vertices $V_1 = \{v(i,j) \mid v \in V$ and $\quad 0 \leq i \leq m$, $0 \leq j \leq n \}$. For a non-negative integer vector (m, n), the **2-dimensional finite periodic toroidal specified graph** $G^{m,n}$ consists of the vertices $V_1 = \{v(i,j) \mid v \in V$ and $\quad 0 \leq i \leq m$, $0 \leq j \leq n \}$. There is an edge between $u(i_1, j_1)$

and $v(i_2, j_2)$ if and only if there is an edge between u and v in the static graph with the associated integer vector satisfying the condition $i_1 \equiv i_2 \bmod (m+1)$ and $j_1 \equiv j_2 \bmod (n+1)$.

A **2-dimensional finite periodic specification**[7] (denoted by 2-F(B,B)P-specification) of a graph $G^{m,n}$ is a three tuple $\Gamma = (G(V,E), m, n)$, where G is a static graph, and m and n, are represented by binary numerals. (It is important to observe that m and n are specified using binary notation.) The size of the 2-F(B,B)P-specification $\Gamma = (G(V,E), m, n)$ (denoted by $size(\Gamma)$) is given by $size(\Gamma) = |V| + |E| + bits(m) + bits(n)$, where $bits(m)$ and $bits(n)$ are number of bits in m and n. A 2-dimensional periodic specification $\Gamma = (G(V,E), m, n)$, is **narrow** if for all integer vectors $y =< y_1, y_2 >$ associated with the edges of G, $y_i \in \{-1, 0, 1\}$, $1 \leq i \leq 2$. We denote 2 dimensional finite periodic narrow specifications by 2-F(B,B)PN-specifications. Figure 1(a) shows a static graph G and the Figure 1(b) shows the periodically specified graph $G^{2,1}$ specified by $\Gamma = (G, 10, 01)$.[8]

It is sometimes useful to imagine a 2-F(B,B)PN-specified graph as being obtained by placing a copy of the vertex set V at each non-negative integral grid point (x, y), $0 \leq x \leq m$ and $0 \leq y \leq n$, and joining vertices placed at neighboring grid points in a manner specified by the edges of the static graph.

Due to space limitations the definition of 2-F(B,B)PN-specified 3CNF formula is given in the Appendix. We only give an example of 2-F(B,B)PN-specified 3CNF formula to illustrate the concept.
Example 1: Let the set of static variables $U = \{x, y, z\}$. The static clauses C is specified by $C(i,j) = [x(i,j) + y(i,j) + z(i,j)] \wedge [x(i+1,j) + y(i,j) + z(i+1,j)] \wedge [x(i,j+1) + z(i,j)]$. The clauses $C^{1,1}$ specified by the specification $(U, C(i,j), 01, 01)$ is
$[x(0,0)+y(0,0)+z(0,0)] \wedge [x(0,1)+y(0,1)+z(0,1)] \wedge [x(1,0)+y(1,0)+z(1,0)] \wedge [x(1,1)+y(1,1)+z(1,1)] \wedge [x(1,0)+y(0,0)+z(1,0)] \wedge [x(1,1)+y(0,1)+z(1,1)] \wedge [x(0,1)+z(0,0)] \wedge [x(1,1)+z(1,0)]$

4 Approximation Algorithms for 2-F(B,B)PN-specified problems

4.1 Basic Technique

The basic idea behind our approximation algorithms involves the conversion of solutions obtained from a *local* algorithm on small sub-grids to a solution of the *global* problem. The method of partial expansion involves the application of a divide and conquer algorithm iteratively by considering different subsets of the given graph; solving each subset by a local algorithm, constructing a global solution and finally choosing the best solution among these iterations as the solution to Π. The method is similar to the shifting strategy devised by Baker [Ba83], for finding efficient approximation algorithms for several combinatorial problems.

We outline the basic technique by discussing our NC-approximation scheme for the maximum independent set problem (2-F(B,B)PN-IS) . Consider a 2-F(B,B)PN specification of a graph G, and an integer $k > 1$. To begin with, for each i, $0 \leq i \leq k$, we partition the graph G into l disjoint sets $G_1, \cdots G_l$ by removing vertices with horizontal coordinates congruent to $i \mod (k+1)$. For each subgraph G_p, $1 \leq p \leq l$, we find an independent set of size at least $\frac{k}{k+1}$ times the optimal value of the independent set in G_p. The independent set for this partition is just the union of independent sets for each of G_p. By an averaging argument, it follows that the partition which yields the largest solution value contains at least $(\frac{k}{k+1})^2 \cdot OPT(G)$ nodes, where $OPT(G)$ denotes the value of the maximum independent set in G. (For simplicity, we use a symbol to denote a set as well as its cardinality. The intended meaning will be clear from the context.)

[7] In the appendix, we briefly discuss the naming convention used to name the problems considered.

[8] The reader is strongly encouraged to go through the example of periodically specified graph given in Figure 1 and an example of periodically specified formula in the Appendix.

It is important to note that the size of the graph we are dealing with is in general exponential in the size of the specification. Hence a naive application of the above idea will lead to algorithms that take an exponential amount of time. However, as we shall see, the "regular" structure of the graph allows us to solve the problems considered here in time polynomial in the size of the specification.

4.2 Approximation Scheme for 2-F(B,B)PN-MAX-IS

The details of the algorithm for solving 2-F(B,B)PN-MAX-IS problem are given in Figure 1. The proof of correctness of the algorithm and its performance guarantee follows from the following lemmas.

Lemma 4.1 For each iteration of loop 2(c)i, the graphs $G_{i,j}^{i_1,j_k}$, $2 \leq j \leq r-1, 2 \leq j_k \leq s_j - 1$ (i.e. the graphs $G_{i,2}^{i_1,2}$, $G_{i,r-1}^{i_1,3,}\cdots G_{i,r-1}^{i_1,s_j-1}$) are isomorphic.

Proof Idea: Follows from the definition of periodic specification. ■

Let us define two subgraphs obtained in iteration 2.(a).i.A to be in the same equivalence class if they are isomorphic. Then it is easy to see that the maximum independent set problem need only be solved for exactly one member of each equivalence class. As a corollary of the above lemma and by definition of periodic specifications we get that the number of equivalence classes are finite. Furthermore, as result of our partitioning step, it can be shown that the size of the individual pieces is $O(k^2 \cdot |G|)$. These crucial facts allow us to bound the running time of our algorithm by $O(RT_{\Pi}(k^2 \cdot |G|))$.

Lemma 4.2 Each of the subgraphs $G_{i,j}^{i_1,j_1}$ obtained in Step 2.(c).i.B is disjoint.

Proof Sketch: Follows from the property of instances specified by 2-F(B,B)PN specifications; namely a vertex defined at grid point (i,j) is adjacent only to vertices that are defined at grid points (l,m) such that $|l-i|, |m-j| \leq 1$. ■

Next, we prove that the algorithm given above indeed computes a near optimal independent set. That is, given any $k > 1$ the algorithm will compute an independent set whose size is at least $(\frac{k}{k+1})^2$ times that of an optimal independent set.

First, we prove that of all the different iterations for i, at least one iteration has the property that the number of nodes that are **not** considered in the independent set computation is a **small fraction** of an optimal independent set.

Recall that for each i we did not consider the vertices which were placed at lattice points with horizontal coordinates $j_1, j_2 \cdots j_p$ such that $j_l \equiv i \ mod(k+1)$, $1 \leq l \leq p$. Let $S_0, S_1, \cdots S_l$ be the set of vertices which were **not** considered for each iteration i. Let $IS_{opt}(S_i)$ denote the vertices in the set S_i which were chosen in the optimal independent set $OPT(G)$.

Lemma 4.3

$$\max_{0 \leq i \leq k} |OPT(G_i)| \geq (\frac{k}{(k+1}) |OPT(G)|$$

Proof: The proof follows by observing that the following equations hold:

$$0 \leq i,j \leq l, \ i \neq j, \ S_i \cap S_j = \phi; \quad \cup_{t=o}^{t=l} S_t = V(G). \ ■$$

Algorithm ALG-2-FPN-MAX-IS:

- *Input:* An instance (G, m, n) of a periodic graph $G^{m,n}$ and an $\epsilon > 0$
- *Output:* A periodic specification of a near optimal independent set in $G^{m,n}$ whose size is at least $(1-\epsilon)^2 \cdot FBEST \cdot OPT$ where OPT is the size the maximum independent set in $G^{m,n}$ and $FBEST$ denotes the best possible factor achievable by any polynomial time approximation algorithm for the maximum independent set problem specified using a standard specification.

1. Let $k = \lceil 1/\epsilon \rceil - 1$.
2. **For** each i, $0 \leq i \leq k$ **do**
   (a) Partition the graph into r_i disjoint sets $G_{i,1} \cdots G_{i,r_i}$ by removing all the vertices at grid points with X-coordinate congruent to i mod $(k+1)$.
   (b) $G_i = \bigcup_{1 \leq j \leq r} G_{i,j}$
   (c) **For** each j, $1 \leq j \leq r_i$ **do**
      i. **For** each i_1, $0 \leq i_1 \leq k$ **do**
         A. Partition the graph $G_{i,j}$ into s_j disjoint sets $G_{i,j}^{i_1,1} \cdots G_{i,j}^{i_1,s_j}$ by removing vertices at grid points with Y-coordinate congruent to i_1 mod $(k+1)$.
         B. $G_{i,j}^{i_1} = \bigcup_{1 \leq j_1 \leq s_j} G_{i,j}^{i_1,j_1}$
         C. For each $G_{i,j}^{i_1,j_1}$, $1 \leq j_1 \leq s_j$ compute the optimal (near optimal) value of the independent set denoted by $IS(G_{i,j}^{i_1,j_1})$.
            Remark: This can be done by running the algorithm on just three graphs namely; $G_{i,j}^{i_1,1}$, $G_{i,j}^{i_1,2}$ and $G_{i,j}^{i_1,s_j}$
         D. $IS(G_{i,j}^{i_1}) = \bigcup_{1 \leq j_1 \leq s_j} IS(G_{i,j}^{i_1,j_1})$
   (d) $IS(G_{i,j}) = \max_{0 \leq i_1 \leq k} IS(G_{i,j}^{i_1})$
   (e) $IS(G_i) = \bigcup_{1 \leq j \leq r_i} IS(G_{i,j})$
3. $IS(G) = \max_{0 \leq i \leq k} IS(G_i)$

Figure 1: Details of the algorithm to solve an instance of 2-F(B,B)PN-MAX-IS.

The proofs of the next two theorems follow by an averaging argument. We omit the proofs due to the lack of space.

Theorem 4.4 $|IS(G_{i,j})| \geq (\frac{k}{k+1}) \cdot FBEST \cdot |OPT(G_{i,j})|$.

Theorem 4.5 $|IS(G)| \geq (\frac{k}{k+1})^2 \cdot FBEST \cdot |OPT(G)|$. Here $FBEST$ denotes the performance guarantee of the best algorithm known to solve the independent set problem.

5 Hardness of 2-F(B,B)PN-3SAT

Theorem 5.1 2-F(B,B)PN-3SAT is NEXPTIME-complete.

Proof: Membership in NEXPTIME follows easily by observing that the size of the expanded formula is $2^{c(|U+C(i,j,i+1,j+1)|+bits(m)+bits(n))}$, where $\Gamma = (U, C(i,j), m, n)$, is the specification of $F^{m,n}$.

Hence a NEXPTIME bounded TM can guess an assignment to the variables and then verify in DEXPTIME that the assignment satisfies all the clauses.

Next, we discuss the reduction which shows the NEXPTIME-hardness of the problem. It is worth pointing out the basic technique used behind the reductions. Since the static formula associated with 2-F(B,B)PN-3SAT instance is the same for each time period, it is not possible to write a 3CNF formula which says that the machine has the correct starting ID. This makes the task of constructing the 3SAT instance more difficult. In order to overcome this difficulty, our reduction consists of two phases. In the first phase, we start with a given Turing machine ϕ with input $x = (x_1, \ldots, x_p)$ and construct a new Turing machine ϕ_x which simulates ϕ on x and has the following additional properties that

1. If Turing machine ϕ does not accept x, then every possible computation of ϕ_x halts within $2^{c_0 p}$ moves, else

2. If Turing machine ϕ accepts x, then ϕ_x has a cycling computation, where the length of an ID never exceeds $2^{d_0 p}$, for some given d_0.

The second phase consists of constructing an instance $(U_x(t,y), G_x(t,y,t+1,y+1), m, n))$ of 2-FPN-3SAT by a polynomial time reduction from ϕ_x. Now we know that each ID of the Turing machine ϕ_x is of length $2^{d_0 p}+1$. From Property 2 above, we need to consider only $2^{d_0 p}$ different ID's for our reduction. In order to understand the construction imagine each ID of the Turing machine ϕ_x being placed vertically in the plane. Two consecutive ID's of ϕ_x are placed vertically next to each other. For the sake of exposition we will refer to the X-axis as the *time line*. In the following discussion, each grid point is referred to as (t,y). We now define the set of variables $U_x(t,y)$ and their intended meaning. $U_x(t,y)$ consists of the following three different types of variables. (i) $TAPE \subset U_x(t,y)$, such that $TAPE(t,y)$ encodes the y^{th} symbol in the t^{th} ID. $TAPE(t,y)$ takes values from the set $\{\#\} \cup \Gamma \cup (Q \times \Gamma)$, where Γ denotes the tape symbols and Q denotes the set of states of ϕ_x. The number of variables needed to encode $TAPE(t,y)$ depends only on the machine ϕ_x. (ii) In order to simulate the behavior of ϕ_x properly we need to have two set of counter variables; c_y and c_t. The counter c_y keeps track of the particular tape cell in a given ID. Let $q = d_0 p$. The counter can be simulated by means of $(d_0 p + 1)$ Boolean variables $tc_q, tc_{q-1}, \ldots tc_0$. tc_0 represents the least significant bit and tc_q represents the most significant bit. The counter c_t keeps track of the number of ID's. The counter c_t can be simulated by means of $(d_0 p + 1)$ Boolean variables. We use Boolean variables $yc_0, yc_1, yc_2, \ldots, yc_q$ to simulate the counter c_y. (iii) Auxiliary variables for making the resulting static formula narrow and in the 3CNF form.

The initial ID is of the form $\#(q_0, x_1) \ldots x_n B^{2^{d_0 p} - p}$, where B denotes a blank. The static formula CNF formula $G_x(t,y)$ is given by $G_x(t,y) = f_1(t,y,t+1,y+1) \wedge f_2(t,y,t+1,y+1) \wedge f_3(t,y,t=1,y+1)$. The counter updating formula f_1 is designed for maintaining an implicit counter. The implicit initialization formula f_2 serves as a way to implicitly initialize the clauses to reflect that the machine starts out right whenever the counters are reset to 0. The initialization condition say that if both the counter values are 0, then we have # as the tape symbol and so on. The consistency checking formula f_3 ensures the consistency of the tape symbols, i.e. that the contents of the tape cells i, $i+1$ and $i+2$ in ID_t determine the contents of the tape cells i, $i+1$ and $i+2$ in ID_{t+1}. The details of each of these formulas is given in the Appendix. The expanded finite periodic 3SAT instance is $\bigwedge_{y=0,t=0}^{t=m,y=n} G_x(t,y,t+1,y+1)$, where $m = 2^{2d_0 p}$ and $n = 2^{2d_0 p}$.

We now prove the correctness of our reduction. If the Turing machine ϕ accepts x then we know that ϕ_x has a cycling computation. Hence by setting the counters $c_t(0,0) = c_y(0,0) = 0$ we get that the first column of the grid contains the right initial ID. From then on, the consistency

conditions ensures that the formula $\bigwedge_{t=0,y=0}^{t=m,y=n} G_x(t,y,t+1,y+1)$ is satisfied. Conversely, assume that the formula is satisfiable. Since m and n are suitably large integers, it is guaranteed that the following two conditions hold:

1. Since n is large enough, the simulation must be carried out for enough steps so that the Turing machine ϕ_x goes through the sequence $c_t = 0, c_t = 1, c_t = 2, \cdots c_t = 2^{dop}$. This implies that the formulas $f_2(t,y)$ and $f_3(t,y)$ would be true from the time when the value of $c_t = 0$.

2. Similarly, since m is large enough, the grid is sufficiently long in the Y-direction so that the counter value c_y goes through a sequence of values $c_y = 0, c_y = 1, c_y = 2, \cdots c_y = 2^{dop}$. This implies that the first part of the implication in f_2 is true and from then on, it is ensured that the TM ϕ_x goes through the simulation correctly.

The above two conditions imply that if the formula $\bigwedge_{t=0,y=0}^{t=m,y=n} G_x(t,y,t+1,y+1)$ is satisfied then the Turing machine ϕ accepts x. ■

6 Contiguous Versions of the problems

Definition 6.1 CONT-2-F(B,B)PN-MAX-3SAT: Given an instance 3CNF formula $F^{m,n}(U^{m,n}, C^{m,n})$ specified by $\Gamma = (U, C(i,j), m, n)$ and integers $k, l \geq 1$, do there exist integers i_1, i_2, j_1, j_2 such that
(i) $1 \leq i_1 \leq i_2 \leq M$
(ii) $1 \leq j_1 \leq j_2 \leq N$, such that
$i_2 - i_1 = k-1$, and $j_2 - j_1 = l-1$, and the subformula $\bigwedge_{i_1 \leq i \leq i_2, j_1 \leq j \leq j_2} C(i,j,i+1,j+1)$ is satisfiable.

The above definition can be extended to define other CONT-2-F(B,B)PN-Max-Sat(S) problems and to graph theoretic problems. In the latter case, we insist that the the projection of the solution on some connected contiguous subgraph be maximized/minimized.

Theorem 6.2
1. If k, l are specified using Unary notation then CONT-2-F(B,B)PN-MAX-3SAT is NP-complete.

2. If one of k, l is specified using Binary notation and the other is specified using Unary notation then CONT-2-F(B,B)PN-MAX-3SAT is PSPACE-complete.

3. If k, l are specified using Binary notation then CONT-2-F(B,B)PN-MAX-3SAT is NEXPTIME-complete.

Proof Sketch for Part (3): Membership in NEXPTIME holds for *arbitrary* instances of these problems. The indicated hardness results follows by reduction from the acceptance problems for a 2^n time bounded NDTM M Starting from M with input x we first construct a Turing machine M_1 that cycles if and only if M accepts x. Now we do a standard reduction from M_1 to the problem say 1-F(B,B)PN-3SAT. If M accepts x, then M_1 cycles and 1-F(B,B)PN-3SAT has a solution in which clauses in a $m^k \times n^k$ area, for suitably large k, can be simultaneously satisfied. If M does not accept x then, M_1 does not cycle on x, and hence the maximum contiguous area of clauses that can be simultaneously satisfied is no more than $O(m) \times O(n)$. ■

The extension of the above proof also shows that finding any ϵ approximation for these problems is NP-, PSPACE- and NEXPTIME-complete respectively. Exactly analogous hardness results hold for the contiguous versions of many of the problems Π considered here.

We can show exactly analogous results hold for natural problems in MAX SNP such as MAX 3SAT. Thus we can show that all approximations for the maximum number of *simultaneously consecutive satisfiable* clauses of a 3CNF formula F are NP-hard. Note that this is a natural definitions of the contiguous problems for standardly specified formulas.

Acknowledgments

We thank Sanjeev Arora, Pascal Brisset, José Balcázar, Anne Condon, Ashish Naik, V. Radhakrishnan, S. S. Ravi and Egon Wanke for several useful discussions and pointers to literature throughout the course of writing this paper.

References

[AC94] S. Agarwal and A. Condon, "On Approximation Algorithms for Hierarchical MAX-SAT," *Proc. of the 10th IEEE Conference on Structures in Complexity Theory,* July, 1995.

[Ar94] S. Arora, "Proof Verification and Hardness of Approximation Problems", *Ph.D. Thesis*, Department of Computer Science, University of California, Berkeley, 1994.

[AL+92] S. Arora, C. Lund, R. Motwani, M. Sudan and M Szegedy, "Proof Verification and Hardness of Approximation Problems", *Proc. 33rd IEEE Symposium on Foundations of Computer Science (FOCS)*, 1992, pp. 14-23.

[Ba83] B.S. Baker, "Approximation Algorithms for NP-complete Problems on Planar Graphs," *24th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1983, pp 265-273. (Journal version in *J. ACM,* Vol. 41, No. 1, 1994, pp. 153-180.)

[Br95] P. Brisset, "Algorithms for the PLacement and Routing of FPGA's," Technical Report, Ecole Polytechnic April 1995.

[BOW83] J.L. Bentley, T. Ottmann, P. Widmayer, "The Complexity of Manipulating Hierarchically Defined set of Intervals," *Advances in Computing Research, ed. F.P. Preparata* Vol. 1, (1983), pp. 127-158.

[Bu62] J.R. Büchi, "Turing Machines and the Entscheidungsproblem," *Math. Ann.* 148, 1962, pp. 201-213.

[CS81] P.R. Cappello and K. Steiglitz, "Digital Signal Processing Applications of Systolic Algorithms," *Proc. CMU Conference on VLSI Systems and Computations,* H.T Kung, B. Sproull and G Steele eds. 1981.

[CM93] E. Cohen and N. Megiddo, "Strongly Polynomial-time and NC Algorithms for Detecting Cycles in Dynamic Graphs," *Journal of the ACM (J. ACM)* Vol. 40, No. 4, September 1993, pp. 791-830.

[CM91] E. Cohen and N. Megiddo, "Recognizing Properties of Periodic graphs", *Applied Geometry and Discrete Mathematics, The Victor Klee Festschrift* Vol. 4, P. Gritzmann and B. Strumfels, eds., ACM, New York, 1991, pp. 135-146.

[Co95] A. Condon, "Approximate Solutions to Problems in PSPACE," *SIGACT News: Introduction to Complexity Theory Column 9, Guest Column*, July, 1995.

[CF+93] A. Condon, J. Feigenbaum, C. Lund and P. Shor, "Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions", in *Proc. 25th ACM Symposium on Theory of Computing (STOC)*, 1993, pp. 305-313.

[CF+94] A. Condon, J. Feigenbaum, C. Lund and P. Shor, "Random Debaters and the Hardness of Approximating Stochastic functions for PSPACE-Hard Functions," *Proc. 9th IEEE Annual Conference on Structure in Complexity Theory*, June 1994, pp. 280-293.

[GJ79] M.R. Garey and D.S. Johnson, *Computers and Intractability. A Guide to the Theory of NP-Completeness*, Freeman, San Francisco CA, 1979.

[GW94] M.X. Goemans and D.P. Williamson ".878 Approximation Algorithms for MAX CUT and MAX 2SAT," *Proc. 26th Annual ACM Symposium on Theory of Computing, (STOC)*, May 1994, pp. 422-431.

[Ha86] D. Harel, "Effective Transformations on Infinite Trees, with Applications to High Undecidability, Dominoes, and Fairness," *Journal of the ACM (J. ACM)* , Vol. 33, No. 1, January, 1986, pp. 224-248.

[Ha85] D. Harel, "Recurring Dominos: Making Highly Undecidable Highly Understandable," *Annals of Discrete Math.* , Vol. 24, 1985, pp. 51-72. A preliminary version of the paper appeared in *Proc. Conference on Foundations of Computation Theory,* LNCS Vol. 158, pp. 177-194.

[HLW92] F. Höfting, T. Lengauer and E. Wanke, "Processing of Hierarchically Defined Graphs and Graph Families," in *Data Structures and Efficient Algorithms* (Final Report on the DFG Special Joint Initiative), Springer-Verlag, LNCS 594, 1992, pp. 44-69.

[HW95] F. Höfting and E. Wanke, "Minimum Cost Paths in Periodic Graphs," *SIAM J. on Computing*, Vol. 24, No. 5, Oct. 1995, pp. 1051-1067

[HW94] F. Höfting and E. Wanke, "Polynomial Time Analysis of Toroidal Periodic Graphs," *Proc. of 22nd International Colloquium on Automata, Programming and Languages,* 1994.

[HMS94] H. B. Hunt III, M. V. Marathe and R. E. Stearns, "Generalized CNF Satisfiability Problems and Non-Efficient Approximability," *Proc. 9th ACM Conf. on Structure in Complexity Theory*, June-July 1994, pp. 356-366.

[IS86] K. Iwano and K. Steiglitz, "Optimization of one-bit full adders embedded in regular structures," *IEEE Transactions on Acoustics, Speech and Signal Processing,* 1986.

[IS87] K. Iwano and K. Steiglitz, "Testing for Cycles in Infinite Graphs with Periodic Structure," *Proc. 19th Annual ACM Symposium on Theory of Computing, (STOC)*, 1987, pp. 46-53.

[IS88] K. Iwano and K. Steiglitz, "Planarity Testing of Doubly Connected Periodic Infinite Graphs," *Networks*, No. 18, 1988, pp. 205-222.

[KMW67] R.M. Karp, R.E. Miller and S. Winograd, "The Organization of Computations for Uniform Recurrence Equations," *Journal of the ACM (J. ACM)*, Vol. 14, No. 3, 1967, pp. 563-590.

[KO91] M. Kodialam and J.B. Orlin, "Recognizing Strong Connectivity in Periodic graphs and its relation to integer programming," *Proc. 2nd ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1991, pp. 131-135.

[KS88] K. R. Kosaraju and G.F. Sullivan, "Detecting Cycles in Dynamic Graphs in Polynomial Time," *Proc. 27th IEEE Symposium on Foundations of Computer Science (FOCS)*, 1988, pp. 398-406.

[LW93] T. Lengauer and E. Wanke, "Efficient Decision Procedures for Graph Properties on Context-Free Graph Languages ," *Journal of the ACM (J. ACM)* , Vol. 40, No. 2, 1993, pp. 368-393.

[Le86] T. Lengauer, "Exploiting Hierarchy in VLSI Design," *Proc. AWOC '86*, Springer-Verlag, LNCS 227, 1986, pp. 180-193.

[MH+94] M.V. Marathe, H.B. Hunt III, R.E. Stearns and V. Radhakrishnan, "Approximation Schemes for PSPACE-Complete Problems for Succinct Specifications," *Proc. 26th ACM Annual Symposium on Theory of Computing (STOC)*, 1994, pp. 468-477.

[MH+95] M.V. Marathe, H.B. Hunt III, D. J. Rosenkrantz, R.E. Stearns and V. Radhakrishnan, "Periodically Specified Satisfiability Problems with Applications: An Alternative to Domino Problems," submitted, November, 1995.

[Or82a] J.B. Orlin, "The Complexity of Dynamic/Periodic Languages and Optimization Problems," Sloan W.P. No. 1679-86 July 1985, Working paper, Alfred P. Sloan School of Management, MIT, Cambridge, MA 02139. A Preliminary version of the paper appears in *Proc. 13th ACM Annual Symposium on Theory of Computing (STOC)*, 1978, pp. 218-227.

[Or84b] J.B. Orlin, "Some Problems on Dynamic/Periodic Graphs," *Progress in Combinatorial Optimization*, Academic Press, May 1984, pp. 273-293.

[PY86] C. Papadimitriou and M. Yannakakis, "A note on Succinct Representation of Graphs," *Information and Computation* No. 71, 1986, pp. 181-185.

[PY91] C. Papadimitriou and M. Yannakakis, "Optimization, Approximation and Complexity Classes," *Journal of Computer and System Sciences (JCSS)*, No. 43, 1991, pp. 425-440.

[Sc78] T. Schaefer, "The Complexity of Satisfiability Problems," *Proc. 10th ACM Symposium on Theory of Computing (STOC)*, 1978, pp. 216-226.

[Su95] K. Sutner, " On the Computational Complexity of Finite Cellular Automata," *Journal of Computer and System Sciences (JCSS)*, No. 50, 1995, pp. 87-97.

[vEB83] van Emde Boas, "Dominoes Forever," *Proc. 1st GTI Workshop*, Paderborn, 1983, pp. 76-95.

[Wa93] E. Wanke, "Paths and Cycles in Finite Periodic Graphs," *Proc. 20th Symposium on Math. Foundations of Computer Science (MFCS)*, LNCS 711, Springer-Verlag, 1993, pp. 751-760.

[Wo84] S. Wolfram *Theory and Applications of Cellular Automata*, World Scientific, Singapore, 1987.

Appendix

7 Additional Definitions

Generalized satisfiability problems

The generalized satisfiability problems SAT(S) were first introduced in Schaefer [Sc78]. This concept is also discussed in [MH+95]. The reader is assumed to be familiar with the problem 3SAT. Many other types of satisfiability problems have also been studied in the literature.

Definition 7.1 (Schaefer [Sc78])
Let $S = \{R_1, R_2, \cdots, R_m\}$ be a finite set of finite arity boolean relations. (A boolean relation is defined to be any subset of $\{0,1\}^p$ for some integer $p \geq 1$. The integer p is called the **rank** of the relation.) An S-**formula** is a conjunction of clauses each of the form $\hat{R}_i(\xi_1, \xi_2, \cdots)$, where $\xi_1, \xi_2, \cdots$ are distinct, unnegated variables whose number matches the rank of $R_i, i \in \{1, \cdots m\}$ and $\hat{R}_i$ is the relation symbol representing the relation R_i. The S-*satisfiability problem* is the problem of deciding whether a given S-formula is satisfiable.
Given a S-formula F, the problem MAX SAT(S) is to determine a (0,1) assignment to the variables of F so as to maximize the number of clauses that can be satisfied.

Definition 7.2 Let S be a finite set of finite arity boolean relations and let f be a SAT(S) formula with set of variables V and set of clauses C.

1. The *bipartite graph* of f, denoted by $\mathbf{BG}(f)$, is the bipartite graph $(V \cup C, E)$, where $e = (c, v) \in E$ if and only if the variable v occurs in the clause c.
2. f is said to be *planar* if and only if the graph $\mathbf{BG}(f)$ is planar.

7.1 Periodically specified satisfiability problems

For the rest of the paper, let $\mathbf{Z}$ and $\mathbf{N}$ denote the set of integers and natural numbers respectively.

Let $U = \{u_1, \ldots, u_n\}$ be a finite set of variables (referred to as static variables). A literal of U is an element of the set $U_1 = \{u_1, \ldots, u_n, \overline{u_1}, \ldots, \overline{u_n}\}$. Define the following sets.

$$\forall t \in \mathbf{Z},\ U_1(t,y) = \{u_k(t,y) \mid 1 \leq k \leq n,\}$$

$$U^{\mathbf{Z}} = \bigcup_{t,y \in \mathbf{Z}} U_1(t,y)$$

$$U^{\mathbf{N}} = \bigcup_{t,y \in \mathbf{N}} U_1(t,y)$$

$$U^{m,n} = \bigcup_{t,y \in \mathbf{N},\ t \leq m,\ y \leq n} U_1(t,y)$$

(In our proofs, variable $u_k(t,y)$ denotes the variable u_k at grid point (t,y).)

If w is a literal of U, then $w(t,y)$, is a literal of $U^{\mathbf{Z},\mathbf{Z}}$. Let $C(t,y)$ be a parameterized conjunction of 3 literal clauses such that each clause in $C(t,y,t+1,y+1)$ consists of variables $u_k(t,y), u_k(t+1,y+1), u_k(t,y+1), u_k(t+1,y)$ with the constraint that at least one variable is of the form $u_k(t,y)$. We refer to the clauses $C(t,y,t+1,y+1)$ as *static narrow clauses*. Let $C^{\mathbf{Z},\mathbf{Z}} = \bigwedge_{t,y \in \mathbf{Z}} C(t,y,t+1,y+1)$ and $C^{\mathbf{N},\mathbf{N}} = \bigwedge_{t,y \in \mathbf{N}} C(t,y,t+1,y+1)$. Given $U^{m,n}$ and $C^{\mathbf{Z},\mathbf{Z}}$, let

$$C^{m,n} = \{(w_1(i_1) \vee w_2(i_2) \vee w_3(i_3)) \mid (w_1(i_1) \vee w_2(i_2) \vee w_3(i_3)) \in C^{\mathbf{Z},\mathbf{Z}}\ \&\ w_1(i_1), w_2(i_2), w_3(i_3) \in U^{m,n}\}$$

Definition 7.3 A 2-dimensional two way infinite (finite) periodic narrow specification (denoted by 1-PN (FPN)-specification) of a 3CNF formula $F^{\mathbf{Z},\mathbf{Z}}(U^{\mathbf{Z},\mathbf{Z}}, C^{\mathbf{Z},\mathbf{Z}})$ ($F^{m,n}(U^{m,n}, C^{m,n})$) is given by ($\Gamma = (U, C(t, y, t+1, y+1))$, ($\Gamma = (U, C(t, y, t+1, y+1), m, n)$), where, U is a finite set of variables, $C(t, y, t+1, y+1)$ is a collection of static narrow 3 literal clauses. (In case of finite specifications m, n are non-negative integers specified in binary.) The size of the specification denoted by $size(\Gamma) = |U| + |C(t, y, t+1, y+1)|$. (In case of finite specifications $size(\Gamma) = |U| + |C(t, y, t+1, y+1)| + bits(m) + bits(n)$, where $bits(m)$ and $bits(n)$ denote the number of bits used to represent m and n respectively.)

The problem 1-P(Z, Z)N-3SAT (problem 3SAT when instances are specified using 1-FPN-specifications) is the problem of determining if a 3CNF formula $F^{\mathbf{Z},\mathbf{Z}}(U^{\mathbf{Z},\mathbf{Z}}, C^{\mathbf{Z},\mathbf{Z}})$ specified by $\Gamma = (U, C(t, y))$ is satisfiable.

Similarly, the problem 1-F(B,B)PN-3SAT (problem 3SAT specified using 1-FPN-specifications) is the problem of determining if a 3CNF formula $F^{m,n}(U^{m,n}, C^{m,n})$ specified by $\Gamma = (U, C(t, y), m, n)$ is satisfiable.

As in the case of periodically specified graphs, it is useful to imagine a narrow periodically specified formula $G^{\mathbf{Z},\mathbf{Z}}$ as being obtained by placing a copy of the variable set U at each grid point in the X-Y plane (or the time line). Furthermore, assume that the clauses $C(t, y, t+1, y+1)$ are placed at grid point (t, y). With this notation, we can now refer to variables $U(t, y)$, as the set of variables at grid point $(t, y,)$ and the clauses $C(t, y, t+1, y+1)$ as the set of clauses at grid point (t, y).

For each finite set S of finite arity Boolean relations, it is straightforward to extend the above definition so as to define the problem 2-F(B,B)PN-SAT(S) and hence we omit these definitions. Observe that 2-F(B,B)PN-specified graphs or formulas can be exponentially larger than their input specifications. The above definition can be applied to define satisfiability problems for variants of periodic specification. We do this in the next paragraph. We can now define various other satisfiability problems.

7.2 Note on Naming Convention

Since we have a large number of parameters, it is necessary to state the notation used throughout this paper for naming problems. The naming convention follows the one given in [MH+95]. We use F and I to denote *finite* or *infinite* graphs respectively. Observe that while this is the property of the expanded object, we choose to use this as a way to classify the specification itself. The symbols U, B in the brackets following F specify, whether the finite bounds are specified in unary or binary notation. The symbols **N**, **Z** following I specify whether the graph is infinite in one direction or both the directions. We have already explained the concept of narrow and wide specifications. We use N and W to denote **narrow** and **wide** specifications respectively. *Dimensions of the expanded Graph:* $\{1, 2, \ldots d\}$ denote the dimensions in which the static graph is translated. Some instances of problems arising in practice have a periodic specification of the graph or a formula along with explicit initial and final conditions. We call such periodic specifications as periodic specifications with boundary conditions (BC).

7.3 Meaning of Approximation Algorithms for Periodically Specified Problems

It is important to understand what we mean by a *polynomial time approximation algorithm* for a problem Π, when Π's instances are specified by 2-F(B,B)PN-specifications. We illustrate this by the following example:

Example: Consider the maximum independent set problem, when graphs are specified be 2-F(B,B)PN-specifications. We provide efficient algorithms for the following versions of the Approximate Maximum Independent Set Problem:
(1) **The Approximation Problem:** Compute the size of a near-maximum independent set in G.
(2) **The Query Problem:** Given any vertex v of G determine whether v belongs to the approximate independent set so computed.
(3) **The Construction Problem:** Output a 2-F(B,B)PN specification of the set of vertices in the approximate independent set.
(4) **The Output Problem:** Output the approximate independent set computed.

Our algorithms for (1), (2) and (3) above run in time **polynomial in the size of the 2-F(B,B)PN-specification** rather than the size of the graph obtained by expanding the specification. Our algorithm for (4) runs in time linear in the size of the expanded graph but uses space which is linear in the size of the periodic specification. ■

We provide results with similar time bounds for the Approximation, Query, Construction, and Output problems associated with the problems Π in Table 1, when instance are specified by 2-F(B,B)PN-, 2-F(B,B)PN(BC)- or 2F(B,B)PTN-specifications.

8 Details of the Proof for Theorem 5.1

Counter Updating: Formula f_1

$f_1 \equiv f_1^1 \wedge f_1^2 \wedge f_1^3 \wedge f_1^4$, where each f_1^i, $1 \leq i \leq 3$ is given as follows:
$f_1^1 \equiv [c_t(t+1,y) = (c_t(t,y)+1) \pmod{2^{d_0 p}+1},]$ $f_1^2 \equiv [0 \leq c_y(t,y) < 2^{d_0 p} \Rightarrow c_t(t,y+1) = c_t(t,y)]$
$f_1^3 \equiv [c_y(t,y+1) = (c_y(t,y)+1) \pmod{2^{d_0 p}+1}]$ $f_1^4 \equiv [0 \leq c_t(t,y) < 2^{d_0 p} \Rightarrow c_y(t+1,y) = c_y(t,y)]$

f_1^1 says that the value of the counter c_t at grid point $(t+1,y)$ is 1 more than the value of the counter at the grid point (t,y). Moreover, the counter is reset after every $2^{d_0 p}+1$ time units. f_1^2 says that the counter value for a given value of t is the same for all y. Conjuncts f_1^3 and f_1^4 describe the desired properties of the counter c_y in a manner similar to f_1^1 and f_1^2.

Implicit Initialization: Formula f_2

$[(c_y(t,y) = 0) \wedge (c_t(t,y) = 0) \Rightarrow TAPE(t,y) = \#] \wedge [(c_y(t,y) = 1) \wedge (c_t(t,y) = 0) \Rightarrow TAPE(t,y) = (q_0 x_1)] \bigwedge$

$\vdots$

$[(c_y(t,y) = n) \wedge (c_t(t,y) = 0) \Rightarrow TAPE(t,y) = x_n] \bigwedge [(n+1 \leq c_y(t,y) \leq 2^n) \wedge (c_t(t,y) = 0) \Rightarrow TAPE(t,y) = B]$

f_2 can be thought of as a way to implicitly initialize the clauses to reflect that the machine starts out right whenever the counters are reset to 0. The initialization condition say that if both the counter values are 0, then we have # as the tape symbol and so on.

Consistency Checking: Formula f_3

$$(0 \leq c_y(t,y) \leq 2^{d_0 p}) \wedge (2^n + 1 \leq c_t(t,y) \leq 2^{d_0 p}) \Rightarrow$$

$$Consistent(TAPE(t,y), TAPE(t,y+1), TAPE(t,y+2), TAPE(t+1,y), TAPE(t+1,y+1), TAPE(t+1,y+2))$$

f_3 ensures the consistency of the tape symbols, i.e. that the contents of the tape cells i, $i+1$ and $i+2$ in ID_t determine the contents of the tape cells i, $i+1$ and $i+2$ in ID_{t+1}. The Consistency function of course depends on the state transition relation.

Although, the above formula contains clauses which are not narrow, it is easy to transform them to a narrow set of clauses by adding temporary variables. We omit the details in this abstract. Now, it is easy to see that these equations can be transformed into an equivalent narrow 3CNF formula $G_x(t, y)$ whose size is polynomial is in n, (recall that $n = |x|$.)

9 Approximation Algotihms for Other Specifications

9.1 Toroidal Specifications

We now briefly discuss how the ideas of Section 4 can be refined to obtain a polynomial time approximation schemes for problems specified by 2-F(B,B)PTN-specifications. Again, we consider the MAX-IS problem for purposes of illustrations. The basic algorithm for solving 2-F(B,B)PTN-MAX-IS remains the same as given in Figure 1. Therefore, we only point out the essential differences. In Step 2(a) of the algorithm, we may get $(r-1)$ disjoint pieces instead of r. Specifically the pieces $G_{i,1}$ and $G_{i,r}$ are not necessarily disjoint due to the wrap around edges in the Y-direction. Moreover these are the only two subgraphs that are connected due to the fact that the specification is *narrow.* Also observe that since we remove vertices at grid points that are distance $(k+1)$ apart, we have that the "width" (number of rows) of the connected graph $G_{i,1} \cup G_{i,r}$ is no more than $2(k+1)$. Next, consider Step 2(c)iA of the algorithm outlined in Figure 1. Again for a given value of i and j, the graphs $G_{i,j}^{i_1,1}$ and $G_{i,j}^{i_1,s_j}$ are possibly connected due to the wrap around edges in X-direction. Therefore, we get $s_j - 1$ connected subgraphs to work with. Moreover, the size of each subgraph can be as big as $2(k+1) \times 2(k+1)$ in terms of the numbers of grid points. The rest of the analysis remains the same as in the case of 2-F(B,B)PN-MAX-IS.

9.2 Extension to d-dimensional specifications

By applying the shifting strategy once for each dimension. we can obtain similar results for d-dimensional periodic narrow specifications, for any fixed d. The performance guarantee grows exponentially in d. Thus we have the following theorem.

Theorem 9.1 For all $d > 0$, $\epsilon > 0$, each d-dimensional periodic specified problems Π in Table 1 has a polynomial time approximation algorithm with performance guarantee $(1+\epsilon)^d FBEST_\Pi$.

10 Domino Problems

Domino (or Tiling) problems were introduced in the early 1960's have been studied extensively in the literature. (see [vEB83, Ha85, Ha86, Bu62] and the references therein.) They have proven useful in obtaining hardness results, especially for decision problems for various logical theories. Usually a *domino system* is described as a finite set of *tiles* or *dominoes*, every tile being a unit square with a fixed orientation and colored edges; we have an unlimited supply of copies of every tile. A *domino problem asks* whether it is possible to tile a prescribed subset of the Cartesian plane with elements of a given domino system, such that adjacent tiles have matching colors on their common edge and perhaps with certain constraints on the tiles that are allowed on certain specific places (e.g. the origin).

We consider the following two optimization versions of the domino or tiling decision problems studied in [Ha85, Ha86, vEB83].

1. Maximum Area Tiled (MAT): Given a fixed collection of dominoes of size 1×1, the aim is to tile the plane so as to maximize the area covered by the tiles or the dominoes. (Observe that we do not demand the area be contiguous.)

2. Contiguous Maximum Area Tiled (CMAT): Given a fixed collection of dominoes of size 1×1, the aim is to tile the plane so as to maximize the contiguous area covered by the tiles or the dominoes.

When the bounds on the area to be tiled are specified in unary, we show that

1. the problem MAT has a polynomial approximation scheme, and that
2. there exists $\epsilon > 0$ such that approximating the problem CMAT within performance guarantee of n^ϵ is NP-hard.

The contrast between the complexities of approximating the problems MAT and CMAT becomes more pronounced when bounds on the area to be tiles are specified in binary. In this case, we show that

1. the problem MAT still has a polynomial approximation scheme and that
2. there exists $\epsilon > 0$ such that the approximating the problem CMAT with performance guarantee of n^ϵ is NEXPTIME-hard; thus CMAT **provably** does not have a polynomial time approximation algorithm with "good" performance guarantee.

Non-approximability results similar to those obtained for CMAT can also be obtained for periodically specified graph and satisfiability problems. We omit the discussion due to lack of space.

We start with some preliminary definitions. Let S be a finite square $\{0, \ldots m\} \times \{0, \ldots m\}$. The classical domino problem as describe in the literature may be formulated as follows:
Instance: $\mathcal{D}$ consisting of a finite set D and a binary relation $H, V \subset D \times D$.
Question: Is there a tiling $\tau : S \to D$ such that for all $(x, y) \in S$ we have

$$\tau(x, y) = d_i \wedge \tau(x + 1, y) = d_j \Rightarrow (d_i, d_j) \in H \quad \textit{and} \quad \tau(x, y) = d_i \wedge \tau(x, y + 1) = d_j \Rightarrow (d_i, d_j) \in V$$

The formulation is equivalent to the more intuitive description of the tiling problem in terms of unit squares with colored edges. We consider the following decision versions of the above domino problem.

B1: Given $\mathcal{D}$ and n specified in Unary; can $\mathcal{D}$ tile an $n \times n$ square.

B2: Given $\mathcal{D}$ and integers n and m, where n denotes the width of the board (width in terms of Y-axis) is specified in unary and m the length of the board (in terms of X-axis) which is specified in binary, can $\mathcal{D}$ tile the $n \times m$ rectangle ?

B3: Given $\mathcal{D}$ and integers n and m, where n denotes the width of the board (width in terms of Y-axis) is specified in binary and m the length of the board (in terms of X-axis) which is specified in binary, can $\mathcal{D}$ tile the $n \times m$ rectangle ?

It is shown in [Ha85, Ha86, vEB83] that

1. **B1** is NP-complete;
2. **B2** is PSPACE-complete;
3. **B3** is NEXPTIME-complete.

Given a problem $\Pi \in \{\mathbf{B1}, \mathbf{B2}, \mathbf{B3}\}$, MAT-$\Pi$ denotes the non-contiguous version of the above problems and CMAT-Π the contiguous version of the optimization problems.

Theorem 10.1 (1) The problems MAT-B1, MAT-B2, and MAT-B3 have polynomial time approximation schemes.
In contrast, there exists $\epsilon > 0$ such that getting an approximation within n^ϵ times optimum is

(2) NP-hard for the problem CMAT-B1,

(3) PSPACE-hard for the problem CMAT-B2, and

(3) NEXPTIME-hard for the problem CMAT-B3.

Proof Sketch:
Part 1: The idea is similar to the one described for solving the independent set problem for periodically specified graphs. We describe the ideas for MAT-B1. Extension to MAT-B2, MAT-B3 follows along the lines of Algorithm in Figure 1. Given an $\epsilon > 0$, we first find the corresponding integer k. To begin with, for each i, $0 \leq i \leq k$, we partition the plane S into l disjoint sets $GS_1, \cdots GS_l$ by removing strips 1 unit thick with horizontal coordinates congruent to $i \mod (k+1)$.

Since the bound on X-axis is specified in unary, it follows that the length of each strip is polynomial in the size of the input and the width of each strip is a constant for a given $\epsilon > 0$. Using this fact, it is easy to devise a dynamic programming algorithm running in linear time that finds if the strip can be tiled or not. The same algorithm can also be used to find the way to tile the strip so as to maximize the tiled area. We can also choose to further subdivide each strips so that we are left with disjoint squares of size $(k+1) \times (k+1)$. This is useful to solve problems where the bounds on the plane to be tiled are specified in binary notation. Thus for each strip GS_p, $1 \leq p \leq l$, we solve the problem optimally. The solution for this iteration is just the union of the solutions for each strip GS_p. By an averaging argument, it follows that the partition which yields the largest solution value contains at least $(\frac{k}{k+1})^2 \cdot OPT(G)$ nodes, where $OPT(G)$ denotes the value of the optimal solution for tiling S.
Part 2: Again, we give an argument for **B1**. Starting with a NDTM M that is $O(n)$ time bounded and an input string x, we construct a Turing machine M_1 that cycles if and only if M accepts x. Now we do a standard reduction from M_1 to the domino problem. If M accepts x, then M_1 cycles and CMAT-B1 has a solution in which $n^k \times n^k$ area, for suitably large k, can be tiled contiguously[9]. If M does not accept x then, M_1 does not cycle on x, and hence the maximum contiguous area that can be tiled is no more than $O(n) \times O(n)$. ∎

[9] A $n \times n$ area is said to be tiled contiguously if the area is completely tiled