LA-UR 97-5201

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

TITLE: **THEORY OF PERIODICALLY SPECIFIED PROBLEMS: COMPLEXITY AND APPROXIMATION**

AUTHOR(S): M. Marathe, H. B. Hunt, D. J. Rosenkrantz, R. E. Stearns

SUBMITTED TO: IEEE Annual Conference on Computational Complexity
June, 1998

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

DISCLAIMER

Theory of Periodically Specified Problems: Complexity and Approximability

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

December 5, 1997

Abstract

We study the complexity and the efficient approximability of graph and satisfiability problems when specified using various kinds of periodic specifications studied in [Or82a, HT95, Wa93, HW94, Wa93, MH+94]. The general results obtained include the following:

1. We characterize the complexities of several basic generalized CNF satisfiability problems SAT(S) [Sc78], when instances are specified using various kinds of 1- and 2-dimensional periodic specifications [Or82a, Wa93, HW94, HW95, CM91, CM93]. We outline how this characterization can be used to prove a number of new hardness results for the complexity classes DSPACE(n), NSPACE(n), DEXPTIME, NEXPTIME, EXPSPACE etc. These results can be used to prove in a *unified* way the hardness of a number of combinatorial problems when instances are specified succinctly using various succinct specifications considered in the literature [LW92, Ga82, BLT92, BG89, HH93]. As one corollary, we show that a number of basic NP-hard problems become EXPSPACE-hard when inputs are represented using 1-dimensional infinite periodic **wide** specifications. This answers a long standing open question posed by Orlin [Or82a].
2. We outline a simple yet a general technique to devise approximation algorithms with provable worst case performance guarantees for a number of combinatorial problems specified periodically. Our efficient approximation algorithms and schemes are based on extensions of the ideas in [Ba83,HM85,MH+94] and represent the first non-trivial characterization of a class of problems having an ϵ-approximation (or PTAS) for periodically specified NEXPTIME-hard problems. Two of properties of our results are: (i) For the first time, efficient approximation algorithms and schemes have been developed for **natural** NEXPTIME-complete problems. (ii) Our results are the first polynomial time approximation algorithms with good performance guarantees for "hard" problems specified using various kinds of periodic specifications considered in this paper.

The results presented significantly extend the known results for succinctly specified problems in [LW92, Ga82, BLT92, Or82a, Pa94, Wa93, HW94, HW95, MH+94].

[1]Current Address: Los Alamos National Laboratory P.O. Box 1663, MS B265 Los Alamos NM 87545. Email: marathe@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,djr,res}@cs.albany.edu. Supported by NSF Grants CCR 90-06396 and CCR94-06611.

1 Introduction

Periodic specifications can be used to define large scale systems with highly regular structures. Using periodic specifications, large objects are described as repetitive connection of a basic module. Frequently, the modules are connected in a straight line but the basic modules can also be repeated in two or higher dimensional patterns. 1-dimensional specifications were studied by Orlin [Or82a], Wanke [Wa93], Hoppe and Tardos [HT95], Ford and Fulkerson [FF58] and Gale [Ga59]. Two dimensional periodic specifications arise naturally in the study of regular systolic arrays and VLSI signal processing arrays [HW94, CS81, IS86, IS86, IS87], discrete dynamical systems such as the cellular automata [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) [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. Other researchers have studied 2-dimensional and more generally d-dimensional periodic specifications. (See [CM91, IS87, KO91, KS88, Wa93, HW94, HW95].) In [Or82a, Pa94], the problem 3SAT, for 1-dimensional infinite narrow periodically specified formulas (denoted here by 1-PN-3SAT), is defined and shown to be PSPACE-complete. Apart from this single result, the complexity of periodically specified generalized satisfiability problems have not been studied previously.

2 Summary of results

We study the complexity and the efficient approximability of graph and satisfiability problems when specified using various kinds of succinct specifications with emphasis on various kinds of periodic specifications studied in [Or82a, HT95, Wa93, HW94, Wa93, MH+94]. We present general techniques for proving both hardness results as approximately solving problems so specified. To obtain our results, we systematically define various kinds of periodic specifications. For uniformity and space reasons, we will concentrate on generalized CNF satisfiability problems specified using various types of periodic specifications. The various kinds of periodic specifications considered depend upon the answers to the following questions: *(1) Is the specified instance* **1- or 2-dimensional** *? (2) Is the specified instance* **finite or infinite** *? (3) Are specifications* **narrow or wide** *? (4) Are* **explicit boundary conditions** *allowed in the specifications ? (5) Are bounds on finite dimensions specified in* **unary (U)** *or* **binary (B)** *? (6) Do the infinite dimensions range over* **natural numbers (N)** *or* **integers (Z)** *?* For the purposes of illustration, we limit our attention in this section to the following specifications: (A) The 2-dimensional finite periodic narrow specifications of Wanke [Wa93], (referred as 2-F(B,B)PN-specifications), (B) The 2-dimensional finite periodic narrow specifications with explicit boundary conditions (referred as 2-F(B,B)PN(BC)-specifications) (C) The 2-dimensional finite toroidal periodic narrow specifica-

tions of Höfting and Wanke [HW94], (referred as 2-F(B,B)PTN-specifications) and (D) the c-uniform 1-dimensional finite wide periodic specifications of Orlin [Or84b] (referred as 1-F(B)PW(c) specifications).

In Section 3.1 we detail the naming convention used to specify problems using various kinds of periodic specifications.

Our work is motivated in part by two basic observations: *(i) the lack of known general techniques to characterize the complexity and approximability of succinctly specified problems and (ii) lack of understanding of the inter-relationship between various succinct specifications considered here.* For instance, Quoting Orlin [Or82a],

> It is of interest to know if there are general properties that an NP-complete problem may have so as to guarantee the corresponding periodic problems to be PSPACE-complete.

Our general results on characterizing the complexity of succinctly specified satisfiability problems and its applications in proving hardness/easiness results for other succinctly specified combinatorial problems can be seen as a first step towards answering the above question raised by Orlin.

We also make progress in the direction suggested by the second observation above and show that

1. Certain simple repetitive structures can be specified (using small specifications) by *all* the succinct specifications studied here.

2. We show that basic combinatorial problems are "hard" (for the respective complexity classes) even for such simple repetitive structures specified succintly; thus the problems are "hard" when specified by *any* of the succinct specifications studied.

These result immediately imply (and provide *alternative and unified proofs* for) the hardness results obtained in the past. We elaborate on this further below.

2.1 Complexity of Periodically Specified Problems

A summary of our results, for the two problems 3SAT and 3SATWP appears in Table 1. Using the notation of Schaefer [Sc78], all of the hardness results for the problem 3SAT, also hold for each of the problems SAT(S) and SAT_c(S) shown to be NP-complete in [Sc78].

We can show that efficient reductions involving *local replacement* (possibly augmented with fixed size enforcers) [GJ79] of the problem 3SAT 1-3SAT, NAE-3SAT 3SATWP[3] etc, to a problem Π can be extended to obtain efficient reductions of the problems 3SAT 1-3SAT, NAE-3SAT 3SATWP, etc, to the problem Π, when instances are specified using the kinds kinds of periodic specifications considered here. These problems include most of the basic problems in [Ka72, GJ79] as well as several basic P-complete problems [JL77]. These results yield a number of new hardness results for the complexity classes DSPACE(n), NSPACE(n), DEXPTIME, NEXPTIME, EXPSPACE etc. depending on the kind of periodic specification used. To our knowledge, previously no DEXPTIME, NEXPTIME, EXPSPACE-hardness or undecidability results were known for periodically specified problems. The periodic languages that can be formalized can be seen as a characterization of various space/time complexity classes. For instance 1-dimensional periodic specifications of Orlin [Or82a]

[3] Horn formula satisfiability problem is the restriction of the problem 3SAT, in which each clause has at most one *positive literal*. This is the same as the problem SATWN, studied by [Sc78]. The problem 3SATWP is similar to 3SATWN except that each clause has at most one negated literal.

constitute an alternative characterization of PSPACE. In a similar fashion, the wide 1-dimensional periodic specifications constitute an alternative characterization of EXPSPACE. Our study generalizes Orlin's result [Or82a] that the problem 1-I(Z)PN-3SAT is PSPACE-complete and Schaefer's characterization [Sc78] of the complexity of generalized CNF satisfiability problems SAT(S), where S is a finite set of finite arity Boolean relations. As one corollary, we prove the EXPSPACE-hardness of a large class of combinatorial problems when specified by 1-dimensional wide periodic specifications. answering the following open question posed by Orlin [Or82a]: *"It is an interesting open question as to whether the non-narrow periodic graph problems are in the class* PSPACE."

We consider representative applications of the above mentioned results. Most of these results were not known earlier. First, note that since 2-dimensional finite periodic specifications can be viewed as simple types of S.C.R. specifications, our hardness for problems specified using 2-dimensional finite periodic narrow specifications, strengthen the hardness results in [PY86, BLT92][4] Second, the hardness results can be easily transformed to obtain hardness results for problems specified using hierarchical specifications of Lengauer and Wagner [LW92] or the G.C.R. specifications of Galperin [Ga82]. The NEXPTIME-hardness results for problems specified using G.C.R. specifications constitute first such results in the literature. Finally, our undecidability results for 2-dimensional infinite periodically specified satisfiability problems imply a number of undecidability results for problems specified using recursive graph specifications. Since 2-dimensional infinite periodic graph specifications are clearly a *simple type* of recursive graph specifications, these results strengthen several undecidability results in Harel et. al. and Beigel and Gasarch [Ha91, BG89, HH93]. An important corollary of our result that the 2-F(B,B)PN-MCVP [5] is DEXPTIME-complete. As pointed out in [LW87a], the circuit value problem is at the basis of most simulation algorithms for integrated circuits. These ideas easily extend to yield similar lower bounds for the simulation or evaluation of all classes of strongly acyclic periodically specified functions, for which the allowed modules can emulate monotone Boolean logic. Thus our reduction and lower bounds apply to strongly acyclic periodically specified functions on many different algebraic structures with 0 and 1. These include all the following algebraic structures (provided that they have at least two elements): lattices, rings with multiplicative identity, idempotent semirings with a multiplicative identity, finite semirings with a multiplicative identity they are not rings, etc. For examples of these structures see [MB, BHR84]. In particular, our techniques and corresponding lower bounds apply to the various lattice-theoretical structures used to simulate faults, errors, transients, unknown states, variable strength signals, etc., in digital logic both at the gate and transistor levels [Ha]. Thus these results significantly extends the earlier results by Lengauer and Wagner [LW92] and Rosenkrantz and Hunt [RH93] on the complexity of simulating hierarchically specified acyclic circuits; moreover, previously no results were known for the complexity of simulating periodically specified acyclic circuits.

In [LW92] Lengauer and Wagner state that:

> We find that the complexity of the nonhierarchical version of a graph problem says practically nothing about the complexity of its hierarchical version.

In [Or84b], Orlin states that

> There are no general conditions which guarantee that the dynamic variants of an NP-complete problem is PSPACE-complete.

[4] Although we strengthen several of the results in [PY86, BLT92], we do not have general metatheorems such as those given in [PY86, BLT92].

[5] MCVP denote the Monotone Circuit Value Problem.

Assuming P $\neq$ PSPACE and/or NP $\neq$ PSPACE, these assertions are true. But our proofs in this paper show that

> Our characterization of succinctly specified SAT(S) problems, combined with our knowledge about known *local reductions* in the non-succinct case can be used to predict the complexity of succinctly specified problems.

The results also demonstrate that using known reductions in the non-succinct case that are *merely resource bounded are inadequate* for inferring the hardness of succinctly specified problems. But, the locality of reductions between problems specified using standard specifications is useful in lifting the reductions to the succinct case.

2.2 Approximation Algorithms

Given the hardness results in the previous section for solving the problems exactly when specified by one of the above specifications, we investigate the existence of polynomial time approximation algorithms for these problems. We present a uniform approach for developing the *first* efficient approximation algorithms and/or schemes for a number of optimization problems when specified using one of the specifications α. To this end we present a fairly simple yet general technique consisting of two main steps. First, by an extension of ideas in Baker [Ba83] we show that for each fixed finite set S there is polynomial time approximation algorithm (and a scheme for planar instances[6]) for the problems MAX SAT(S) specified periodically using one of the specifications mentioned earlier in the section. In the next step, we show that a number of important class of problems when specified periodically can be reduced in an approximation preserving way to appropriate problems MAX SAT(S) specified using the same type of periodic specifications. We all such reductions *structure preserving* L-reductions. The general approximation algorithms and schemes for the problems MAX SAT(S)are an attempt to answer the fundamental question *which "hard" periodically specified optimization problems have efficient approxmations ?* In this direction the general theory developed here (and discussed above) provides a sufficient condition:

> Periodically specified graph and other optimization problems have a ϵ-approximation algorithm (or PTAS) when the the semantics of the problem can be described by a SAT(S) formula such a way that the formula interaction graph inherits the structure of the graph.

In the recent years (see [CK97, KT94, KM96], etc) there has been a significant interest in providing syntactic characterization of optimization problems in an attempt to explain and provide a uniform framework for solving such problems. Our results provide a syntactic (algebraic) class of problems, namely, MAX SAT(S) whose closure under L-reductions (or other appropriate approximation preserving reductions) define one such characterization for problems that have ϵ-approximations (or PTAS). This represents the *first non-trivial characterization* of a class of problems having ϵ-approximations (or a PTAS) when restricted to periodically specified problems. The algebraic model (characterization) is general enough to express the optimization version of (i) the generalized satisfiability problems of Schaefer [Sc78], (ii) feasibility of systems of linear equations over a variety of algebraic structures, (iii) a class of nonlinear optimization problems (iv) several well known graph theoretic problems. We refer the reader to [KM96] for more details. Our main result in this context can be stated as follows:

[6] corresponding bipartite graphs are planar.

Theorem 2.1 For each fixed $l \geq 1$, and for each of the problems Π listed in Table 1,[7] the problem α-Π, has a polynomial time approximation algorithm with running time $O(RT_\Pi(l^2 \cdot |G|))$ with performance guarantee[8] $(\frac{l+1}{l})^2 \cdot FBEST_\Pi$. 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 1-F(B)PW(c)-MAX 2SAT 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, using the recent nonapproximability results of [AL+92] we get the following:

Theorem 2.2 For all the problems Π listed in Table 1, the problems α-Π have polynomial time approximation schemes if and only if P = NP.

As a second result which follows from the proof of Theorem 2.1, we get that all of the above problems Π have a polynomial time approximation scheme (PTAS), when restricted to planar instances. *We can show that many of these problems remain* NEXPTIME*-complete, even when restricted to planar instances.*

Theorem 2.3 For all the problems Π listed in Table 1, the problems α-Π have polynomial time approximation schemes when restricted to planar instances.

The approximation algorithms 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.* To our knowledge this is the first time, polynomial time approximation algorithms are developed for **natural** NEXPTIME-hard problems. Thus our results provide the first natural problems for which there is a proven exponential (and possibly doubly exponential) gap [9] between the time complexities of finding exact and approximate solutions[10]. Only very recently has there even been work on the efficient approximability of PSPACE-hard problems (See [AC94, Co95, CF+93, CF+94, MH+94]). The NEXPTIME-hardness show that the very regular structure of problems specified periodically do **not** suffice to make problems easy. But, the efficient approximation algorithms and schemes developed here show the following:

> The very regular structures of problem instances specified by the periodic specifications of A-D (mentioned earlier) suffice to make approximating many basic optimization problems easy.

Approximating many of the optimization problems considered here, when instances are specified using he *small circuit specifications* of [PY86,LB89] can be shown to be NEXPTIME-hard by extensions

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

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

[9] Previous non-approximability results show that many optimization problems are NP-hard or PSPACE-hard to approximate beyond a certain factor. While these hardness results point out that it is *unlikely* in general to find "good" polynomial time approximation algorithms, they does not rule out this possibility. The results presented here show a **provable** gap between approximation and decision since the decision problems are NEXPTIME-complete and hence requires **at least** 2^{cn} steps and **possibly** $2^{2^{cn}}$ steps (if NEXPTIME $\neq$ DEXPTIME) to solve.

[10] Of course, it is easy to construct artificial problems, whose decision versions are NEXPTIME-hard, that have polynomial time approximation algorithms with good performance guarantees.

of the arguments in [Ar94]. Thus our results high-light one *important difference* between multiple-dimension finite periodic specifications and small circuit specifications.

Due to lack of space, the remainder of this paper consists of preliminary definitions and selected proof sketches. Additional results are discussed in the Appendix.

3 Preliminary Definitions

We first review, the concept of periodically specified instances. In what follows we discuss the concept of 2-dimensional periodically specified satisfiability problems. The notion of periodically specified graphs is given in [CM93, CM91, Wa93, HW94, HW95]. Figure 1 shows an example of periodic specification and the associated expanded graph.

For the rest of the paper, let **Z** and **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). $U^{M,N} = \{u_k(i,j) : 1 \leq k \leq n,\ i \in \{0,1,2,\cdots,M\},\ j \in \{0,1,2,\ldots,N\}\}$. (In our proofs, variable $u_k(i,j)$ denotes the variable u_k at grid point (i,j).) A literal of U is an element of $\{u_1, \ldots, u_n, \overline{u_1}, \ldots, \overline{u_n}\}$. If w is a literal of U, then $w(i,j)$, $0 \leq i \leq M$ and $0 \leq j \leq N$ is a literal of $U^{M,N}$. Let $C(i,j,i+1,j+1)$ be a parameterized conjunction of 3 literal clauses such that each clause in $C(i,j,i+1,j+1)$ consists of variables $u_k(i,j), u_k(i+1,j), u_k(i,j+1), u_k(i+1,j+1)$ with the constraint that at least one variable is of the form $u_k(i,j)$. We refer to the clauses $C(i,j,i+1,j+1)$ as *static narrow clauses.* ($C(i,j)$ is called narrow because for all $(w_1(i_1,j_1) \vee w_2(i_2,j_2) \vee w_3(i_3,j_3)) \in C(i,j,i+1,j+1)$, $|i_s - i_r|, |j_s - j_r| \leq 1$ for $1 \leq r \leq s \leq 3$.) The conjuction of static narrow clauses is referred to as static narrow formula. Let $\Gamma = (U, C(i,j,i+1,j+1), M, N)$. Let $\mathcal{C} = \bigwedge_{i=0,j=0}^{i=M,j=N} C(i,j,i+1,j+1)$. Then $\mathcal{C}$ is the 3CNF formula specified by Γ. Given $U^{M,N}$ and $\mathcal{C}$, let $C^{M,N}$ be a subset of $\mathcal{C}$ with the following property: for each clause $(w_1(i_1,j_1) \vee w_2(i_2,j_2) \vee w_3(i_3,j_3)) \in C^{M,N}$, $w_1(i_1,j_1), w_2(i_2,j_2), w_3(i_3,j_3) \in U^{M,N}$.

Definition 3.1 A 2-dimensional finite periodic narrow specification (2-F(B,B)PN-specification) of a 3CNF formula $F^{M,N}(U^{M,N}, C^{M,N})$ is a four tuple $\Gamma = (U, C(i,j,i+1,j+1), M, N)$, where, U is a a finite set of variables, $C(i,j)$ is a collection of static narrow 3 literal clauses, and M, N are non-negative integers specified in binary. The size of the specification denoted by $size(\Gamma) = |U| + |C(i,j,i+1,j+1)| + bits(M) + bits(N)$, where $bits(M)$ and $bits(N)$ denote the number of bits used to represent M and N respectively.

The problem 2-F(B,B)PN-3SAT (problem 3SAT specified using 2-dimensional finite periodic narrow specifications with both bounds in binary i.e. 2-F(B,B)PN-specifications) is the problem of determining if a 3CNF formula $F^{M,N}(U^{M,N}, C^{M,N})$ specified by $\Gamma = (U, C(i,j,i+1,j+1), M, N)$ is satisfiable.

2-I(**Z**, B)PN-3SAT is the problem of, given a 2-dimensional periodic specification $\Gamma = (F(U, C(i,j,i+1,j+1), m)$, where m denotes the width (in terms of Y-axis) is specified in binary, determining whether the CNF formula, $\wedge_{i=-\infty,j=0}^{i=\infty,j=m} C(i,j,i+1,j+1)$ satisfiable.

3.1 Note on Naming Convention

Since we have a large number of parameters, it is necessary to state the notation used throughout this abstract for naming problems. 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).

Example 1: Let the set of static variables $U = \{x, y, z\}$. The static clauses C is specified by $C(i, j, i{+}1, j{+}1) = [x(i,j){+}y(i,j){+}z(i,j)] \wedge [x(i{+}1,j){+}y(i,j){+}z(i{+}1,j)] \wedge [x(i,j{+}1){+}z(i,j)]$. The set of clauses $C^{1,1}$ is given by

$[x(0,0){+}y(0,0){+}z(0,0)] \wedge [x(0,1){+}y(0,1){+}z(0,1)] \wedge [x(1,0){+}y(1,0){+}z(1,0)] \wedge [x(1,1){+}y(1,1){+}$
$z(1,1)] \wedge [x(1,0){+}y(0,0){+}z(1,0)] \wedge [x(1,1){+}y(0,1){+}z(1,1)] \wedge [x(0,1){+}z(0,0)] \wedge [x(1,1){+}z(1,0)]$

4 NEXPTIME-completeness of 2-F(B,B)PN-3SAT

The main idea involves the construction of a static formulas, force the satisfiability of the expanded formulas to correspond to the existence of legal computations of Turing machines. Intuitively, we have one column for each step of a computation and one row for each tape cell of the Turing machine. Proving hardness results for satisfiability problems without explicit boundary conditions is subtle, since there is no obvious way to force the the Turing machine to start correctly.

Theorem 4.1 2-F(B,B)PN-3SAT is NEXPTIME-complete.

Proof Sketch: 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-FPN-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 $x = (x_1, \ldots, x_n)$ 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 $2^{c_0 n}$ moves, else
2. If Turing machine ϕ accepts x, then ϕ_x has a cycling computation, where the length of an ID never exceeds $2^{d_0 n}$, 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 $2^{d_0 n} + 1$. From Property 2 above, we need to consider only $2^{d_0 n}$ 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 $q = d_0 n$. The counter can be simulated by means of $(d_0 n + 1)$ 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 $(d_0 n+1)$ 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 $\#(q_0, x_1) \ldots x_n B^{2^{d_0 n} - n}$, where B denotes a blank. The static formula CNF formula $G_x(t,y)$ is given by $G_x(t,y) = f_1(t,y) \wedge f_2(t,y) \wedge f_3(t,y)$. We now describe each of the subformulas $f_i, 1 \leq i \leq 3$ separately. Each f_i is described in terms of variables at coordinates $y, y+1, t, t+1$. **Counter Updating formula** f_1 is used to simulate a counter to achieve implicit initialization. **Implicit Initialization formula** 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 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 these formulas is given in the appendix.

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 $\bigwedge_{y=0,t=0}^{y=N,t=M} G_x(t,y,t+1,y+1)$ 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 $c_t = 0, c_t = 1, c_t = 2, \cdots c_t = 2^{d_0 n}$. 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 $c_y = 0, c_y = 1, c_y = 2, \cdots c_y = 2^{d_0 n}$. 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 $\bigwedge_{y=0,t=0}^{y=N,t=M} G_x(t,y,t+1,y+1)$ is satisfied then the Turing machine ϕ accepts x. ■

4.1 Polynomial time solvability of 2-F(Z,Z)PN-3SATWP

Next, we consider the problems 2-F(B,B)PN-3SATWP, 2-I(N,N)PN-3SATWP and 2-I(Z,Z)PN-3SATWP. Extending our results for these problems to similar problems involving Horn formula satisfiability is straightforward and is omitted here. In contrast to the undecidability of solving 2-I(N,N)PN(BC)-3SATWP, we show that each of the above three problems has a polynomial time algorithm. This points out a major difference between these variants of periodic specifications.

We first consider the problem 2-I(Z,Z)PN-3SATWP. Recall that a relation R is weakly positive if R is equivalent to some CNF formula having at most one negated variable in each conjunct. The algorithm for solving the problem 2-I(Z,Z)PN-3SATWP is relatively easy, and is based on the following two observations. The first observation is that if there is a clause with only one literal, all copies of the corresponding variable must have the same value. For instance, if there is a clause consisting of the single literal $\overline{x_i(t+1,y)}$, then all copies of variable x_i have to be set to false. The second observation is that after simplifying the set of clauses as much as possible on the basis of the first observation, every remaining clause has either no literals or more than one literal. Weak positivity implies that each clause with more than one literal contains at least one positive literal, so setting all remaining variables to true will satisfy all such clauses. Since each simplification of the set of clauses based on the first observation assigns a value to a variable in the static formula which has not been previously assigned a value, the algorithm will terminate in polynomial time. Note that if the expanded formula for the given instance of 2-I(Z,Z)PN-3SATWP is satisfiable, there exists a satisfying assignment that assigns the same value to all the copies of a given variable in the static formula.

Next consider the problems 2-I(N,N)PN-3SATWP and 2-F(B,B)PN-3SATWP. Any algorithm for solving these problems must deal with subtle issues created by the presence of a "boundary" in the expanded formula. A clause of the form $x_i(t,y)$ implies that x_i is set to true for all time periods. However, a clause of the form $x_i(t+1,y)$ does not imply anything about the value of the variable $x_i(0,y)$ in a satisfying assignment of the formula. Similar arguments hold for clauses of the form $x_i(t,y+1)$ and $x_i(t+1,y+1)$ (the second clause might arise after the elimination of other variables.) The following simple example, shows that even for 1-dimensional specifications, there are cases where all satisfying assignments to the expanded formula assign different values to the copies of a particular variable.

Example 2: Let $F = (U, C(t,t+1), 1)$ be an instance of 1-F(B)PN-3SATWP where the set of static clauses are given by $(x_1(t) + x_2(t+1)) \wedge (\overline{x_2(t)}) \wedge (x_2(t) + \overline{x_1(t+1)})$. The set of variables are $U = \{x_1, x_2\}$. The expanded formula is $(x_1(0) + x_2(1)) \wedge (\overline{x_2(0)}) \wedge (x_2(0) + \overline{x_1(1)}) \wedge (x_2(1))$.

By inspection it is clear that any assignment to the variables of the expanded formula such that $v[x_1(0)] = v[x_1(1)]$ and $v[x_2(0)] = v[x_2(1)]$ cannot satisfy the formula. But the assignment $v[x_1(0)] = 1$, $v[x_1(1)] = 0$, $v[x_2(0)] = v[x_2(1)] = 0$ satisfies the expanded formula. A similar example can be constructed for the 2-dimensional case. ■

Example 2 suggests that a polynomial time algorithm for solving 1-I(N,N)PN-3SATWP should distinguish between the copy of each variable at $t = 0, y = 0$ and the copies of the same variable at time $t, y \geq 0$. Our polynomial time algorithm for 2-I(N,N)PN-3SATWP considers four groups of variables, corresponding to $(t = 0, y = 0)$, $(t = 0, y > 0)$, $(t > 0, y = 0)$, and $(t > 0, y > 0)$.

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

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 can be seen as an extension of 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. 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.)

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.

ALGORITHM 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. $IS(G) = \max_{0 \leq i \leq k} IS(G_i)$

We illsutrate the basic technique by discussing an algorithm for solving 2-F(B,B)PN-MAX-IS problem (ALGORITHM ALGORITHM ALG-2-FPN-MAX-IS). The correctness and the performance guarantee through a series of intermediate results is established in the Appendix.

Acknowledgements

We thank A. Condon, T. Lengauer, S. Shukla, R. Sundaram, S.S. Ravi and E. Wanke for valuable discussions on succinct specifications. We also thank J. Balcázar, van Emde Boas, M. Fürer, J. Orlin and E. Cohen for making available copies of their papers.

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,* June, 1995, pp. 214-226.

[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.)

[BLT92] J.L. Balcazar, A. Lozano, and J. Toran " The Complexity of Algorithmic Problems for Succinct Instances," in *Computer Science* Ed. R. Baeza-Yates, 1992, pp. 351-377.

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

[BG89] R. Beigel, W. I. Gasarch, " On the Complexity of Finding the Chromatic Number of Recursive Graphs," Parts I and II, *Annals of Pure and Applied Logic*, 45, 1989, pp. 1-38 and 227-247.

[BHR84] P.A. Bloniarz, H.B. Hunt III and D.J. Rosenkrantz "Algebraic Structures with Hard Equivalences and Minimization Problems," *J. ACM*, 31, (1984), pp. 879-904.

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

[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.

[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.

[CK97] P. Crescenzi and V. Kann, "A Compendium of NP-Optimization Problems," preliminary version, September 1997.

[CM93] E. Cohen and N. Megiddo, "Strongly Polynomial-time and NC Algorithms for Detecting Cycles in Dynamic Graphs," *Proc. 21st ACM Annual Symposium on Theory of Computing (STOC)*, 1989, pp. 523-534. Journal version appears in *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.

[FF58] L.R. Ford and D.R. Fulkerson, "Constructing Maximal Dynamic Flows from Static Flows," *Operations Research,* No. 6, 1958, pp. 419-433.

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

[Ga59] D. Gale, "Transient Flows in Networks," *Michigan Mathematical Journal*, No. 6, 1959 , pp. 59-63.

[Ga82] H. Galperin "Succinct Representation of Graphs," Ph.D. Thesis, Princeton University, 1982.

[GW83] H. Galperin and A. Wigderson, "Succinct Representation of Graphs," *Information and Control* , Vol. 56, 1983, pp. 183-198.

[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.

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

[Ha] J.P. Hayes, "Digital Simulation with Multiple Logic values," *IEEE Transactions on Computer Aided Design,* CAD-5, 1986, pp. 274-283.

[Ha91] D. Harel, "Hamiltonian Paths in Infinite Graphs," *Israel J. Math.* 76: 3, 1991, pp. 317-336. A preliminary version of the paper appears in *Proc. 23rd Annual ACM Symposium on Theory of Computing, (STOC)*, 1991, pp. 220-229.

[HH93] T. Hirst, D. Harel, "Taking it to the Limit: On Infinite Variants of NP-Complete Problems," *Proc. 8th IEEE Annual Conference on Structure in Complexity Theory*, June, 1993, pp. 292-304.

[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 International Colloquium on Automata, Programming and Languages,* LNCS, Oct. 1994, pp.

[HT95] B. Hoppe, E. Tardos, "The quickest Transhipment Problem," *Proc. 6th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1995, pp. 512-521.

[Hu73a] H.B. Hunt III, "On The Time and Tape Complexity of Languages," Ph.D. thesis, Department of Computer Science, Cornell University, August, 1973.

[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.

[JL77] N.D. Jones and W.T. Laaser, "Complete Problems for Deterministic Polynomial Time," *Theoretical Computer Science*, No. 3, 1977, pp. 105-117.

[KM96] S. Khanna and R. Motwani "Towards a Syntactic Characterization of PTAS" *Proc. 28th Annual ACM Symposium on Theory of Computing, (STOC)*, pp. 329-337, Philadelphia, PA May 1996.

[KT94] P. G. Kolaitis and M.N. Thakur, "Logical Definability of NP Optimization Problems," *Information and Computation*, No. 115, 1994, pp. 321-353.

[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.

[Ka72] R.M. Karp, "Reducibility Among Combinatorial Problems," in R.E. Miller and J.W. Thatcher (eds) *Complexity of Computer Computations*, Plenum Press, N.Y. 1972, pp. 85-103.

[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.

[LW87a] T. Lengauer and E. Wanke, "Efficient Solutions for Connectivity Problems for Hierarchically Defined Graphs ," *SIAM Journal on Computing*, Vol. 17, No. 6, 1988, pp. 1063-1080.

[LW92] T. Lengauer and K.W. Wagner, "The Correlation Between the Complexities of Non-Hierarchical and Hierarchical Versions of Graph Problems," *Journal of Computer and System Sciences (JCSS)*, Vol. 44, 1992, pp. 63-93.

[MB] S. MacLane and G. Birkhoff, "Alegbra," Macmillan, NY 1967.

[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.

[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.

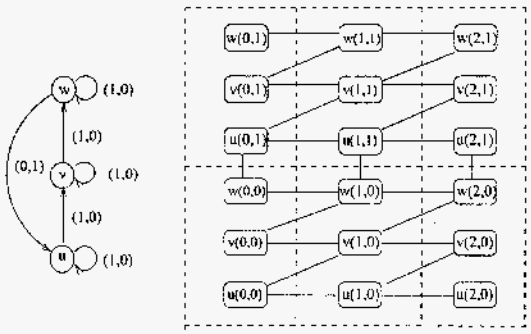
[Pa94] C. Papadimitriou, *Computational Complexity* Addison-Wesley, Reading, Massachusetts, 1994.

[RH93] D.J. Rosenkrantz and H.B. Hunt III, "The Complexity of Processing Hierarchical Specifications," *SIAM Journal on Computing*, Vol. 22, No. 3, 1993, pp. 627-649.

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

[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.



(a) (b)

Figure 1: (a)The static graph with 2-dimensional integer vectors associated with each edge. (b) The graph $G^{2,1}$ specified by $\Gamma = (G, 10, 01)$.

Appendix

Details for the Proof of Theorem 4.1:

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:
$f_1^1 \equiv [c_t(t+1,y) = (c_t(t,y)+1) \pmod{2^{d_0 n}+1},]$ $f_1^2 \equiv [0 \leq c_y(t,y) < 2^{d_0 n} \Rightarrow c_t(t,y+1) = c_t(t,y)]$
$f_1^3 \equiv [c_y(t,y+1) = (c_y(t,y)+1) \pmod{2^{d_0 n}+1}]$ $f_1^4 \equiv [0 \leq c_t(t,y) < 2^{d_0 n} \Rightarrow c_y(t+1,y) = c_y(t,y)]$

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 $2^{d_0 n}+1$ 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

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

$\vdots$

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

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

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

Problems		Specification Type					Complete
Type	Name	Dimension(s) X,Y	Finite (F) or Infinite (I)	Boundary Conditions Present Yes (Y) or No (N)	Edges Narrow (N) or Wide (W)	F/I Bounds F,Unary (U) F, Binary (B) I, Int. (Z) I, Nat. (N)	
3SAT	1-F(B)PN(BC)-3SAT	1	F	Y	N	B	NSPACE(n)-C
	1-F(B)PN-3SAT	1	F	N	N	B	NSPACE(n)-C
	1-I(Z)PN-3SAT	1	I	N	N	Z	NSPACE(n)-C
	1-I(N)PN-3SAT	1	I	N	N	N	NSPACE(n)-C
	1-F(B)PW(BC)-3SAT	1	F	Y	W	B	NEXPTIME-C
	1-F(B)PW(BC)-3SAT	1	F	N	W	B	NEXPTIME-C
	1-I(Z)PW-3SAT	1	I	-	W	Z	EXPSPACE-C
	2-F(B, U)PN-3SAT	2	F,F	-	N,N	B, U	NSPACE(n)-C
	2-F(B, B)PN-3SAT	2	F,F	-	N,N	B, B	NEXPTIME-C
	2-I(Z, B)PN-3SAT	2	I,F	-	N,N	B, Z	NEXPTIME-C
	2-I(Z, Z)PN-3SAT	2	I,I	-	N,N	Z,Z	undecidable
	2-I(N, N)PN-3SAT	2	I,I	-	N,N	N,N	undecidable
3SATWP	1-F(B)PN-3SATWP	1	F	N	N	B	Poly
	1-I(Z)PN-3SATWP	2	I	N	N	Z	Poly
	2-I(Z, Z)PN-3SATWP	2	I	N	N	Z,Z	Poly
	2-I(N, N)PN-3SATWP	2	I	N	N	N,N	Poly
	1-F(B)PN(BC)-3SATWP	1	F	Y	N	B	DSPACE(n)-C
	1-F(B)PW(BC)-3SATWP	1	F	Y	W	B	DEXPTIME-C
	2-F(B, U)PN(BC)-3SATWP	2	F,F	Y	N,N	B,U	DSPACE(n)-C
	2-F(B,B)PN(BC)-3SATWP	2	F,F	Y	N,N	B,B	DEXPTIME-C
	2-F(Z, B)PN(BC)-3SATWP	2	I,F	Y	N,N	Z,B	EXPSPACE-C
	2-F(N, N)PN(BC)-3SATWP	2	I,I	Y	N,N	N, N	undecidable

Table 1: Table summarizing the results for the problems 3SAT and 3SATWP when instances are specified using various kinds of periodic specifications. For example, the 8th row in the table specifies that the problem 3SAT when specified using 2-dimensional finite periodic narrow specifications, with the bounds on the X-axis specified in binary and the bounds on the Y-axis specified in unary is PSPACE-complete. **Z, N** stand for integers and natural numbers respectively.

Problem	2-F(B,B)PN Specs		Standard Specs	
	Planar	Arbitrary	Planar	Arbitrary
MAX-3SAT	$(\frac{l+1}{l})^3$	$(\frac{l+1}{l})^2 \cdot 4/3$	$(\frac{l+1}{l})$	$4/3$
MAX-SAT(S)	$(\frac{l+1}{l})^3$	$(\frac{l+1}{l})^2 \cdot 2^p$	$(\frac{l+1}{l})$	2^p
MIN-VC	$(\frac{l+1}{l})^3$	$(\frac{l+1}{l})^2 \cdot 2$	$(\frac{l+1}{l})$	2
MAX-IS	$(\frac{l+1}{l})^3$	$(\frac{l+1}{l})^2 \cdot b$	$(\frac{l+1}{l})$	b
MIN-DOM-SET	$(\frac{l+1}{l})^3$	$(\frac{l+1}{l})^2 \cdot b$	$(\frac{l+1}{l})$	$\log b$
MAX ED- DOM-SET	$(\frac{l}{l-1})^3$		$(\frac{l}{l-1})$	2
MAX-PART- INTO-TRIAN	$(\frac{l+1}{l})^3$	$(\frac{l-1}{l}) \cdot 3$	$(\frac{l+1}{l})$	3
MAX-H- MATCH	$(\frac{l+1}{l})^3$	$(\frac{l+1}{l}) \cdot (\lvert V_H \rvert/2 + \epsilon)$	$(\frac{l+1}{l})$	$(\lvert V_H \rvert/2 + \epsilon)$
MAX-CUT	$(\frac{l+1}{l})^2$	$(\frac{l+1}{l}) \cdot 1.137$	polynomial	1.137

Table 2: Performance Guarantee Results for Optimization Problems for problems specified using 2-F(B,B)PN-specifications. Problem names are clear from the abbreviations. All the problems can be shown to be NEXPTIME-hard using the method outlined in the paper. Similar results hold for problems specified using 2-F(B,B)PN(BC) 2-F(B,B)PTN and 1-F(B)PW(c)-specifications respectively. b denotes the degree bound. p denotes the maximum arity of a relation in S. The approximation results for the standard case for arbitrary and planar instances can be found in [CK94].

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 $n = |x|$.) The expanded finite periodic 3SAT instance is $\bigwedge_{y=0,t=0}^{y=N,t=M} G_x(t, y, t+1, y+1)$, where $M = 2^{2d_0 n}$ and $N = 2^{2d_0 n}$.

5.1 Proving EXPSPACE-hardness of 2-I(N,B)PN-3SAT

Although there are technical difficulties, the basic idea behind the proof is similar to the idea used to prove NEXPTIME-hardness of 2-F(B,B)PN-3SAT. Therefore, we only point out essential differences. Recall that we used two counters to keep a track of the length of each ID and also to keep track of the number of ID's. Since in the proofs of NEXPTIME-hardness, we need only consider singly exponential many ID's we were able to use a counter which had only polynomial number of bits. This in particular implied that the variables constituting the counter can occur together **explicitly** in the static formula. In this case, we want to simulate an 2^{cn} space-bounded Turing machine and this means that we need to keep track of roughly $2^{2^{cn}}$ ID's. To do this we need a counter with roughly 2^{cn} bits. Thus all the variables constituting the counter cannot occur together **explicitly** in the static formula.

It is easy to see that the above ideas can be extended to prove hardness results for wide specifications. We only give the basic idea behind the reduction. For this, it is useful to imagine each ID being rotated horizontally on the X-axis. Now observe that the narrow clauses used to describe the relationship between variables of consecutive ID's in case of 2-dimensional specifications can be replaced by wide clauses in the 1-dimensional specifications. Summarizing the discussion, we get the following theorem.

6 Approximation Scheme for 2-F(B,B)PN-MAX-IS: Performance

Lemma 6.1 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 6.2 The number of equivalence classes is no more than 9. Furthermore, The number of elements in each equivalence class is a polynomial time computable function f (in the size of the specification) of M and N, denoted by $f(M, N)$.

Proof Sketch: For the purposes of understanding, assume that the periodic graph as a large square which is partitioned into small square pieces. Figure 2 (given in the appendix) shows the possible different equivalence classes. ■

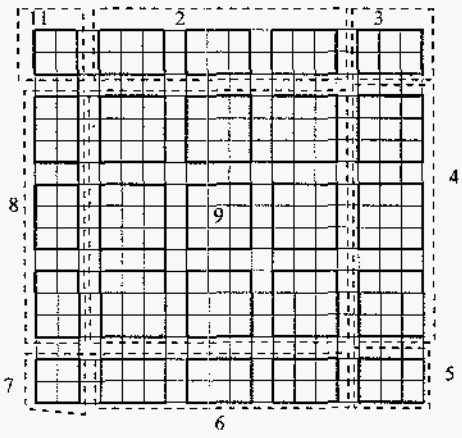


Figure 2: Figure showing the possible equivalence classes as a result of decomposition. The black squares denote subgraphs and the black dotted lines denote the equivalence classes.

Lemma 6.3 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 6.4

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

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

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

The proofs of the theorem follows by an averaging argument. We omit the proofs due to the lack of space.

Theorem 6.5 $|IS(G)| \geq (\frac{k}{k+1})^2 \cdot FBEST \cdot |OPT(G)|$. Here $FBEST$ denotes the performance guarantee of the best algorithm known to solve the independent set problem.