LA-UR 95-20

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: **PERIODICALLY SPECIFIED PROBLEMS: AN EXPONENTIAL COMPLEXITY GAP BETWEEN EXACT AND APPROXIMATE SOLUTIONS**

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

SUBMITTED TO: 27th ACM Annual Symposium on Theory of Computing (STOC) 1995

DISCLAIMER

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

Periodically Specified Problems : An Exponential Complexity Gap Between Exact and Approximate Solutions

(Extended Abstract)

H.B. Hunt III [1,2,3] M.V. Marathe[1,2,3,5] V. Radhakrishnan[2,4]
D.J. Rosenkrantz [1,2,3] R.E. Stearns[1,2,3]

November 28, 1994

Abstract

We study both the complexity and approximability of various graph and combinatorial problems specified using two dimensional narrow periodic specifications (see [CM93, HW92, KMW67, KO91, Or84b, Wa93]). The following two general kinds of results are presented.

1. We prove that a number of natural graph and combinatorial problems are NEXPTIME- or EXPSPACE-complete when instances are so specified.
2. In contrast, we prove that the optimization versions of several of these NEXPTIME-, EXPSPACE-complete problems have polynomial time approximation algorithms with constant performance guarantees. Moreover, some of these problems even have polynomial time approximation schemes.

We also sketch how our NEXPTIME-hardness results can be used to prove analogous NEXPTIME-hardness results for problems specified using other kinds of succinct specification languages.

Our results provide the **first natural** problems for which there is a proven exponential (and possibly doubly exponential) gap between the complexities of finding exact and approximate solutions.

[1]Email addresses of authors: {hunt,madhav,djr,res}@cs.albany.edu

[2]Supported by NSF Grants CCR 89-03319, CCR 89-05296, CCR 90-06396 and CCR9406611.

[3]Department of Computer Science, University at Albany - SUNY, Albany, NY 12222.

[4]Current Address: Mailstop 47LA, Hewlett-Packard Company, 19447 Pruneridge Avenue, Cupertino, California 95014-9913. Email: rven @cup.hp.com

[5]Current Address: Los Alamos National Laboratory P.O. Box 1663, MS M986, Los Alamos NM 87545. Email: madhav@gardener.lanl.gov. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

1 Introduction and Motivation

A 2-dimensional periodic graph is an infinite graph described succinctly by a **static graph**. A static graph is a directed graph with 2-dimensional integer displacement vectors associated with its edges. The periodic graph G^{∞} is obtained by repeating the static graph G in a 2-dimensional orthogonal grid as follows: For each edge from u to v in G with vector weight y, the copy of u at grid point z is connected with a copy of v at grid point $z + y$. For an integer vector (M, N), the **finite periodic graph** $G^{(M,N)}$ is the subgraph of the infinite periodic graph G^{∞} induced by the vertices associated with nonnegative grid points less than or equal to (M, N). Here M denotes the largest X coordinate and N denotes the largest Y coordinate. A finite periodic graph $G^{(M,N)}$ is **narrow** if for all integer displacement vectors $y =< y_1, y_2 >$ of its static graph, $y_i \in \{-1, 0, 1\}$, $1 \leq i \leq 2$. Figure 1 shows an example of a 2-dimensional periodic graph. It is possible to define periodically specified formulas in a similar manner. A formal definition of periodically specified boolean formulas is given in Section 3.

Periodic graphs (specifications) have applications in a variety of areas such as transportation planning, communication networks, VLSI layout, etc, and consequently have been widely studied (see [CM93, HW92, KMW67, KO91, Or84b, Wa93]) . A second specification language studied in this paper is the *Graph Construction Representation* (G.C.R) specification language of Galperin [Ga82]. The G.C.R specification language is motivated by practical applications in the area of VLSI design.

An important feature of these specifications is that they permit more concise description of objects than the standard (non-succinct) descriptions. Consequently, the complexity of analyzing or otherwise processing a given succinctly specified object can be different from the complexity when the object is presented using a standard description.

The contributions of this paper include the following.

1. For the first time in the literature we prove that a number of natural graph and combinatorial problems are NEXPTIME- or EXPSPACE-complete when instances are specified using 2-dimensional narrow periodic specifications. As an example, we show that the satisfiability problem when instances are specified using 2-dimensional finite narrow periodic specifications is NEXPTIME-complete.

   Previously, Wanke [Wa93] and Höfting and Wanke [HW92] proved the NP-, PSPACE-hardness of a number of combinatorial problems specified using 2-dimensional periodic specifications. Apart from their work, we are not aware of any lower bound results for problems specified using a 2-dimensional narrow periodic specification. Our results should be contrasted with those of Orlin [Or82a] who proved PSPACE-hardness results for a number of problems when specified using 1-dimensional periodic specifications.

   Our results show that problems specified using a two dimensional periodic specification can be "harder" to solve than the corresponding problems specified using 1-dimensional periodic specifications.

2. In contrast, we prove that the optimization versions of several of these NEXPTIME-complete problems have polynomial time approximation algorithms with constant performance guarantees. Some of these problems, even have polynomial time approximation schemes. In particular, these problems have polynomial time approximation algorithms with performance

guarantees asymptotically equal to the best possible performance guarantees for the same problem when instances are specified using standard specifications. The ideas can also be extended to obtain polynomial time approximation algorithms for EXSPACE-hard problems specified using a 2-dimensional narrow periodic specifications.

This is the **first natural** class of problems for which there is an exponential (and possibly a double exponential)[6] complexity gap between solving the problem exactly versus approximately.

3. We prove that a 2-dimensional narrow periodic specification of a finite graph can be translated in polynomial time into a G.C.R. specification of an isomorphic graph. Together with our NEXPTIME-hardness results in (1) above for 2-dimensional narrow periodically specified problems, this yields the first NEXPTIME-hardness results for a number of natural problems specified using the G.C.R. specification language. The G.C.R. specification language is strictly less powerful than the small circuit representation (S.C.R.) [Ga82, GW83, PY86, LB89] used to succinctly represent inputs. Thus our results significantly strengthen several lower bound results proved by Papadimitriou and Yannakakis [PY86], Galperin and Wigderson [GW83] and Lozano and Balcazar [LB89] (See Section 2.3 for a more detailed discussion of this). In Galperin [Ga82] **no** hardness results were presented for graph problems when graphs are represented using G.C.R. specification. The results here are the first such results for problems specified using G.C.R. specifications.

2 Summary of Results

Let Π be a problem posed for instances specified in the standard (non-succinct) manner. We use 2-FPN-Π to denote the problem Π, when instances are specified using 2-dimensional finite periodic narrow specifications. An instance of a periodically specified graph (formula) is a three tuple (G, M, N), where G denotes a static graph (formula),[7] and M and N, denote binary numerals which give the bounds on the X and the Y coordinates of the grid points. The corresponding periodic graph (formula) is the graph $G^{(M,N)}$.

2.1 Hardness Results for 2-Dimensional Periodically Specified Problems

Our first result shows that the problems 3SAT and Pl-3SAT (3SAT problem restricted to instances in which the associated bipartite graph is planar) for 2-FPN specified instances are NEXPTIME-complete.

Theorem 2.1 *2-FPN-3SAT and 2-FPN-Pl-3SAT are NEXPTIME-complete.*

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

[7]A collection of clauses is *narrow* if for all $(w_1(i_1, j_1) \vee w_2(i_2, j_2) \vee w_3(i_3, j_3)) \in C$, $|i_s - i_r|, |j_s - j_r| \leq 1$ for $1 \leq r \leq s \leq k$.

Theorem 2.1 along with known polynomial time local reductions can be used to prove that many graph, combinatorial and satisfiability problems are NEXPTIME-complete when specified using a 2-FPN specification. We first recall the generalized satisfiability problems introduced by Schaefer [Sc78]. In this framework, we have a finite set $\mathbf{S} = \{R_1, R_2, \cdots, R_m\}$ of finite arity logical relations. An **S-formula** is a conjunction of clauses each of the form $R_i(\xi_1, \xi_2, \cdots)$, where $\xi_1, \xi_2, \cdots$ are distinct, unnegated variables whose number matches the rank of $R_i, i \in \{1, \cdots m\}$. The **S**-*satisfiability problem* is the problem of deciding whether a given **S**-formula is satisfiable. Let SAT(**S**) denote the set of all satisfiable **S**-formulas. Given this framework, we define the problems MAX SAT(**S**) as follows: Given an **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 are satisfied. The following theorem yields an infinite collection of 2-FPN specified satisfiability problems and several graph problems that are NEXPTIME-complete.

Theorem 2.2 *(1) For each finite set* **S** *such that the problem SAT(***S***) is NP-complete, the problem 2-FPN-SAT(***S***) is NEXPTIME-complete.*
(2) The following problems are NEXPTIME-hard for 2-FPN specified planar graphs: independent set, 3 coloring, dominating set, vertex cover, partition into triangles, and Hamiltonian path.

Theorems 2.1 and 2.2 hold when M and N are finite and are specified using binary notation. Instead, now suppose M equals infinity. (That is we consider the graph $G^{(\infty,N)}$ which is a 2-dimensional periodic graph finite in the Y direction but infinite in the X direction.) We call such graphs two dimensional semi-finite specified graphs and the associated specifications as 2-SemiFPN specifications. Extending Theorems 2.1 and 2.2 we can show that several combinatorial problems when instances are specified using 2-SemiFPN specifications are EXPSPACE-complete. Thus for example we can show the following:

Theorem 2.3 *The following problems are EXPSPACE-complete for 2-SemiFPN planar specifications: 3SAT, independent set, 3 coloring, dominating set, vertex cover, partition into triangles.*

Analogous NEXPTIME- and EXPSPACE-completeness results hold, for **most** but **not all** of the problems shown to be NP-complete in [Ka72, Sc78]. An example of this is the CLIQUE problem which remains NP-complete even when graphs are specified using 2-SemiFPN specifications.

2.2 Approximation Algorithms for 2-FPN Specified Problems

The hardness results of Theorem 2.1-2.3 motivate the study of polynomial time approximation algorithms with good performance guarantees for these problems. The search for good approximation algorithms probably is all the more important from a practical standpoint, given that the problems take at least 2^{cn} (and possibly $2^{2^{cn}}$) time to solve exactly. We give *simple* polynomial time approximation algorithms for problems shown to be NEXPTIME-complete in Theorems 2.1 and 2.2. As mentioned earlier, this is the first time polynomial time approximation algorithms have been given for natural NEXPTIME-hard problems. Our approximation algorithms are based on the technique of *partial expansion* which we developed in [MH+94] to devise efficient approximation algorithms for level restricted hierarchically specified graphs (formulas) and one-dimensional finite periodically specified graphs (formulas).

Let Π be one of the following problems: MAX 3SAT, MAX 2SAT, MAX 1-3SAT, MAX SAT(S), MAX Independent Set, MIN Dominating Set, Minimum Vertex Cover, Minimum Partition into Triangles. Following Schaefer [Sc78], S denotes a finite set of finite arity boolean relations.

Theorem 2.4 *For all fixed $l \geq 1$ and for all of the problems Π stated above given a 2-FPN specification of an instance of Π, there exists 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, $RT_\Pi(n)$ denotes the running time of the algorithm with input size n which guarantees a performance of $FBEST_\Pi$ for the problem Π and $FBEST_\Pi$ denotes the best known performance guarantee of an algorithm for the problem Π for non-succinctly specified instances.*

As an example, using recent results in [GW94], we get that the problem 2-FPN-MAX 2SAT has a polynomial time approximation algorithm that outputs a solution which is within a factor of $(1-\epsilon)0.878$ of an optimum solution. As a corollary of Theorem 2.4, using the recent nonapproximability results of [AL+92] we get the following:

Theorem 2.5 *The problems Π stated above when specified using 2-FPN specifications, have polynomial time approximation schemes if and only if $\mathbf{P} = \mathbf{NP}$.*

As a second result which follows from Theorem 2.4, we get that all the above problems Π have a polynomial time approximation scheme (PTAS), when restricted to planar instances.

Theorem 2.6 *Given a 2-FPN specification of a planar graph, there are polynomial time approximation schemes for the problems Π listed above.*

As mentioned before, results analogous to Theorems 2.4 and 2.6 hold for 2-SemiFPN specified optimization problems. We remark that recently, there has been interest in the study of efficient approximability of PSPACE-hard functions (See [AC94, CF+93, CF+94, MH+93a, MR+93, MH+94]).

2.3 Applications to G.C.R Specification Language

Next we discuss the hardness results obtained for problems specified using the G.C.R. specification language. As observed Lengauer and Wanke [LW93], the G.C.R specification language provides a natural and practical way of succinctly representing large objects occurring in VLSI design. For instance using this language one can succinctly represent regular graphs like the meshes of trees, trees of meshes, a complete grid, Petersons graph, Hypohamilton graph etc., in such a way that the size of the specification is logarithmic in the size of the actual graph. The G.C.R specification language uses repeated applications of the operations *cloning* and *glueing* to construct bigger graphs. In this model, a graph is represented by a description of a way to construct it. The description starts with a few basic graphs to which a set of graph operations is applied repeatedly to define new graphs.

We first prove the following translation theorem.

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

Theorem 2.7 *Given an instance (Γ, M, N) of a 2-dimensional narrow finite periodic graph $\Gamma^{(M,N)}$, there is a polynomial time algorithm A that outputs a G.C.R specification Δ_Γ of a graph G' which is isomorphic to $\Gamma^{(M,N)}$.*

As a corollary of Theorem 2.7 we can show that many natural graph and combinatorial problems are NEXPTIME-complete. For example, we can show the following:

Theorem 2.8 *The following problems are NEXPTIME-complete for graphs specified using a G.C.R-specification: independent set, dominating set, vertex cover, 3SAT, 1-3SAT, partition into triangles and maximum cut.*

The translation theorem enables us to obtain hardness results for problems specified using G.C.R specifications by translating the known hardness results for problems specified using 2-FPN specifications. Moreover, the theorem for the first time relates two seemingly unrelated succinct specification languages. In the past each of the models considered here have been studied separately; and as a result no attempt was made to find general ways of obtaining hardness and easiness results for problems specified using these models. The results obtained are a part of our attempt to characterize the complexity of various combinatorial problems for different succinct specifications in a unified way.

Theorem 2.8 yields the **first** NEXPTIME-hardness results in the literature for problems specified using G.C.R specifications. It also strengthens several results in [GW83, LB89, PY86] showing that problems like 3SAT when instances are specified using the Small Circuit model (S.C.R.) [GW83, LB89, PY86] are NEXPTIME-complete. This follows from the results in Galperin [Ga82], showing that the S.C.R. specification language is strictly more powerful than the G.C.R specification language. For instance, given a graph G specified using a S.C.R. specification trivial questions such as "Is there an edge in G ? " is NP-complete. On the other hand the same question can be solved easily in polynomial time when G is specified using a G.C.R specification. It is important to note that although we strengthen several results obtained by Papadimitriou and Yannakakis [PY86] and Lozano and Balcazar [LB89], we do not have analogues of the metatheorems given in [PY86, LB89] which give sufficient conditions on the property of graphs for it to be NEXPTIME-hard.

Due to lack of space the rest of the paper consists of selected proof sketches. A detailed exposition of some of the proofs appears in the Appendix.

3 Hardness Results for Periodic Satisfiability Problems

In this section we outline our proof of the NEXPTIME-hardness of 2-FPN-3SAT (finite periodic non-narrow 3SAT). The EXPSPACE-hardness of 2-SemiFPN-3SAT follows by a similar argument. We first give a formal definition of the two dimensional periodic satisfiability problem. The definition is an extension of 1-FPN-3SAT given in Orlin [Or82a].

Definition 3.1 *An instance of 2-FPN-3SAT consists finite set of variables $U = \{u_1, \ldots, u_n\}$. U^∞ denotes the set $\{u_k(i,j) : 1 \leq k \leq n,\ i \in \{0,1,2,\cdots,N\}\ j \in \{0,1,2,\cdots,M\}\}$. 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 $w(i,j)$ is a literal of $U^{M,N}$. A set of clauses C is periodic if the following is true. $(w_1(i_1,j_1) \vee w_2(i_2,j_2) \vee w_3(i_3,j_3)) \in C$ if and*

only if $(w_1(i_1+x, j_1+y) \vee w_2(i_2+x, j_2+y) \vee w_3(i_3+x, j_3+y)) \in C$. *Here* w_j *denotes a literal. A collection of clauses is* narrow *if for all* $(w_1(i_1, j_1) \vee w_2(i_2, j_2) \vee w_3(i_3, j_3)) \in C$, $|i_s - i_r|, |j_s - j_r| \leq 1$ *for* $1 \leq r \leq s \leq k$. *To specify the input for a periodic collection of clauses it suffices to specify all clauses containing at least one of the literals* $\{u_1(0,0), \ldots, u_n(0,0), \overline{u_1}(0,0), \ldots, \overline{u_n}(0,0)\}$.

An instance of 2-FPN-3SAT consists of a finite set of variables U *and a periodic collection of narrow 3 literal clauses* C *defined on* $U^{M,N}$, *where* M *and* N *are specified in binary.*

Theorem 3.2 *The problem 2-FPN-3SAT is NEXPTIME-complete.*

Proof Sketch: The membership in NEXPTIME follows easily by observing that the size of the expanded formula is $2^{c(|G+M+N|)}$, where (G, M, N) is the specification of $G^{M,N}$. Hence a NEXPTIME bounded TM can guess an assignment to the variables and then verify in EXPTIME 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 SAT instance more difficult. In order to overcome this difficulty, our reduction consists of two phases. In the first phase, we start with the given Turing machine M with input $x = (x_1, \ldots, x_n)$ and construct a new Turing machine M_1 which simulates M on x with the following additional properties that

1. If the Turing machine M does not accept x then M_1 on x halts within $2^{c_0 n}$ moves, else
2. If the Turing machine M accepts x then M_1 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 $f(x)$ of 2-FPN-3SAT by a polynomial time reduction from M_1. Now we know that each ID of the Turing machine M_1 is of length $2^{d_0 n}+1$. Since it is a non-deterministic exponential time bounded Turing machine, 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 M_1 being laid out in the plane along the Y-axis. Two consecutive ID's of M_1 are laid out next to each other in the plane. For the sake of exposition we will refer to the X-axis as the *time line.* In order to simulate the behavior of M_1 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. The counter can be simulated by means of $d_0 n + 1$ boolean variables. 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. 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 $f(x)$ is described as follows. Let $TAPE(y, t)$ denote the y^{th} symbol in the t^{th} ID. $TAPE(y, t)$ 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 M_1. The number of variables needed to encode $TAPE(y, t)$ depends only on the machine M_1. The static formula is given by $R(t, y) = f_1(t, y) \wedge f_2(t, y) \wedge f_3(t, y)$. We now describe each of the subformulas $f_i(t, y), 1 \leq i \leq 3$ separately.

1. **Counter Updating: Formula** $f_1(t, y)$

$$c_t(y, t+1) \equiv (c_t(y, t) + 1) \pmod{2^{d_0 n} + 1}$$

$$c_t(y+1,t) \equiv c_t(y,t) \;\; 0 \leq c_y(t,y) < 2^{d_0 n}$$
$$c_y(y+1,t) \equiv (c_y(y,t)+1) \pmod{2^{d_0 n}+1}$$
$$c_y(y,t+1) \equiv c_y(y,t) \;\; 0 \leq c_t(t,y) < 2^{d_0 n}$$

2. **Implicit Initialization: Formula** $f_2(t,y)$

$$(c_y = 0) \wedge (c_t = 0) \Rightarrow TAPE(y,t) = \#$$
$$(c_y = 1) \wedge (c_t = 0) \Rightarrow TAPE(y,t) = (q_0 x_1)$$
$$\vdots$$
$$(c_y = n) \wedge (c_t = 0) \Rightarrow TAPE(y,t) = x_n$$
$$(n+1 \leq c_y \leq 2^n) \wedge (c_t = 0) \Rightarrow TAPE(c_y, c_t) = B$$

3. **Consistency Checking: Formula** $f_3(t,y)$

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

Next, we explain what each of the $f_i(t,y)$ denote.

Counter Updating Formula: $f_1(t,y)$

Equations 1 and 2 describe the counter c_t. Equation 1 says that the counter c_t is reset after every $2^{d_0 n}+1$ time units. Equation 2 says that the counter value for a given value of t remains the same. Equation 3 and 4 describe the desired properties of the counter c_y. Equation 3 says that the counter c_y resets after every $2^{d_0 n}+1$ steps. Equation 4 says that the counter value c_y has the same value for a given y coordinate.

Implicit Initialization: $f_2(t,y)$

$f_2(t,y)$ can be thought of as a way to implicitly check that the Turing machine starts right. Observe that the counters c_y and c_t are implicitly initialized. The initialization condition say that if both the counter values are 0, then we have # as the tape symbol and so on.

Consistency Check: $f_3(t,y)$

$f_3(t,y)$ ensure the consistency of the tape symbols, i.e. the contents of the tape cells i, $i+1$ and $i+2$ in ID_t determine the contents of the tapecells i, $i+1$ and $i+2$ in ID_{t+1}. The Consistency function of course depends on the state transition relation. $f(x)$ is the conjunction of clauses as defined by the above 3 set of formulas. It is easy to see that these equations can be transformed into an equivalent narrow 3CNF formula $f(x)$ whose size is polynomial in n, where n is the size of the input to x.

The expanded finite periodic 3SAT instance is $\bigwedge_{y=0,t=0}^{y=M,t=N} R(y,t)$, where M and N are suitably chosen large numbers.

We now prove the correctness of our reduction. If the Turing machine M accepts x then we know that M_1 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 force the formula $\bigwedge_{y=0,t=0}^{y=M,t=N} R(y,t)$ 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 M_1 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 $f_2(t, y)$ is true and from then on, it is ensured that the TM M_1 goes through the simulation correctly.

The above two conditions imply that if the formula $\bigwedge_{y=0,t=0}^{y=M,t=N} R(y, t)$ is satisfied then the Turing machine M accepts x. ■

4 Approximation Algorithms and Schemes for 2-FPN-specified problems

4.1 Meaning of Approximation Algorithms for Periodically Specified Problems

First, it is important to understand what we mean by a *polynomial time approximation algorithm* for a graph problem, when the graph is specified using a 2-FPN specification. Our definition of an approximation algorithm is best understood by means of an example.

Example: Consider the maximum independent set problem, where the input is a 2-FPN specification of a graph G, we wish to compute the size of a maximum independent set of vertices in G.

We provide efficient algorithms for the following versions of the 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-FPN 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-FPN 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. ■

This is a natural extension of the definition of approximation algorithms in the flat (i.e. non-succinct) case. This can be seen as follows: In the flat case, the number of vertices is polynomial in the size of the description. Given this, any polynomial time algorithm to determine if a vertex v of G is in the approximate maximum independent set can be modified easily into a polynomial time algorithm that lists all the vertices of G in the approximate maximum independent set. The concept of approximation algorithms for problems specified using 2-FPN specifications can be extended in

a natural way so as to apply to problems specified using 2-SemiFPN specifications. The extension is along the lines of the definition of periodic optimization problems defined in Orlin [Or82a] and Cohen and Megiddo [CM91]. We omit the details in this abstract.

4.2 Basic Technique

The basic idea behind our approximation algorithms involves the conversion of solutions derived by a *local* algorithm on graphs induced by certain bounded grid points to a solution to 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 used by Baker [Ba83], Hochbaum and Maass [HM85, HM87] and Hunt et al. [HM+94a] 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-FPN 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 at hand in time polynomial in the size of the specification.

4.3 Approximation Scheme for the Maximum Independent Set Problem for 2-FPN Specified Graphs

Input: An instance (G, M, N) of a periodic graph $G^{M,N}$ and an $\epsilon > 0$

Algorithm: 2-FPN-MAX-IS

1. Let $k = \lceil 1/\epsilon \rceil - 1$.
2. **For** each i, $0 \leq i \leq k$ **do**
   (a) Partition the graph into r disjoint sets $G_{i,1} \cdots G_{i,r}$ 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, $1 \leq j \leq r$ **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. $G_{i,j}^{i_1} = \bigcup_{1 \le j_1 \le s_j} G_{i,j}^{i_1,j_1}$

C. For each $G_{i,j}^{i_1,j_1}$, $1 \le j_1 \le s_j$ 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. $IS(G_{i,j}^{i_1}) = \bigcup_{1 \le j_1 \le s_j} IS(G_{i,j}^{i_1,j_1})$

(d) $IS(G_{i,j}) = \max_{0 \le i_1 \le k} IS(G_{i,j}^{i_1})$

(e) $IS(G_i) = \bigcup_{1 \le j \le r} IS(G_{i,j})$

3. $IS(G) = \max_{0 \le i \le k} IS(G_i)$

The proof of correctness and the performance guarantee is outlined in the Appendix.

5 Conclusions and Discussion

We have looked at large instances of graph and satisfiability problems created in a natural and simple ways, namely by repeating a single graph or formula in a multidimensional grid and then connecting vertices placed at given grid points to vertices placed at neighboring grid points. The size of large objects so created can be *exponential* in the *size* of the object being replicated. In spite of the simple repetitive nature of the constructed object, the difficulty in solving certain NP-complete problems blows up with the size of the object being specified. Thus, several of these problems are NEXPTIME-complete when complexity is measured in the size of the original (periodic) description. In contrast, the simple repetitive nature also enables us to design efficient polynomial time approximation algorithms with good performance guarantees **even when** complexity is measured in the size of the periodic specification. The complexity of approximation remains polynomial in the size of the description. Thus we have a striking contrast that problems which are NEXPTIME-complete to solve exactly can be efficiently approximated in polynomial time.

The fact that there is an exponential gap between solving the problem exactly and approximately for such succinctly specified objects *may* prove to be useful in trying to tackle other questions in complexity theory. For example, the results obtained in this paper raise the possibility of proving the recent non-approximability results **without** using the machinery of interactive proof systems. Given that the decision problem 2-FPN-3SAT is NEXPTIME-hard it is conceivable that one could prove Theorem 2.5 directly without using the recent results from interactive proof systems. If this were the case, then by Theorem 2.4 one gets a **direct** proof of the fact that MAX 3SAT does not have a polynomial time approximation scheme unless P = NP.

The simplicity of the graph (formula) obtained in the proof of Theorem 2.1 makes it a good candidate for being specified using other kinds of succinct descriptions. In particular, it can be specified using G.C.R. specifications. This shows for the first time that natural problems specified using this model are also NEXPTIME-hard to solve. The G.C.R. model is generally acknowledged as a natural and useful way of describing large real-world objects such as circuits and VLSI designs.

Acknowledgements:

We wish to thank Prasad Chalasani, Ashish Naik, R. Ravi, Ravi Sundaram, for reading the draft and suggesting improvements. We also thank Anne Condon, Thomas Lengauer and Egon Wanke for helpful discussions.

References

[AC94] S. Agarwal and A. Condon, "On Approximation Algorithms for Hierarchical MAX-SAT," Manuscript, November, 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 *Journal of the ACM (J. ACM)*, Vol. 41, No. 1, 1994, pp. 153-180.

[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.

[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 Debators 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.

[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.

[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.

[HM85] D.S. Hochbaum, W. Maass, "Approximation Schemes for Covering and Packing Problems In Image Processing and VLSI," *Journal of the ACM (J. ACM)*, Vol. 32, No. 1, 1985, pp. 130-136.

[HM87] D.S. Hochbaum, W. Maass, "Fast Approximation Algorithms for a Nonconvex Covering Problem," *Journal of Algorithms*, Vol. 8, 1987, pp. 305-323.

[HW92] F. Höfting and E. Wanke, "Minimum Cost Paths in Periodic Graphs," Tech Report 99, Universit ät-Gesamthochschule-Paderborn, FRG, April, 1992.

[HU] J. E. Hopcroft, and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation* Addison Wesley, Reading MA., 1979.

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

[HM+94a] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz and R. E. Stearns, "A Unified Approach to Approximation Schemes for NP- and PSPACE-Hard Problems for Geometric Graphs", to appear in *Proc. 2nd Annual European Symposium on Algorithms (ESA'94)*, September, 1994.

[HM+94b] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, D.J. Rosenkrantz and R.E. Stearns, "A Unified Approach to Prove Both Easiness and Hardness Results for Succinct Specifications," Technical Report No. 94-5 Department of Computer Science, University at Albany, May 1994.

[HM+94c] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, D.J. Rosenkrantz and R.E. Stearns, "Designing Approximation Schemes Using L-Reductions," to appear in *Proc. 13th Foundations of Software Technology and Theoretical Computer Science (FST-TCS'94)*, also available as Technical Report No 94-7, Department of Computer Science, University at Albany, May 1994.

[HM+94d] H.B. Hunt III, M.V. Marathe, V. Radhakrishnan, D.J. Rosenkrantz and R.E. Stearns, "On the Complexity and Approximability of Periodic and Hierarchical Specifications," Technical Report No 94-10, Department of Computer Science, University at Albany, August 1994.

[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.

[Jo74] D.S. Johnson, "Approximation Algorithms for Combinatorial Problems," *Journal of Computer and System Sciences (JCSS)*, Vol. 9, 1974, pp. 256-278.

[Kan92b] V. Kann, "On the Approximability of NP-complete Optimization Problems," Ph.D. Thesis, Dept. of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm, Sweden, May 1992.

[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.

[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.

[Li82] D. Lichtenstein, "Planar Formulae and their Uses", *SIAM Journal on Computing*, Vol 11, No. 2, May 1982 , pp. 329-343.

[LB89] A. Lozano, and J.L. Balcazar, " The Complexity of Graph Problems for Succinctly Represented Graphs," *Proc. 15th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'89)* , Springer Verlag, LNCS, Vol. 411, 1989, pp. 277-286.

[MH+93a] M.V. Marathe H.B. Hunt III, and S.S. Ravi, "The Complexity of Approximating PSPACE-Complete Problems for Hierarchical Specifications," *Proc. 20th International Colloquium on Automata Languages and Programming (ICALP)*, July, 1993, pp. 76-87.

[MR+93] M.V. Marathe, V. Radhakrishnan, H.B. Hunt III, and S.S. Ravi, "Hierarchical Specified Unit Disk Graphs," *Proc. 19th International Workshop on Graph-Theoretic Concepts in Computer Science (WG '93)*, June, 1993, pp. 21-32.

[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.

[Or82b] J.B. Orlin, "Minimizing the Number of Vehicles to Meet a Fixed Periodic Schedule: An Application to Periodic Posets," *Operations Research*, No. 30, 1982, pp. 760-776.

[Or83a] J.B. Orlin, "Dynamic Matching and Quasidynamic Fractional Matchings I," *Networks* Vol. 13, 1983, pp. 551-562.

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

[Or85] J.B. Orlin, "Maximum Throughput Dynamic Network Flows," *Mathematical Programming*, 1985.

[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.

[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.

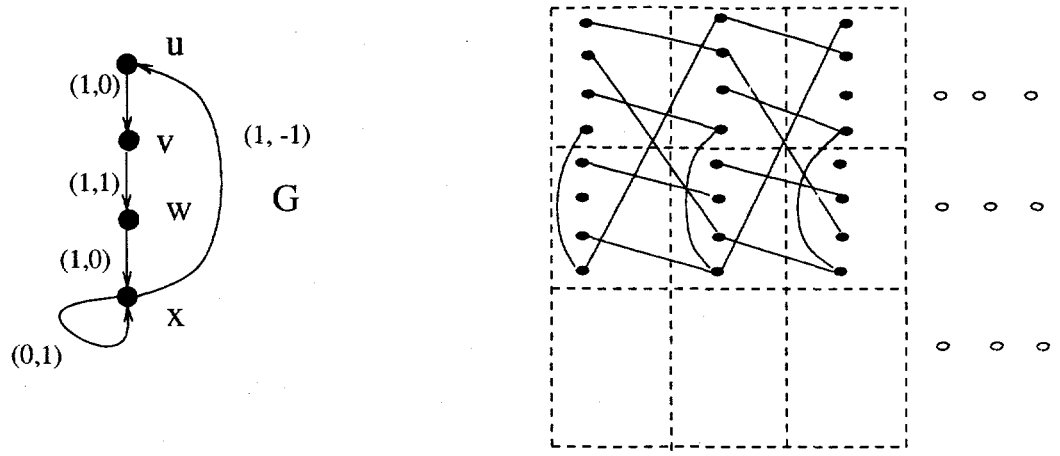


Figure 1: Example of a 2-dimensional narrow periodically specified graph. The static graph with 2-dimensional displacement vectors and its associated expanded graph.

Appendix

Analysis of Algorithm 2-FPN-MAX-IS

We now prove the correctness and the performance guarantee of the algorithm 2-FPN-MAX-IS. To do this we prove a series of lemma which hold for every iteration of the innermost loop of the algorithm.

Lemma 5.1 *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 can 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 5.2 *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 shows the possible different equivalence classes. ■

Lemma 5.3 *Each of the subgraphs* $G_{i,j}^{i_1,j_1}$ *obtained in Step 2.(c).i.B is disjoint.*

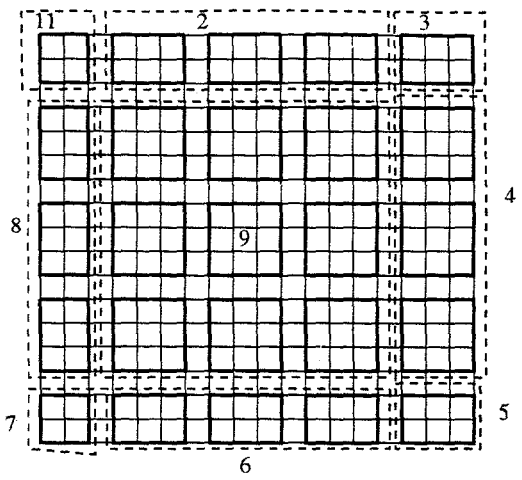


Figure 2: Figure showing the possible equivalence classes as a result of decomposition. The black squares denote subgraphs and the black dotted lines denote the equivalence classes.

Proof Sketch: Follows from the property of instances specified by 2-FPN 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 5.4

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

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

Theorem 5.5 $|IS(G_{i,j})| \geq (\frac{k}{k+1}) \cdot FBEST \cdot |OPT(G_{i,j})|$.

Theorem 5.6 $|IS(G)| \geq (\frac{k}{k+1})^2 \cdot FBEST \cdot |OPT(G)|$. *Here FBEST denotes the performance guarantee of the best algorithm known to solve the independent set problem.*

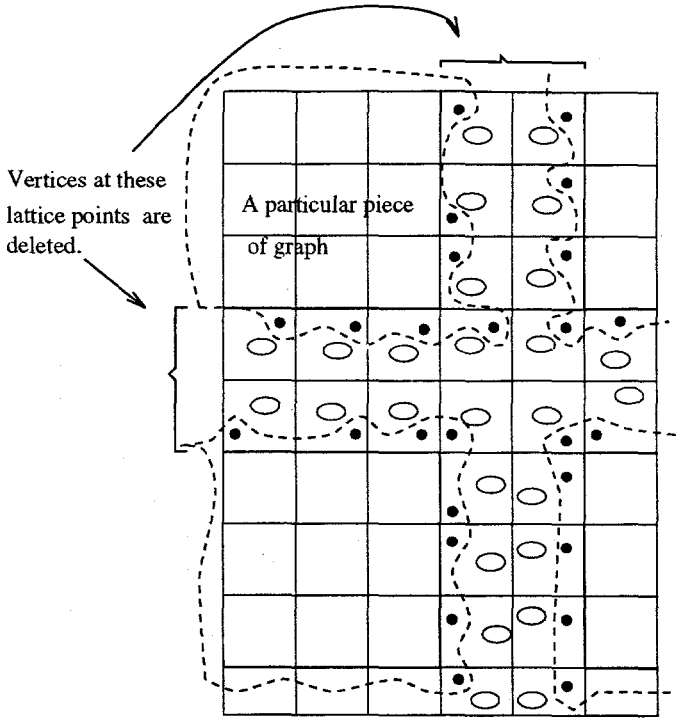


Figure 3: Basic Idea behind the approximation algorithm for 2FPN-MAX Pl-SAT(S). The black dots represent variables and the ellipses denote clauses. The figure depicts which set of clause to remove and the redistribution of the variables.

Approximating 2FPN-MAX Pl-SAT(S)

We now discuss how to obtain a approximation scheme for the problems 2FPN-MAX SAT(S) when the bipartite graph is planar. We first recall a theorem from [HM+94c].

Theorem 5.7 *For any* **S**, *the problem 2FPN-MAX Pl-SAT(S) has a polynomial time approximation scheme.*

The basic idea behind the approximation schemes is as follows: For each $0 \leq i \leq 2l$ in steps of 2, we remove the defined clauses which are at grid points with X coordinate j and $j+1$, such that $j = i \mod (l+1)$. This break the bipartite graph into a number of small disjoint subgraphs. We then repeat the above process by now removing clauses with Y coordinates j and $j+1$, such that $j = i \mod (l+1)$. This break the consecutive vertical strips into smaller disjoint subgraphs. As shown in [HM+94c] the problem MAX Pl-SAT(S) has an approximation scheme, which runs in linear time. Using this algorithm, we find a near optimal solution for each small subgraph. The union of the clauses satisfied for each subgraph gives a solution for a given value of i. We pick the best solution for different values of i. As in the case of maximum independent set problem, we can show that at least for one of the iterations $0 \leq t \leq l$, at most $1/(2l+2)$ clauses are in levels t and $t+1$ such that $t = i \mod (l+1)$. This ensures that the best assignment to the variables over all values of i is at least $(\frac{l}{l+1})^3$ of an optimal assignment to the variables of the 2-FPN-Pl3SAT instance. The running time of the algorithm is now $O(l^2 \cdot |G|)$.

Translation Theorem

In this subsection we discuss our lower bound results for graph problems when graphs are specified using a G.C.R. specification language. In this model, a graph is represented by a description of a way to construct it. The representation starts with a few basic graphs to which a set of graph operations is applied repeatedly to define new graphs. The last graph defined is the graph we consider as the graph specified by the above specification. The graph operations used here are (i) Cloning (replication) (ii) Vertex Glueing and (iii) Edge Glueing.

1. **Cloning:** Given a graph G with a associated set of lists S_G the operation clone(G) defines a new graph G' which is isomorphic to G. The set of lists $S_{G'}$ is defined to be the isomorphic image of S_G.

2. **Vertex Glueing** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs and let $L_1 = (a_1, \ldots, a_k)$, $a_i \in V_1$, $L_2 = (b_1, \ldots, b_k)$, $b_i \in V_2$. A graph G is a vertex glue of G_1 and G_2 via L_1 and L_2 (denoted by $G = G_1(L_1) \circ_v G_2(L_2)$) if it is defined as follows:

   (a) $V(G) = V_1 \cup (V_2 - \{b_1, \ldots b_k\})$

   (b) $E(G) = E_1 \cup \{(v,w) | (v,w) \in E_2$ where $v, w \in V_2 - \{b_1, \ldots b_k\}\} \cup \{(v, a_i) | v, \in V_2 - \{b_1, \ldots b_k\}$ where $(v, b_i) \in E_2\} \cup \{(a_i, a_j) | (b_i -, b_j) \in E_2\}$

3. **Edge Glueing:** Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs and let $L_1 = (a_1, \ldots, a_k)$, $a_i \in V_1$, $L_2 = (b_1, \ldots, b_k)$, $b_i \in V_2$. A graph G is a edge glue of G_1 and G_2 via L_1 and L_2 (denoted by $G = G_1(L_1) \circ_e G_2(L_2)$) if it is defined as follows:

   (a) $V(G) = V_1 \cup V_2$

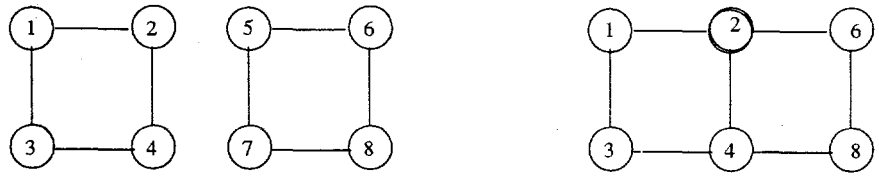
   (b) $E(G) = E_1 \cup E_2 \cup \{(a_i, b_i) | 1 \leq i \leq k\}$

   The set of lists S_G associated with G is defined as follows:
   Let $S_{G_1} = \{F_1, \ldots, F_m\}$ and $S_{G_2} = \{M_1, \ldots, M_n\}$, then a list L in S_G is either $L = F_i$ or $L = M_i$ or $L = F_i \circ M_j$, where $\circ$ denotes the concatenation of two lists. Throughout this discussion we assume that the graphs are simple and we omit duplicates. Figure 4 illustrates the vertex and edge glueing operations described above.

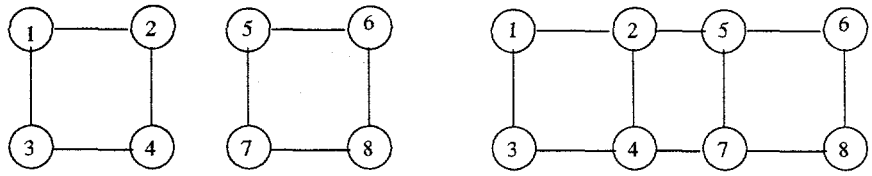
We are now ready to prove our translation theorem.

Proof of Theorem 2.7: In this proof we will construct a GCR specification using the vertex glueing and cloning operations. Consider the static graph Γ specifying the periodic graph. Construct a auxiliary graph Γ_1 which consists of four copies of Γ, denoted by $\Gamma_{0,0}, \Gamma_{0,1}, \Gamma_{1,0}, \Gamma_{1,1}$. There is an edge between two vertices $u, v \in \Gamma_1$ if and only if $u \in \Gamma_{i,j}$, $v \in G_{k,l}$ with the constraint that there is an edge between u and v in Γ with vector weight $(|k-i|, |l-j|)$.
$M = N = 1$: In this case the GCR specification simply consists of the graph Γ_1.
If not we build a GCR specification as follows:

1. $G_1 = \Gamma_1, (L_e, L_w, L_n, L_s)$ where the lists are defined as follows.
   $L_e = V(\Gamma_{0,0}) \cup V(\Gamma_{0,1})$ $L_w = V(\Gamma_{1,0}) \cup V(\Gamma_{1,1})$
   $L_n = V(\Gamma_{0,0}) \cup V(\Gamma_{1,0})$ $L_s = V(\Gamma_{0,1}) \cup V(\Gamma_{1,1})$



(a)



(b)

Figure 4: Example of Vertex and Edge Glueing. (a) Vertex Glueing: Graphs G_1 and G_2. $L_{G_1} = \{2,4\}$ and $L_{G_2} = \{5,7\}$. After the Vertex Glueing operation is performed 2 merges with 5 and 4 merges with 7. (b) Edge Glueing: In this case we add edges $(2,5)$ and $(4,7)$.

2. $G_2 = clone(G_1)$

3. $G_3 = G_1(L_w) \circ_v G_2(L_e)$ and
   $L_e(G_3) = L_e(G_1)$, $L_w(G_3) = L_w(G_2)$,
   $L_n(G_3) = L_n(G_1) \circ L_n(G_2)$, and $L_s(G_3) = L_s(G_1) \circ L_s(G_2)$.

4. $G_4 = clone(G_3)$

5. $G_5 = G_4(L_w) \circ G_3(L_e)$
   $\vdots$

6. $G_{2p} = G_{2p-2}(L_w) \circ G_{2p-1}(L_e)$

   Remark: $G_{2p} = (\Gamma)^{(M,2)}$. Furthermore assume that $S(n)$ denotes the number of copies of Γ in the graph defined by G_n. Then it can be seen that the following set of recurrence equations hold:

$$S(2n) = S(2n-1) + S(2n-2) - 2$$

$$S(2n-1) = S(2n-2)$$

   Combining the two equations the number of copies of Γ obtained as a result of expanding G_{2n} is given by $S(2n) = 2S(2n-2) - 2$. Solving the recurrence we get that in $O(\log |M|)$ steps we can specify the graph $\Gamma^{(M,2)}$.

7. Now we do a similar operation merging the lists L_n and L_s appropriately and in turn getting graph which is big enough in the Y-direction. In particular we do the following.

8. $H_1 = clone(G_{2p})$

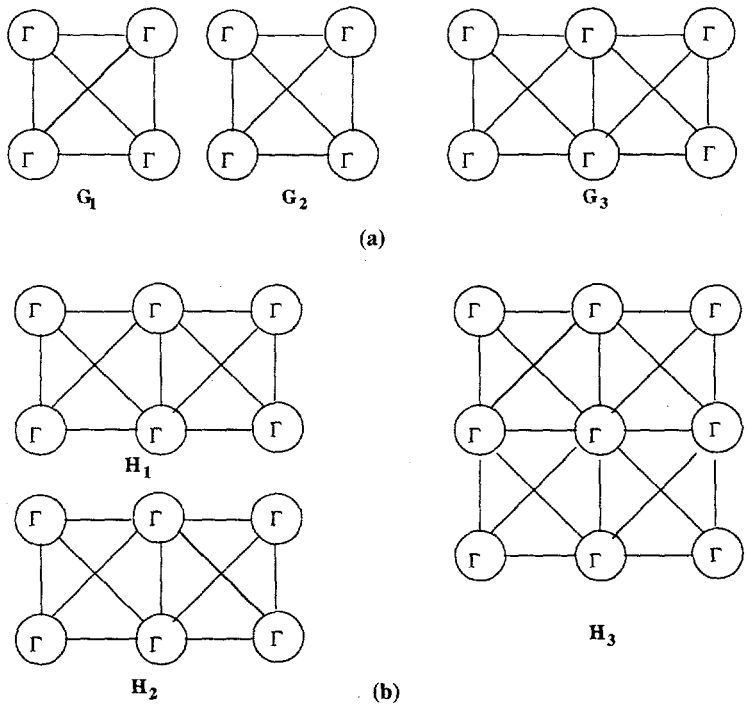


Figure 5: Example illustrating how to obtain a GCR specification. In our example we give a GCR specification of $\Gamma, 2, 2$. (a) The east-west vertex glue to get G_3. (b) H_1 and H_2 are two clones of G_3 and we do a north-south vertex glue to get H_3.

9. $H_2 = clone(H_1)$

10. $H_3 = H_1(L_s) \circ_v H_2(L_n)$ and
$L_n(H_3) = L_n(H_1)$, $L_s(H_3 = L_s(H_2)$, $L_e(H_3) = L_e(H_1) \circ L_e((H_2)$, and $L_w(H_3) = L_w(H_1) \circ L_w((H_2)$.

11. $H_4 = clone(H_3)$

12. $H_5 = H_4(L_w) \circ H_3(L_e)$
$\vdots$

13. $H_{2q} = H_{2q-2}(L_w) \circ H_{2q-1}(L_e)$

Remark: $H_{2q} = \Gamma^{(M,N)}$. By arguments similar to thsoe before we get that for $q = O(\log |N|)$ steps we obtain description of the graph $\Gamma^{(M,N)}$.