In *Congressus Numerantium* **105** (1994), pp. 116–128.

Constant Time Computation of Minimum Dominating Sets

Marilynn Livingston
Dept. of Computer Science
Southern Illinois University
Edwardsville, IL 62026-1653

Quentin F. Stout*
Elec. Eng. and Comp. Sci.
University of Michigan
Ann Arbor, MI 48109-2122

Abstract

Let G be a graph and let $P(n)$ denote an element from a one-parameter family of graphs, such as a path of length n, a cycle of length n, or a complete binary tree of height n. We are concerned with determining minimum dominating sets of graphs of the form $G \times P(n)$. Using dynamic programming and properties of finite state spaces, we show a constant time algorithm to produce a minimum dominating set of $G \times P(n)$, for fixed G and all n, for the one-parameter families mentioned. Previous researchers had used similar techniques but obtained only linear-time algorithms. We also show how a closed form expression can be obtained for the minimum domination number of $G \times P(n)$. We discuss extensions of the algorithm to the determination of all minimum dominating sets for $G \times P(n)$, and to related problems of coverings, packings, and codes. In addition, we discuss algorithm extensions to several different types of domination, including perfect domination, and to other ways of composing graphs.

Key Words: codes, covering, domination, packing, matching, perfect domination, grid graph, product graph, mesh, torus.

1 Introduction

Let $G = (V, E)$ denote an undirected graph. A subset of D vertices is called a *dominating set* of G if for every $v \in V - D$ there is some $u \in D$ such that $(u, v) \in E(G)$. Sometimes a dominating set is referred to as a *vertex-vertex cover*. The minimum cardinality of the dominating sets of G is called the *domination number* of G and is denoted by $\gamma(G)$.

The general problem of determining $\gamma(G)$ for a given graph G, and of finding a dominating set D of G of this minimum cardinality, has been an active area of research for many years [HL90]. When properly stated, this problem has been shown to be NP-complete [GJ79], and remains so even when G is restricted to certain simple classes of graphs. One example of this is the family of grid graphs, formed from products of paths, where P_n denotes the path with n vertices. The $m \times n$ *complete grid graph*, $P_m \times P_n$, has vertex set $V = \{(i, j) \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ and an edge between pairs of vertices (i, j) and (u, v) if and only if $|i - u| + |j - v| = 1$. A *grid graph* is any subgraph of a complete grid graph. T. Leighton proved that determining the domination number of an arbitary grid graph is NP-complete [Jo85]. The complexity of the domination problem for complete grid graphs is not known, however.

*Partially supported by NSF/ARPA grant CCR-9004727

M. Jacobson and L. F. Kinch [JK84] found closed form expressions for $\gamma(P_m \times P_n)$ for $m = 2, 3, 4$. E. Cockayne, E. Hare, S. T. Hedetniemi, and T. Wimer [CHHW] reported that they had inductive proofs for $m = 2, 3$ and all n, and for $m = 4$ with $n = 4k$. Using an IBM 3081 to perform exhaustive search, they found exact values for $P_m \times P_n$ for $m = 4$ and $4 \leq n \leq 10$, $m = 5$ and $5 \leq n \leq 8$, $m = 6$, and $n = 10, 11$, and used 20 CPU hours to determine $\gamma(P_7 \times P_7)$. In addition, they constructed elementary arguments to establish the inequality

$$(n^2 + n - 3)/5 \leq \gamma(P_n \times P_n) \leq (n^2 + 4n - c)/5$$

where c is 16, 17, or 20, depending on the remainder of n modulo 5. E. Hare, S. Hedetniemi, and W. Hare [HHH] gave a $\Theta(n)$ algorithm for computing $\gamma(P_m \times P_n)$ for fixed m. Their algorithm was based on dynamic programming techniques with an associated state table. Using an IBM 3081, one of their implementations took one and a half minutes to construct the state table and then one additional minute to compute $\gamma(P_7 \times P_{300})$. Using a less memory-intensive implementation, they computed $\gamma(P_8 \times P_8)$ in 2.5 minutes and $\gamma(P_8 \times P_{19})$ in approximately 7 minutes. In [SP87], H.G. Singh and R.P. Pargas described a parallel implementation to compute the domination number of $P_m \times P_n$. They obtained results for $m \leq 9$ and $n \leq 10$. Time requirements became prohibitive for $m = 10$, even for the 16-node FPS T series hypercube. More recently, T.Y. Chang and W.E. Clark [CC93] gave a lengthy proof of a closed form expression for $\gamma(P_m \times P_n)$ for $m = 5, 6$, thus extending the results of E. Hare [H89] to all $n \geq 1$ for these values of m.

In this paper we show how to use the properties of finite state spaces, together with dynamic programming, to produce a $\Theta(1)$ time algorithm for computing $\gamma(P_m \times P_n)$ for fixed m. In fact, we show how to obtain closed form expressions for $\gamma(G \times P_n)$ for fixed graph G and all n. Moreover, we show how to explicitly describe a minimum dominating set for $G \times P_n$ in terms of a regular grammar over states derived from G. Using only an IBM PC, we have been able to rapidly replicate the earlier results mentioned above, and obtain closed form expressions for $\gamma(P_m \times P_n)$ for even larger m than was previously considered. Further details of the algorithm implementation, and tables of $\gamma(P_m \times P_n)$, will appear in [LS94].

Many domination-related concepts defined for arbitrary graphs enjoy easy solutions or at least fast algorithms when restricted to the family of trees. S.M. Hedetniemi, S.T. Hedetniemi, and R. Laskar [HHL] give an extensive coverage of domination and domination-related algorithms for trees, most of which are linear time algorithms. The approach we describe in this paper can be easily modified to determine minimum dominating sets for $\gamma(G \times P(n))$ when $P(n)$ is a complete t-ary tree of height n, for fixed t and all n.

Our approach can be adapted to allow different types of domination as well, such as perfect [LS90], efficient [BBHS, BBS] and total domination [HL90], and still retain the $\Theta(1)$ time complexity. We will illustrate with an example of this in Section 3.2.

A closely related concept to dominating sets is that of packing. Let k be a positive integer. A subset $K \subseteq V$ is called a *k-packing* of the graph $G = (V, E)$ if the distance between every pair of vertices in K is greater than k. The *k-packing number* of G is the cardinality of the largest k-packing of G. Note that the 1-packing number of G is also known as the *independence number* of G, the largest size of a subset of nonadjacent vertices of G. E. Hare and W. Hare [HH91] gave a linear time algorithm for determining the 2-packing number of the complete grid graph $P_m \times P_n$ for fixed m. Recently, D.C. Fisher [F93] determined the 2-packing number of $P_m \times P_n$ for all m and n. He established a recursive inequality which enabled him to deal with the cases for $m \geq 8$ with elementary arguments in the same spirit as those in [CHHW]. Several cases had to be treated separately, some of which were handled with a branch and bound algorithm, others by ad hoc arguments. Using the techniques outlined in this paper, we have found a closed form expression for the 2-packing number of $P_m \times P_n$ for each $m < 9$ and all n, which allowed us to produce a considerably shorter and simpler determination of the 2-packing numbers for all m and n. Further, our techniques easily extend to k-packings of $G \times P_n$ for arbitrary G and k.

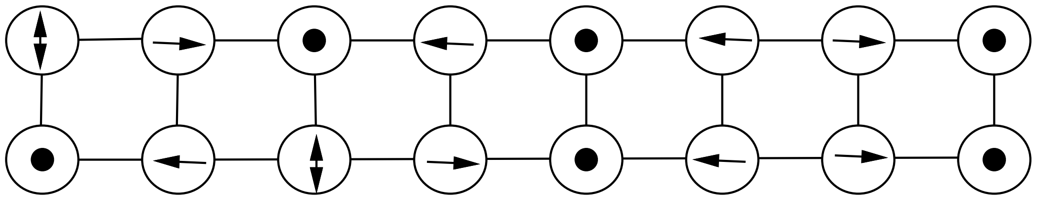
Figure 1: A dominating set for $P_2 \times P_8$

s_1	s_2	s_3	s_4	s_5	s_6	s_7
$\rightarrow$	$\rightarrow$	$\leftarrow$	$\leftarrow$	$\updownarrow$	$\bullet$	$\bullet$
$\rightarrow$	$\leftarrow$	$\rightarrow$	$\leftarrow$	$\bullet$	$\updownarrow$	$\bullet$

Figure 2: $\mathcal{S}_2$, states for $P_2 \times P_n$

2 An Illustrative Example

Before launching into the general description of the method, we illustrate it with a small example, showing how to compute minimal dominating sets for $P_2 \times P_n$. Consider the graph $P_2 \times P_8$ and let $S = \{(2,1),(1,3),(1,5),(2,5),(1,8),(2,8)\}$ be one of its dominating sets. (S is a minimal dominating set, but is not a dominating set of minimum size.) This is shown in Figure 1, where each vertex in S is labeled with a $\bullet$, and all other vertices are labeled with an arrow pointing to an element of S that dominates them.

At vertex $(1,4)$ of Figure 1 there is a choice as to which dominating set element to point to. We force a specific choice through the following interpretation of the labels, which will be critical for the constructions in this paper. Vertex (j,k) labeled

- $\bullet$ means that vertex (j,k) is in S,
- $\updownarrow$ means that vertex (j,k) is not in S but at least one of the vertices $(j \pm 1, k)$ is in S,
- $\leftarrow$ means that none of the vertices $(j,k),(j \pm 1,k)$ are in S, but $(j,k-1)$ is in S.
- $\rightarrow$ means that none of the vertices $(j,k),(j\pm 1,k),(j,k-1)$ are in S, but $(j,k+1)$ is in S.

2.1 States

Note that if we consider any column of the graph we have a copy of P_2 with its vertices labeled by elements of $\{\bullet, \updownarrow, \leftarrow, \rightarrow\}$. Such a labeling of P_2 which can arise from a dominating set in $P_2 \times P_k$ for some k will be called a *state*. There are 4^2 possible labelings, but some reflection upon the interpretation of the labels shows that a labeling is a state if and only if it satisfies the following conditions:

(S–i) if one entry is $\bullet$, then the other entry is $\bullet$ or $\updownarrow$.

(S–ii) if one entry is $\updownarrow$, then the other entry is $\bullet$.

We will let $\mathcal{S}_2$ denote the set of all labelings of P_2 which satisfy these conditions. It is easy to verify that $\mathcal{S}_2$ has 7 elements, given in Figure 2, where we show the states as column vectors of length 2. Note that all of these states occur in Figure 1.

We will be constructing dominating sets for $P_2 \times P_{k+1}$ from dominating sets (and sets that nearly nominate) for $P_2 \times P_k$. To help in this, we use the notion of *state transitions* to describe which states are possible for column $k+1$, given a particular state for column k. In general, it is possible to go from state s_i in column ℓ to state s_j in column $\ell + 1$ if and only if the following conditions hold for all rows p:

	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	0	0	0	0	0	0	1
s_2	0	0	0	0	0	1	1
s_3	0	0	0	0	1	0	1
s_4	1	0	0	0	1	1	1
s_5	0	1	0	0	1	1	1
s_6	0	0	1	0	1	1	1
s_7	0	0	0	1	1	1	1

Figure 3: The State Transition Table $\mathcal{T}_2$

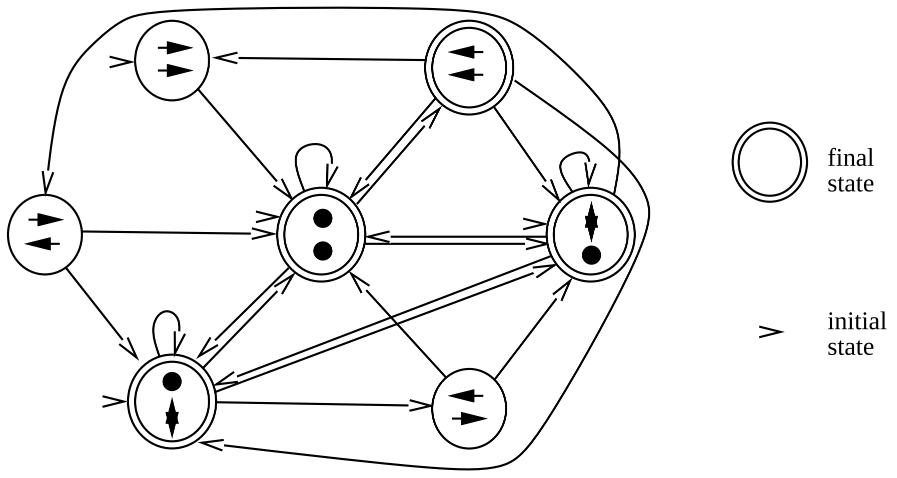

Figure 4: The State Transition Graph for P_2

(T–i) if $s_i(p) = \rightarrow$, then $s_j(p) = \bullet$.

(T–ii) if $s_i(p) = \bullet$, then $s_j(p) \neq \rightarrow$.

(T–iii) if $s_j(p) = \leftarrow$, then $s_i(p) = \bullet$.

This information can be presented in the form of a *state transition table* $\mathcal{T}_2$, given in Figure 3. The state transition table consists of 7 rows and 7 columns in which a 1 appears in row i, column j if it is possible to go directly from state s_i to state s_j, and a 0 otherwise.

States which could be the first column of a dominating set will be called *initial states*, denoted by $\mathcal{I}_2$, while those which could be the last column will be called *final states*, denoted by $\mathcal{F}_2$. Note that a state is initial if and only if it has no $\leftarrow$ entries, while a state is final if and only if it has no $\rightarrow$ entries. We see from Figure 2 that $\mathcal{I}_2 = \{s_1, s_5, s_6, s_7\}$, and $\mathcal{F}_2 = \{s_4, s_5, s_6, s_7\}$.

Let $\mathcal{G}_2$ denote the directed graph (with loops) whose vertices are the states in $\mathcal{S}_2$ and whose edges are determined by the condition that (s_i, s_j) is an edge if and only if $\mathcal{T}_2(s_i, s_j) = 1$, for all $s_i, s_j \in \mathcal{S}_2$. We call $\mathcal{G}_2$ the *state-transition graph* for the pair $\mathcal{S}_2, \mathcal{T}_2$, and illustrate it in Figure 4. Adding the identification of the initial and final states, one has a precursor of a *finite state automaton,* though no alphabet has yet been specified.

2.2 State Sequences

Let S be a dominating set of $G_n = P_2 \times P_n$. S induces a labeling of the vertices of G_n with elements of $\{\bullet, \updownarrow, \leftarrow, \rightarrow\}$. The pair (G_n, S) can be associated with the sequence of states $\alpha = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$, where column j of G_n has the state α_j in $\mathcal{S}_2$ induced by S. We call α the *state sequence induced by the pair* (G_n, S) and express this by writing $\langle G_n, S \rangle = \alpha$. For example, for the graph and dominating set given in Figure 1, the induced state sequence is $\langle s_5, s_2, s_6, s_3, s_7, s_4, s_1, s_7 \rangle$.

For any state sequence $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ induced by a dominating set S, we see that we must have $\mathcal{T}_2(\alpha_i, \alpha_{i+1}) = 1$ for $i = 1, 2, \ldots, n-1$. Furthermore, the first state α_1 must be in $\mathcal{I}_2$, and the final state α_n must be in $\mathcal{F}_2$. Thus α corresponds to a path in G_2 of length $n-1$, starting in $\mathcal{I}_2$ and ending in $\mathcal{F}_2$.

Let U denote a set of vertices of $\mathcal{G}_2$, and, for $j \geq 1$, let $\mathcal{P}(U, j)$ denote the set of all paths in $\mathcal{G}_2$ of length $j-1$ which begin at a vertex in $\mathcal{I}_2$ and end at a vertex in U. We shall be particularly interested in the case when U is a single state or the set of final states. If S is a dominating set of G_n, we see that $\langle G_n, S \rangle$ determines an element p of $\mathcal{P}(\mathcal{F}_2, n)$. Conversely, if p is an element of $\mathcal{P}(\mathcal{F}_2, n)$, then the sequence of n vertices on the path p corresponds to a sequence of states $\langle a_1, a_2, \ldots, a_n \rangle$ which can be identified with a unique dominating set A of $P_2 \times P_n$, where

$$A = \{(i,j) \mid \text{ the element in the } i \text{ th row of state } a_j \text{ is } \bullet\}.$$

We express this relationship by writing $\langle\langle a_1, a_2, \ldots, a_k \rangle\rangle = A$ or, equivalently, $\langle\langle p \rangle\rangle = A$. Thus,

there is a natural 1-1 correspondence between the dominating sets of G_n and the paths of length $n-1$ in the state-transition graph $\mathcal{G}_2$ which begin at an initial state and end at a final state.

Now, if we define the *weight* of the path p, $w(p)$, as the total number of $\bullet$ entries in all the columns of its associated state sequence $\langle a_1, a_2, \ldots, a_n \rangle$, then we see that $w(p)$ is just the cardinality of the set $A = \langle\langle p \rangle\rangle$. Furthermore, the above 1-1 correspondence maps the minimum dominating sets of G_n onto the minimum weight paths in $\mathcal{P}(\mathcal{F}_2, n)$.

2.3 The Cost Matrix

We organize our computational process in a $7 \times n$ matrix, $\mathcal{C}_n$, which we call the *cost matrix*. The element $\mathcal{C}_n(i,j)$, in row i and column j, contains the quantities $c(i,j)$ and $f(i,j)$. The cost $c(i,j)$ is defined as

$$c(i,j) = \begin{cases} \text{minimum weight of the elements of } \mathcal{P}(\{s_i\}, j) & \text{if } \mathcal{P}(\{s_i\}, j) \neq \emptyset \\ \infty & \text{otherwise.} \end{cases}$$

The quantity $f(i,j)$ informs us of the most recent state from which we have made the transition to the present state. More specifically, if $j \geq 2$ and $a = \langle a_1, a_2, \ldots, a_j \rangle$ is an element of $\mathcal{P}(\{s_i\}, j)$ of weight $c(i,j)$, then $f(i,j)$ "points" to the second last vertex on the path a. If there is more than one such state sequence of weight $c(i,j)$, we choose the smallest such k. So, for $j \geq 2$,

$$f(i,j) = k, \text{ where } a_{j-1} = s_k.$$

To complete the definition of $f(i,j)$, we set $f(i,1) = 0$ for $1 \leq i \leq 7$.

Before we describe the recursive relations that exist for $c(i,j)$ and $f(i,j)$, we need some additional notation. For $s \in \mathcal{S}_2$, let

$$Pred(s) = \{w \mid w \in \mathcal{S}_2 \text{ and } \mathcal{T}_2(w,s) = 1\}.$$

	1	2	3	4	5	6	7
s_1	0;0	∞;0	2;4	2;4	3;4	4;4	4;4
s_2	∞;0	1;5	2;5	2;5	3;5	3;5	4;5
s_3	∞;0	1;6	2;6	2;6	3;6	3;6	4;6
s_4	∞;0	2;7	2;7	3;7	4;7	4;7	5;7
s_5	1;0	2;5	2;3	3;3	3;3	4;3	4;3
s_6	1;0	2;5	2;2	3;2	3;2	4;2	4;2
s_7	2;0	2;1	3;2	4;1	4;1	5;1	5;2

Figure 5: The Cost Matrix $\mathcal{C}_7$

Then, since each element of $\mathcal{P}(s_i, j+1)$ can be viewed as consisting of an element of $\mathcal{P}(s_k, j)$, for some $s_k \in Pred(s_i)$, with the edge from s_k to s_i appended to the path, we have the following recursive relation for the $c(i,j)$:

$$\begin{aligned} c(i, j+1) &= w(s_i) + \min\{c(k,j) \mid s_k \in Pred(s_i)\}, \text{ for } j \geq 1, \\ c(i,1) &= \begin{cases} w(s_i) & \text{if } s_i \in \mathcal{I}_2 \\ \infty & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

The cost matrix $\mathcal{C}_7$ is shown in Figure 5, where the entry in row i and column j is exhibited as $c(i,j); f(i,j)$.

For $j \geq 2$, let the sequence $k_1, k_2, \ldots, k_{j-1}$ be defined in terms of $f(i,j)$ as follows:

$$\begin{aligned} k_1 &= f(i,j), \text{ and} \\ k_{t+1} &= f(k_t, j-t), \text{ for } 1 \leq t \leq j-2. \end{aligned} \tag{2}$$

Let $\langle f_{i,j} \rangle$ denote the associated state sequence $\langle s_{k_{j-1}}, \ldots, s_{k_1}, s_i \rangle$. It follows that

$$\langle f_{i,j+1} \rangle = \langle f_{k_1,j} \rangle s_i \text{ for } j \geq 1.$$

To illustrate this notation, let us find a minimum dominating set for $P_2 \times P_6$ from $\mathcal{C}_7$. Arbitrarily choosing s_4 from among the four states in $\mathcal{F}_2$ with minimum $c(i,6)$ values, we use the $f(i,j)$ entries to find the state sequence $\langle f_{4,6} \rangle = \langle s_1, s_7, s_4, s_1, s_7, s_4 \rangle$. The minimum dominating set for $P_2 \times P_6$ corresponding to this minimum weight path is $\langle\langle f_{4,6} \rangle\rangle = \{(1,2),(2,2),(1,5),(2,5)\}$.

2.4 Periodic Behavior

An examination of Figure 5 reveals that

$$c(i,7) = c(i,5) + 1$$

for all i. We will refer to this behavior of the columns of $\mathcal{C}_n$ as *periodic*, for, once two columns, j_1 and j_2 have the property that $c(i,j_1) = c(i,j_2) + b$ for some constant b and all $1 \leq i \leq 7$, it will be the case that columns k and $k + |j_1 - j_2|$ must differ by this constant b for all $k \geq \max(j_1, j_2)$. This relationship, together with Equation 1, implies that

$$c(i,j) = \begin{cases} c(i,5) + \lfloor (j-5)/2 \rfloor & \text{if } j \equiv 1 \pmod 2 \\ c(i,6) + \lfloor (j-5)/2 \rfloor & \text{otherwise} \end{cases}$$

for $j \geq 7$, and all i. It follows that $\gamma(G_n) = \lceil \frac{n-5}{2} \rceil + 3$ for $n \geq 5$. A check of the cost matrix for $1 \leq n \leq 4$ completes the proof of that

$$\gamma(P_2 \times P_n) = \left\lceil \frac{n+1}{2} \right\rceil \text{ for } n \geq 1.$$

Now, let us turn to the construction of a minimum dominating set for $P_2 \times P_n$. The periodic behavior of the $c(i,j)$ guarantees that

$$f(i,j) = \begin{cases} f(i,6) & \text{if } j \equiv 0 \pmod 2 \\ f(i,7) & \text{otherwise} \end{cases}$$

for $j \geq 7$ and all i. Thus, if j is even and $j \geq 8$, then Equation 2 becomes

$$\begin{aligned} k_1 &= f(i,6) \\ k_{2t+\epsilon} &= f(k_{2t+\epsilon-1}, 7-\epsilon) \end{aligned}$$

for $\epsilon = 0,1$ and $t \geq 1$. Similarly, for j odd and $j \geq 7$, we obtain

$$\begin{aligned} k_1 &= f(i,7) \\ k_{2t+\epsilon} &= f(k_{2t+\epsilon-1}, 6+\epsilon) \end{aligned}$$

for $\epsilon = 0,1$ and $t \geq 1$. Consequently, the associated state sequences $\langle f_{i,j} \rangle$ satisfy recurrence relations for each i. For example, taking $i = 5$,

$$\langle f_{5,j} \rangle = \langle f_{5,j-4} \rangle s_2 s_6 s_3 s_5$$

for $j \geq 10$. Let θ denote the sequence $s_2 s_6 s_3 s_5$, then

$$\langle f_{5,j} \rangle = \langle f_{5,6+(j-6 \mod 4)} \rangle \theta^{\lfloor (j-6)/4 \rfloor} \text{ for } j \geq 10.$$

Further, we may express the sequences $\langle f_{5,j} \rangle$ for $6 \leq j \leq 9$ as

$$\begin{aligned} \langle f_{5,9} \rangle &= \langle f_{2,6} \rangle s_6 s_3 s_5 &= s_5 \theta^2 \\ \langle f_{5,8} \rangle &= \langle f_{6,6} \rangle s_3 s_5 &= s_5 s_6 s_3 s_5 \theta \\ \langle f_{5,7} \rangle &= \langle f_{3,6} \rangle s_5 &= s_6 s_3 s_5 \theta \\ \langle f_{5,6} \rangle &= {s_5}^2 \theta \end{aligned}$$

Thus, from the periodic behaviour of the cost matrix, we see that, for all n, not only can we give the value of $\gamma(P_2 \times P_n)$, but we can also describe explicitly a minimum dominating set for $P_2 \times P_n$, namely $\langle \langle f_{5,n} \rangle \rangle$.

3 The Algorithm

To extend our methods from $P_2 \times P_n$ to $G \times P_n$ for an arbitrary graph G, we have to slightly generalize the definition of state. A state is a labeling of the vertices of G with elements of $\{\bullet, \updownarrow, \leftarrow, \rightarrow\}$, satisfying the restrictions

(S′–i) if a vertex is labeled $\bullet$, then its neighbors are all labeled $\bullet$ or $\updownarrow$.

(S′–ii) if a vertex is labeled $\updownarrow$, then at least one of its neighbors is labeled $\bullet$.

The definitions of state transition, final state, and initial states are all exactly as they were in Section 2. In Algorithm 3.1 we outline our algorithm for the cost matrix, from which we can compute domination numbers and dominating sets of minimal size. As can be seen, both the time and space complexity of this algorithm depend only on the graph G, although each will be exponential in the size of G.

The proof that the periodic behavior must occur, forcing the loop in steps 5-13 to terminate, appears in [LS94].

Algorithm 3.1 (Domination Algorithm)
The logical variable periodic remains false until we discover periodic behavior between two columns K_1 and K_2 of the cost matrix.

```
1  Determine $\mathcal{S}$, the set of states, $\mathcal{I}$, the initial states, $\mathcal{F}$, the final states, and $N$, the number of states.
2  Determine the state transition table $\mathcal{T}$.
3  Compute column 1 of the cost matrix $\mathcal{C}(*, 1)$.
4  $j$ := 1,  periodic:= false
5  repeat
6        $j$ :=$j$ + 1.
7        for $i$ := 1 to $N$ do
8              Compute $c(i, j)$ and $f(i, j)$
9        for $t$ := 1 to $j - 1$ do
10             If $c(*, j) - c(*, t)$ is a constant vector then
11                   periodic := true.
12                   $K_1 := t,\ K_2 := j$.
13 until periodic.
```

3.1 Main Theorem

When Algorithm 3.1 terminates, post-processing of the cost matrix gives the following result.

Theorem 3.1 *Let G be an arbitrary graph. Then there are integer constants $m \geq 1$, $n_0 \geq 0$, $a \geq 1$, and b_i, $0 \leq i \leq m-1$, such that, for all $n \geq n_0$,*

$$\gamma(G \times P_n) = a\lfloor n/m \rfloor + b_i, \text{ where } i = n \bmod m.$$

Additionally, for all $0 \leq i \leq m-1$, there are states $s^1_{i,j}$ and $s_{i,k}$ of G, where $1 \leq j \leq n_0 + i$, and $1 \leq k \leq m$, such that, for any $n \geq n_0$, the set

$$S_n = \langle\langle s^1_{i,1}, \ldots, s^1_{i,n_0+i}, (s_{i,1}, \ldots, s_{i,m})^{\lfloor (n-n_0)/m \rfloor}\rangle\rangle,$$

where $i = n \bmod m$, is a dominating set of $G \times P_n$, with cardinality $\gamma(G \times P_n)$.

Further, these constants and states are determined by Algorithm 3.1 in a time which depends solely on G. □

We note that, while Theorem 3.1 emphasizes the asymptotic behavior, we also obtain the corresponding results for $n < n_0$ during the running of the algorithm, and hence determine $\gamma(G \times P_n)$ for all n.

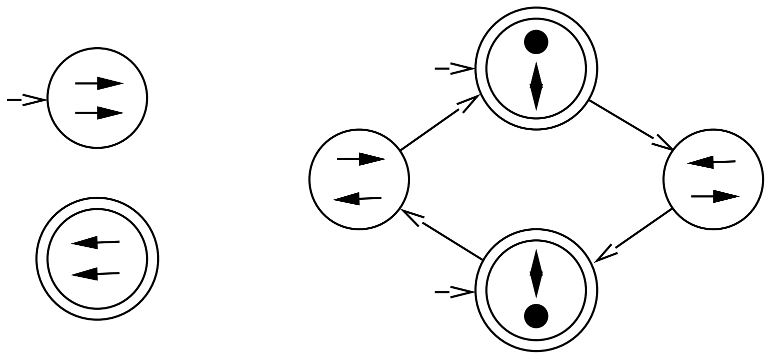

Figure 6: Finite State Automaton for Perfect Domination

3.2 Perfect Domination

While we do not have space to show all of the variations that can be solved by this approach, we will outline another example. One form of domination is *perfect domination*, where a subset S of vertices V of a graph G is a perfect dominating set if it is a dominating set, and for every pair of vertices u, v in S, $N(u) \cap N(v) = \emptyset$, where $N(v)$ denotes the neighborhood set of v, i.e., v and all vertices adjacent to v. Not all graphs have perfect dominating sets: the smallest counterexample is a square of 4 vertices.

To determine those k for which $G \times P_k$ has a perfect dominating set, we need to modify the definitions of states and of transitions. Still using a vertex labeling with the labels $\{\bullet, \updownarrow, \leftarrow, \rightarrow\}$, with the same interpretations, we impose the following restrictions on states:

(S″–i) if a vertex is labeled $\bullet$, then all of its neighbors are labeled $\updownarrow$.

(S″–ii) if a vertex is labeled $\updownarrow$, then exactly one neighbor is labeled $\bullet$.

For state transitions, we impose the following restrictions on going from state s_i to s_j, for all vertices p of G:

(T″–i) $s_i(p) = \rightarrow$ if and only if $s_j(p) = \bullet$.

(T″–ii) $s_i(p) = \bullet$ if and only if $s_j(p) = \leftarrow$.

Figure 6 shows the automaton for $G = P_2$. From this, it is easy to see that $P_2 \times P_n$ has a perfect dominating set if and only if $(n \bmod 2) = 1$. We also see that states in which a pair of neighbors are both labeled $\leftarrow$, or are both labeled $\rightarrow$, can never contribute to a solution because they can have no predecessor or successor, respectively. Thus we can add the following restrictions to states without changing our results:

(S″–iii) if a vertex is labeled $\rightarrow$, then none of its neighbors are labeled $\rightarrow$

(S″–iv) if a vertex is labeled $\leftarrow$, then none of its neighbors are labeled $\leftarrow$

Using this automaton specially constructed for perfect domination, we obtain:

Theorem 3.2 *Let G be an arbitrary graph. Then there are integer constants $n_0 \geq 1$, $m \geq 1$, and (possibly empty) sets $I \subseteq \{1, \ldots, n_0 - 1\}$ and $J \subseteq \{0, \ldots, m-1\}$, such that $G \times P_n$ has a perfect dominating set if and only if*

$$\begin{array}{rl} n \in I & \textit{if } n < n_0 \\ (n \bmod m) \in J & \textit{otherwise.} \end{array}$$

Further, these constants and sets can be determined by an algorithm whose running time depends solely on G.

Proof: $G \times P_n$ has a perfect dominating set if and only if there is a path of length $n-1$, starting at an initial state and ending at a final state, in the state transition graph described above. It is well known that the lengths of paths of accepting sequences in a finite state automaton can be written in the form given in the theorem, and that they can be determined from the state transition graph. $\Box$

4 Conclusion

We have shown that, for any graph G, there is a closed-form formula for $\gamma(G \times P_n)$ as a function of n, and that our algorithm finds this formula in time depending only on G. Further, dominating sets of minimal size can be given as a regular grammar over states derived from G. We showed this by reducing the original problem to one involving paths in a state space, and then solving this automata problem for all n by utilizing dynamic programming and the periodic nature of the solution. While others had also noted that a state space and dynamic programming could be used for this problem, apparently none had noticed that the periodic properties of finite state spaces could be exploited to eliminate the time dependence on n.

By changing the definitions of the states (perhaps including labels of edges) and their transitions, this approach can be extended to a great many other problems involving domination and domination-related concepts such as packings, coverings, matchings, etc. For example, we can solve problems such as perfect domination, domination involving distances greater than 1, domination of nonconvex regions such as knight's moves ([HH87]), independent domination, edge-edge domination, etc. For packings, we can solve problems such as k-packings, vertex-disjoint (or edge-disjoint) packings of subgraphs, etc. Covers can include vertex covers, edge covers, covers by subgraphs, etc.

We can also change some of the information kept along with the dynamic programing, and use it to answer various counting problems. For example, we can develop formulas for the number of dominating sets, number of minimal dominating sets, number of dominating sets of minimal size, number of perfect dominating sets, etc. We can count number of matchings, number of perfect matchings, and so on. Other variations, still producing closed-form solutions, include replacing P_n by complete t-ary trees of height n (for fixed t), or by cycles of n vertices. One can also vary the definition of product used, allowing us to analyze, for example, grid graphs where each vertex is connected to its eight nearest neighbors, rather than its four nearest neighbors.

Several of the extensions mentioned above will be explored in [LS94] and subsequent papers. Those papers will include more details of material outlined here, including tables of $\gamma(P_m \times P_n)$.

References

[BBHS] D.W. Bange, A.E. Barkauskas, L.H. Host, and P.J. Slater, "Efficient near-domination of grid graphs", *Congressus Numerantium* 58 (1987) 83–92.

[BBS] D.W. Bange, A.E. Barkauskas, and P.J. Slater, "Efficient dominating sets in graphs", *Applications of Discrete Mathematics*, R.D. Ringeisen and F.S. Roberts, eds., SIAM (1988).

[CC93] T.Y. Chang and W.E. Clark, "The domination numbers of the $5 \times n$ and $6 \times n$ grid graphs", *J. Graph Theory* 17 (1993) 81–107.

[CHHW] E.J. Cockayne, E.O. Hare, S.T. Hedetniemi and T.V. Wimer, "Bounds for the domination number of grid graphs", *Congressus Numerantium* 47 (1985) 217–228.

[F93] D.C. Fisher, "The 2-packing number of complete grid graphs", *Ars Combinatoria* 36 (1993) 261–270.

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

[H89] E.O. Hare, "Algorithms for grid and grid-like graphs". Ph. D. thesis, Dept. Comp. Sc., Clemson University, Clemson, SC (1989).

[HH91] E.O. Hare and W.R. Hare, "k-Packing of $P_m \times P_n$", *Congressus Numerantium* 84 (1991) 33–39.

[HH87] E.O. Hare and S.T. Hedetniemi, "A linear algorithm for computing the knight's domination problem of a $k \times n$ chessboard", *Congressus Numerantium* 59 (1987) 115–130.

[HHH] E.O. Hare, S.T. Hedetniemi and W.R. Hare, "Algorithms for computing the domination number of $k \times n$ complete grid graphs", *Congressus Numerantium* 55 (1986) 81–92.

[HL90] S.T. Hedetniemi and R.C. Laskar, "Bibliography on domination in graphs and some basic definitions of domination parameters", *Discrete Math.* 86 (1990) 257–277.

[HHL] S.M. Hedetniemi, S.T. Hedetniemi, and R. Laskar, "Domination in trees: models and algorithms", *Graph Theory with Applications to Algorithms and Computer Science*, Y. Alavi, G. Chartrand, L. Lesniak, D. Lick, and C. Wall, eds., (1985) Wiley, 423–442.

[JK84] M.S. Jacobson and L.F. Kinch, "On the domination number of products of graphs", *Ars Combinatoria* 18 (1984) 33–44.

[Jo85] D.S. Johnson, "The NP-Completeness column: an ongoing guide", *J. Algorithms* 6 (1985) 434–451.

[KYK] T. Kikuno, N. Yoshida, and Y. Kakkuda, "A linear algorithm for the domination number of a series-parallel graph", *Discrete Appl. Math.*, 37 (1983) 299–311.

[LS90] M. Livingston and Q.F. Stout, "Perfect dominating sets", *Congressus Numerantium* 79 (1990) 187–203.

[LS94] M. Livingston and Q.F. Stout, "Constant time computation of properties of product graph families", in preparation

[SP87] H.G. Singh and R.P. Pargas, "A parallel implementation for the domination number of a grid graph", *Congressus Numerantium* 59 (1987) 297–311.