A domination algorithm for generalized Cartesian products

S. Benecke* and C. M. Mynhardt†
Department of Mathematics and Statistics
University of Victoria, P.O. Box 3060 STN CSC, Victoria, B.C.
CANADA V8W 3R4
stephen@math.uvic.ca, mynhardt@math.uvic.ca

February 2009

Abstract

Let $G \,\square\, H$ denote the Cartesian product of two graphs G and H. In 1994, Livingston and Stout [Constant time computation of minimum dominating sets, Congr. Numer., **105** (1994), 116–128] introduced a linear time algorithm to determine $\gamma(G \,\square\, P_n)$ for fixed G, and claimed that P_n may be substituted with any graph from a one-parameter family, such as a cycle of length n or a complete t-ary tree of height n for fixed t. We explore how the algorithm may be modified to accommodate such graphs and propose a general framework to determine $\gamma(G \,\square\, H)$ for any graph H. Furthermore, we illustrate its use in determining the domination number of the generalized Cartesian product $G \,\boxdot\, H$, as defined by Benecke and Mynhardt [Domination of Generalized Cartesian Products, preprint (2009)].

Keywords: Cartesian product graph, generalized Cartesian product, domination number, domination algorithm.

AMS Classification: 05C69, 05C99.

*Supported by the Skye Foundation and the National Research Foundation of South Africa.

†Supported by the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

In 1994, Livingston and Stout [6] introduced a linear time algorithm to determine $\gamma(G \,\square\, P_n)$ for a fixed graph G, using the notion of finite state spaces. They observed that the complexity may be reduced to constant time through an observation of periodicity in the solution. Besides illustrating the applicability to other types of domination, the authors mention the algorithm's use to determine $\gamma(G \,\square\, P(n))$, where $P(n)$ is a graph from a one-parameter family of graphs, such as a cycle of length n or a complete t-ary tree of height n for fixed t. We explore how the algorithm may be modified to accommodate such graphs and propose a general framework to determine $\gamma(G \,\square\, H)$ for any graph H. Furthermore, we illustrate its use in determining the domination number of the generalized Cartesian product $G \,\boxdot\, H$, defined by Benecke and Mynhardt [1].

We generally follow the notation and terminology in [4]. For two graphs G and H, the *Cartesian product* $G \,\square\, H$ is the graph with vertex set $V(G) \times V(H)$ and vertex (v_i, u_j) adjacent to (v_k, u_l) if and only if (a) $v_i v_k \in E(G)$ and $u_j = u_l$, or (b) $v_i = v_k$ and $u_j u_l \in E(H)$. As usual, $\gamma(G)$ denotes the domination number of G. The set $S \subseteq V(G)$ is called a *γ-set* if it is a dominating set with $|S| = \gamma(G)$. In 1967, Chartrand and Harary [3] defined the *generalized prism* πG of G as the graph consisting of two copies of G, with edges between the copies determined by a permutation π acting on the vertices of G. Serving as a generalization of both the Cartesian product and the generalized prism, Benecke and Mynhardt [1] defined the *generalized Cartesian product* as follows. For two labelled graphs G and H and a permutation π of $V(G)$, the *generalized Cartesian product* $G \,\boxdot\, H$ is the graph with vertex set $V(G) \times V(H)$, and vertex (v_i, u_j) adjacent to (v_k, u_l) if and only if (a) $v_i v_k \in E(G)$ and $u_j = u_l$, or (b) $v_k = \pi^{l-j}(v_i)$ and $u_j u_l \in E(H)$. If $\pi \in \mathrm{Aut}(G)$, then $G \,\boxdot\, H$ is isomorphic to the Cartesian product $G \,\square\, H$, whereas $G \,\boxdot\, H$ is a generalized prism if $H = K_2$.

The basic algorithm of Livingston and Stout [6] is explained in Section 2 by way of an example on $K_2 \,\square\, P_n$. A similar example is used in [6], and a more detailed explanation may be found there. In Section 3 we introduce a general framework for evaluating the domination number of $G \,\square\, H$ for a fixed graph G and any H. The problem of determining $\gamma(G \,\square\, H)$ is shown to be equivalent to a conditional, weighted homomorphism problem. We provide an algorithm in Section 4 to determine $\gamma(G \,\square\, T)$ for any tree T, and observe that it is polynomial for trees of bounded maximum degree. Lastly, in Section 5 we discuss how the general framework for $G \,\square\, H$ may be modified to accommodate the generalized Cartesian product $G \,\boxdot\, H$, and provide an example on $G \,\boxdot\, T$.

For $A, B \subseteq V(G)$, we abbreviate "A dominates B" to "$A \succ B$"; if $B = V(G)$ we write $A \succ G$ and if $B = \{b\}$ we write $A \succ b$. Further, $N(v) = \{u \in V(G) : uv \in E(G)\}$ and $N[v] = N(v) \cup \{v\}$ denote the *open* and *closed neighbourhoods*, respectively, of a vertex v of G. The *closed neighbourhood* of $S \subseteq V(G)$ is the set $N[S] = \bigcup_{s \in S} N[s]$, while $N\{S\}$ denotes the set $N[S] - S$. For a directed graph D we denote an arc $(u, v) \in E(D)$ as uv for convenience. The (open) out-neighbourhood of a vertex v is the set $N_{\text{out}}(v) = \{u \in V(D) : vu \in E(D)\}$, while the (open) in-neighbourhood is the set $N_{\text{in}}(v) = \{u \in V(D) : uv \in E(D)\}$.

Consider two graphs G and H with vertex sets labelled $v_1, v_2, \ldots, v_m$ and $u_1, u_2, \ldots, u_n$ respectively. Vertices (v_i, u_j) of the Cartesian product $G \Box H$ are labelled $v_{i,j}$ for convenience. The subgraph induced by all vertices that differ from a given vertex $v_{i,j}$ only in the first [second] coordinate, is known as the (Cartesian) *G-layer* [*H-layer*] through $v_{i,j}$. The G-layer through a vertex $v_{i,j}$ is denoted G_j. To specify the appropriate G-layer, we often write $N_{G_j}(v_{i,j})$ for the neighbourhood $\{v_{k,j} : v_k \in N_G(v_i)\}$. Other types of neighbourhoods restricted to a specific G-layer may be defined similarly. Lastly, if H is a path or a cycle, then a *canonical labelling* of H is a labelling of the vertices of H such that $u_1, u_2, \ldots, u_n$ is the vertex sequence along the path or cycle.

2 The Cartesian product $G \Box P_n$

In this section, an efficient algorithm, introduced by Livingston and Stout [6], to determine $\gamma(G \Box P_n)$ is discussed. Similar to [6], the graph $K_2 \Box P_7$ is used as an example to explain the algorithm.

Let $V(G) = \{v_1, \ldots, v_m\}$, $V(P_n) = \{u_1, \ldots, u_n\}$ (labelled canonically), and let D be a dominating set of $G \Box P_n$. We label the vertices according to a mapping $l_D : V(G \Box P_n) \mapsto \{\leftarrow, \rightarrow, \bullet, \updownarrow\}$, where

$$l_D(v_{i,j}) = \begin{cases} \bullet & \text{if } v_{i,j} \in D \\ \updownarrow & \text{if } v_{i,j} \in N_{G_j}\{D\} \\ \leftarrow & \text{if } v_{i,j-1} \in D \text{ and } v_{i,j} \notin N_{G_j}[D] \\ \rightarrow & \text{if } v_{i,j+1} \in D,\ v_{i,j-1} \notin D \text{ and } v_{i,j} \notin N_{G_j}[D]. \end{cases}$$

For example, if $D = \{v_{1,1}, v_{2,1}, v_{1,3}, v_{1,5}, v_{2,5}, v_{2,7}\}$ is a dominating set of $K_2 \Box P_7$, then its (unique) labelling is illustrated in Figure 1.

For any graph G and dominating set D of $G \Box P_n$, a labelling l_D of a G-layer G_j satisfies the following two conditions for any $v \in V(G_j)$:

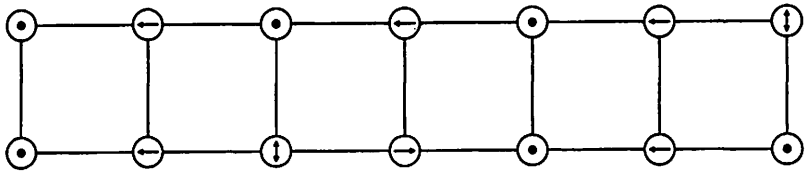

Figure 1: The labelling l_D for $K_2 \,\square\, P_7$ corresponding to the dominating set $D = \{v_{1,1}, v_{2,1}, v_{1,3}, v_{1,5}, v_{2,5}, v_{2,7}\}$.

s_0	s_1	s_2	s_3	s_4	s_5	s_6
$\rightarrow$	$\rightarrow$	$\leftarrow$	$\leftarrow$	$\bullet$	$\bullet$	$\updownarrow$
$\rightarrow$	$\leftarrow$	$\rightarrow$	$\leftarrow$	$\bullet$	$\updownarrow$	$\bullet$

Table 1: All possible states of K_2.

- if $l_D(v) = \bullet$, then $l_D(u) = \updownarrow$ or $l_D(u) = \bullet$ for every $u \in N_{G_j}(v)$;
- if $l_D(v) = \updownarrow$, then $l_D(u) = \bullet$ for some $u \in N_{G_j}(v)$.

A labelling of $V(G_j)$, written as $l_D(G_j)$, is called a *state* of G. Clearly the number of valid states depends on the structure of G. For the graph $G = K_2$, there are seven valid states, listed in Table 1 as column vectors $s_0, s_1, \ldots, s_6$. We write $[s]_i = l_D(v_{i,j})$ for the i^{th} entry in the state $s = l_D(G_j)$.

For a canonical labelling of $V(P_n)$ and the resulting grid representation of $G \square P_n$, *state transitions* are defined according to which states are allowed to "follow" each other in the grid representation of the Cartesian product $G \,\square\, P_n$. More precisely, a state s in a G-layer G_j may be "followed" by a state t in G_{j+1} if and only if the following conditions hold for all i:

- if $[s]_i = \rightarrow$, then $[t]_i = \bullet$;
- if $[s]_i = \bullet$, then $[t]_i \neq \rightarrow$;
- if $[t]_i = \leftarrow$, then $[s]_i = \bullet$.

From these state transitions a digraph $\mathcal{G}$, called the *state-transition graph*, is obtained with vertex set the set of all possible states of G, and $st \in E(\mathcal{G})$ if and only if state s may be followed by t for some dominating set D of $G \,\square\, P_n$. For the graph $G = K_2$, the state-transition graph $\mathcal{G}(K_2)$

is illustrated in Figure 2, with states labelled $s_0, s_1, \ldots, s_6$ according to Table 1.

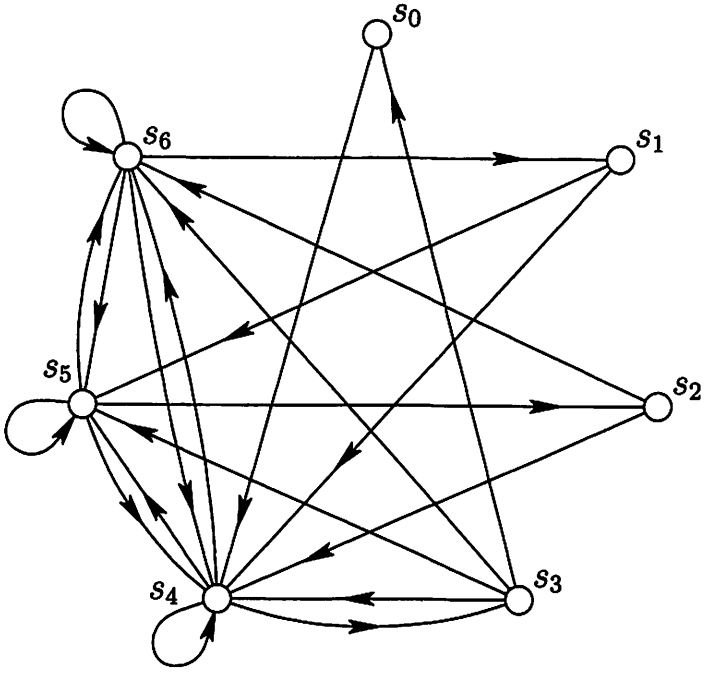

Figure 2: The state-transition graph of K_2.

Special consideration is needed for the first and last G-layers, i.e. G_1 and G_n corresponding to the end-vertices of the path P_n. Let I denote the set of states (vertices of $\mathcal{G}$) that may be assigned to G_1, and F be the set of possible states for G_n. For example, if $G = K_2$, then $I = \{s_1, s_4, s_5, s_6\}$ and $F = \{s_3, s_4, s_5, s_6\}$.

Consider a dominating set D of $G \,\square\, P_n$. The unique labelling l_D associated with the dominating set D induces a sequence of states $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, where α_i is the state associated with $l_D(G_i)$, $i = 1, 2, \ldots, n$. Since $\alpha_i \in V(\mathcal{G})$ for any i, and α_i is followed by α_{i+1}, the state sequence $\underline{\alpha}$ corresponds to a directed walk of length $n-1$ in $\mathcal{G}$, starting with a state in I and ending with a state in F. For example, the dominating set $D = \{v_{1,1}, v_{2,1}, v_{1,3}, v_{1,5}, v_{2,5}, v_{2,7}\}$ of $K_2 \,\square\, P_7$ yields the walk $L : s_4, s_3, s_5, s_2, s_4, s_3, s_6$.

Conversely, a directed walk in $\mathcal{G}$ that starts in I and ends in F, corre-

sponds to a state sequence that yields a unique dominating set of $G \,\square\, P_n$. So there is a one-to-one correspondence between dominating sets of $G \,\square\, P_n$ and directed walks of length $n-1$ in $\mathcal{G}$ starting in I and ending in F.

The vertices of $\mathcal{G}$ can be weighted according to the number of $\bullet$-labels in the state. If the weight $w(L)$ of a walk L in $\mathcal{G}$ is defined as the sum of the weights of the states in the walk, then the domination number of $G \,\square\, P_n$ is given by

$$\gamma(G \,\square\, P_n) = \min\{w(L) : L \text{ is a directed walk of length } n-1 \text{ in } \mathcal{G}(G) \text{ starting in } I \text{ and ending in } F\}.$$

Livingston and Stout [6] discussed an efficient algorithm to find a minimum weight walk in a fixed state-transition graph, thereby determining the domination number of the Cartesian product $G \,\square\, P_n$ for fixed G. Considering the graph $K_2 \,\square\, P_7$, the walk $s_6, s_1, s_5, s_2, s_6, s_1, s_5$ has minimum weight 4, and the corresponding labelling of $V(K_2 \,\square\, P_7)$, from which the minimum dominating set $D = \{v_{2,1}, v_{1,3}, v_{2,5}, v_{1,7}\}$ follows, is shown in Figure 3.

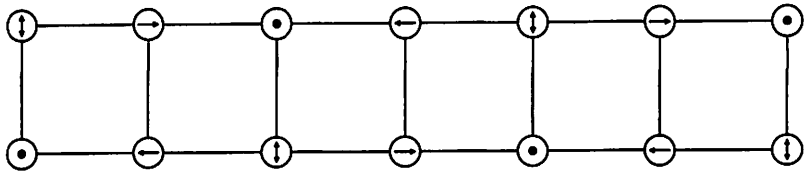

Figure 3: The labelling for $K_2 \,\square\, P_7$ corresponding to a γ-set $D = \{v_{2,1}, v_{1,3}, v_{2,5}, v_{1,7}\}$.

The algorithm has a linear time complexity in the order n of the path. However, Livingston and Stout observed a periodicity in the output of the algorithm. Once this periodic behaviour occurs, the domination number is determined without further computation. From this observation they deduced a constant time complexity for the problem of determining $\gamma(G \,\square\, P_n)$ for fixed G.

When considering the product $G \,\square\, C_n$, the algorithm can be modified easily without changing the complexity. Since the state-transition graph $\mathcal{G}$ simply depends on G and the type of graph product, it is obtained in exactly the same manner. However, in this case there is a one-to-one correspondence between dominating sets of $G \,\square\, C_n$ and directed circuits of length n in $\mathcal{G}$. Repeated application of the algorithm can be used to determine minimum weight directed walks of length $n-1$, thereby determining a minimum weight circuit in $\mathcal{G}$.

It is unclear whether the algorithm can be modified easily to accommodate graphs $G \,\square\, H$ if H has many vertices of degree greater than 2. We require a more general framework for determining $\gamma(G \,\square\, H)$.

3 A general framework for $\gamma(G \,\square\, H)$

In this section we generalize the algorithm by Livingston and Stout [6] in order to determine $\gamma(G \,\square\, H)$ for any graph H.

Consider the Cartesian product $G \,\square\, H$, with the vertices of G and H denoted by $v_1, v_2, \ldots, v_n$ and $u_1, u_2, \ldots, u_m$ respectively. Also, let D be a dominating set of $G \,\square\, H$ and $\overrightarrow{H}$ an orientation of H. As in Section 2, a state-transition graph $\mathcal{G}$ is constructed with vertex set the set of all possible states of G. Once again, we write $[s]_i = l_D(v_{i,j})$ for the i^{th} entry in the state $s = l_D(G_j)$.

The arc set of the digraph $\mathcal{G}$ is now given by the following condition:

- $st \in E(\mathcal{G})$ if and only if for every i: if $[s]_i = \bullet$, then $[t]_i \neq \rightarrow$.

Next, we assign a *binary colour vector* $\underline{c} = (c_0, c_1, \ldots, c_m)$ to each arc st in $\mathcal{G}$. This assignment satisfies the following conditions, which hold for all i. Let $[\underline{c}]_k$ denote the $(k+1)^{\text{st}}$ entry in the vector.

- $[\underline{c}(st)]_0 = 1$ if and only if $st \in E(\mathcal{G})$;
- for any $t \in N_{\text{out}}(s)$ with $[t]_i = \bullet$, $[\underline{c}(st)]_i = 1$ if and only if $[s]_i = \rightarrow$;
- for any $t \in N_{\text{out}}(s)$ with $[s]_i = \bullet$, $[\underline{c}(st)]_i = 1$ if and only if $[t]_i = \leftarrow$.

For the example $G = K_2$, with corresponding vertex set $\{s_0, s_1, \ldots, s_6\}$ of $\mathcal{G}$ listed in Table 1, there are only seven arcs not in $\mathcal{G}$, namely s_6s_0, s_6s_2, s_5s_0, s_5s_1, s_4s_0, s_4s_1 and s_4s_2. If the colours c_0, c_1 and c_2 are viewed as "no colour", "blue" and "red" respectively, then

- $c_0(e) = 1$ for all $e \in E(\mathcal{G})$ (all arcs have the no-colour attribute);
- $\underline{c}(e) = (1,1,0)$ for $e = s_0s_5, s_1s_4, s_1s_5, s_5s_2, s_5s_3$ (the arcs s_0s_5, s_1s_4, s_1s_5, s_5s_2 and s_5s_3 have a blue attribute but not red);
- $\underline{c}(e) = (1,0,1)$ for $e = s_0s_6, s_2s_4, s_6s_1, s_6s_3$ (the arcs s_0s_6, s_2s_4, s_6s_1 and s_6s_3 have a red attribute but not blue);

- $\underline{c}(e) = (1,1,1)$ for $e = s_0s_4, s_4s_3$ (the arcs s_1s_5 and s_5s_4 have both a blue and red attribute).

This state-transition graph $\mathcal{G}$ is illustrated in Figure 4. Dashed arcs have colour vector $(1,1,0)$ (the blue attribute only), dashed-dotted arcs have colour vector $(1,0,1)$ (the red attribute only), solid arcs have colour vector $(1,1,1)$ (both the red and blue attribute). The arcs not shown are in $\mathcal{G}$ and have colour vector $(1,0,0)$, while the dotted arcs are not present in the graph.

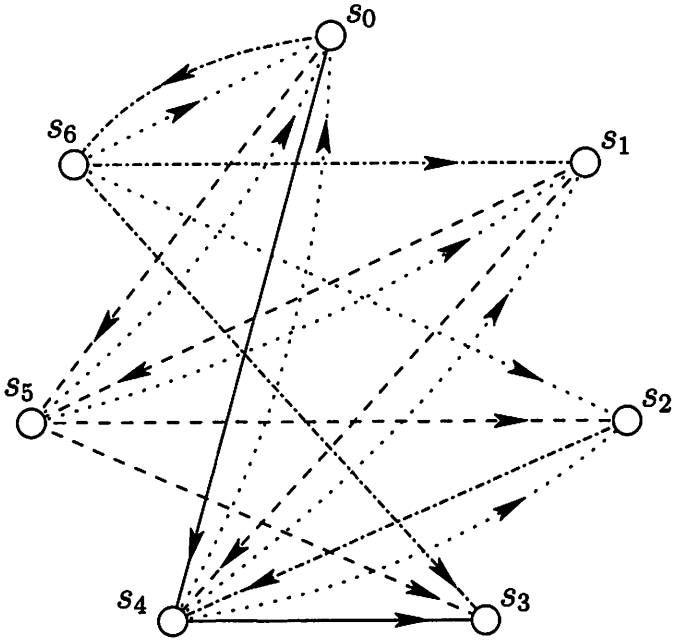

Figure 4: The state-transition graph $\mathcal{G}$ of K_2. Dashed arcs have colour vector $(1,1,0)$ (the blue attribute only), dashed-dotted arcs have colour vector $(1,0,1)$ (the red attribute only), solid arcs have colour vector $(1,1,1)$ (both the red and blue attribute). The arcs not shown are in $\mathcal{G}$ and have colour vector $(1,0,0)$, while the dotted arcs are not present in the graph.

Let $P, Q \subseteq \{1,2,\ldots,m\}$. The unique labelling l_D associated with the dominating set D induces a sequence of states $\alpha_1, \alpha_2, \ldots, \alpha_n$, where $\alpha_j = l_D(G_j)$, $j = 1,2,\ldots,n$. This corresponds uniquely to a homomorphism $f : \overrightarrow{H} \mapsto \mathcal{G}$, with $f(u_j) = \alpha_j$, satisfying the following condition:

- if $f(u) = s \in V(\mathcal{G})$, $[s]_i = \leftarrow$ for $i \in P$ and $[s]_k = \rightarrow$ for $k \in Q$, then

$$\text{for any } i \in P,\ [\underline{c}(f(wu))]_i = 1 \text{ for some } w \in N_{\text{in}}(u);$$
$$\text{for any } k \in Q,\ [\underline{c}(f(uw))]_k = 1 \text{ for some } w \in N_{\text{out}}(u).$$

For example, for the product $K_2 \,\square\, H$, the homomorphism $f : \overrightarrow{H} \mapsto \mathcal{G}(K_2)$ has the property that if s_0 is an image under f, then over all arcs *from* s_0 in the image of $f(E(\overrightarrow{H}))$, there is a blue and a red attribute represented. If s_1 is an image under f, then over all arcs *from* s_1 in the image of $f(E(\overrightarrow{H}))$, there is a blue attribute represented, while over all arcs *to* s_1 in the image of $f(E(\overrightarrow{H}))$, there is a red attribute represented.

Conversely, a conditional homomorphism $f : \overrightarrow{H} \mapsto \mathcal{G}$ satisfying the property above corresponds to a state sequence that yields a unique dominating set of $G \,\square\, H$. There is a one-to-one correspondence between dominating sets of $G \,\square\, H$ and conditional homomorphisms $f : \overrightarrow{H} \mapsto \mathcal{G}$ for some orientation of H. Denote the set of such homomorphisms by $\text{HOM}(\overrightarrow{H}, \mathcal{G})$. Note that the orientation of H is used only to establish the one-to-one correspondence between the dominating sets of $G \,\square\, H$ and the state sequence.

Similar to Section 2, the vertices of $\mathcal{G}$ can be weighted according to the number of $\bullet$-labels in the state. Let the weight $w(f(\overrightarrow{H}))$ of the homomorphic image of $\overrightarrow{H}$ under f in $\mathcal{G}$ be defined as the sum of the weights of the images, i.e.

$$w(f(\overrightarrow{H})) = \sum_{v \in V(\overrightarrow{H})} w(f(v)).$$

Then the domination number of $G \,\square\, H$ is given by

$$\gamma(G \,\square\, H) = \min\{w(f(\overrightarrow{H})) :\ f \in \text{HOM}(\overrightarrow{H}, \mathcal{G})\}.$$

For $H = P_n$ a canonical labelling of H yields a directed path $\overrightarrow{H}$ in which every vertex has in-degree and out-degree at most 1. This simplifies the state-transition graph $\mathcal{G}$, since homomorphisms satisfying the required property only include arcs with colour vectors $(1, 0, \ldots, 0)$ and/or $(1, 0, \ldots, 0, 1)$ (only "no-colour" or both "blue"and "red"arcs in the case of $G = K_2$). In this case the colour vectors play no role in the correspondence, and the state-transition graph reduces to the graph defined in Section 2. The method introduced by Livingston and Stout [6] is a special case of the general method discussed in this section.

4 An algorithm for $\gamma(G \,\square\, T)$

We illustrate the implementation of this general framework by providing an algorithm to find the domination number of $G \square T$ for any tree T. We follow the algorithm format used in [5]. Throughout the following algorithms, external functions are used in an intuitive manner. These functions are named so that their use is clear. Let $V(T) = \{v_0, v_1, \ldots, v_{n-1}\}$ and denote the tree rooted at a vertex v by T_v. The main algorithm GAMMA(), shown as Algorithm 4.1, starts by constructing the state-transition graph $\mathcal{G}$ of G by way of the function STATEGRAPH(). In addition to the state-transition graph, this function also returns information about the colour vectors of the arcs that can be used by all subalgorithms. Let the states (vertices in $\mathcal{G}$) be denoted $s_0, s_1, \ldots, s_{N-1}$. The colour vector of an arc in $\mathcal{G}$ is returned by the function CVECT(). For each state in $\mathcal{G}$, there is an associated incoming colour requirement vector (a binary vector with entry 1 corresponding to rows with label "$\leftarrow$"). Similarly, an outgoing colour requirement vector is associated with each state in $\mathcal{G}$, according to the "$\rightarrow$" labels in the state. The incoming colour requirement vector of a vertex in $\mathcal{G}$ is returned by the function INC(), while OUTC() returns the outgoing colour requirement vector. For a vertex $u \in V(T_v)$, let T_u denote the subtree of T_v induced by u and its descendents.

The algorithm GAMMA() visits the vertices of T_v in a reversed breadth-first-search (BFS) order to construct two $N \times n$ matrices, denoted W and P, one column at a time. The $(i+1, j+1)$-entry of W is the minimum weight of a conditional homomorphism $\phi_u : T_u \mapsto \mathcal{G}$ such that $\phi_u(u) = s_i$, where $u = v_j$. The corresponding entry in P contains such a minimum homomorphism ϕ_u, restricted to $N_{\text{out}}(u)$, the children of $u = v_j$ (for efficiency). Denote this restricted homomorphism by $\phi_u^{(c)}$. For each $v_j \in V(T_v)$ and each $s_i \in V(\mathcal{G})$, the subalgorithm MINHOM() is called to determine the $(i+1, j+1)$-entry in both W and P. This algorithm is shown as Algorithm 4.3.

Since $\deg_{\text{in}}(v) = 0$ for the root v of T_v, only a subset of the states in $\mathcal{G}$ are valid images for v. The subalgorithm ROOTIMAGELIST(), shown as Algorithm 4.2, determines this subset $F \subseteq V(\mathcal{G})$. It follows that if $v = v_j$ is the root of T_v, then $\gamma(G \square T) = \min_{s_i \in F} W(i+1, j+1)$. This is done through the function MIN(), which returns the minimum weight and an image of v that yields such a weight. The minimum conditional homomorphism $\phi_v : T_v \mapsto \mathcal{G}$ can be obtained easily from the appropriate entries in P, and the minimum dominating set corresponding to ϕ_v follows directly.

Algorithm 4.1: GAMMA(G, T_v, v)

comment: { Returns $\gamma(G \,\square\, T)$ and a minimum dominating set D.
T_v is the tree T rooted at vertex v.

external { STATEGRAPH(), MINHOM(), ROOTIMAGELIST()
BFS(), REVERSE(), MIN(), DOMSET(), MAKEHOM()

$\mathcal{G} \leftarrow$ STATEGRAPH(G)
treelist $\leftarrow$ BFS(T_v)
revtreelist $\leftarrow$ REVERSE(treelist)

for each u **in** *treelist*
do {
  for each s **in** $V(\mathcal{G})$
  do {
    $g_u, \phi_u^{(c)} \leftarrow$ MINHOM$(T_v, \mathcal{G}, u, s, W)$
    $W(s,u) \leftarrow g_u$
    $P(s,u) \leftarrow \phi_u^{(c)}$

$F \leftarrow$ ROOTIMAGELIST$(\mathcal{G})$
$\gamma, vimg \leftarrow$ MIN$(W(F, v))$
$\phi_v \leftarrow$ MAKEHOM$(v, vimg, P)$
$D \leftarrow$ DOMSET(ϕ)

return (γ, D)

Algorithm 4.2: ROOTIMAGELIST$(\mathcal{G})$

comment: { Returns the set of valid states for the root of a tree.
$\mathcal{G}$ is the state–transition graph.

external INC(), EMPTYLIST(), APPEND()

list $\leftarrow$ EMPTYLIST()
for each s **in** $V(\mathcal{G})$
do {
  if INC$(\mathcal{G}, s) = \underline{0}$
  then *list* $\leftarrow$ APPEND$(list, s)$

return (*list*)

Algorithm 4.3: MINHOM$(T_v, \mathcal{G}, u, s, W)$

comment: Returns the weight of a minimum homomorphism $\phi_u : T_u \mapsto \mathcal{G}$ as well as $\phi_u^{(c)} = \phi_u | C(u)$, such that $\phi_u(u) = s$. T_v is the tree T rooted at vertex v. $\mathcal{G}$ is the state–transition graph and W the weight matrix.

external OUTC(), ISLEAF(), WEIGHT(), CHILDREN(), LENGTH(), EMPTYHOM(), TUPLE(), CHILDIMAGELIST(), COLOURCHECK()

$g_u \leftarrow \infty$
$\phi_u^{(c)} \leftarrow$ EMPTYHOM()

if ISLEAF(T_v, u)
then **if** OUTC$(\mathcal{G}, s) = \underline{0}$
then $g_u \leftarrow$ WEIGHT(s)
else *childlist* $\leftarrow$ CHILDREN(T_v, u)
$k \leftarrow$ LENGTH(*childlist*)
slist $\leftarrow$ CHILDIMAGELIST$(\mathcal{G}, s)$
for each X **in** TUPLE(*slist*, k)
do **if** COLOURCHECK$(T_v, \mathcal{G}, u, s,$ *childlist*, $X)$
then *gtmp* $\leftarrow$ WEIGHT(s)
for $i \leftarrow 1$ **to** LENGTH(*childlist*)
do *gtmp* $\leftarrow$ *gtmp* $+ W($*childlist*$(i), X(i))$
if *gtmp* $< g_u$
then $g_u \leftarrow$ *gtmp*
$\phi_u^{(c)}$(*childlist*) $\leftarrow X$

return $(g_u, \phi_u^{(c)})$

When considering $T \cong P_n$, this algorithm is the same as the one used by Livingston and Stout [6] and described in Section 2. Selecting an end-vertex as the root v of the tree, the reversed BFS ordering simply corresponds to a canonical labelling of the path ending in v. The set F contains the valid images (or final states) of the right-most vertex v in the path.

We now briefly explain the subalgorithm MINHOM() that is used to determine a single entry in each of the matrices W and P. As input it takes the rooted tree T_v, the state-transition graph $\mathcal{G}$, a vertex $u \in V(T_v)$, a state

$s \in V(\mathcal{G})$ and the weight-matrix W. As mentioned previously, it determines the minimum weight g_u of a conditional homomorphism $\phi_u : T_u \mapsto \mathcal{G}$ such that $\phi_u(u) = s$, and returns this weight and $\phi_u^{(c)} = \phi_u | N_{\text{out}}(u)$, the homomorphism ϕ_u restricted to the set of children of u (if it exists).

If u is a leaf in T_v, then the algorithm verifies whether s is a valid image for u, since only states with no outgoing colour requirements are allowed to be images of leaves. If s is a valid image, g_u is set to the weight of s. Otherwise it is set to some large value, which we denote here by ∞. If u is a leaf in T_v, no homomorphism $\phi_u^{(c)}$ is returned.

Suppose u is not a leaf in T_v and let $C(u)$ denote the set of k children of u, i.e. $C(u) = N_{\text{out}}(u)$ and $k = \deg_{\text{out}}(u)$. The subalgorithm CHILDIMAGELIST() reduces the set of valid images for $C(u)$, and is shown as Algorithm 4.4. It checks that a state $t \in V(\mathcal{G})$ is adjacent from $s = \phi_u(u)$ and that the colour vector of the arc $st \in E(\mathcal{G})$ satisfies the incoming colour requirement of t. The algorithm MINHOM() calls this subset of valid images *slist*. For every k-tuple X of *slist*, it is verified by way of COLOURCHECK() that the corresponding homomorphism $C(u) \mapsto X$ is valid, in terms of the outgoing colour requirement on $s = \phi_u(u)$ and the additional requirement on the images of leaves. This subalgorithm is shown as Algorithm 4.5. Lastly, the minimum weight over all valid k-tuples is determined and returned as the value of g_u, with $\phi_u^{(c)}$ the corresponding homomorphism $T_u \mapsto \mathcal{G}$ restricted to $C(u)$.

Algorithm 4.4: CHILDIMAGELIST($\mathcal{G}, s$)

comment: { Returns the set of valid images
for the children of a vertex mapping to s.
$\mathcal{G}$ is the state–transition graph.

external INC(), CVECT(), EMPTYLIST(), APPEND()

list ← EMPTYLIST()
for each t **in** $V(\mathcal{G})$
do { **if** $(s,t) \in E(\mathcal{G})$ **and** CVECT($\mathcal{G}, s, t$) $\geq$ INC($\mathcal{G}, t$)
then *list* ← APPEND(*list*, t)

return (*list*)

Algorithm 4.5: COLOURCHECK$(T_v, \mathcal{G}, u, s, childlist, X)$

comment: { Returns **true** if the mapping given by $u \mapsto s$ and *childlist* $\mapsto X$ is valid for some homomorphism $T_v \mapsto \mathcal{G}$. T_v is a rooted tree and $\mathcal{G}$ is the state-transition graph.

external CVECT(), OUTC(), ORSUM(), ISLEAF(), LENGTH()

vect $\leftarrow \underline{0}$
for each t **in** X
  do *vect* $\leftarrow$ ORSUM(*vect*, CVECT($\mathcal{G}, s, t$))
if OUTC($\mathcal{G}, s$) $>$ *vect*
  then return (**false**)
for $i \leftarrow 1$ **to** LENGTH(*childlist*)
  do { $w \leftarrow$ *childlist*(i)
         $t \leftarrow X(i)$
         if ISLEAF(T_v, w) **and not** OUTC($\mathcal{G}, t$) $= \underline{0}$
           then return (**false**)

return (**true**)

s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8
→	→	→	→	→	←	←	←	←
→	→	←	←	↕	→	→	←	←
→	←	→	←	•	→	←	→	←

s_9	s_{10}	s_{11}	s_{12}	s_{13}	s_{14}	s_{15}	s_{16}
←	•	•	•	•	•	↕	↕
↕	•	•	↕	↕	↕	•	•
•	•	↕	→	←	•	•	↕

Table 2: All possible states of P_3, i.e. the vertices of $\mathcal{G}(P_3)$.

As an example, consider the tree T with vertces $v_0, v_1, \ldots, v_5$, rooted at v_0, as shown in Figure 5. The states in the state-transition graph $\mathcal{G}$ of P_3 are listed in Table 2 as column vectors. Calling the algorithm GAMMA(P_3, T_v, v_0) to determine $\gamma(P_3 \square T)$ yields the information shown in Table 3. This table shows corresponding entries of the matrices W and P. The set of valid states for the root v_0 is $F = \{s_0, s_4, s_{10}, s_{11}, s_{12}, s_{14}, s_{15}, s_{16}\}$. The images s_4, s_{12} and s_{16} of v_0 all yield a minimum weight of 5 for a conditional homomorphism $\phi : T_v \mapsto \mathcal{G}$, so that $\gamma(P_3 \square T) = 5$. For example,

the image s_4 of v_0 yields $\phi = \{v_0 \mapsto s_4, v_1 \mapsto s_{13}, v_2 \mapsto s_5, v_3 \mapsto s_4, v_4 \mapsto s_{13}, v_5 \mapsto s_{16}, \}$, shown in Figure 6. The corresponding minimum dominating set of $P_3 \,\square\, T$ is shown in Figure 7.

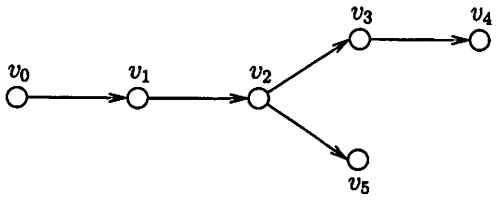

Figure 5: The tree T rooted at vertex v_4.

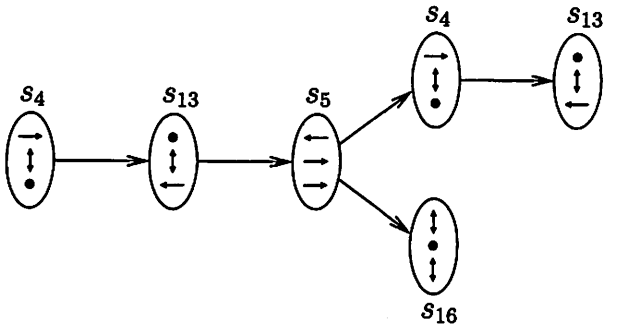

Figure 6: A minimum weight conditional homomorphism $\phi : T_v \mapsto \mathcal{G}$.

We briefly discuss the complexity of the algorithm GAMMA(). For a fixed graph G, the state-transition graph can be constructed in constant time complexity. A breadth-first search through the tree T_v of order n has complexity $\mathcal{O}(n^2)$. For each of the n vertices in the tree, each of the N states is considered, and the subalgorithm MINHOM() called. If the out-degree of a vertex u in T_v is k, then this subalgorithm considers every k-tuple of a set of at most N states, possibly looking through all the k weight-matrix entries corresponding to the children of u. Thus the complexity is $\mathcal{O}(n\Delta N^{\Delta+1} + n^2)$. For a family of trees of bounded maximum degree, this yields a polynomial time algorithm.

	v_4	v_5	v_3	v_2	v_1	v_0
s_0	$\infty \mid -$	$\infty \mid -$	$3 \mid \{v_4 \mapsto s_{10}\}$	$4 \mid \{v_3 \mapsto s_4, v_5 \mapsto s_{11}\}$	$4 \mid \{v_2 \mapsto s_{10}\}$	$6 \mid \{v_1 \mapsto s_{10}\}$
s_1	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_{11}\}$	$3 \mid \{v_3 \mapsto s_{12}, v_5 \mapsto s_{16}\}$	$4 \mid \{v_2 \mapsto s_{10}\}$	$5 \mid \{v_1 \mapsto s_{11}\}$
s_2	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_{14}\}$	$4 \mid \{v_3 \mapsto s_4, v_5 \mapsto s_{11}\}$	$4 \mid \{v_2 \mapsto s_{10}\}$	$5 \mid \{v_1 \mapsto s_{14}\}$
s_3	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_{11}\}$	$3 \mid \{v_3 \mapsto s_{12}, v_5 \mapsto s_{16}\}$	$4 \mid \{v_2 \mapsto s_{10}\}$	$5 \mid \{v_1 \mapsto s_{11}\}$
s_4	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_{13}\}$	$4 \mid \{v_3 \mapsto s_1, v_5 \mapsto s_{13}\}$	$5 \mid \{v_2 \mapsto s_{10}\}$	$5 \mid \{v_1 \mapsto s_{13}\}$
s_5	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_{15}\}$	$3 \mid \{v_3 \mapsto s_4, v_5 \mapsto s_{16}\}$	$4 \mid \{v_2 \mapsto s_{10}\}$	$5 \mid \{v_1 \mapsto s_{15}\}$
s_6	$\infty \mid -$	$\infty \mid -$	$1 \mid \{v_4 \mapsto s_{16}\}$	$3 \mid \{v_3 \mapsto s_4, v_5 \mapsto s_{16}\}$	$4 \mid \{v_2 \mapsto s_{10}\}$	$5 \mid \{v_1 \mapsto s_{11}\}$
s_7	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_{14}\}$	$3 \mid \{v_3 \mapsto s_4, v_5 \mapsto s_{16}\}$	$4 \mid \{v_2 \mapsto s_4\}$	$5 \mid \{v_1 \mapsto s_4\}$
s_8	$0 \mid -$	$0 \mid -$	$1 \mid \{v_4 \mapsto s_{16}\}$	$3 \mid \{v_3 \mapsto s_4, v_5 \mapsto s_{16}\}$	$4 \mid \{v_2 \mapsto s_0\}$	$4 \mid \{v_1 \mapsto s_0\}$
s_9	$1 \mid -$	$1 \mid -$	$2 \mid \{v_4 \mapsto s_{13}\}$	$4 \mid \{v_3 \mapsto s_1, v_5 \mapsto s_{13}\}$	$4 \mid \{v_2 \mapsto s_1\}$	$5 \mid \{v_1 \mapsto s_1\}$
s_{10}	$3 \mid -$	$3 \mid -$	$3 \mid \{v_4 \mapsto s_8\}$	$4 \mid \{v_3 \mapsto s_8, v_5 \mapsto s_8\}$	$6 \mid \{v_2 \mapsto s_8\}$	$7 \mid \{v_1 \mapsto s_8\}$
s_{11}	$2 \mid -$	$2 \mid -$	$3 \mid \{v_4 \mapsto s_9\}$	$5 \mid \{v_3 \mapsto s_7, v_5 \mapsto s_9\}$	$5 \mid \{v_2 \mapsto s_7\}$	$6 \mid \{v_1 \mapsto s_7\}$
s_{12}	$\infty \mid -$	$\infty \mid -$	$2 \mid \{v_4 \mapsto s_9\}$	$4 \mid \{v_3 \mapsto s_5, v_5 \mapsto s_9\}$	$5 \mid \{v_2 \mapsto s_9\}$	$5 \mid \{v_1 \mapsto s_9\}$
s_{13}	$1 \mid -$	$1 \mid -$	$2 \mid \{v_4 \mapsto s_9\}$	$4 \mid \{v_3 \mapsto s_5, v_5 \mapsto s_9\}$	$4 \mid \{v_2 \mapsto s_5\}$	$5 \mid \{v_1 \mapsto s_5\}$
s_{14}	$2 \mid -$	$2 \mid -$	$3 \mid \{v_4 \mapsto s_9\}$	$4 \mid \{v_3 \mapsto s_6, v_5 \mapsto s_9\}$	$5 \mid \{v_2 \mapsto s_6\}$	$6 \mid \{v_1 \mapsto s_6\}$
s_{15}	$2 \mid -$	$2 \mid -$	$3 \mid \{v_4 \mapsto s_{13}\}$	$5 \mid \{v_3 \mapsto s_3, v_5 \mapsto s_{13}\}$	$5 \mid \{v_2 \mapsto s_3\}$	$6 \mid \{v_1 \mapsto s_3\}$
s_{16}	$1 \mid -$	$1 \mid -$	$2 \mid \{v_4 \mapsto s_{16}\}$	$4 \mid \{v_3 \mapsto s_2, v_5 \mapsto s_{16}\}$	$5 \mid \{v_2 \mapsto s_2\}$	$5 \mid \{v_1 \mapsto s_2\}$

Table 3: The corresponding entries of the matrices W and P used to determine $\gamma(P_3 \square T)$.

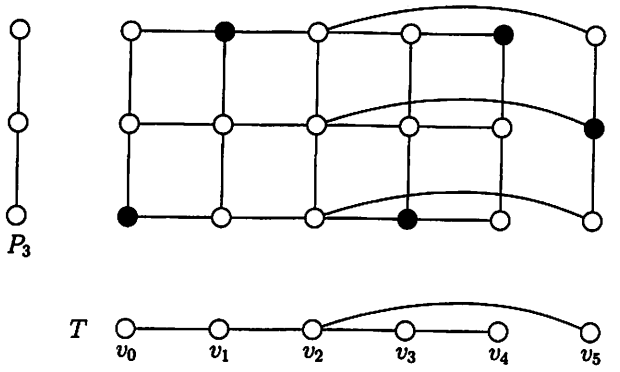

Figure 7: A minimum dominating set of $P_3 \,\square\, T$.

5 The generalized Cartesian product $G \,\boxdot\, H$

The general framework presented in Section 3 to determine the domination number of $G \,\square\, H$ need only be modified slightly to accommodate the generalized Cartesian product $G \,\boxdot\, H$. Let G and H be graphs with vertices labelled $v_1, v_2, \ldots, v_m$ and $u_1, u_2, \ldots, u_n$ respectively, and let π be any permutation of $V(G)$. For convenience we view $\pi \in S_m$ acting on the subscripts of the vertex labels of G. Also, let D be a dominating set of $G \,\boxdot\, H$ and the mapping $l_D : V(G \,\boxdot\, H) \mapsto \{\leftarrow, \rightarrow, \bullet, \updownarrow\}$ be defined as in Section 3. We write $[s]_i = l_D(v_{i,j})$ for the i^{th} entry in the state $s = l_D(G_j)$. For $a = 1, 2, \ldots, n-1$, let $\mathcal{G}_a$ denote a state-transition graph of G with vertex set the set of valid states of G, and $st \in E(\mathcal{G}_a)$ if and only if for any i:

- if $[s]_i = \bullet$, then $[t]_{\pi^a(i)} \neq \rightarrow$.

Depending on the permutation π, many of these digraphs $\mathcal{G}_a$ may be the same, but for the sake of simplicity they will be considered to be distinct here. Let $N^{(a)}(s)$ denote the neighbourhood of s in $\mathcal{G}_a$. Colour vectors $\underline{c}_a$ for the arcs of $\mathcal{G}_a$ are defined similar to those in Section 3 according to the following conditions, which hold for all i:

- $[\underline{c}_a(st)]_0 = 1$ if and only if $st \in E(\mathcal{G}_a)$;
- for any $t \in N^{(a)}_{\text{out}}(s)$ with $[t]_{\pi^a(i)} = \bullet$, $[\underline{c}_a(st)]_i = 1$ if and only if $[s]_i = \rightarrow$;

- for any $t \in N^{(a)}_{\text{out}}(s)$ with $[s]_i = \bullet$, $[\underline{c}_a(st)]_i = 1$ if and only if $[t]_{\pi^a(i)} = \leftarrow$.

Let $\overrightarrow{H}$ be an acyclic orientation of H such that $j > i$ if $u_iu_j \in E(\overrightarrow{H})$ (i.e. all the arcs are forward arcs). The labelling associated with a dominating set D of $G \boxdot H$ induces a sequence of states $\alpha_1, \alpha_2, \ldots, \alpha_n$ that corresponds uniquely to a mapping $f : V(\overrightarrow{H}) \mapsto V(\mathcal{G})$, with $f(v_i) = \alpha_i$, satisfying the following:

- if $u_iu_j \in E(\overrightarrow{H})$, then $f(u_i)f(u_j) \in E(\mathcal{G}_{j-i})$;
- if $f(u_i) = s \in V(\mathcal{G})$, $[s]_p = \leftarrow$ for $p \in P$ and $[s]_q = \rightarrow$ for $q \in Q$, $P, Q \subseteq \{1, 2, \ldots, m\}$, then

  for any $p \in P$, $[\underline{c}_{i-j}(f(u_j)f(u_i))]_p = 1$ for some $u_j \in N_{\text{in}}(u_i)$;
  for any $q \in Q$, $[\underline{c}_{j-i}(f(u_i)f(u_j))]_q = 1$ for some $u_j \in N_{\text{out}}(u_i)$.

Conversely, a mapping $f : V(\overrightarrow{H}) \mapsto V(\mathcal{G})$ satisfying the properties above corresponds to a state sequence and therefore a unique dominating set of $G \boxdot H$. The minimum weight of such a mapping yields the domination number of the generalized Cartesian product $G \boxdot H$.

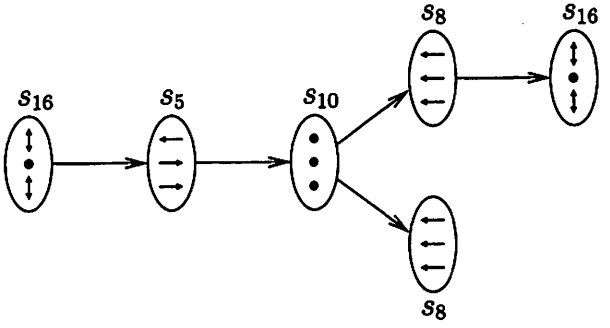

Figure 8: A minimum weight mapping $\phi : V(T_v) \mapsto V(\mathcal{G})$.

As an example, consider $P_3 \boxdot T$, where $\pi = (v_1, v_2)$ and T is the tree with vertex set $V(T) = \{v_0, v_1, \ldots, v_5\}$, rooted at v_0, as shown in Figure 5. A slight modification of the algorithm used in Section 4, in accordance with the above discussion, may be used to determine the domination number of this generalized Cartesian product (and in fact any product $G \boxdot T$ for any

tree T). The mapping $\phi = \{v_0 \mapsto s_{16}, v_1 \mapsto s_5, v_2 \mapsto s_{10}, v_3 \mapsto s_8, v_4 \mapsto s_{16}, v_5 \mapsto s_8\}$ yields a minimum weight of 5, showing that $\gamma(P_3 \boxtimes T) = 5$. This mapping and the corresponding minimum dominating set of $P_3 \boxtimes T$ are shown in Figures 8 and 9 respectively.

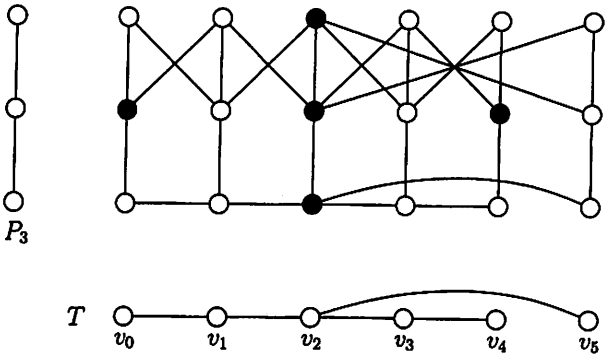

Figure 9: A minimum dominating set of $P_3 \boxtimes T$.

ACKNOWLEDGEMENTS

This paper is based on part of the first author's Ph.D. dissertation and he gratefully acknowledges financial assistance from the South African NRF prestige awards programme and the Skye Foundation. Work towards this paper was also supported financially by NSERC.

References

[1] S. BENECKE AND C.M. MYNHARDT, *Domination of generalized Cartesian products*, preprint.

[2] A.P. BURGER, C.M. MYNHARDT AND W.D. WEAKLEY, *On the domination number of prisms of graphs*, Dicuss. Math. Graph Theory, **24**(2) (2004), 303–318.

[3] G. CHARTRAND AND F. HARARY, *Planar permutation graphs*, Ann. Inst. H. Poincaré Sect. B (N.S.), **3** (1967), 433–438.

[4] T.W. HAYNES, S.T. HEDETNIEMI AND P.J. SLATER, *Fundamentals of Domination in Graphs*, Marcel Dekker, New York, 1998.

[5] W. Kocay and D. L. Kreher, *Graphs, Algorithms, and Optimization*, Chapman & Hall/CRC, Boca Raton, Florida, 2005.

[6] M. LIVINGSTON AND Q.F. STOUT, *Constant time computation of minimum dominating sets*, Congr. Numer., **105** (1994), 116–128.