Maximization Versions of "Lights Out" Games in Grids and Graphs

Article · September 2000

Source: CiteSeer

CITATIONS
20

READS
59

1 author:

John Goldwasser
West Virginia University
48 PUBLICATIONS **369** CITATIONS

SEE PROFILE

Maximization Versions of "Lights Out" Games in Grids and Graphs

John Goldwasser† and William Klostermeyer‡
†Dept. of Mathematics ‡Dept. of Computer Science
West Virginia University
Morgantown, WV 26506
E-mail: jgoldwas@math.wvu.edu, wfk@cs.wvu.edu

April 17, 2007

Abstract

Complexity results and algorithms are given for the problem of maximizing the number of off vertices (switches) in graphs and $m \times n$ rectangular grids. When a switch is toggled, it and its neighbors change state. It is shown that the problem is NP-complete in graphs and a simple approximation algorithm is given as well as a non-approximability result. Fixed-parameter problems are studied in grids and graphs and an algorithm given for $m \times n$ grids that turn at least $mn - \frac{m}{2}$ vertices off, $m \leq n$. It is shown that $m \times n$ grids exist for which at most $mn - \frac{m}{log_2 m}$ vertices can be turned off.

1 Introduction

Consider a simple, undirected graph in which each vertex represents a switch. Each switch can be in the "on" or "off" position. We have an *activation* operation which consists of changing the state of a switch. However, when the state of a switch is changed, the state of each adjacent switch also changes. Given an initial setting, or configuration, of the switches, we seek to maximize the number of switches that are in the off state by activating a subset of the vertices. Note that it is never necessary to activate a switch more than once. The problem has been previously studied in the context of determining when it is possible to get all the switches off and in this context is known as the *all-parity dominating set problem.* The problem in 3×3 grids was studied in [12], [7] and the $m \times n$ grid graph problem was studied in [15], [2], [9], and [10]. A 5×5 example of the problem may be found in the hand-held electronic computer game "Lights

Out." The all-parity dominating set problem has also been studied for classes of graph such as trees, cycles, and series-parallel graphs by Sutner [15] and by Amin, Clark and Slater [2-4]. Alon and Spencer study a related maximization problem, called "unbalancing lights" in [1].

In section 2 we show that maximizing the number of off switches is NP-complete, in general, and a simple approximation algorithm is given with a performance ratio of less than two. It is also shown that there exists an $\epsilon > 0$ such that no polynomial time approximation algorithm can have performance ratio less than $1 + \epsilon$ unless $P = NP$. In section 3 we discuss the problem of maximizing the number of off switches in grid graphs. A fast algorithm is given that achieves at least $mn - \lceil \frac{m}{2} \rceil$ switches off in a grid graph with mn vertices, where $n \geq m$. A simple polynomial time algorithm is given that determines if a graph (with some initial on/off configuration) can be transformed so that at least $mn - c$ vertices are off for a constant c. Another polynomial time algorithm is given to decide if all initial configurations of a graph can be transformed so that at least $mn - c$ vertices are off for a constant c. The latter algorithm is based on coding theory and enables us to show that there exist grid graphs of dimension $m \times n, m \leq n$, for which at most $mn - \frac{m}{log_2 m}$ vertices can be turned off. Some conjectures and open problems, both combinatorial and algorithmic, are suggested.

2 NP-completeness result

2.1 NP-completeness proof

Let $G = (V, E)$ be a simple graph with $|V| = n$. We formulate the decision problem as follows. "Using a sequence of activation operations, can G (with some initial on/off configuration) be transformed to a graph with at least k switches in the off state." Certain restricted cases are solvable in polynomial time, for example, $k = n$ [9], [15]. We call this the *Maximizing Off Switches (MOS)* problem.

Theorem 1 *Maximizing Off Switches is NP-complete.*

Proof: The problem is clearly in NP, since we can guess a subset of the vertices to activate and count if at least k vertices are in the off state as a result.

To show the problem is NP-hard, we do a reduction from MAX-3SAT [cf. 8] and utilize the NP-complete variant of MAX 3-SAT in which each variable occurs in exactly five clauses [5]. Simpler reductions of the same flavor exist (from 1-in-3 3SAT with no negated literals, for example) but our reduction has an added benefit, as we shall see in section 2.2.

Let S be an instance of MAX-3SAT with m clauses and n variables. Assume without loss of generality that no clause contains both a variable and its negation. For each clause U_i create seven *clause* vertices $c_{i_1}, \ldots, c_{i_7}$ and also create seven *extra* vertices $e_{i_1}, \ldots, e_{i_7}$ and connect e_{i_j} to c_{i_j} with an edge.

Let U_i be a clause with literals (in positive or negated form) u, v, w. Create *variable vertices* u_i, v_i, w_i (labeled as negated, if necessary) and one other vertex x_i. Connect u_i, v_i, w_i, x_i into a complete subgraph. Call this K_4 a *variable component subgraph.* Now connect each of u_i, v_i, w_i to c_{i_1}, connect c_{i_2} to both u_i and v_i, connect c_{i_3} to u_i and w_i, and connect c_{i_4} to v_i and w_i. Then connect u_i to c_{i_5}, v_i to c_{i_6} and w_i to c_{i_7}. Together call these eighteen vertices the *clause component for* U_i Finally, for each variable u do the following. Let u (or its negation) be in clauses $U_{i_1}, U_{i_2}, \ldots, U_{i_5}$. For each u_{i_j}, u_{i_q} pair, $j \neq q$ and where u_{i_j} and u_{i_q} are literals corresponding to the same variable, construct a *truth setting component* consisting of ten vertices (five pair of adjacent vertices) and connect u_{i_j} and u_{i_q} each to the same one of each pair of truth setting vertices, as shown in Figure 1. Note that in Figure 1, the truth setting component is not shown for the a variables, due to lack of space. Thus five of the truth setting vertices in each truth setting component will have degree three and five will have degree one. Let the five truth setting vertices of degree one be initially "off" and the five truth setting vertices of degree three be initially "on" if u_{i_j} and u_{i_q} are complements of one another and initially "off" otherwise. In Figure 1, vertices that are initially "off" are shaded. Call the resultant graph G and set all the vertices in G to be in the on state, except for the extra vertices which are all initially off and the five vertices in each truth setting component that are initially off. Note that G contains $18m + z$ vertices where z is the number of truth-setting vertices. Since each variable appears in five clauses, G has $18m + 60m$ vertices. G can obviously be constructed in polynomial time.

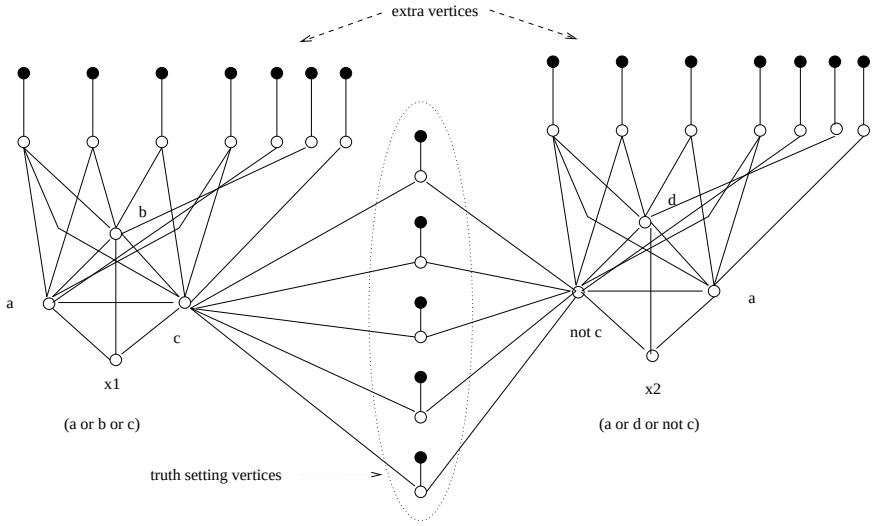

Figure 1. Reduction from MAX 3-SAT to MOS

Truth Setting Component Not Shown for a

Claim: S can have c or more clauses satisfied if and only if we can get at least $f = 15c + 11(m - c) + z$ vertices to be in the off state using the activate operation, where z is the number of truth-setting vertices in G.

First suppose there is a truth assignment for S satisfying at least c clauses. Let T be the set of variables in S that are assigned "true" in this assignment. Clearly, by activating the vertices corresponding to the variables in T, we cause all the variable vertices to be off (since we can activate the x_i vertices if need be) and four of the seven clause vertices of the clause component U_i to be off, for each i such that U_i is a satisfied clause in S, as well as all the extra vertices and truth setting vertices in G being off. Using this truth assignment, each clause component corresponding to an unsatisfied clause in S can be made to have eleven vertices off (i.e., the extra and variable vertices).

Now suppose G can be transformed so that at least f vertices are off. First observe that it is not possible for there to be more than fifteen off vertices in any clause component (out of the total of eighteen per clause component) in G after a sequence of activations, regardless of which vertices are activated. Second, if at least one variable vertex is activated in a clause component then four clause vertices will be off as a result. Also note that clause components in which no variable vertices are activated can be made to have eleven vertices off, since we can activate the x_i vertex in that component. It is easy to see that activating clause or extra vertices is of no use in trying to create additional off vertices in a clause component, since extra vertices have degree one and are initially off. That is, it is not

possible to get more than eleven vertices off in a clause component without activating at least one variable vertex in that component. Finally, note that we must use a legal truth assignment, i.e. we cannot get f or more vertices off by activating a variable vertex and its negation since that would not enable us to get all ten vertices off in some truth setting component. An attempt to activate both u_i and $\overline{u_i}$ can create four off clause vertices in a clause component, but it creates five on vertices the corresponding truth setting component. The theorem then follows since it is only productive to activate variable and x_i vertices and we may assign positive (non-negated) variables in S corresponding to the activated variable vertices the value "true" and vice versa for activated negated variable vertices. □

2.2 Approximation Algorithms

It was shown in [15] and in [6], [9] that every graph contains a subset of vertices (called an *odd dominating set*) whose activation changes the state of every vertex. Such a subset can be found in polynomial time, although finding the smallest such subset is NP-complete [15]. Hence given G, we can always ensure that at least one-half of the vertices are off by activating an odd dominating set if the number of off vertices is initially less than $\lceil \frac{n}{2} \rceil$. This implies the existence of a polynomial time approximation algorithm with performance ratio $2 - \lfloor \frac{2}{n} \rfloor$, since we can initially test the graph in polynomial time to see if all the vertices can be switched off. (Using the results from section 3.2 this can be improved to $2 - \lfloor \frac{c}{n} \rfloor$ for any constant c). We can now state a corollary of Theorem 1.

Corollary 2 *There exists a fixed $\epsilon > 0$ such that no polynomial time algorithm for Maxizing Off Switches can have a performance ratio of less than $1 + \epsilon$ unless $P = NP$.*

Proof: Observe that the reduction given in Theorem 1 is a *gap-preserving reduction* [5] with parameters $(1, (1-\delta)^{-1})$, $(\frac{75}{78}, (1 - \frac{7\delta}{78})^{-1})$ from MAX 3-SAT to MOS. That is, if all the clauses in the MAX-3SAT instance, S, are satisfiable, then at least $\frac{75}{78}N$ of the vertices in the MOS graph can be turned off, where N is the number of vertices in the graph produced by the reduction, G. This is because there are $78m$ vertices in G and if S is satisfiable, at most $3m$ vertices (3 per clause) will be left on in G by an optimal set of activations. If, for each $\delta > 0$, the fraction of clauses in S that can be satisfied is less than $1 - \delta$, then the number of vertices that can be turned off is less than $\frac{71\delta}{78}N$. This is because there are 78 vertices in G for each of the m clauses in S (including all the truth-setting vertices) and because each unsatisfied clause corresponds to at least seven on vertices in G, by the argument given in the proof of Theorem 1. □

We believe it to be quite difficult to devise a better approximation algorithm than the simple one given above.

Open Problem 1 *Devise a polynomial time approximation algorithm for MOS with performance ratio less than* $2 - \lfloor \frac{c}{n} \rfloor$*, where* c *is a constant.*

Evidence of the difficulty of this question is given by Hastad [11], where it is shown that for the more general problem of finding the largest subset of consistent equations from a set of linear equations over $GF(2)$, no polynomial time approximation can have a performance ratio less than $2 - \epsilon$, for any $\epsilon > 0$, unless $P = NP$.

3 Two Fixed-Parameter Problems

We now consider two fixed-parameter versions of the MOS problem. Let c be a constant and G a graph with N vertices. Denote by *c-MOS* the decision problem, "can G (with some initial on/off configuration) be made to have at least $N - c$ vertices off." Obviously, the c-MOS problem can be answered in polynomial time by exhaustive search techniques that work for other NP-complete problems such as minimum dominating set. To do this, one simply has to solve at most $\begin{pmatrix} N \\ c \end{pmatrix}$ sets of linear equations. Each equation in a set of linear equations specifies a vertice's neighbors (including itself) and the objective: either having an even or odd number of neighbors activated. For $c = 1$, for example, the running time of this algorithm is $O(N^4)$, using standard methods such as Gaussian elimination to solve the linear equations.

In [10], an algorithm is given that decides in $O(n\ log^2 n)$ time whether all initial configurations of an $m \times n$ grid can be transformed to the all-off configuration, $m \leq n$. Note that this is faster than traditional methods of determining if an $mn \times mn$ matrix (representing the neighborhoods of the vertices of G) is non-singular (see Theorem 3, below). We now ask, can each initial configuration of an $m \times n$ grid be transformed to a configuration with at most c vertices on, for a constant c. We give an algorithm for this problem below, which also answers the same question for arbitrary graphs. This algorithm will then enable us to prove that there exist $m \times n$ grids, $m \leq n$, such that at most $mn - \frac{m}{log_2 m}$ vertices can be turned off, for certain initial configurations.

3.1 Background

Some background is needed for subsequent sections of the paper. The definitions and terminology are from [10]. All arithmetic in this and subsequent sections is done over the binary field $GF(2)$. We number the vertices in an $m \times n$ grid graph from 1 to $N = mn$ from top left to bottom right increasing across each row. Define A to be the $mn \times mn$ binary *neighborhood* matrix

$$A = \begin{bmatrix} B & I_n & 0 & 0 & \dots & 0 & 0 \\ I_n & B & I_n & 0 & \dots & 0 & 0 \\ 0 & I_n & B & I_n & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & I_n & B & I_n \\ 0 & 0 & \dots & 0 & 0 & I_n & B \end{bmatrix}$$

Figure 2: Matrix A

shown in block form in Figure 2, where I_n is the $n \times n$ identity matrix and $B = [b_{ij}]$ is the $n \times n$ tridiagonal matrix defined by $b_{ij} = 1$ if $|i - j| \leq 1$ and $b_{ij} = 0$ otherwise. We note that $A - I_N$ is an adjacency matrix for an $mn \times mn$ grid graph and that A is a symmetric matrix. Denote the N-tuple space over $GF(2)$ by F^N and define $y = (y_1, ..., y_N) \in F^N$ by $y_j = 1$ if the initial state of the jth switch/vertex is on and $y_j = 0$ otherwise. It is not hard to see that the initial state y can be transformed to the all-off state by a sequence of activations if and only if the equation $Ax = y$ has a solution over $GF(2)$ [9, 15]. If it does have a solution $x = (x_1, ..., x_N)$ then $\{i | x_i = 1\}$ gives a set of switches whose activation (in any order) results in the all-off state. Let $f_i(x)$ be the i^{th} Fibonacci polynomial over $GF(2)$. That is, $f_1 = 1$, $f_2 = x$, and $f_i = xf_{i-1} + f_{i-2}$ for $i \geq 3$.

If x is a non-zero binary N-vector in the nullspace of A, we can form an $m \times n$ matrix $M(x)$ by putting x_i in the ith cell of $M(x)$ (numbering the cells of $M(x)$ from 1 to N from top left to bottom right, increasing across the rows) for $i = 1, 2, ..., N$. We call such a matrix a *nullspace matrix*. A non-zero $m \times n$ binary matrix is clearly a nullspace matrix if and only if the sum of the entries of each cell and all of its rectilinearly neighboring cells is even. We can then restate the following result from [9, 10].

Theorem 3 *Each initial configuration of an $m \times n$ grid graph can be turned to all-off if and only if there does not exist an $m \times n$ nullspace matrix.*

Of course, the analog of Theorem 3 is true if G is an arbitrary graph: each initial configuration of a graph G can be turned to all-off if and only if the neighborhood matrix of G is non-singular.

3.2 Fixed-Parameter Algorithm

We now describe an algorithm to determine if every initial configuration of a graph with N vertices can be transformed using a series of activations so that at least $N - c$ vertices are off, for a constant c. This algorithm

$$H = \begin{bmatrix} f_1(B)x_1 & f_2(B)x_1 & \dots & f_n(B)x_1 \\ f_1(B)x_2 & f_2(B)x_2 & \dots & f_n(B)x_2 \\ \vdots & \vdots & \ddots & \vdots \\ \dots & \dots & \dots & \dots \\ f_1(B)x_m & f_2(B)x_m & \dots & f_n(B)x_m \end{bmatrix}$$

Figure 3: Matrix H

applies to arbitrary graphs also, but we assume for sake of terminology and exposition that G is a grid graph.

Let G be an $m \times n$ grid graph such that there exists an $m \times n$ nullspace matrix. As usual, assume $m \leq n$. Let $y_1, y_2, \dots, y_k$ be length mn row vectors which form a basis for the nullspace of A. These vectors can be found in $O((mn)^3)$ time by standard methods [cf. 14, Chapter 2.4]. Let H be the $k \times (mn)$ matrix whose i^{th} row is y_i.

In [10] it is shown that if G is a grid graph and if $x_1, x_2, \dots, x_k$ are the vectors (of length m each) resulting from taking the first m components (elements) of $y_1, y_2, \dots, y_k$, respectively, then $x_1, x_2, \dots x_k$ are linearly independent. Likewise, given the beginning segments $x_1, x_2, \dots, x_k$ (not any linearly independent set will do, unless $k = m$), there is a unique way to extend them to length mn vectors, $y_1, y_2, \dots, y_k$ in the nullspace of A: y_i is formed by concatenating $f_1(B)x_i, \dots f_n(B)x_i$ (and $f_{n+1}(B)x_i = 0$), as shown in Figure 3.

Since A is symmetric, its rangespace and nullspace are orthogonal complements, so the rangespace of A, which is the set of initial configurations which can be brought to the all-off configuration by a series of activations, is equal to the nullspace of H. In the language of linear codes, H is a parity check matrix for the code $\mathcal{C}_{m,n}$ consisting of all initial configurations reducible to all-off by a series of activations. $\mathcal{C}_{m,n}$ has length mn and dimension $mn - k$. Let w be the length mn vector with 1's in positions corresponding to vertices of G which are initially on. The $k \times 1$ vector $s = Hw$ is called the *syndrome* of w. The coset $w + \mathcal{C}_{m,n}$ of $\mathcal{C}_{m,n}$ consists of all vectors representing configurations obtainable from w by a series of activations. Each vector in the coset has the same syndrome. The weight of a coset is the smallest number of 1's of any vector in the coset. It is equal to smallest number of columns of H whose sum is s. The maximum weight of a coset is called the *covering radius* of $\mathcal{C}_{m,n}$ which gives the following result.

Proposition 4 *Let m and n be positive integers such that $A_{mn \times mn}$ is singular. Let H be a $k \times (mn)$ matrix whose rows form a basis for the nullspace*

of A. Then H is a parity check matrix for the code $\mathcal{C}_{m,n}$ consisting of all vectors representing starting configurations of the $m \times n$ grid graph G which can be brought to all-off by a series of activations. The covering radius of $\mathcal{C}_{m,n}$ is the largest integer r such that there exists a configuration with r on vertices which cannot be brought to a configuration with fewer than r in the on state, by a series of activations.

For a constant c, we can determine in polynomial time whether every starting configuration of a graph can be reduced to one with at most c on vertices by a series of activations. We just calculate the sums of the vectors in the

$$\sum_{i=1}^{c} \begin{pmatrix} mn \\ i \end{pmatrix}$$

subsets of the columns of H of size at most c. The answer is "yes" if and only if the set of sums include all 2^k possible syndromes. If each non-zero syndrome appears as a column of H, then we can achieve the configuration in which only one switch is on. Note that this does not require time exponential in mn: if 2^k is larger than mn, the number of columns of H, we simply output "no." Otherwise 2^k is obviously polynomial in mn and we then count if there are $2^k - 1$ distinct columns. If there are not 2^k distinct columns, we consider all possible pairs of columns of H. If each non-zero syndrome occurs as the sum (mod 2) of at most two columns of H, then we can achieve a configuration in which at most two switches are on. Thus for a constant c, we can determine in polynomial time whether we can always reach a configuration in which at most c switches are on. For example, if $c = 1$, the running time of the algorithm is $O((mn)^3)$.

We leave the following as an open problem.

Open Problem 2 *Characterize the precise complexity of the MOS problem in grid graphs.*

In particular, we do not think the MOS problem in grid graphs is NP-complete, nor do we know of a polynomial time algorithm to solve it. More generally, one can also ask about the complexity of the question "can all $m \times n$ grids, $m \leq n$, be made to have at least $mn - f(m)$ off vertices," for some function f (e.g. $f(m) = log\ m$). If G is an arbitrary graph, this question is equivalent to asking about the covering radius of an arbitrary binary code; hence the problem is Π_2^p complete [13] – i.e. is thought to be "harder" than NP-complete. But if G is restricted to be a grid, the complexity is not so evident. In section 3.4, it is explained why the algorithms above cannot be used to solve Open Problem 2.

As a related note, it is a straightforward exercise to derive a polynomial time algorithm for the MOS problem in trees. In particular, for a tree with

n vertices and t leaves, it is always possible to get at least $n - \lfloor \frac{t}{2} \rfloor$ vertices off.

3.3 Basic Properties of Grids

In this section, we describe a fast algorithm to get "most" of the vertices off in an $n \times m$ grid graph, given an initial on/off configuration of vertices. Assume $m \leq n$. Consider the m vertices on row 1. For each vertex that is on, activate the vertex just beneath it in row 2. These activations will change the states of certain vertices in rows 1, 2, and 3 and will leave all of the vertices in row 1 in the off state. Iterate this procedure on rows 2 through n. When complete, the only on vertices will be in row n.

At this point, consider the arrangement of the vertices in row n. Proceeding from 1 to m, if vertex i and vertex $i-1$ are both on, then activate vertex i. Doing so will turn off vertices i and $i-1$ and produce an on vertex in row $n-1$. It will also change the state of vertex $i+1$. It is now easy to count the maximum number of on vertices on row n and row $n-1$ combined, yielding the following.

Fact 5 *In an $n \times m$ grid graph, $m \leq n$, at least $mn - \lceil \frac{m}{2} \rceil$ vertices can be turned off using the activate operation.*

In fact, with a slight modification, it can be shown that for $n > 2$ this procedure produces the minimum possible number of on vertices in the last two rows, given that all vertices are off in the first $n-2$ rows and that no activations were performed in row 1. The modification is that once all vertices are off in the first $n-1$ rows, the first vertex in row n is activated if the first (i.e. leftmost) off vertex in row n is the i^{th}, where $i > 2$, but not congruent to 1 (mod 3). The number of on vertices in the last two rows will then be at most $\lfloor \frac{m}{2} \rfloor$, except if m is odd and all vertices in the first $n-1$ rows are off and the vertices in row n alternate on/off beginning with on.

We remark that the above algorithm can be applied after any set of activations are performed in row 1. Thus given any initial configuration of on vertices, there are precisely 2^m subsets of the mn vertices whose activation leaves all vertices in the first $n-1$ rows in the off state (2^m ways to choose the activations in row 1, with all subsequent activations prescribed by the row by row algorithm described above). It is not hard to show that the number of distinct configurations of ons obtainable in the last row (with all other vertices off) is 2^{m-k} where k is the dimension of the nullspace of the neighborhood matrix A of Figure 2. In particular, if A is nonsingular then there are 2^m distinct possible last row configurations, one of them being all off. If $k = m$ then for a given starting configuration the last row will come out the same, no matter what activations are performed in row 1 before applying the algorithm. For example, if $m = 5$ and $n = 23$ then (as shown

in [10]) $k = 5$, so for certain starting configurations the algorithm would produce a last row of "on off on off on" and there would be no way to get fewer ons, given that all vertices in the first 21 rows are off. Note that it turns out that this configuration can be reduced to vertex number 36 (the first vertex in the eighth row) in the on state and all others off. This can be seen by choosing x_1, x_2, x_3, x_4, x_5 in the matrix H of Figure 3 to be the standard basis for F^5, in which case the 36^{th} column of H is 10101, which is equal to the syndrome.

3.4 Better Bounds for Grid Graphs

What can we say in general about the covering radius, $r(m, n)$ of $\mathcal{C}_{m,n}$? The algorithm given in section 3.3 shows that $r(m, n) \leq \lceil \frac{m}{2} \rceil$, assuming $n \geq m$. But a tighter lower bound is desirable. That is, we would like to know if there exist initial configurations of an $m \times n$ grid graph that cannot reach a state with $mn - f(m)$ vertices off, for some "large" $f(m)$. For example, in the 11×47 grid graph, using the algorithm given in section 3.2, one can show that it is not always possible to get at least 515 vertices off (i.e., leave at most two vertices on), but you can always get at least 514 vertices off. We answer the question more precisely in the next theorem. All logarithms are base 2 in this section.

Theorem 6 *For each positive integer t and each real number ϵ, there exist positive integers m and n with $t < m \leq n$ such that some starting configuration of an $n \times m$ grid cannot be changed to a configuration with fewer than $(1 - \epsilon)\frac{m}{\log m}$ switches on.*

Proof: If a parity check matrix H has a relatively small number of columns for its number of rows, then the following counting argument shows that $r(m, n)$ cannot be small. In particular, as shown in [10, Proposition 8], if $m = 5 \cdot 2^i - 1$, where i is any positive integer, and if $n = 3m + 2$ then there exist m linearly independent $m \times n$ nullspace matrices. If $N = mn$, then a parity check matrix for the associated code $C_{m,n}$ is $m \times N$ and the covering radius $r = r(m, n)$ of $\mathcal{C}_{m,n}$ satisfies

$$\sum_{j=0}^{r} \binom{N}{j} \geq 2^m \tag{1}$$

Using Stirling's approximation and some simple algebra we get

$$\sum_{j=0}^{r} \binom{N}{j} \leq r \binom{N}{r} \leq r \left(\frac{Ne}{r}\right)^r \leq r \left(\frac{4}{5}\right)^r \left(\frac{5em^2}{r}\right)^r \leq \left(\frac{5em^2}{r}\right)^r$$

We let

$$g(r) = \left(\frac{5em^2}{r}\right)^r$$

Suppose by way of contradiction that $r \leq (1-\epsilon)\frac{m}{log\, m}$ for some ϵ satisfying $0 < \epsilon < 1$. Since $g(r)$ is an increasing function for $r \in (0, 5m^2)$ we have

$$log(g(r)) \leq (1-\epsilon)\frac{m}{log\, m} log\left(\frac{5em\, log\, m}{(1-\epsilon)}\right)$$

$$= (1-\epsilon)m\left[\frac{log\frac{5e}{1-\epsilon} + log\, log\, m}{log\, m} + 1\right]$$

$$= (1-\epsilon)m[h(m)+1]$$

We can choose m large enough such that $h(m) < \epsilon$. Then $log(g(r)) \leq (1-\epsilon^2)m$ and $\sum_{j=0}^{r}\left(\begin{array}{c} N \\ j \end{array}\right) < 2^m$, which is a contradiction. □

We make the following conjecture.

Conjecture 1 *For each positive real number ϵ, there exists a positive integer N such that any $m \times n$ grid graph with $N \leq m \leq n$ can reach a configuration with at most $(1+\epsilon) \cdot \frac{m}{log_2 m}$ switches on, by a series of activations.*

Theorem 6 shows that this conjecture, if true, is the best possible.

3.5 A Conjecture

Theorem 6 shows the existence of an infinite class of $m \times n$ grid graphs, $m \leq n$, where some initial configurations cannot be reduced to fewer than roughly $\frac{m}{log\, m}$ vertices on. It is natural to look at the other extreme: is there an infinite class of grid graphs such that for each graph in the class, every starting configuration can be reduced to a "small" number of on vertices by a sequence of activations, but some configuration cannot be reduced to all-off?

In [10] it is shown that if the Fibonacci polynomial f_{n+1} is the square if an irreducible polynomial (in which case $n+1$ must be prime) and if m is the smallest positive integer such that $f_{m+1}(B)e_1 = 0$ where B is the tridiagonal $n \times n$ matrix of Figure 2 and e_1 is the first length n standard basis vector, then there are n linearly independent $m \times n$ nullspace vectors.

Conjecture 2 *Let f_{n+1} be a Fibonacci polynomial over $GF(2)$ which is equal to the square of an irreducible polynomial and let m be the smallest positive integer such that $f_{m+1}(B)e_1 = 0$. Then all initial configurations of the $m \times n$ grid graph can be made to have at most two vertices on by a sequence of activations.*

We used the algorithm of Section 3.2 to verify the conjecture for grid graphs of size 8×6, 30×10, 62×12, and 512×18. We think the conjecture can be proved by a careful analysis of the Fibonacci polynomials and the coding theory tools discussed above, but as of yet have been unable to prove the conjecture.

Acknowledgements The authors wish to thank George Trapp and Franz Delahan for their many helpful discussions.

References

[1] N. Alon, J. Spencer, and P. Erdös (1992), The Probabilistic Method, John Wiley and Sons, New York

[2] A. Amin, L. Clark, and P. Slater (1996), Parity Dimension for Graphs, manuscript

[3] A. Amin and P. Slater (1992), Neighborhood Domination with Parity Restriction in Graphs, *Congressus Numerantium*, vol. 91, pp. 19-30

[4] A. Amin and P. Slater (1996), All Parity Realizable Trees, *J. Comb. Math. and Comb. Comput.*, to appear

[5] S. Arora and C. Lund (1996), Hardness of Approximation, in Approximation Algorithms for NP-Hard Problems, D. Hochbaum, editor, PWS Publishing, Boston, MA.

[6] Y. Caro (1996), Simple Proofs to Three Parity Theorems, *Ars Combinatoria*, 42, pp. 175-180

[7] F. Delahan, W. Klostermeyer, and G. Trapp (1995), Another Way to Solve Nine-Tails, *SIGCSE Bulletin*, 27, 4, pp. 27-28

[8] M. Garey and D. Johnson (1979), Computers and Intractability: A Guide to the Theory of NP-completeness, W. H. Freeman, San Francisco

[9] J. Goldwasser, W. Klostermeyer, G. Trapp, and C.-Q. Zhang (1995), Setting Switches on a Grid, *Technical Report TR-95-20*, Dept. of Statistics and Computer Science, West Virginia University

[10] J. Goldwasser, W. Klostermeyer, and G. Trapp (1997), Characterizing Switch-Setting Problems, *Linear and Multilinear Algebra*, to appear

[11] J. Hastad (1997), Some Optimal In-Approximability Results, *Proc.* 29^{th} *ACM Symp. on Theory of Computing*, to appear

[12] P. Heck (1995), Dynamic Programming for Pennies a Day, *SIGCSE Bulletin*, 26, 1, pp. 213-217

[13] A. McLoughlin (1984), The Complexity of Computing the Covering Radius of a Code, *IEEE Trans. Info. Theory*, IT-30, 6, pp. 800-804

[14] G. Strang (1976), Linear Algebra and its Applications, Academic Press, New York

[15] K. Sutner (1989), Linear Cellular Automata and the Garden-of-Eden, *The Mathematical Intelligencer*, vol. 11, no. 2, pp. 49-53