Powers of Large Matrices on GPU Platforms to Compute the Roman Domination Number of Cylindrical Graphs

J. A. MARTÍNEZ [1,3], **E. M. GARZÓN** [1,3], **AND M. L. PUERTAS** [2,3]
[1]Department of Computer Science, Universidad de Almería, 04120 Almería, Spain
[2]Department of Mathematics, Universidad de Almería, 04120 Almería, Spain
[3]Agrifood Campus of International Excellence (ceiA3), Universidad de Almería, 04120 Almería, Spain

Corresponding author: J. A. Martínez (jmartine@ual.es)

ABSTRACT The Roman domination in a graph G is a variant of the classical domination, defined by means of a so-called Roman domination function $f : V(G) \rightarrow \{0, 1, 2\}$ such that if $f(v) = 0$ then, the vertex v is adjacent to at least one vertex w with $f(w) = 2$. The weight $f(G)$ of a Roman dominating function of G is the sum of the weights of all vertices of G, that is, $f(G) = \sum_{u \in V(G)} f(u)$. The Roman domination number $\gamma_R(G)$ is the minimum weight of a Roman dominating function of G. In this paper we propose algorithms to compute this parameter involving the (min, +) powers of large matrices with high computational requirements and the GPU (Graphics Processing Unit) allows us to accelerate such operations. Specific routines have been developed to efficiently compute the (min, +) product on GPU architecture, taking advantage of its computational power. These algorithms allow us to compute the Roman domination number of cylindrical graphs $P_m \square C_n$ i.e., the Cartesian product of a path and a cycle, in cases $m = 7, 8, 9 \ n \geq 3$ and $m \geq 10, n \equiv 0 \pmod 5$. Moreover, we provide a lower bound for the remaining cases $m \geq 10, n \not\equiv 0 \pmod 5$.

INDEX TERMS Cylindrical graphs, GPU platforms, (min, +) matrix multiplication, Roman domination.

I. INTRODUCTION

The efficient location of resources in a network is a well-known optimization problem that is usually approached by using graphs. The domination parameters in graphs play a central role in such problems since they can represent a wide variety of additional properties required in the distribution of resources. A dominating set in a graph G is a vertex subset S such that every vertex not in S has at least one neighbor in it. The domination number of G is the cardinal of a minimum dominating set. Applications of this parameter and some of its variations to the optimal location of radio stations or land surveying sensors can be found in [15]. Moreover, linear algorithms using domination parameters for the resource allocation in trees are studied in [16].

In this paper we focus on the Roman domination that models a classical optimization problem (see [39]). The Roman Emperor Constantine the Great, in the 4th century AD, ordered that no legion be sent out of its usual place if such place was left unprotected. Therefore, a pair of legions must be placed in some locations so that one of them could be sent to an adjacent one. Meanwhile, locations with just one legion just protect themselves. The goal is selecting the appropriate locations to place either one or two legions in order to minimize the needed forces.

Following this approach, the Roman domination in graphs was introduced in [4]. A *Roman dominating function* in a graph G is a function $f : V(G) \rightarrow \{0, 1, 2\}$ such that every vertex v with $f(v) = 0$ is adjacent to at least a vertex w satisfying $f(w) = 2$. The weight of a Roman dominating function is $f(G) = \sum_{u \in V(G)} f(u)$. The minimum weight of a Roman dominating function of G is the *Roman domination number* $\gamma_R(G)$. We denote $S_i^f = \{v \in V(G) : f(v) = i\}$ (and we will omit f if there is no confusion). Therefore, $f(G) = |S_1^f| + 2|S_2^f|$. We say that vertices in S_1 just dominate themselves while every vertex $w \in S_2$ dominates itself and its neighborhood $N(w)$.

The Cartesian product of two graphs $G \square H$ is the graph with vertex set $V(G) \times V(H)$ such that two vertices $(g_1, h_1), (g_2, h_2)$ are adjacent in $G \square H$ if either $g_1 = g_2$

J. A. Martínez *et al.*: Powers of Large Matrices on GPU Platforms to Compute the Roman Domination Number of Cylindrical Graphs

IEEE *Access*

and h_1, h_2 are adjacent in H, or g_1, g_2 are adjacent in G and $h_1 = h_2$ (see [18]). The Roman domination number remains unknown for general Cartesian product graphs while the Roman domination number of particular cases of Cartesian product of paths and cycles have been computed. The Roman domination number of the Cartesian product of two paths has been recently obtained in [36] meanwhile the problem is still open for the Cartesian product of a path and a cycle and the Cartesian product of two cycles. In both of them, solutions for small cases have been provided by using an algorithmic approach (see [33]).

Regarding the computational complexity of these problems, the computation of the domination number is NP-complete in general graphs (see [10]). Moreover, it remains NP-complete when restricted to bipartite or chordal graphs and it is polynomial in a few graphs classes such as trees or interval graphs (see [15]).

In a similar way, the computation of the Roman domination number is also an NP-complete problem, even in bipartite, planar or chordal graphs and it can be computed in linear time in trees (see [4]). In the same paper, the authors also stated that there is a $2 \log n$ approximation algorithm for the Roman domination number. More recently, a $2(1 + \log(\Delta + 1))$-approximation algorithm, with Δ the maximum degree of the graph, has been presented in [32] to find a Roman domination function with minimum weight.

The great interest to express graph algorithms in terms of tropical algebra operations is well known [22]. From a computational point of view, this approach involves several challenges due to the high dimension of the matrices that take part in such algorithms. This way, many works have focused on taking advantage of the sparsity of the matrices and the regularity of specific graphs to reduce the complexity of the corresponding matrix computations [5], [6], [8]; the goals of other works are the optimal computational implementations of the primitive matrix operations related to this field to exploit modern multicore and GPU platforms [17], [23], [41].

In this work, the study of the Roman domination in graphs relies on the computation of matrix powers. It is expressed as a $(\min, +)$ product sequence which starts with sparse matrices which are filled as new products are computed. Therefore, the best option is to compute the matrix power with dense data structures. However, it implies a very high computational complexity in terms of run-time and memory requirements since the size of matrices strongly increases with the dimensions of the cylindrical graphs.

The cylinder $P_m \square C_n$ is the Cartesian product of the path with m vertices P_m and the cycle with n vertices C_n, that is, the graph with vertex set $V(P_m) \times V(C_n)$ such that two vertices (u_1, v_1), (u_2, v_2) are adjacent in $P_m \square C_n$ if $u_1 = u_2$ and v_1, v_2 are adjacent in C_n or u_1, u_2 are adjacent in P_m and $v_1 = v_2$. As we have said, there is no general formula for the Roman domination number in this graph family and just exact values of $\gamma_R(P_m \square C_n)$ with $2 \le m \le 6$ or $2 \le n \le 8$ are known (see [33]).

In this paper we provide an algorithm involving powers of large matrices, that allows us to compute the exact values of $\gamma_R(P_7 \square C_n)$ and $\gamma_R(P_8 \square C_n)$. To this end, we have developed a GPU version of $(\min, +)$ powers of dense matrices relying on the routine MatrixMult of [30]. Moreover, graphs of higher dimensions could be studied on GPUs with larger memory capacity than used in the experimental study. Finally, we will use a modification of the algorithm to obtain a lower bound of $\gamma_R(P_m \square C_n)$, for $m \ge 9$, that gives the exact value of this parameter if $n \equiv 0 \pmod 5$.

II. RELATED WORK

The Roman domination in graphs has been widely studied since it was formally defined in 2004 (see [4]) and several hundreds of papers about it can be found in literature. We quote some references as an example: for instance, general upper bounds of the Roman domination number in terms of the number of vertices of the graphs were first obtained in [2] and this problem has also been studied in [20], [27].

A different point of view is the study of the relationship between the Roman domination number and other domination parameters, for instance with the domination number in [7] or with variations of the Roman domination itself, such as the Roman-{2}-domination in [28]. Moreover, the Roman domination number can be defined in directed graphs and in [14] the relationship between the domination number and the Roman domination number in such graphs is considered. The algorithmic point of view has also been studied, as an example in [26] authors showed that the Roman domination number of cographs and interval graphs can be computed in linear time.

Furthermore, a number of variations of the original definition have recently been analyzed. As an example, the Italian domination number is computed in some Cartesian product graphs with algorithmic procedures in [9]; the sum and the product of the double Roman domination numbers of a graph and its complement are considered in [21]; the Roman-{2}-bondage number is computed in [29] for some graph families such as paths, cycles, complete bipartite graphs, trees, unicyclic graphs and planar graphs; the total double Roman domination number is studied in [37], where some general upper bounds in terms of the number of vertices and the maximum degree are obtained.

It is well-known that the domination properties are difficult to handle in the Cartesian product graphs and computing such parameters in them has attracted attention since the Vizing Conjecture, which remains open, was formulated in 1968 (see [40]). As an example, the problem of computing the domination number of the Cartesian product of two paths was open for almost thirty years and it was finally solved in [11], where authors provide the value of this parameter in terms of the number of vertices of both paths. In previous works, upper and lower bounds can be found, as in [3], [12], [19]. Moreover, in [19], [38] the exact values of some small cases were computed. This problem is still open in other Cartesian products of two graphs involving paths,

cycles or more general graphs. Also, some partial results about the domination number of the Cartesian product of k paths can be found in [12] and for the case of k cycles in [24].

Among the techniques used to compute domination parameters in Cartesian product graphs, an algorithmic approach using matrix powers has provided significant results in cases involving paths and cycles. The final paper for the domination number of the Cartesian product of two paths [11] uses this technique, which was presented in [25] for fasciagraphs and rotagraphs, of which Cartesian products of paths and cycles are particular cases. Later, in [34] the authors followed these ideas to compute the domination number of the Cartesian product of two cycles and the Cartesian product of a path and cycle, but just in some small cases. The same authors have adapted the technique to obtain the Roman domination number of some Cartesian product graphs involving small paths and/or cycles in [33]. Recently, in [36] the Roman domination number of the Cartesian product of two paths has been completely computed following the same ideas.

III. PRELIMINARY RESULTS
In this section we present the needed tools to provide the algorithms that we will use to compute the Roman domination number in selected cylinders.

The first tool is the $(\min, +)$ matrix algebra over the semi-ring $\mathcal{P} = (\mathbb{R} \cup \{\infty\}, \min, +, \infty, 0)$ of tropical numbers in the minimum convention (see [35]). The $(\min, +)$ matrix multiplication $\bigotimes$ is defined by $C = A \bigotimes B$, being the matrix where for all i, j, $c_{i,j} = \min_k(a_{i,k} + b_{k,j})$.

Moreover, the $(\min, +)$ product of a matrix A and $\alpha \in \mathbb{R} \cup \{\infty\}$ is defined by $(\alpha \bigotimes A)_{i,j} = \alpha + a_{i,j}$. Therefore, $(A \bigotimes (\alpha \bigotimes B))_{i,j} = \min_k(a_{i,k} + (\alpha \bigotimes B)_{k,j}) = \min_k(a_{i,k} + (\alpha + b_{k,j})) = \min_k(\alpha + (a_{i,k} + b_{k,j})) = \alpha + \min_k(a_{i,k} + b_{k,j}) = \alpha + (A \bigotimes B)_{i,j}$. Hence $A \bigotimes (\alpha \bigotimes B) = \alpha \bigotimes (A \bigotimes B)$.

The second ingredient that we need is the following result (see [1]) that we quote from [25]. We just need the particular case related to the tropical semi-ring.

Let G be a digraph with $V(G) = \{v_1, v_2, \ldots, v_s\}$ together with a labeling function ℓ which assigns an element of $\mathcal{P}$ to every arc of G. A path of length k in G is a sequence of k consecutive arcs $Q = (v_{i_0} v_{i_1})(v_{i_1} v_{i_2}) \ldots (v_{i_{k-1}} v_{i_k})$ and Q is a *closed path* if $v_{i_0} = v_{i_k}$. The labeling ℓ can be easily extended to paths

$$\ell(Q) = \ell(v_{i_0} v_{i_1}) + \ell(v_{i_1} v_{i_2}) + \cdots + \ell(v_{i_{k-1}} v_{i_k}).$$

Theorem 1: Let S_{ij}^k be the set of all paths of length k from v_i to v_j in G and let $A(G)$ be the matrix defined by

$$A(G)_{ij} = \begin{cases} \ell(v_i, v_j) & \text{if } (v_i, v_j) \text{ is an arc of } G, \\ \infty & \text{otherwise.} \end{cases}$$

If $A(G)^k$ is the k-th $(\min, +)$ power of $A(G)$, then

$$(A(G)^k)_{ij} = \min\{\ell(Q) : Q \in S_{ij}^k\}.$$

Finally, we will also use the following lemma about the $(\min, +)$ matrix multiplication. It is a standard argument and we include its proof here for the sake of completeness.

Lemma 2: Let A be a square matrix and suppose that there exist natural numbers n_0, a, b such that $A^{n_0+a} = b \bigotimes A^{n_0}$ then, $A^{n+a} = b \bigotimes A^n$, for every $n \geq n_0$.

Proof: We proceed by induction. By hypothesis, $A^{n_0+a} = b \bigotimes A^{n_0}$. Let $n \geq n_0$ be such that $A^{n+a} = b \bigotimes A^n$ then, $A^{(n+1)+a} = A \bigotimes A^{n+a} = A \bigotimes (b \bigotimes A^n) = b \bigotimes (A \bigotimes A^n) = b \bigotimes A^{n+1}$, as desired. □

IV. COMPUTATION OF $\gamma_R(P_7 \square C_n)$ AND $\gamma_R(P_8 \square C_n)$
In this section we provide an algorithm to compute the exact values of $\gamma_R(P_m \square C_n)$, for $m = 7, 8$, by using Theorem 1. Our approach follows the ideas in [33], [36]. In the first paper, authors compute the Roman domination number of $P_m \square C_n$, with $m \leq 6$, and we use a modification of their algorithm, following the techniques shown in the second reference to compute the following two cases.

Our algorithm constructs a matrix for each value of m and computes some $(\min, +)$ powers of it. The sizes of such matrices grow exponentially with m and it is not expected that it could compute cases much larger than $m = 8$, even by using additional computing resources. However, these small cases play an important role in formulating a conjecture about the behavior of the general case. Although it does not happen for very small m, a regular behavior could appear as in the case of the Cartesian product of two paths. This is the reason for our interest to compute as many small cases as possible.

A. THEORETICAL RESULTS
First of all, we encode the vertex set of $P_m \square C_n$. We say that it has m rows and n columns, each row being a cycle with n vertices and each column being a path with m vertices (see Figure 1). The m rows are numerated from top to bottom and the n columns from left to right. We will consider that the last column is the previous column of the first one (and the first column follows the last one).
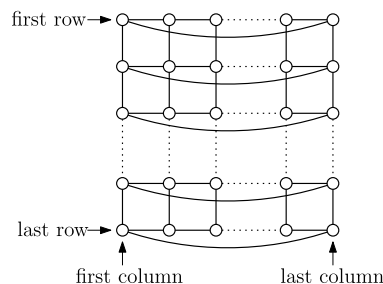

FIGURE 1. The cylinder $P_m \square C_n$.

Let f be a Roman dominating function of $P_m \square C_n$, with associated vertex partition $S_i = \{v \in V(P_m \square C_n) : f(v) = i\}$, for $i = 0, 1, 2$. We assign a label to each vertex in the following way.

- $v = a$ if $v \in S_2$,
- $v = b$ if $v \in S_1$,

J. A. Martínez *et al.*: Powers of Large Matrices on GPU Platforms to Compute the Roman Domination Number of Cylindrical Graphs

IEEE *Access*

- $v = c$ if $v \in S_0$ and it has at least one neighbor in S_2 in its column or in the previous one,
- $v = d$ if $v \in S_0$ and it has no neighbor in S_2 in its column nor in the previous one.

Now each column is a word of length m in the alphabet $\{a, b, c, d\}$ and the function f can be uniquely identified with a sequence of n consecutive words. Note that not any word can appear associated to such functions because the definition of the labeling implies that letter sequences ad, da are not possible.

Remark 3: If a word associated to a Roman dominating function contains any of the sequences ab, ba, bb, we can replace them with ac, ca, ac respectively to obtain another Roman dominating function with, at most, the same weight.

Bearing in mind these considerations, we pose the following definition.

Definition 4: A correct word of length m in the alphabet $\{a, b, c, d\}$ is a sequence of m letters not containing ad, da, ab, ba, bb.

Note that some correct words cannot appear together associated to a Roman dominating function, again because of the definition of the labeling. We now list the needed conditions for a pair of correct words $\mathbf{p} = (p_1, \ldots p_m)$ and $\mathbf{q} = (q_1, \ldots q_m)$ (we will denote words in bold font and their letter in italic font), in order to make it possible that $\mathbf{p}$ can follow $\mathbf{q}$. Note that we also avoid combinations ab, ba, bb in the same row (see Remark 3), in order to reduce the number of suitable sequences of words.

1) conditions for the first row
   - if $q_1 = a$, then $p_1 = a$ or $p_1 = c$
   - if $q_1 = b$, then $\big(p_1 = c$ and $p_2 = a\big)$ or $p_1 = d$
   - if $q_1 = c$, then $p_1 = a$ or $p_1 = b$ or $\big(p_1 = c$ and $p_2 = a\big)$ or $p_1 = d$
   - if $q_1 = d$, then $p_1 = a$

2) conditions for the intermediate rows $2 \leq i \leq m - 1$
   - if $q_i = a$, then $p_i = a$ or $p_i = c$
   - if $q_i = b$, then $\big(p_i = c$ and $p_{i-1} = a\big)$ or $\big(p_i = c$ and $p_{i+1} = a\big)$ or $p_1 = d$
   - if $q_i = c$, then $p_i = a$ or $p_i = b$ or $\big(p_i = c$ and $p_{i-1} = a\big)$ or $\big(p_i = c$ and $p_{i+1} = a\big)$ or $p_1 = d$
   - if $q_i = d$, then $p_i = a$

3) conditions for the last row
   - if $q_m = a$, then $p_m = a$ or $p_m = c$
   - if $q_m = b$, then $\big(p_m = c$ and $p_{m-1} = a\big)$ or $p_m = d$
   - if $q_m = c$, then $p_m = a$ or $p_m = b$ or $\big(p_m = c$ and $p_{m-1} = a\big)$ or $p_m = d$
   - if $q_m = d$, then $p_m = a$

Remark 5: Denote by $\mathcal{S}$ the set of all the sequences $\mathbf{p_1}, \mathbf{p_2} \ldots \mathbf{p_n}$ of n correct words of length m such that $\mathbf{p_1}$ can follow $\mathbf{p_n}$ and $\mathbf{p_{i+1}}$ can follow $\mathbf{p_i}$, for $1 \leq i \leq n - 1$. From Remark 3, the set $\mathcal{R_S}$ of Roman dominating functions associated to the sequences in $\mathcal{S}$ contains at least one function with minimum weight.

In order to apply Theorem 1, we consider the digraph G whose vertex set is $V(G) = \{$correct words of length m in the alphabet $\{a, b, c, d\}\}$, and such that there is an arc from the word $\mathbf{q}$ to the word $\mathbf{p}$ if $\mathbf{p}$ can follow $\mathbf{q}$. Moreover, we define the following labeling function ℓ which assigns to every arc of G an element of the semi-ring of tropical numbers $\mathcal{P}$: $\ell(\mathbf{q}, \mathbf{p}) = 2\mathbf{p}(a) + \mathbf{p}(b)$, where $\mathbf{p}(a) =$ number of $a's$ in the word $\mathbf{p}$ and $\mathbf{p}(b) =$ number of $b's$ in the word $\mathbf{p}$.

Proposition 6: Let $f \in \mathcal{R_S}$ be a Roman dominating function of $P_m \square C_n$ and let $\mathbf{p_1 p_2} \ldots \mathbf{p_n}$ be the sequence of correct words associated to f. Then, $Q = (\mathbf{p_1 p_2})(\mathbf{p_2 p_3}) \ldots (\mathbf{p_{n-1} p_n})(\mathbf{p_n p_1})$ is a closed path in the digraph G such that $\ell(Q) = f(P_m \square C_n)$, the weight of f.

Proof: Clearly Q is a closed path in the digraph G, because there is an arc from $\mathbf{p_i}$ to $\mathbf{p_{i+1}}$ (for $1 \leq i \leq n-1$) and from $\mathbf{p_n}$ to $\mathbf{p_1}$. Moreover, $\ell(Q) = \ell(\mathbf{p_1 p_2}) + \ell(\mathbf{p_2 p_3}) + \cdots + \ell(\mathbf{p_{n-1} p_n}) + \ell(\mathbf{p_n p_1}) = (2\mathbf{p_2}(a) + \mathbf{p_2}(b)) + (2\mathbf{p_3}(a) + \mathbf{p_3}(b)) + \cdots + (2\mathbf{p_n}(a) + \mathbf{p_n}(b)) + (2\mathbf{p_1}(a) + \mathbf{p_1}(b)) = 2(\mathbf{p_1}(a) + \mathbf{p_2}(a) + \ldots \mathbf{p_n}(a)) + (\mathbf{p_1}(b) + \mathbf{p_2}(b) + \ldots \mathbf{p_n}(b)) = 2|S_2^f| + |S_1^f| = f(P_m \square C_n)$. □

Corollary 7: Let $A(G)$ be the matrix defined by

$$A(G)_{\mathbf{qp}} = \begin{cases} \ell(\mathbf{q}, \mathbf{p}) & \text{if } (\mathbf{qp}) \text{ is an arc of } G, \\ \infty & \text{otherwise.} \end{cases}$$

Then $\min_{\mathbf{p}}(A(G)^n)_{\mathbf{pp}} = \gamma_R(P_m \square C_n)$.

Proof: From Theorem 1 we know that $(A(G)^n)_{\mathbf{pp}} = \min\{\ell(Q): Q \in S_{\mathbf{pp}}^n\}$, where $S_{\mathbf{pp}}^n$ is the set of all closed paths of length n from $\mathbf{p}$ to $\mathbf{p}$ in G. Clearly, the closed path $Q = (\mathbf{pp_2}) \ldots (\mathbf{p_{n-1} p_n})(\mathbf{p_n p})$ belongs to $S_{\mathbf{pp}}^n$ if and only if the sequence of words $\mathbf{p}, \mathbf{p_2}, \ldots \mathbf{p_n}$ belong to the set $\mathcal{S}$. Therefore, from Remark 5 and Proposition 6, $\min_{\mathbf{p}}(A(G)^n)_{\mathbf{pp}} = \min_{\mathbf{p}}(\min\{\ell(Q): Q \in S_{\mathbf{pp}}^n\}) = \min_{\mathbf{p}}(\min\{f(P_m \square C_n): f \in \mathcal{R_S}, \mathbf{p} \text{ is the first column}\}) = \gamma_R(P_m \square C_n)$. □

The Corollary above allows us to compute the Roman domination number of a cylinder $P_m \square C_n$, where both m and n are fixed. The recurrence argument for $(\min, +)$ matrix powers shown in Lemma 2 gives the opportunity to compute the Roman domination number of $P_m \square C_n$, just fixing one of the sizes but not both of them, as we show in the following result.

Proposition 8: Let m be a natural number and consider the digraph G and the matrix $A(G)$ constructed above. If $A(G)^{n_0 + \alpha} = \beta \bigotimes A(G)^{n_0}$ for natural numbers n_0, α, β, then $\gamma_R(P_m \square C_{n+\alpha}) - \gamma_R(P_m \square C_n) = \beta$, for $n \geq n_0$.

Proof: If $A(G)^{n_0 + \alpha} = \beta \bigotimes A(G)^{n_0}$ then, by using Lemma 2 we obtain that $A(G)^{n+\alpha} = \beta \bigotimes A(G)^n$, for $n \geq n_0$. Moreover, Corollary 7 gives $\gamma_R(P_m \square C_{n+\alpha}) = \min_{\mathbf{p}}(A(G)^{n+\alpha})_{\mathbf{pp}} = \min_{\mathbf{p}}(\beta \bigotimes A(G)^n)_{\mathbf{pp}} = \beta + \min_{\mathbf{p}}(A(G)^n)_{\mathbf{pp}} = \beta + \gamma_R(P_m \square C_n)$. □

The unique solution of the finite difference equation $\gamma_R(P_m \square C_{n+\alpha}) - \gamma_R(P_m \square C_n) = \beta, n \geq n_0$, with the boundary values $\gamma_R(P_m \square C_k), n_0 \leq k \leq n_0 + (\alpha - 1)$, provides $\gamma_R(P_m \square C_n)$, for m fixed and $n \geq n_0$.

B. THE ALGORITHMS

We can now describe the basic algorithm to compute the Roman domination number of the cylinder $P_m \square C_n$, with m and n fixed, by using Corollary 7.

Algorithm 1 Computation of $\gamma_R(P_m \square C_n)$, for m, n Fixed

Require: m, n natural numbers
Ensure: $\gamma_R(P_m \square C_n)$
1: compute all correct words of length m
2: compute matrix $A(G)$
3: compute the (min, +) matrix power $A(G)^n$
4: **return** $\min_{\mathbf{p}}(A(G)^n)_{\mathbf{pp}}$

This algorithm allows us to compute the Roman domination number for cylinders $P_m \square C_n$ where m, n are small. On the one hand, the size of matrix $A(G)$ is the number of correct words, which is in the order of 4^m. This size grows exponentially thus, the algorithm is useful only if m is small enough. On the other hand, it is needed to compute the $n - th$ power of the matrix $A(G)$ and, in spite of being a sparse matrix, it becomes dense after a small number of multiplications. This gives that parameter n can not be large so then it is possible to run the algorithm in a short time.

We now present the algorithm to compute the Roman domination number of $P_m \square C_n$, where just m is fixed. We use Proposition 8, so the algorithm compute the integers n_0, α, β that we need to solve the finite difference equation that gives the formula for $\gamma_R(P_m \square C_n)$.

Algorithm 2 Computation of $\gamma_R(P_m \square C_n)$, for m Fixed

Require: m a natural number
Ensure: the finite difference equation
  $\gamma_R(P_m \square C_{n+\alpha}) - \gamma_R(P_m \square C_n) = \beta$, for $n \geq n_0$,
  or finite difference equation not found
1: compute all correct words of length m
2: compute matrix $A(G)$
3: compute every (min, +) power $A(G)^k$, for $k \leq K$ big enough
4: **if** $A^{n_0+\alpha} = \beta \bigotimes A^{n_0}$ for n_0, α, β natural numbers **then**
5:     **return** n_0, α, β
6: **else**
7:     **return** recurrence not found
8: **end if**

There are some sufficient conditions to ensure that Step 4 in Algorithm 2 is true (see [38]). However, these theoretical results provide a huge value for n_0, in the order of the square of the matrix size, and it is not practical. We have looked for the desired relationship just by checking the powers computed in Step 3, with $K = 50$.

C. COMPUTATIONAL RESULTS

In this subsection we present the details of the implementation in C programming language of the algorithms and the results we have obtained. As we said before, for each m the

number C_m of correct words is of the order of 4^m, which is the number of all words of length m that can be done with 4 letters. The first step in both algorithms is the computation of correct words, having a computational complexity of $O(4^m)$. Due to the small values of m considered, this computing time is not relevant. The second step is the computation of the matrix $A(G)$, whose size C_m grows exponentially with m so, this step is $O((C_m)^2)$. Moreover, this computation has plenty of control flow instructions therefore, it can take better advantage of CPU architecture than GPU architecture.

In Table 1 we quote the matrix sizes for $2 \leq m \leq 11$, the memory requirements and the computing time of such matrices if $m \leq 9$. In cases $m = 10$ and 11, we have estimated the memory requirements to store each matrix bearing in mind the matrix sizes and that each entry is an int data (4 bytes). An Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz with 8 cores (16 threads) and 64 GB of RAM has been used to compute the matrices in both algorithms.

TABLE 1. Matrix sizes, memory requirements and GPU running time of matrix computation line 2 in Algorithm 1 and Algorithm 2.

m	Matrix size C_m	Memory size (approx.)	Computing time of matrix $A(G)$
2	11	< 1MB	< 1 s
3	33	< 1MB	< 1 s
4	97	< 1MB	< 1 s
5	287	< 1 MB	< 1 s
6	848	2.7 MB	< 1 s
7	2507	23.9 MB	3.21 s
8	7411	209.5 MB	30.04 s
9	21909	1.78 GB	4 min 47 s
10	64769	estimated 15 GB	–
11	191476	estimated 136 GB	–

Regarding the third step in both algorithms, the complexity of the matrix multiplication operation is $O((C_m)^3)$ so, it consumes most of the running time of both algorithms and it is computed K times to get $A(G)^K$. The (min, +) matrix product consists of a modification of the usual product that keeps the complexity.

To explore the computational requirements of these operations, we have carried out some examples of matrix multiplication by computing one product with an OpenMP CPU implementation (same CPU as before) by using 16 number of threads. In the largest case $m = 9$, the matrix $A(G)$ has a size of 21909 and the computation of just one (min, +) product takes 7 hours and 30 minutes.

Due to the long running time, we have used a GPU to accelerate the matrix powers computation in both algorithms. Such powers have been carried out by a modification of the routine MatrixMul, available in the NVIDIA CUDA TOOLKIT 11 [30] and described in the CUDA C Programming Guide (see [31], Chapter 3), to adapt it to the (min, +) multiplication. We call this new routine CuMatrixTrop. Such code takes advantage of shared memory to optimize the memory access. We have adapted the kernel code by changing the

J. A. Martínez et al.: Powers of Large Matrices on GPU Platforms to Compute the Roman Domination Number of Cylindrical Graphs

IEEE *Access*

usual operations in the matrix multiplication: the mapping operation, which is the multiplication, is replaced by the addition and the reduction operation, which is the addition, is replaced by the minimum.

We have used a CUDA block size 32×32 and a padding matrix approach to optimize the performance of CuMatrixTrop. The padding consists of filling the matrix with additional rows and columns of the neutral element, i.e. infinite, until the matrix size is a multiple of 32. These strategies give an efficient exploitation of the GPU, which we have analyzed with the NVIDIA profiling tool *nvprof*. In all cases, the achieved occupancy, the global memory load/store efficiency and the warp execution efficiency are near to 100%. The shared memory efficiency is 66% due to the lower parallelism of the reduction operation.

As we said before, we have run Algorithm 2 with $K = 50$ and this is enough in cases $2 \leq m \leq 9$, for which the algorithm returns the integers n_0, α and β. We have used an NVIDIA Tesla K80 GPU, with 12GB of memory, 13 multiprocessors with 192 cores in each multiprocessor (2496 cores CUDA) and shared memory of 49152 bytes, to compute $A(G)^k$ with $k \leq 50$. In Table 2 we show the results for $2 \leq m \leq 9$.

TABLE 2. Running times on K80 GPU to compute the matrix powers in Algorithm 2.

m	Computing time of 50 powers	n_0	α	β
2	< 1 s	6	4	4
3	< 1 s	11	4	6
4	< 1 s	16	5	10
5	1.19 s	16	5	12
6	8.42 s	19	5	14
7	1 min 11 s	21	5	16
8	11 min 14 s	21	5	18
9	76 min 27 s	22	5	20

Note that it is necessary to save every matrix $A(G)^k$ with $2 \leq k \leq 50$, in order to complete steps 4 to 8 in Algorithm 2. We have evaluated the profiling of the $A(G)^{50}$ computation based on CuMatrixTrop in terms of GPU running time, disk storage time and CPU-GPU communications. The results are shown in Table 3.

TABLE 3. Profiling results of computing $A(G)^{50}$ in Algorithm 2.

m	CuMatrixTrop (% over total time)	CPU-GPU communication times (% over total time)	Disk storage (% over total time)
6	0.30 s (3.53%)	0.02 s (0.24%)	8.12 s (96.23%)
7	4.92 s (6.94%)	0.12 s (0.17%)	66.08 s (92.89%)
8	139.34 s (20.67%)	1.01 s (0.15%)	534.66 s (79.18%)
9	3100.00 s (67.58%)	8.78 s (0.19%)	1487.00 s (32.23%)

The disk storage needed in the algorithm consumes a relevant amount of resources. The disk storage is carried out on a Network File System (NFS) that slows down with the heavy network traffic due to the communication of large matrices. So, this is a bottleneck in our experimental environment. However, the penalties due to the CPU-GPU communications are not significant so, the overload due to GPU exploitation has a negligible impact on performance.

Finally, we have solved the finite difference equations provided by Proposition 8, for $2 \leq m \leq 9$, with the parameters shown in Table 2 and boundary values (that is, the cases $n_0 \leq n \leq n_0 + (\alpha - 1)$) computed with Algorithm 1.

The remaining values (cases $3 \leq n < n_0$), have also been computed with Algorithm 1. We just show the final formula for $m = 7, 8, 9$ which are the new cases, for $m \leq 6$ we have obtained the same results as [33].

$$\gamma_R(P_7 \square C_n) = \begin{cases} \lceil \frac{16n}{5} \rceil & \text{if } n \equiv 0 \pmod 5 \\ \lceil \frac{16n}{5} \rceil + 1 & \text{otherwise} \end{cases}$$

$$\gamma_R(P_8 \square C_n) = \begin{cases} \lceil \frac{18n}{5} \rceil & \text{if } n \equiv 0 \pmod 5 \\ \lceil \frac{18n}{5} \rceil + 1 & \text{if } n \equiv 2, 3, 4 \pmod 5, \\ & \text{or } n = 6 \\ \lceil \frac{18n}{5} \rceil + 2 & \text{otherwise} \end{cases}$$

$$\gamma_R(P_9 \square C_n) = \begin{cases} 4n & \text{if } n \equiv 0 \pmod 5 \\ 4n + 2 & \text{otherwise} \end{cases}$$

V. A LOWER BOUND OF $\gamma_R(P_m \square C_n)$ AND THE COMPUTATION OF $\gamma_R(P_m \square C_{5k})$

As we have said before, the size of the matrix is a decisive factor when running Algorithm 1 and Algorithm 2, and matrices for values $m \geq 10$ are too big to allocate two of them in the GPU memory, to perform the matrix product operation. So we have developed a different approach that we show in this section. We have adapted both algorithms to compute the loss of a Roman dominating function in a cylinder. The loss, originally called wasted domination in [13], has been used to study several domination parameters in the Cartesian product of two paths (see [11], [36]). We follow these ideas to compute a lower bound of the Roman domination number for cylinders of large size and with a small computational cost. Although this bound is not tight in general, it provides the exact value when $n \equiv 0 \pmod 5$.

We would also like to point out that both versions of the algorithms are complementary. The computation of small cases is needed to conjecture the lower bound, that will later be confirmed by the modified algorithm. Moreover, this second version covers, at least with a lower bound, those cases that are not within the scope of the original algorithm.

Our conjecture for the lower bound of the Roman domination number in cylinder $P_m \square C_n$ comes from the regular pattern followed by values of α, β in Table 2 for $4 \leq m \leq 9$, that is $\alpha = 5$ and $\beta = 2(m + 1)$. Assuming that such pattern remains the same for larger values of m, we conjecture that $\lceil \frac{2(m+1)n}{5} \rceil \leq \gamma_R(P_m \square C_n)$.

Definition 9: Let f be a Roman dominating function of $P_m \square C_n$, $m \geq 10$, with weight $f(P_m \square C_n)$. We define the loss of f as $L(f) = \frac{5}{2}(f(P_m \square C_n)) - mn$ and we denote the minimum loss of a Roman dominating function in $P_m \square C_n$ by $\mathcal{L}(m, n)$.

The following lemma is the key result to obtain our lower bound and its proof follows the techniques shown in Section IV, with the needed modifications. This proof is quite long and we present it in Appendix, in order to make this section easier to follow.

Lemma 10: $\mathcal{L}(m, n) \geq n$, for $m, n \geq 10$.

We can now present the announced result with a general lower bound of the Roman domination number in cylinders.

Theorem 11: $\left\lceil \frac{2(m+1)n}{5} \right\rceil \leq \gamma_R(P_m \square C_n)$, for $m, n \geq 10$.

Proof: $\mathcal{L}(m, n) = \min_f L(f) = \min_f(\frac{5}{2}(f(P_m \square C_n)) - mn) = \frac{5}{2}\gamma_R(P_m \square C_n) - mn$. This gives that $\gamma_R(P_m \square C_n) = \frac{2}{5}(mn + \mathcal{L}(m, n))$. Finally, from Lemma 10, $\gamma_R(P_m \square C_n) = \frac{2}{5}(mn + \mathcal{L}(m, n)) \geq \frac{2}{5}(mn + n) = \frac{2(m+1)n}{5}$. Therefore $\left\lceil \frac{2(m+1)n}{5} \right\rceil \leq \gamma_R(P_m \square C_n)$, as desired. $\square$

This bound provides the exact value if $n \equiv 0 \pmod 5$.

Corollary 12: If $m, n \geq 10$ and $n \equiv 0 \pmod 5$, then $\gamma_R(P_m \square C_n) = \frac{2(m+1)n}{5}$.

Proof: In Figure 2 we show a Roman dominating function f in $P_{10} \square C_{10}$: black vertices have image 2, grey ones have image 1 and white ones have image 0. This regular construction can be repeated in any cylinder with $m, n \geq 10$ and $n \equiv 0 \pmod 5$ and it has $\frac{n}{5}$ black vertices in each row and $\frac{n}{5}$ grey vertices in the first and the last rows. Therefore, its weight is $f(P_m \square C_n) = 2m\frac{n}{5} + 2\frac{n}{5} = \frac{2(m+1)n}{5}$. This means that $\gamma_R(P_m \square C_n) \leq \frac{2(m+1)n}{5}$ and Theorem 11 gives the desired equality. $\square$
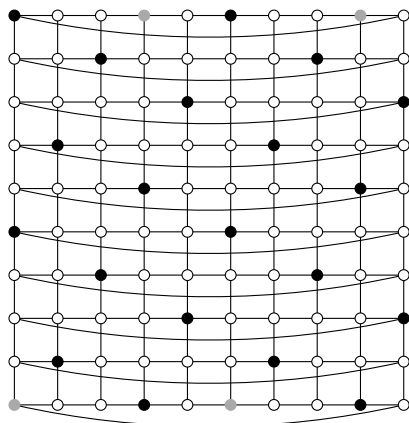

FIGURE 2. Roman domination in $P_{10} \square C_{10}$.

Remark 13: The corollary above and the cases computed in Section IV and in [34] give that $\gamma_R(P_m \square C_n) = \frac{2(m+1)n}{5}$ if $m, n \geq 4$ and $n \equiv 0 \pmod 5$. Meanwhile, the Roman domination number is smaller if $m = 2, 3$.

VI. CONCLUSION

Algorithms 1 and 2 provide the exact value of $\gamma_R(P_m \square C_n)$ where m is fixed and their high computational requirements

restrict the maximum value of m for which the algorithms are useful. In this work we have computed $\gamma_R(P_m \square C_n)$ for $m = 7, 8, 9$ according to the available platforms and these results could be extended by using more powerful platforms. Additionally, we have computed the Roman domination number of cylinders $P_m \square C_n$ in cases $m \geq 10$, $n \equiv 0 \pmod 5$, expanding the family of such graphs whose Roman domination number is known. We have also provided a lower bound of this parameter if $m \geq 10$ and $n \not\equiv 0 \pmod 5$. We have implemented algorithms that allow to compute the exact values for selected cases and the general lower bound, by using the (min, +) powers of large matrices in GPU platforms.

Our technique involves the computation of the loss of Roman dominating sets. Computing the loss has provided the exact values of several domination parameters in the Cartesian product of two paths (see [11], [36]). We have applied similar techniques to the Roman domination number in cylinders and we have obtained the exact values only in some cases, but a lower bound in other ones. This different behaviour is due to several reasons. On the one hand, this technique computes the loss located at the border of the graph. Therefore, in the cylinder $P_m \square C_n$ just upper and lower borders play a role, showing information about the loss that depends on the parameter n but avoiding what depends on m. However, in $P_m \square P_n$ the loss depending on both m and n can be computed by using the four borders.

On the other hand, we think that the formula for the general case will depend on the parity of $n \pmod 5$. Meanwhile, the Cartesian product of two paths presents a unique formula for big enough m and n. It is likely that a technique other than the computation of the loss will be necessary to completely solve this problem.

Moreover, it would be interesting to compute the exact values in additional small cases, in order to conjecture formulas for any value of both m and n.

APPENDIX
PROOF OF LEMMA 10

Our strategy to prove this lemma is similar to [13], so we are going to compute the loss just in both borders of the cylinder.

Denote $V(P_m) = \{u_1, u_2, \ldots u_m\}$, $V(C_n) = \{v_1, v_2, \ldots v_m\}$ and consider the following partition of $V(P_m \square C_n)$: $V_1 = \{(u_i, v_j): 1 \leq i \leq 4 \text{ and } 1 \leq j \leq n\}$, $V_2 = \{(u_i, v_j): 5 \leq i \leq m - 4 \text{ and } 1 \leq j \leq n\}$ and $V_3 = \{(u_i, v_j): m - 3 \leq i \leq m \text{ and } 1 \leq j \leq n\}$ (see Figure 3). We call G_i the subgraph of $P_m \square C_n$ induced by the vertex set V_i, so G_1 and G_3 are both isomorphic to the cylinder $P_4 \square C_n$. Meanwhile, G_2 is isomorphic to $P_{m-8} \square C_n$. If f is a Roman dominating function, we denote by $f_i = f|_{V_i}$, the restriction of f to the set V_i. Note that $f(P_m \square C_n) = f_1(G_1) + f_2(G_2) + f_3(G_3)$ (the functions f_i are not necessarily Roman dominating functions, but the concept of weight can obviously be extended to them).

For $i = 1, 2, 3$ we denote $D_i = (S_1 \cap V_i) \cup N[(S_2 \cap V_i)]$, which is the set of vertices dominated by $(S_1 \cup S_2) \cap V_i$. Note that the closed neighborhood is $N[(S_2 \cap V_i)]$ computed in the
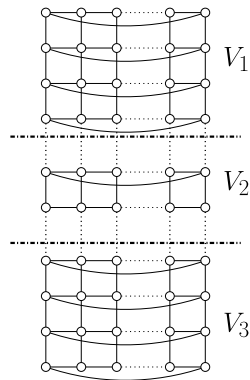
J. A. Martínez *et al.*: Powers of Large Matrices on GPU Platforms to Compute the Roman Domination Number of Cylindrical Graphs

IEEE *Access*

FIGURE 3. Partition of $V(P_m \square C_n)$, $m, n \geq 10$.

cylinder $P_m \square C_n$, so it is not necessarily contained in V_i, and

$$|D_i| \leq |(S_1 \cap V_i)| + 5|(S_2 \cap V_i)| \qquad (1)$$

Moreover, it is clear that $D_1 \cup D_2 \cup D_3 = V(P_m \square C_n)$ and D_1, D_2, D_3 are not necessarily disjoint.

Using this notation, we can rewrite the loss of a Roman dominating function as follows:
$L(f) = \frac{5}{2}f(P_m \square C_n) - mn = \frac{5}{2}(f_1(G_1) + f_2(G_2) + f_3(G_3)) - |D_1 \cup D_2 \cup D_3| \geq (\frac{5}{2}f_1(G_1) - |D_1|) + (\frac{5}{2}f_2(G_2) - |D_2|) + (\frac{5}{2}f_3(G_3) - |D_3|) \geq (\frac{5}{2}f_1(G_1) - |D_1|) + (\frac{5}{2}f_3(G_3) - |D_3|)$.

The first inequality comes from the sets D_i not being necessarily disjoint and the second one comes from the fact that each term of the sum is positive or zero, by using the inequality shown in Equation 1:

$$\frac{5}{2}f_i(G_i) - |D_i|$$
$$= \frac{5}{2}(|S_1 \cap V_i| + 2|S_2 \cap V_i|) - |D_i|$$
$$\geq \frac{5}{2}(|S_1 \cap V_i| + 2|S_2 \cap V_i|) - (|S_1 \cap V_i| + 5|S_2 \cap V_i|)$$
$$= \frac{3}{2}|S_1 \cap V_i|.$$

In fact, we expect $|S_1 \cap V_2|$ to be zero or close to zero in optimal cases, and this is why we discard the second term of the sum.

Vertex sets V_1 and V_3 play a similar role, so we will focus on V_1. The function f_1 satisfies that every vertex $v \in V_1$, not in row number four, with $f_1(v) = 0$ is adjacent to at least a vertex w satisfying $f_1(w) = 2$. Meanwhile, vertices in the fourth row do not need to be dominated by vertices in V_1. This leads us to the following definition:

Definition 14: Consider $P_4 \square C_n$ as the subgraph consisting of the fourth top (or bottom) rows of $P_m \square C_n$, $m \geq 10$. An almost Roman dominating function on $P_4 \square C_n$ is $g : V(P_4 \square C_n) \rightarrow \{0, 1, 2\}$ such that every vertex v, not in the fourth row, with $f(v) = 0$ is adjacent to at least a vertex w satisfying $f(w) = 2$. We denote by $R_i^g = \{v \in V(P_4 \square C_n) : g(v) = i\}$ (we will omit g if no confusion is possible).

The set of dominated vertices of an almost Roman dominating function is $D(g) = R_1 \cup N[R_2]$, where the closed

neighborhood is computed in $P_m \square C_n$. The loss of an almost dominating function is $L_a(g) = \frac{5}{2}g(P_4 \square C_n) - |D(g)|$ and the minimum loss of an almost Roman dominating function is $\mathcal{L}_a(n) = \min_g L_a(g) = \min_g(\frac{5}{2}g(P_4 \square C_n) - |D(g)|)$.

With this notation, the relationship between the loss of a Roman dominating function of $P_m \square C_n$ and the loss in the borders of the cylinder is as follows:
$L(f) \geq (\frac{5}{2}f_1(G_1) - |D_1|) + (\frac{5}{2}f_3(G_3) - |D_3|) \geq 2\mathcal{L}_a(n)$, therefore the minimum loss satisfies $\mathcal{L}(m, n) \geq 2\mathcal{L}_a(n) = \min_g(5g(P_4 \square C_n) - 2|D(g)|)$. We now modify Algorithm 1 and Algorithm 2, in order to compute the value of $\min_g(5g(P_4 \square C_n) - 2|D(g)|)$.

We keep the definition of correct words, words of length 4 in the alphabet $\{a, b, c, d\}$ not containing ac, ca, ab, ba, bb and we need to modify the conditions for a word $\mathbf{p} = (p_1, p_2, p_3, p_4)$ to follow another word $\mathbf{q} = (q_1, q_2, q_3, q_4)$. First and intermediate row conditions are the same, but the fourth row, which is the last one in this case, behaves in a different way:

1) conditions for the first row: the same as before
2) conditions for the intermediate rows $2 \leq i \leq 3$: the same as before
3) conditions for the last row (fourth row):
   - if $q_4 = a$, then $p_4 = a$ or $p_4 = c$
   - if $q_4 = b$, then $(p_4 = c$ and $p_3 = a)$ or $p_3 = d$
   - if $q_4 = c$, then $p_4 = a$ or $p_4 = b$ or $(p_4 = c$ and $p_3 = a)$ or $p_4 = d$
   - if $q_4 = d$, then $p_4 = a$ or $p_4 = b$ or $(p_4 = c$ and $p_3 = a)$ or $p_4 = d$

Just conditions in case $q_4 = d$ are new, because vertices in the fourth row do not need to be dominated. Therefore, in this case $p_4 = a$ is possible but not compulsory and other labels for p_4 are suitable.

Every almost Roman dominating function is uniquely identified with an ordered list of correct words such that each word can follow the previous one and the first word can follow the last one.

The directed graph G is defined as before: vertices are the correct words and there is an arc from $\mathbf{q}$ to $\mathbf{p}$ if $\mathbf{p}$ can follow $\mathbf{q}$. We need an appropriate labeling ℓ of the arcs such that, $\ell(Q) = 5g(P_4 \square C_n) - 2|D(g)|$, for $Q = (\mathbf{p_1 p_2}) \ldots (\mathbf{p_{n-1} p_n})(\mathbf{p_n p_1})$ a closed path in G and g the almost Roman dominating function represented by Q.

For an arc $(\mathbf{qp})$, we define $\ell(\mathbf{q}, \mathbf{p}) = 10\mathbf{p}(a) + 5\mathbf{p}(b) - 2nd(\mathbf{q}, \mathbf{p})$, where $\mathbf{p}(a)$ is the number of $a's$ of $\mathbf{p}$, $\mathbf{p}(b)$ is the number of $b's$ of $\mathbf{p}$ and $nd(\mathbf{q}, \mathbf{p})$ is the number of newly dominated vertices (see [13]). That is, the number of vertices dominated by $\mathbf{p}$ which are not dominated by $\mathbf{q}$, when $\mathbf{q}, \mathbf{p}$ are consecutive columns of an almost Roman dominating function. We compute such number with Algorithm 3.

Note that the computational complexity of Algorithm 3 is of the order of the square of the number of correct words, so it does not increase the complexity of the computation of the matrix $A(G)$.

Algorithm 3 Computation of $nd(\mathbf{q}, \mathbf{p})$

Require: $\mathbf{q}, \mathbf{p}$ correct words such that $\mathbf{p}$ can follow $\mathbf{q}$
Ensure: $nd(\mathbf{q}, \mathbf{p})$

1: $nd(\mathbf{q}, \mathbf{p}) = 0$
2: **for** $i = 1$ to 4 **do**
3:     **switch** (q_i, p_i)
4:     **case** $q_i = a$ and $p_i = a$:
5:       $nd(\mathbf{q}, \mathbf{p}) = nd(\mathbf{q}, \mathbf{p}) + 1$
6:     **case** $q_i = b$ and $p_i = c$:
7:       $nd(\mathbf{q}, \mathbf{p}) = nd(\mathbf{q}, \mathbf{p}) + 1$
8:     **case** $q_i = c$ and $p_i = a$:
9:       $nd(\mathbf{q}, \mathbf{p}) = nd(\mathbf{q}, \mathbf{p}) + 2$
10:     **case** $q_i = c, d$ and $p_i = b, c$:
11:       $nd(\mathbf{q}, \mathbf{p}) = nd(\mathbf{q}, \mathbf{p}) + 1$
12:     **case** $q_i = d$ and $p_i = a$:
13:       $nd(\mathbf{q}, \mathbf{p}) = nd(\mathbf{q}, \mathbf{p}) + 3$
14:     **end switch**
15: **end for**
16: **if** $p_4 = a$ **then**
17:     $nd(\mathbf{q}, \mathbf{p}) = nd(\mathbf{q}, \mathbf{p}) + 1$
18: **end if**
19: **return** $nd(\mathbf{q}, \mathbf{p})$

If g is the almost dominating function represented by the closed path $Q = (\mathbf{p_1}\,\mathbf{p_2})(\mathbf{p_2}\,\mathbf{p_3})\ldots(\mathbf{p_{n-1}}\mathbf{p_n})(\mathbf{p_n}\mathbf{p_1})$ then $\left(\sum_{i=1}^{n-1} nd(\mathbf{p_i}, \mathbf{p_{i+1}}) \right) + nd(\mathbf{p_n}\mathbf{p_1}) = |D(g)|$, the number of vertices dominated by g. Therefore, we obtain that $\ell(Q) = \left(\sum_{i=1}^{n-1} \ell(\mathbf{p_i}\mathbf{p_{i+1}}) \right) + \ell(\mathbf{p_n}\mathbf{p_1}) = \left(\sum_{i=1}^{n-1} 10\mathbf{p_{i+1}}(a) + 5\mathbf{p_{i+1}}(b) - 2nd(\mathbf{p_i}, \mathbf{p_{i+1}}) \right) + 10\mathbf{p_1}(a) + 5\mathbf{p_1}(b) - 2nd(\mathbf{p_n}, \mathbf{p_1}) = 10 \sum_{i=1}^{n} \mathbf{p_i}(a) + 5 \sum_{i=1}^{n} \mathbf{p_i}(b) - 2\left(\left(\sum_{i=1}^{n-1} nd(\mathbf{p_i}, \mathbf{p_{i+1}}) \right) + nd(\mathbf{p_n}\mathbf{p_1}) \right) = 5g(P_4 \square C_n) - 2|D(g)|$,

The construction of matrix $A(G)$ is as follows:

$$A(G)_{\mathbf{qp}} = \begin{cases} \ell(\mathbf{q}, \mathbf{p}) & \text{if } (\mathbf{qp}) \text{ is an arc of } G, \\ \infty & \text{otherwise.} \end{cases}$$

Algorithm 2 for $m = 4$, using the new rules for construction of the matrix $A(G)$, gives $n_0 = 30$, $a = 1$, $b = 1$. This means $2\mathcal{L}_a(n) = \min_g(5g(P_4 \square C_n) - 2|D(g)|)$ satisfies the finite difference equation $2\mathcal{L}_a(n + 1) - 2\mathcal{L}_a(n) = 1$, for $n \geq 30$. Moreover, by using Algorithm 1 (again with the new matrix $A(G)$), we have obtained $2\mathcal{L}_a(n) = n$, for $10 \leq n \leq 30$. This gives $2\mathcal{L}_a(n) = n$, for $n \geq 10$, and finally, $\mathcal{L}(m, n) \geq 2\mathcal{L}_a(n) = n$, for $m, n \geq 10$, as desired.

REFERENCES

[1] B. Carré. *Graphs and Networks*. Oxford, U.K.: Clarendon Press, 1979.
[2] E. W. Chambers, B. Kinnersley, N. Prince, and D. B. West, "Extremal problems for roman domination," *SIAM J. Discrete Math.*, vol. 23, no. 3, pp. 1575–1586, Jan. 2009.
[3] E. J. Cockayne, E. O. Hare, S. T. Hedetniemi, and T. V. Wimer, "Bounds for the domination number of grid graphs," *Congr. Numer.*, vol. 47, pp. 217–228, 1985.
[4] E. J. Cockayne, P. A. Dreyer, S. M. Hedetniemi, and S. T. Hedetniemi, "Roman domination in graphs," *Discrete Math.*, vol. 278, nos. 1–3, pp. 11–22, Mar. 2004.
[5] W. Dobosiewicz, "A more efficient algorithm for the min-plus multiplication," *Int. J. Comput. Math.*, vol. 32, nos. 1–2, pp. 49–60, Jan. 1990.
[6] R. Duan and S. Pettie, "Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths," in *Proc. 20th Annu. ACM-SIAM Symp. Discrete Algorithms*, Jan. 2009, pp. 384–391.
[7] O. Favaron, H. Karami, R. Khoeilar, and S. M. Sheikholeslami, "On the roman domination number of a graph," *Discrete Math.*, vol. 309, no. 10, pp. 3447–3451, May 2009.
[8] P. F. Felzenszwalb and J. J. McAuley, "Fast inference with min-sum matrix product," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 12, pp. 2549–2554, Dec. 2011.
[9] H. Gao, T. Xu, and Y. Yang, "Bagging approach for Italian domination in $C_n \square P_m$," *IEEE Access*, vol. 7, pp. 105224–105234, Aug. 2019.
[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Com.Pleteness*. New York, NY, USA: Freeman, 1979.
[11] D. Gonçalves, A. Pinlou, M. Rao, and S. Thomassé, "The domination number of grids," *SIAM J. Discrete Math.*, vol. 25, no. 3, pp. 1443–1453, Jan. 2011.
[12] S. Gravier and M. Mollard, "On domination numbers of Cartesian product of paths," *Discrete Appl. Math.*, vol. 80, nos. 2–3, pp. 247–250, Dec. 1997.
[13] D. R. Guichard, "A lower bound for the domination number of complete grid graphs," *J. Combin. Math. Combin. Comput.*, vol. 49, pp. 215–220, Jan. 2004.
[14] G. Hao, Z. Xie, and X. Chen, "A note on Roman domination of digraphs," *Discuss. Math. Graph Theory*, vol. 39, no. 1, pp. 13–21, 2019.
[15] T. W. Haynes, S. T. Hedetniemi, and P. J. Slater, *Fundamentals of Domination in Graphs*. New York, NY, USA: Marcel Dekker, 1998.
[16] S. M. Hedetniemi, S. T. Hedetniemi, and T. Wimer, "Linear time resource allocation algorithms for trees," Dept. Math. Sci., Clemson Univ., Clemson, SC, USA, Tech. Rep. URI-014, 1986.
[17] A. Humayun, M. Asif, and M. K. Hanif, "BTAS: A library for tropical algebra," *Int. J. Comput. Sci. Inf. Secur. (IJCSIS)*, vol. 14, no. 12, pp. 220–225, 2016.
[18] W. Imrich and S. Klavžar, *Product Graphs: Structure and Recognition*. New York, NY, USA: Wiley, 2000.
[19] M. S. Jacobson and L. F. Kinch, "On the domination number of products of graphs: I," *Ars Combin.*, vol. 18, pp. 33–44, Dec. 1983.
[20] N. J. Rad and H. Rahbani, "A Nordhaus–Gaddum bound for roman domination," *Discrete Math., Algorithms Appl.*, vol. 11, no. 5, Oct. 2019, Art. no. 1950055.
[21] N. Jafari Rad and H. Rahbani, "Some progress on the double Roman domination in graphs," *Discuss. Math. Graph Theory*, vol. 39, no. 1, pp. 41–53, 2019.
[22] J. Kepner and J. Gilbert, Eds., *Graph Algorithms in the Language of Linear Algebra*. Philadelphia, PA USA: SIAM, 2011.
[23] J. Kepner, D. Bader, A. Buluç, J. Gilbert, T. Mattson, and H. Meyerhenke, "Graphs, matrices, and the GraphBLAS: Seven good reasons," *Procedia Comput. Sci.*, vol. 51, pp. 2453–2462, Jan. 2015.
[24] S. Klavžar and N. Seifter, "Dominating Cartesian products of cycles," *Discrete Appl. Math.*, vol. 59, no. 2, pp. 129–136, May 1995.
[25] S. Klavžar and J. Žerovnik, "Algebraic approach to fasciagraphs and rotagraphs," *Discrete Appl. Math.*, vol. 68, nos. 1–2, pp. 93–100, Jun. 1996.
[26] M. Liedloff, T. Kloks, J. Liu, and S.-L. Peng, "Roman domination over some graph classes," in *Graph-theoretic concepts in Computer Science* (Lecture Notes in Computer Science), vol. 3787, Berlin, Germany: Springer, 2005, pp. 103–114.
[27] C.-H. Liu and G. J. Chang, "Upper bounds on roman domination numbers of graphs," *Discrete Math.*, vol. 312, no. 7, pp. 1386–1391, Apr. 2012.
[28] A. C. Martínez and I. G. Yero, "A characterization of trees with equal Roman 2-domination and Roman domination numbers," *Commun. Comb. Optim.*, vol. 4, no. 2, pp. 95–107, Mar. 2019.
[29] A. Moradi, D. A. Mojdeh, and O. Sharifi, "Roman 2-bondage number of a graph," *Discuss. Math. Graph Theory*, vol. 40, no. 1, pp. 255–268, 2020.
[30] *NVIDIA CUDA Toolkit*. Accessed: May 1, 2020. [Online]. Available https://developer.nvidia.com/cuda-math-library
[31] *NVIDIA CUDA Documentation*. Accessed: May 1, 2020. [Online]. Available: https://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf
[32] C. Padamutham and V. S. R. Palagiri, "Algorithmic aspects of roman domination in graphs," *J. Appl. Math. Comput.*, vol. 64, nos. 1–2, pp. 89–102, Oct. 2020.
[33] P. Pavličm and J. Žerovnik, "Roman domination number of the Cartesian products of paths and cycles," *Electron. J. Combinatorics*, vol. 19, no. 3, pp. 19–37, Aug. 2012.

J. A. Martínez *et al.*: Powers of Large Matrices on GPU Platforms to Compute the Roman Domination Number of Cylindrical Graphs

IEEE *Access*

[34] P. Pavlič and J. Žerovnik, "A note on the domination number of the Cartesian products of paths and cycles," *Kragujevac J. Math*, vol. 37, no. 2, pp. 275–285, 2013.

[35] J.-E. Pin, "Tropical semirings" in *Publications of the Newton Institute*, vol. 11, G. Jeremy, Ed. Cambridge, U.K.: Cambridge Univ. Press, 1998, pp. 50–69.

[36] M. Rao and A. Talon, "The 2-domination and Roman domination numbers of grid graphs," *Discrete Math. Theor. Comput. Sci.*, vol. 21, no. 1, pp. 9–14, May 2019.

[37] Z. Shao, J. Amjadi, S. M. Sheikholeslami, and M. Valinavaz, "On the total double roman domination," *IEEE Access*, vol. 7, pp. 52035–52041, Apr. 2019.

[38] A. Spalding, "Min-plus algebra and graph domination," Ph.D. dissertation, Dept. Appl. Math., Univ Colorado, Denver, CO, USA, 1998.

[39] I. Stewart, "Defend the roman empire!," *Scientific Amer.*, vol. 281, no. 6, pp. 136–138, Dec. 1999.

[40] V. G. Vizing, "Some unsolved problems in graph theory," (in Russian), *Uspekhi Mat. Nauk*, vol. 23, no. 6, pp. 117–134, 1968.

[41] C. Yang, A. Buluc, and J. D. Owens, "GraphBLAST: A high-performance linear algebra-based graph framework on the GPU," 2019, *arXiv:1908.01407*. [Online]. Available: http://arxiv.org/abs/1908.01407

E. M. GARZÓN was born in Granada, Spain, in 1962. She received the B.Sc. degree in physics from the University of Granada, Spain, in 1985, and the Ph.D. degree in computer engineering from the University of Almería, Spain, in 2000.

She is currently a Full Professor with the Department of Computer Sciences and the Head of the Supercomputing-Algorithms research group, University of Almería. Her research interests include high-performance computing, GPU computing, and scientific computation.

J. A. MARTÍNEZ was born in Almería, Spain, in 1969. He received the B.Sc. degree in computer sciences from the University of Granada, Spain, in 1992, and the Ph.D. degree in computer engineering from the University of Almería, Spain, in 2007.

He has been an Associate Professor with the Department of Computer Sciences, University of Almería, since 2008. His research interests include high performance computing, GPU computing, and cloud computing.

M. L. PUERTAS was born in Almería, Spain, in 1971. She received the B.Sc. degree in mathematics from the Universidad de Granada, Spain, in 1994, and the Ph.D. degree in mathematics from the University of Almería, Spain, in 1997.

She has been an Associate Professor with the Department of Mathematics, University of Almería, since 2003. Her research interests include the domination properties and the distance related parameters in graph theory.

· · ·