mathematics

MDPI

Article

Networks of Picture Processors with Filtering Based on Evaluation Sets as Solvers for Cryptographic Puzzles Based on Random Multivariate Quadratic Equations

Karina Paola Jiménez [1,2], Sandra Gómez-Canaval [2,*], Ricardo Villanueva-Polanco [3] and Silvia Martín Suazo [2]

[1] Facultad de Ingenierías, Universidad Simón Bolívar, Calle 58 No 55-132, Barranquilla 080001, Colombia; kjimenez6@unisimonbolivar.edu.co

[2] Departamento de Sistemas Informáticos, Universidad Politécnica de Madrid, Calle Alan Turing s/n, 28031 Madrid, Spain; silvia.marsuazo@alumnos.upm.es

[3] Computer Science Department, Universidad del Norte, Barranquilla 080001, Colombia; rpolanco@uninorte.edu.co

* Correspondence: sm.gomez@upm.es

Received: 10 November 2020; Accepted: 30 November 2020; Published: 4 December 2020

Abstract: Networks of picture processors is a massively distributed and parallel computational model inspired by the evolutionary cellular processes, which offers efficient solutions for NP-complete problems. This bio-inspired model computes two-dimensional strings (pictures) using simple rewriting rules (evolutionary operations). The functioning of this model mimics a community of cells (pictures) that are evolving according to these bio-operations via a selection process that filters valid surviving cells. In this paper, we propose an extension of this model that empowers it with a flexible method that selects the processed pictures based on a quantitative evaluation of its content. In order to show the versatility of this extension, we introduce a solver for a cryptographic proof-of-work based on the hardness of finding a solution to a set of random quadratic equations over the finite field $\mathbb{F}_2$. This problem is demonstrated to be NP-hard, even with quadratic polynomials over the field $\mathbb{F}_2$, when the number of equations and the number of variables are of roughly the same size. The proposed solution runs in $O(n^2)$ computational steps for any size (n, m) of the input pictures. In this context, this paper opens up a wide field of research that looks for theoretical and practical solutions of cryptographic problems via software/hardware implementations based on bio-inspired computational models.

Keywords: bio-inspired computational model; networks of picture processors; combinatorial optimization problems; massively parallel and distributed algorithms; cryptographic proof-of-work; multivariate quadratic equations; post-quantum blockchain

1. Introduction

Networks of bio-inspired processors (NBP) is a family of computational models facing NP-complete problems by mimicking, from a syntactical perspective, the manner by which cell communities evolve via gene mutations in DNA molecules. Each cell is represented by a word that describes their DNA sequences. Mutation and cellular division are defined by operations based on string rewritings. Cells form colonies, which belongs to specific species, and a colony can be interpreted as an evolutionary system that evolves according to these operations. This dynamic abstracts the natural process of evolution and it is defined

over a highly parallel and distributed computing architecture for symbolic processing. NBP can process one or two-dimensional arrays of symbols belonging to a predefined alphabet. The specific NBP model processing one-dimensional arrays of symbols (strings) are: network of [evolutionary/splicing/genetic] processors [1–3]. On the other hand, NBP models processing pictures (two-dimensional strings) are named networks of picture processors (NPP) [4]. The NBP model family defines a computational process based on the application of rewriting operations over strings, which can be explained as an abstraction of the behavior of mutating genes in chromosomes. For the NPP model, this process is applied to pictures representing a generalization of gene mutations in chromosomes over two-dimensional strings [4].

The NBP architecture consists of a set of processors, each of which occupies a node of a virtual graph, employed to set up a network of evolutionary processors. An evolutionary processor is a theoretical device that applies simple rewriting operations to data. These rules, called evolutionary rules, are specialized in either deleting a symbol, inserting a symbol or changing a symbol for other symbol, which simulates operations over a pair of nucleotides. It is important to stress that each processor can apply a single type of rule (insertion, deletion or substitution) to its local data and that all processors in the network simultaneously applies their predefined rules (this is what is known as an evolutionary step).

Data (syntactical representation of cells) may evolve by these rules (evolutionary process). Then, data may be interchanged through the network by means a predefined protocol. This protocol is a filtering strategy based on enforcing a set of conditions, defined in each processor, in order to control the dispatch and the reception of data (even simultaneously). The navigation of data following a filtering strategy is defined as the communication process. It is important to stress that the filtering strategies mimic the process of selection from a Darwinian perspective.

As computing devices, the functioning of NBP networks occurs in turn repeatedly by intermixing the application of the evolutionary and communication processes. The computation is finished only if a predefined stopping condition is fulfilled. All processors modify their local data at the same time, defining a processing step and then local data that satisfies some filtering condition are allowed to flow from a processor to another according to the network topology (communication step).

NBP models have been extensively studied as language generators, language-accepting devices and problem solvers. Additionally, other properties of them have been extensively investigated, e.g., their computational power, their complexity aspects, their efficiency as problem solvers, the existence of universal networks, etc. For a better comprehension of these studies, we refer the reader to [5].

Particularly, networks of picture processors (NPP) [4,6,7] is a specialized model for computing two-dimensional strings, called pictures. There, a picture is a two-dimensional array of elements belonging a finite set of symbols (alphabet). Traditionally, the NPP model is applied to problems concerning image processing, picture recognizability or pattern matching [8–10].

Recently, an NPP model named networks of polarized evolutionary picture processors (NPEPP) was proposed in [11]. NPEPP defines its communication protocol based on the polarization concept. This filtering strategy generalizes the protocol defined by networks working over strings [12,13] in such a way that it does work over rectangular pictures. In particular, the polarization is an abstraction of the electric charge (negative, positive or neutral) that may be applied to both pictures and nodes. Whereas the polarization of a node is predetermined and not subject to or able to be changed during the entire computing process, the polarization of a picture is dynamically computed by a valuation mapping that determines the picture's polarity based on its symbols and its predefined values. The valuation mapping returns the sign of this value but not its exact value. The movement of a picture from one node to other node through the communication protocol (filtering strategy) relies on both the picture polarization and the node polarization. Thus, this flow of a picture from one node to other node mimics the bio-electrical communication between cells.

A generalization of the polarization concept using evaluation sets to enhance the valuation mapping to calculate the exact value (not only the sign of this value) was proposed in [14] and was used to empower the network of splicing processors [15] and the networks of evolutionary processors [16] models. In these works, the valuation mapping is used to give the precise numerical value of a string. In addition, the polarization is redefined to provide a filtering strategy able to allow the communication between two nodes, simulating the movement of molecules or particles from an area to other along a concentration gradient in a solution. Finally, it is demonstrated that this kind of communication protocol is more flexible and is able to deal with hard optimization problems with integer restrictions in an efficient way.

In this context, we propose an extension of the NPEPP model proposed in [11], enhancing this model with the evaluation sets as filtering strategy for improving the communication protocol. In particular, we adapted the communication protocol working over strings proposed in [14,15] to work over pictures.

With this proposal, we enhanced the general NPP model with a more versatile protocol to include some quantitative features in a simple way, keeping rules for pictures processing. To show the computational power of our proposed model as well as its flexibility, we show how to leverage it to solve systems of quadratic polynomial equations over finite fields, e.g., the so called multivariate quadratic (MQ) problem.

Based on the hardness of the MQ problem (or variants of it), several cryptographic primitives have been constructed over the years. For instance, it is the security underpinning of the multivariate public key cryptosystems, a new family of post-quantum cryptographic schemes that withstand the action or effect of quantum computer attacks. In general terms, the hardness of solving the MQ problem over $\mathbb{F}_2$ is directly connected to the security of these cryptographic schemes [17]. Recently, the authors of [18] proposed a proof-of-work (PoW) based on random multivariate quadratic equations for a post-quantum blockchain [19], which requests a solver (miner) to solve a set of random multivariate quadratic equations (RMQE) over the finite field $\mathbb{F}_2$.

The main aim of this research paper was to provide a solver for the MQ problem over $\mathbb{F}_2$ using our extended and adapted model from NPP, named network of picture processors with evaluation sets (NPPES). As far as we know, this is the first occasion in which a bio-inspired computational model, such as the NBP model, was used to deal with solving systems of random quadratic equations (SRQE/RMQE) and therefore this paper proposes a unconventional solver for the PoW proposed in [18] running in $O(n^2)$ computational (processing and communication) steps, where n is the number of indeterminates. We remark that some results reported in the literature solving SRQE/RMQE problem require exponential time [20–23] for specific n, m input values. Therefore, our results suggest that the NPPES model faces such problems adequately and that the proposed network might be potentially deployed into a massively distributed and highly parallel platform without any modification, extension or adaptation.

This paper is structured as follows. In Section 2, we introduce the background concepts as well as the formal definition for the NPPES model. Section 3 discusses the capability of NPPES to solve the RMQE problem. We designed an NPEPP algorithm that works in $O(n^2)$ time. In addition, in Section 4.6, we discuss the obtained results, the reasons we believe our solution is more suitable for such problems and how the proposed algorithm may be transformed to be deploy into an ultra-scalable computational platform without any modification, extension or adaptation. Finally, conclusions and forthcoming works are presented in Section 5.

2. Networks Picture Processors with Filtering by Evaluation Sets

In this section, we introduce the network of picture processors with evaluation sets (NPPES) proposed model. First, we start off by summarizing the notions used throughout this paper, which were previously

introduced in [7]. This is the theoretical background underlying our proposed model. We later introduce the formal definitions for the NPPES model.

2.1. Preliminary Concepts

We define an alphabet as a set of non-empty symbols. The number of elements of a finite set S is denoted by $card(S)$ (this is also known as the cardinality of S). Also, we define a *picture* as a two-dimensional array of symbols taken from an alphabet V. The set V_*^* denotes the set of all pictures over V and ε denotes the empty picture. The pair $(\overline{\pi}, |\pi|)$ defines the size of the picture π, where $\overline{\pi}$ denotes the number of rows and $|\pi|$ denotes the number of columns of π. The size of the empty picture ε is given by (n, m) with $nm = 0$. The notation $\pi(i, j)$ denotes the symbol placed at the intersection of the i_{th} row with the j_{th} column of the picture π.

Let π be a picture of size (m, n) over V, $1 \leq i \leq k \leq m$ and $1 \leq j \leq l \leq n$, then ${}^{[i,j]}\pi_{[k,l]}$ is a sub-picture of π that consist of all symbols $\pi(k_i, k_j)$ with $i \leq k_i \leq k$ and $j \leq k_j \leq l$. Following this definition, π is ${}^{[1,1]}\pi_{[m,n]}$. The minimal alphabet containing all visible symbols appearing in a picture π is denoted by $alph(\pi)$. Also, we define a *valuation* of V^* in $\mathbb{Z}$ as a homomorphism from the monoid V^* to the monoid of additive integers $\mathbb{Z}$. For any alphabet V and a symbol $a \in V$, $\not{a}$ denotes the invisible copy of a and $\not{V} = \{\not{a} | a \in V\}$. A picture $\pi \in (V \cup \not{V})_m^n$ is said to be a well-defined picture if there exist $1 \leq i \leq k \leq m$ and $1 \leq j \leq l \leq n$ such that all elements of ${}^{[i,j]}\pi_{[k,l]}$ are from V and all the other elements of π are from $\not{V}$. In this case, the maximal visible subpicture of π is denoted as ${}^{[i,j]}\pi_{[k,l]}$. In concrete, a well-defined picture π may be interpreted as one having some rows and/or columns on its border hidden but not deleted. We stress that any picture over the alphabet V is a well-defined picture and that, from this point forward, we deal with only well-defined pictures.

We use the rewriting operations or evolutionary rules such as they are introduced and used in [6,7]. Concretely, we use substitution, mask and unmask rules. We summarize the definitions of these rules below. In order to know the rest of rules (insertion and deletion) as well as the formal definitions for all evolutionary rules, we refer the reader to [7].

An evolutionary rule is defined as $r = s_1 \rightarrow s_2$, with $s_1, s_2 \in V \cup \{\varepsilon\}$. Particularly, r is defined as a substitution rule if neither s_1 nor s_2 is ε.The set $Sub_V = \{s_1 \rightarrow s_2 \mid s_1, s_2 \in V\}$ is the set of substitution rules. On the other hand, a mask rule is a rule able to hide a column or a row. Similarly, an unmask rule is able to make visible a column or a row. The *action mode* is the element that defines the picture's position in which a rule r can be applied. Let π be a picture $\pi \in (V \cup \not{V})_m^n$ and let π_{max} be the maximal visible subpicture of π, then the actions of r on π are $(\leftarrow, \rightarrow, \uparrow, \downarrow, +)$. These actions are defined as following:

- Substitution rules:
  - $r^{\leftarrow}(\pi)$ is the set of all pictures obtained from π by replacing an occurrence of s_1 by s_2 in the leftmost column of π_{max}. We remark that r is applied to all instances of the symbol s_1 in the leftmost column of the π_{max}'s different copies.
  - Similarly as above, $r^{\rightarrow}(\pi)$ is the set of all pictures obtained by applying r to the rightmost column of π; $r^{\uparrow}(\pi)$ is the set of all pictures obtained by applying r to the first row of π; $r^{\downarrow}(\pi)$ is the set of all pictures obtained by applying r to the last row of π; $r^{+}(\pi)$ is the set of all pictures obtained by applying r to any column/row of the π_{max}.
- Mask and unmask rules:
  - mask$^{\leftarrow}(\pi)$ returns a picture obtained from π, by transforming all visible symbols from the leftmost column of the π_{max} into their invisible copies. Analogously, it is defined as the mappings mask$^{\rightarrow}$, mask$^{\uparrow}$ and mask$^{\downarrow}$.

– unmask$^{\leftarrow}(\pi)$ returns a picture from π, by making its leftmost column visible. If ${}^{[i,j]}\pi_{[k,l]}$ is π_{max}, then all invisible symbols $\pi(s, j-1), i \leq s \leq k$, become visible. If $j = 1$, then unmask$^{\leftarrow}(\pi) = \pi$. Similarly, it occurs for unmask$^{\rightarrow}$, unmask$^{\uparrow}$, and unmask$^{\downarrow}$.

For every rule r, symbol $\beta \in \{\uparrow, \downarrow, \rightarrow, \leftarrow, +\}$ and $L \subseteq (V \cup \not{V})^{**}$, the β-action of r on L is defined by $r^{\beta}(L) = \bigcup_{\pi \in L} r^{\beta}(\pi)$. For a finite set of rules M, we define the β-action of M on the picture π and the language L by:

$$M^{\beta}(\pi) = \bigcup_{r \in M} r^{\beta}(\pi) \text{ and } M^{\beta}(L) = \bigcup_{\pi \in L} M^{\beta}(\pi)$$

Analogously, for every $\beta \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, it defined mask$^{\beta}(L) = \{\text{mask}^{\beta}(\pi) \mid \pi \in L\}$ and unmask$^{\alpha}(L) = \{\text{unmask}^{\alpha}(\pi) \mid \pi \in L\}$.

2.2. NPPES: Formal Model Definition

We start defining a new function that is required to extend the NPP model with the filtering strategy using evaluation sets. This function is necessary to extend or more precisely generalize the NPP variant proposed in [11].

Definition 1. *Let V be a finite alphabet and π be a picture over V with size (m, n). We define the projection function $\Phi : V_{**} \times \mathcal{P}(V) \rightarrow V^{**}$ as follows*

$$\Phi(\pi, S) = \pi'_{1,1}\pi'_{1,2}\ldots\pi'_{m,n}, \textit{where } \pi'_{i,j} = \begin{cases} \pi(i,j) & \textit{if } \pi(i,j) \in S \\ \varepsilon & \textit{otherwise} \end{cases}$$

This function is required because we need to calculate, for a picture, the "concentration" of the symbols included in S. Therefore, this function ignores the other symbols that do not belong to S. With this definition and other components required to propose the evaluation sets filtering strategy, we define our extension of a picture processor as follows.

Definition 2. *Let V be an alphabet. A generalized picture processor over V is a triple (M, S, α) such that:*

- *M is a set of evolutionary rules containing either substitution or deletion rules over V ($M \subseteq Sub_V$ or $M \subseteq Del_V$).*
- *$S \subseteq \mathcal{P}(V)$.*
- *α is a set of mutually disjoint intervals of $\mathbb{Z}$, which gives the generalized polarity of the node. In particular, the polarity of a node, that is $-, 0, +$, in our generalization, it can be viewed as the intervals $(-\infty, 0)$, $\{0\}$, $(0, \infty)$, respectively.*

Definition 3. *A hiding generalized picture processor over $V \cup \not{V}$ is a triple (M, S, α) such that M is a set of either mask or unmask rules over V. The rest of the other parameters are identical to those in the Definition 2.*

The set of (hiding) generalized picture processors over V is denoted by GPP_V. Evidently, the generalized picture processor introduced here is a mathematical concept comparable to that of an evolutionary algorithm, both of which are motivated by the Darwinism. Compared to evolutionary algorithms, the evolutionary rules may be understood as a two-dimensional generalization of gene mutations and the evaluation sets (supported by means of S and α elements, Definitions 1 and 2) might be seen as a selection process similar to, as we have mentioned in the Section 1, the bio-electrical properties of the concentration gradients in a solution.

Definition 4. *A network of picture processors with evaluation sets (NPPES) is a* 10*-tuple* $\Gamma = (V, U, G, R, \beta, \rho, \varphi, \underline{In}, \underline{Halt}, \underline{Accept})$*, where:*

- V *is the input alphabet and* U *is the network alphabet.*
- $G = (X_G, E_G)$ *is an undirected graph (which is called the underlying graph of the network) such that* X_G *is the set of vertices and* E_G *is the set of edges.*
- $R : X_G \to GPP_U$ *defines a mapping for associating each node* $x \in X_G$ *with the generalized picture processor* $R(x)$ *over* U*, such that* $R(x) = (M_x, S_x, \alpha_x)$*.*
- $\beta : X_G \to \{\leftarrow, \rightarrow, \uparrow, \downarrow, +\}$ *such that* $\beta(x)$ *defines the action mode of the rules of node* x *on the pictures existing in that node.*
- ρ *is a mapping which allows to associate each subset of* U *with an interval (possibly empty) of* $\mathbb{Z}$ *satisfying that* $\rho(X) \cap \rho(Y) = \emptyset$ *for any* $X \neq Y$ *subsets of* U*.*
- $\varphi : U^* \to \mathbb{Z}$ *is a valuation of* U^* *in* $\mathbb{Z}$*.*
- $\underline{In}, \underline{Halt}, \underline{Accept}$ *are nodes of* X_G *such that they are the input, halting and accepting nodes of* Γ*, respectively.*

Definition 5. *A configuration of an NPPES* Γ *such as it is defined in the Definition 4, is a mapping* $C : X_G \to \mathcal{P}(U_*^*)$ *linking every node of the network's graph with a set of pictures. Informally, a configuration represents the content (the set of pictures) of each node at a given moment. Let* $\pi \in V_*^*$ *a picture, the initial configuration of* Γ *on* π *is formally denoted by* $C_0^{(\pi)}(\underline{In}) = \{\pi\}$ *and* $C_0^{(\pi)}(x) = \emptyset$*,* $\forall x \in X_G \backslash \{\underline{In}\}$*.*

An NPPES network evolves through two steps: *processing* and *communication steps* over the configurations. On a processing step for instance, each element belonging to a configuration C evolves according to the set of rules defined for the node x (M_x). These steps are defined as follows.

Definition 6. *One processing step that produces a configuration* C' *is obtained from the previous configuration* C*, which is denoted by* $C \Longrightarrow C'$*, iff*

$$C'(x) = M_x^{\beta(x)}(C(x)) \; \forall x \in X_G \tag{1}$$

Definition 7. *One communication step, denoted by* $C \vdash C'$ *produces a configuration* C' *denoted by*

$$C'(x) = (C(x) \backslash H_x) \cup \bigcup_{\{x,y \in E_G\}} T_y, \tag{2}$$

where

$$\begin{aligned} H_x &= \{\pi \in C(x) \mid \forall X \in S_x((\varphi(\Phi(\pi, X)) \notin \bigcup_{Y \in \alpha_x} Y) \vee \\ &\quad ((\exists Y \in \alpha_x \text{ such that } \varphi(\Phi(\pi, X)) \in Y) \to (Y \neq \rho(X))))\}, \\ T_y &= \{w \in C(y) \mid \exists X \in S_y \text{ such that } \varphi(\Phi(\pi, X)) \in \rho(X) \text{ and} \\ &\quad \rho(X) \in \alpha_x\}. \end{aligned}$$

Assuming that a picture within node x has its valuation with respect to some $X \in S_x$ in $\rho(X)$, if $\rho(X)$ belongs to α_x, then a copy of the picture is kept within the node x, else it is forced out. If $\rho(X) \in \alpha_y$, $y \in X_G$, then a copy of the picture gets into y as long as x is adjacent to y in G. On condition that a picture is forced out of a node and is not able to get in any node, then it is lost. We remark that a copy of some picture π can stay in x and the same time other copies of π can move from x to other nodes that are adjacent to x.

Definition 8. *A computation for an NPPES* Γ *on the input picture* $\pi \in V_*^*$ *is defined as a sequence of configurations* $C_0^{(\pi)}, C_1^{(\pi)}, C_2^{(\pi)}, \ldots$ *such that* $C_0^{(\pi)}$ *is the initial configuration and* $C_{2i}^{(\pi)} \Longrightarrow C_{2i+1}^{(\pi)}$ *and* $C_{2i+1}^{(\pi)} \vdash C_{2i+2}^{(\pi)}$, *for all* $i \geq 0$ *are the processing and communication steps performed in an alternate way. Note that the configuration* $C_{i-1}^{(\pi)}$ *determines the configuration* $C_i^{(\pi)}$.

Finally, the stop condition for an NPPES Γ occurs when the halting node is non-empty at some point of the computation. Then, the picture language decided by Γ is given by

$$L(\Gamma) = \{\pi \in V_*^* \mid \text{the computation of } \Gamma \text{ on } \pi \text{ stops with a non-empty accepting node}\}.$$

3. Random Multivariate Quadratic Equations

In this section, we formally describe a well-known NP-hard problem regarding the hardness of solving a set of m random quadratic equations in n indeterminates over a finite field F. This is called the MQ problem [24].

Let $\mathbb{F}_q$ be the finite field with q elements. The multivariate quadratic (MQ) problem is defined as follows. Given m quadratic polynomials $P^{(1)}, P^{(2)}, \ldots, P^{(m)}$ in the n indeterminates $x_1, \ldots, x_n$ as shown below

$$\begin{array}{rcl} P^{(1)}(x_1, x_2, \ldots, x_n) & = & \sum_{i=1}^{n} \sum_{j=i}^{n} p_{i,j}^{(1)} x_i x_j + \sum_{i=1}^{n} p_i^{(1)} x_i + p_0^1 \\ P^{(2)}(x_1, x_2, \ldots, x_n) & = & \sum_{i=1}^{n} \sum_{j=i}^{n} p_{i,j}^{(2)} x_i x_j + \sum_{i=1}^{n} p_i^{(2)} x_i + p_0^{(2)} \\ \vdots & \vdots & \vdots \\ P^{(m)}(x_1, x_2, \ldots, x_n) & = & \sum_{i=1}^{n} \sum_{j=i}^{n} p_{i,j}^{(m)} x_i x_j + \sum_{i=1}^{n} p_i^{(m)} x_i + p_0^{(m)} \end{array} \tag{3}$$

where $p_{i,j}^{(k)}, p_i^{(k)}, p_0^{(k)} \in \mathbb{F}_q$, with $1 \leq k \leq m$ and $1 \leq i \leq j \leq n$, are chosen uniformly and independently in a random way. The MQ problem asks to find $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n) \in \mathbb{F}_q^n$ such that

$$P^{(1)}(\hat{x}_1, \ldots, \hat{x}_n) = P^{(2)}(\hat{x}_1, \ldots, \hat{x}_n) = \ldots = P^{(m)}(\hat{x}_1, \ldots, \hat{x}_n) = 0 \tag{4}$$

The MQ problem is proven to be NP hard, even with quadratic polynomials over the field $\mathbb{F}_2$ (MQ_2) [24]. Moreover, the hardness of the MQ problem is the security underpinning for multivariate public key cryptosystems [25], a new family of post-quantum cryptosystems that can withstand the action or effect of quantum computer attacks. For instance, several MQ-based signature schemes were submitted to the National Institute of Standardization of Technology post-quantum cryptographic standardization process [23,25–28]. In particular, Rainbow has been included in the third round of this process [29].

The MQ problem has also been used to construct a proof of work for post-quantum blockchains [19], and is part of the cryptocurrency called ABC Mint [18]. In this setting, the solver's task is to find a solution to a given instance of the problem within a reasonable period of time in order for the solver to include a block of transactions in the blockchain. In particular, this instance of the MQ problem is defined over $\mathbb{F}_q \cong \mathbb{F}_2$ with n being a suitable but variable chosen integer and m being set to $n - 8$. The reason of this choice is to guarantee that a solution can be found with overwhelming probability within a reasonable period of time [18]. To perform the task, a solver first generates a bit string of $(n-8) \cdot (n \cdot (n-1)/2 + n + 1)$ bits uniformly and independently at random, by applying a cryptographic hash function to a random input formed from the block to be included in the blockchain. This long string is then partitioned into $(n-8)$ strings of $n \cdot (n-1)/2 + n + 1$ bits, each of which represent the coefficients of a polynomial

in the equations system. The solver then performs a computational task consisting in finding a vector $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ such that the Equation (4) holds [18].

As an additional remark, a coefficient-based representation of a polynomial $P^k(\mathbf{x})$ requires storing $n \cdot (n-1)/2 + n + 1$ elements of $\mathbb{F}_q$. In total, $m \cdot (n \cdot (n-1)/2 + n + 1)$ elements of $\mathbb{F}_q$ are required to represent all polynomials. That is, a complete coefficient-based representation of all polynomials in the equation system requires $\log(q) \cdot m \cdot (n \cdot (n-1)/2 + n + 1)$ bits in a classical computer.

Another contribution of this paper is that we construct a solver based on NPPES for solving an instance of the MQ problem, assuming the polynomials $P^{(1)}(\mathbf{x}), P^{(2)}(\mathbf{x}), \ldots, P^{(n-8)}(\mathbf{x})$ are given in a special form. We introduce such solver based on NPPES, which is able to find a vector $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ that satisfies the Equation (4), in the following section.

4. A Solver for the Random Multivariate Quadratic Equations (RMQE) Problem Using NPPES

In this section we discuss how to construct a solver for the MQ problem described in the previous section.

Let $n, m \in \mathbb{N}$ be defined as the number of variables and equations respectively. Following the definitions introduced in the Section 2, we define the next components of an NPPES, named Γ, for our proposed solver.

1. Let $V = P \cup X \cup C \cup B \cup \{\$\}$ be the input alphabet such that:
   - $P = P_Q \cup P_L \cup P_0$, where
     - $P_Q = \underline{P_Q} \cup \overline{P_Q}$, where $\underline{P_Q} = \{\underline{p_{i,j}^{(k)}}\ s.t.\ 1 \leq k \leq m\ ,\ 1 \leq i \leq j \leq n\}$ and $\overline{P_Q} = \{\overline{p_{i,j}^{(k)}}\ s.t.\ 1 \leq k \leq m\ ,\ 1 \leq i \leq j \leq n\}$. The two symbols $\overline{p_{i,j}^{(k)}}$ and $\underline{p_{i,j}^{(k)}}$ represent the two values that $p_{i,j}^{(k)}$ can assume. In other words, the symbol $\overline{p_{i,j}^{(k)}}$ represents that $p_{i,j}^{(k)}$ assumes the value 1, while the symbol $\underline{p_{i,j}^{(k)}}$ represents that $\underline{p_{i,j}^{(k)}}$ assumes the value 0.
     - P_L is the set of all symbols $\overline{p_i^{(k)}}$, $\underline{p_i^{(k)}}$ for $1 \leq k \leq m$ and $1 \leq i \leq n$. The two symbols $\overline{p_i^{(k)}}$ and $\underline{p_i^{(k)}}$ represent the two values that $p_i^{(k)}$ can assume. In other words, the symbol $\overline{p_i^{(k)}}$ represents that $p_i^{(k)}$ assumes the value 1, while the symbol $\underline{p_i^{(k)}}$ represents that $p_i^{(k)}$ assumes the value 0.
     - $P_0 = \{\overline{p_0^{(i)}}, \underline{p_0^{(i)}}\ s.t.\ 1 \leq i \leq m\}$. The two symbols $\overline{p_0^{(i)}}$ and $\underline{p_0^{(i)}}$ represent the two values that $p_0^{(i)}$ can assume. In particular, the symbol $\overline{p_0^{(i)}}$ represents that $p_0^{(i)}$ assumes the value 1, while the symbol $\underline{p_0^{(i)}}$ represents that $p_0^{(i)}$ assumes the value 0.
   - $X = \{x_1, x_2, \ldots x_n\}$, represents the n indeterminates $x_1, x_2, \ldots x_n$.
   - $C = \{c_0, c_1, c_2, \ldots c_n\}$, is the set of control symbols used through the computation of the NPPES algorithm.
   - $B = \{\boxplus^1, \boxplus^2, \boxplus^k, \ldots \boxplus^m\}$, with $\boxplus^k$ representing a signaling symbol for the polynomial $P^{(k)}$ from Equation (3).

   Let Π be the set of pictures $\pi^{(k)}$, for $1 \leq k \leq m$, where the size of each picture $\pi^{(k)}$ is $(n+3, 3 \cdot n)$. Each picture may be seen as a matrix of which rows are indexed starting from one. The n first rows of $\pi^{(k)}$ encode the terms $p_{i,j}^{(k)}\ x_i x_j$ for $1 \leq i \leq j \leq n$, having the empty entries filled with the symbol \$. In particular, the l_{th}-row, with $1 \leq l \leq n$, has the first $3 \cdot (l-1)$ entries, from left to right, filled with the symbol \$, the entry $3 \cdot l$ filled with a symbol from $\{\underline{p_{i,j}^{(k)}}, \overline{p_{i,j}^{(k)}}\}$, then the entry $3 \cdot l + 1$ filled with x_l, then the entry $3 \cdot l + 2$ filled with x_l and so on. The $n+1$ row of $\pi^{(k)}$ encodes the terms $p_i^{(k)} x_i$ for $1 \leq i \leq n$,

having the empty entries filled with the symbol \$. The $n+2$ row encodes the independent term $p_0^{(k)}$ and the $\boxplus^k$ term, having the empty entries filled with the symbol \$. Finally, the last row encodes the control symbols from the set C, which are required for internal computation of Γ. For example, given the polynomial $P^{(k)}(x_1,x_2,x_3) = \sum_{i=1}^{3}\sum_{j=i}^{3} p_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{3} p_i^{(k)} x_i + p_0^{(k)}$, it is then encoded as the picture shown by Table 1.

We remark that our network receives m input pictures encoding the m polynomials $P^{(1)}(\mathbf{x}),\ldots,P^{(m)}(\mathbf{x})$ from the instance of the problem to be solved, and then our network Γ will obtain values (if they exists) for the indeterminates $x_1,x_2,x_3,\ldots,x_n$ such that Equation (4) holds.

Table 1. Input picture for the polynomial $\sum_{i=1}^{3}\sum_{j=i}^{3} p_{i,j}^{(k)} x_i x_j + \sum_{i=1}^{3} p_i^{(k)} x_i + p_0^{(k)}$.

$\overline{p_{1,1}^{(k)}} \mid \underline{p_{1,1}^{(k)}}$	x_1	x_1	$\overline{p_{1,2}^{(k)}} \mid \underline{p_{1,2}^{(k)}}$	x_1	x_2	$\overline{p_{1,3}^{(k)}} \mid \underline{p_{1,3}^{(k)}}$	x_1	x_3
\$	\$	\$	$\overline{p_{2,2}^{(k)}} \mid \underline{p_{2,2}^{(k)}}$	x_2	x_2	$\overline{p_{2,3}^{(k)}} \mid \underline{p_{2,3}^{(k)}}$	x_2	x_3
\$	\$	\$	\$	\$	\$	$\overline{p_{3,3}^{(k)}} \mid \underline{p_{3,3}^{(k)}}$	x_3	x_3
$\overline{p_1^{(k)}} \mid \underline{p_1^{(k)}}$	x_1	\$	$\overline{p_2^{(k)}} \mid \underline{p_2^{(k)}}$	x_2	\$	$\overline{p_3^{(k)}} \mid \underline{p_3^{(k)}}$	x_3	\$
$\underline{p_0^{(k)}} \mid \overline{p_0^{(k)}}$	\$	\$	\$	\$	\$	\$	\$	$\boxplus^k$
c_0	\$	c_1	\$	\$	c_2	\$	\$	c_3

2. Let $U = V \cup C' \cup C'' \cup \ddot{C} \cup \underline{X} \cup \underline{X'} \cup \overline{X} \cup \overline{X'} \cup \dot{X} \cup \ddot{X} \cup R \cup B' \cup \{\$', \blacksquare, \blacklozenge, \ddot{k}\} \cup \{1,0\} \cup \{c_0^0, c_0^1\}$ be the network alphabet, where:

- $C' = \{c_0', c_1', c_2', \ldots c_n'\}$.
- $C'' = \{c_0'', c_1'', c_2'', \ldots c_n''\}$.
- $\ddot{C} = \{\ddot{c_1}, \ddot{c_2}, \ldots \ddot{c_n}\}$.
- $\underline{X} = \{\underline{x_1}, \underline{x_2}, \ldots \underline{x_n}\}$.
- $\underline{X'} = \{\underline{x_1'}, \underline{x_2'}, \ldots \underline{x_n'}\}$.
- $\ddot{X} = \{\ddot{x_1}, \ddot{x_2}, \ldots, \ddot{x_n}\}$.
- $\overline{X} = \{\overline{x_1}, \overline{x_2}, \ldots \overline{x_n}\}$.
- $\overline{X'} = \{\overline{x_1'}, \overline{x_2'}, \ldots \overline{x_n'}\}$
- $\dot{X} = \{\dot{x_1}, \dot{x_2}, \ldots, \dot{x_n}\}$
- $R = \{R_0, R_1, \ldots, R_{n-1}\}$
- $B' = \{\boxtimes^1, \boxtimes^2, \ldots \boxtimes^m\}$

In particular, the sets $C', C'', \ddot{C}$ are needed to control the transition between the different Γ's phases, namely: generation, linear evaluation, quadratic evaluation and validation. These phases are represented by different subnetworks as explained below. The set X is the set of symbols representing the indeterminates $x_1, x_2, \ldots x_n$. Similarly, the sets $\underline{X}, \underline{X'}, \overline{X}, \overline{X'}, \dot{X}, \ddot{X}$ and the set $\{1,0\}$ encapsulate the symbols with the values assigned to the indeterminates $x_1, x_2, \ldots x_n$ when they are evaluated in each phase of Γ and require some evaluation. The symbols $\{\$', \blacksquare, \blacklozenge, \ddot{k}\}$ are trash symbols required for some transformations at different phases of the computation. Finally, $\{c_0^0, c_0^1\}$ are punctual transformations of the control symbol c_0 in order to indicate the final evaluation of the Equation (4) for a picture.

3. The underlying graph G of our proposed Γ is depicted in the Figure 1. The nodes **In**, **H** and **A** are the input, halting and accept nodes, respectively, while the boxes **Gen**, $\mathbf{Eval_1}$, $\mathbf{Eval_2}$, **Val** are subnetworks representing the different stages or phases for Γ computation. In particular, the box **Gen** represents the subnetwork in charge of the combinatorial task for generating all pictures π containing the different values for the indeterminates $x_1, x_2, \ldots x_n$ for each input picture. Also, both boxes $\mathbf{Eval_1}$ and $\mathbf{Eval_2}$ represent the subnetworks for the phases of evaluation. The former represents the subnetwork that computes the evaluation of the $n+1$ row of a given picture, namely, the evaluation of the terms $p_i^{(k)} x_i$,

while the latter represents the subnetwork that computes the evaluation of the first n-rows of a given picture, namely, the evaluation of the terms $p_{i,j}^{(k)} x_i x_j$. Finally, the box **Val** represents the subnetwork in charge of the phase for selecting the pictures that satisfy the Equation (4). Detailed description for each subnetwork as well as the In, H and A nodes are introduced in the Sections 4.1–4.5. In addition, the internals for each subnetwork are given in the Figures 2–5.

4. With regards to the mapping R for Γ, it is specified for each node in the Tables 2–5.
5. With regards to the action mapping β for Γ, it is specified for each node in the Tables 2–5.

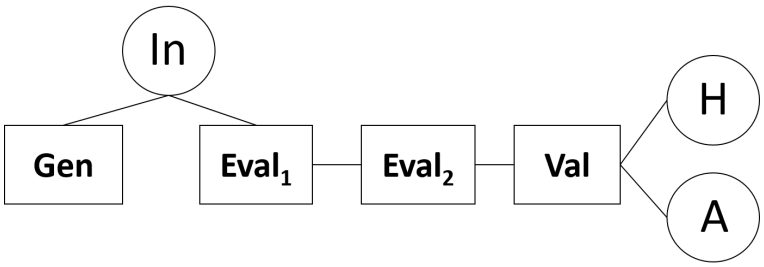

Figure 1. Underlying graph G.

6. The mapping function ρ is defined as follows:

- $\rho(\underline{X} \cup \underline{X'} \cup \overline{X} \cup \overline{X'} \cup \{1\} \cup \{0\}) = [0, n(2n+1)]$
- $\rho(\overline{C} \cup \overline{C''}) = [2n(n+1), 2n^2(n+1)]$
- $\rho(\{\blacklozenge\} \cup \{\$\}) = [2n^2(n+2), \infty)$
- $\rho(\dot{X} \cup \dot{X} \cup \dot{X} \cup \{\$'\}) = [-(2^{(n+1)}+1), -1]$
- $\rho(\dot{C'} \cup \{\blacksquare\} \cup \{\dot{k}\} \cup \dot{B}) = [-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$
- $\rho(\ddot{C} \cup B' \cup \{c_0^0\} \cup \{c_0^1\}) = \{-4mn(n+1)(2^{(n+1)})\}$

We assume that any symbol in U not appearing in the previous mapping has an empty interval for evaluation since it is not relevant for the proposed algorithm.

7. The function φ is defined as follows:

- $\varphi(\overline{p_i^{(k)}}) = \varphi(\overline{p_{i,j}^{(k)}}) = \varphi(\overline{p_0^{(k)}}) = \varphi(\overline{x_i}) = \varphi(1) = \varphi(\overline{x_i'}) = 1$, for all $1 \leq k \leq m, 1 \leq i, j \leq n$
- $\varphi(\dot{x}_i) = \varphi(\dot{x}_i) = \varphi(\$') = -1$, for all $1 \leq i \leq n$
- $\varphi(\underline{p_i^{(k)}}) = \varphi(\underline{p_{i,j}^{(k)}}) = \varphi(\underline{p_0^{(k)}}) = \varphi(\underline{x_i}) = \varphi(\underline{x_i'}) = \varphi(0) = 0$, for all $1 \leq k \leq m, 0 \leq i, j \leq n$
- $\varphi(\ddot{x}_i) = -(2^{(i-1)})$, for all $1 \leq i \leq n$
- $\varphi(c_i) = \varphi(c_i'') = 2n(n+1)$, for all $0 \leq i \leq n$
- $\varphi(\boxplus^k) = \varphi(c_i') = \varphi(\dot{k}) = \varphi(\blacksquare) = -n(2^{(n+1)})$, for all $1 \leq k \leq m, 0 \leq i \leq n$
- $\varphi(\blacklozenge) = \varphi(\$) = 2n^2(n+2)$
- $\varphi(\ddot{c}_i) = \varphi(\boxtimes^k) = \varphi(c_0^0) = \varphi(c_0^1) = -2kn(n+1)(2^{(n+1)})$, for all $1 \leq i \leq n, 1 \leq k \leq m$

We remark that the values for elements in P are given as input values.

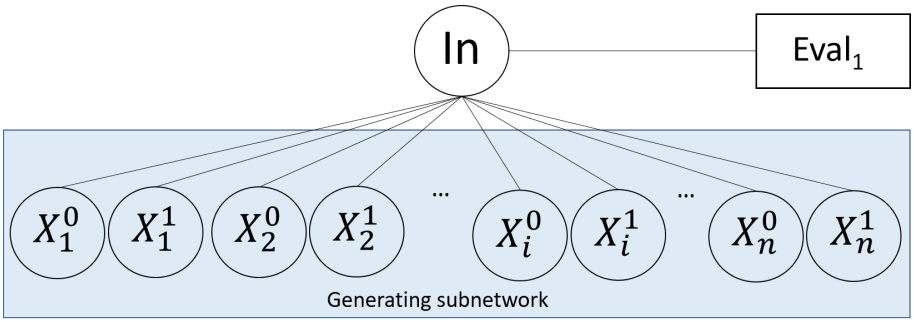

Figure 2. Subnetwork **Gen**.

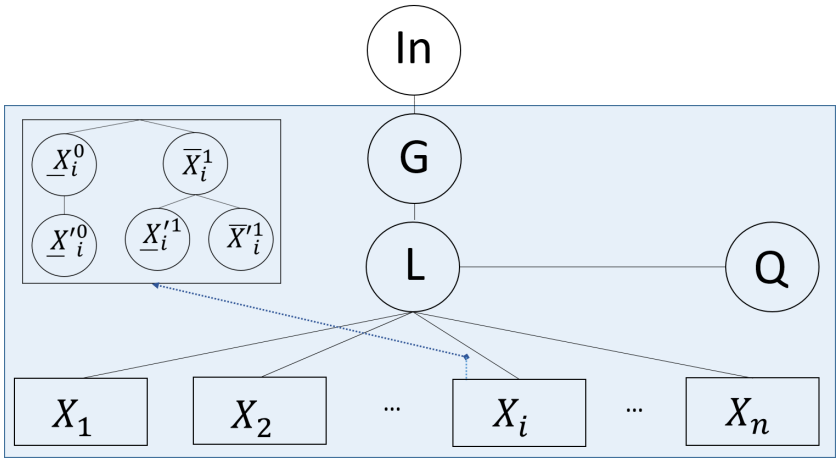

Figure 3. Subnetwork **Eval$_1$**.

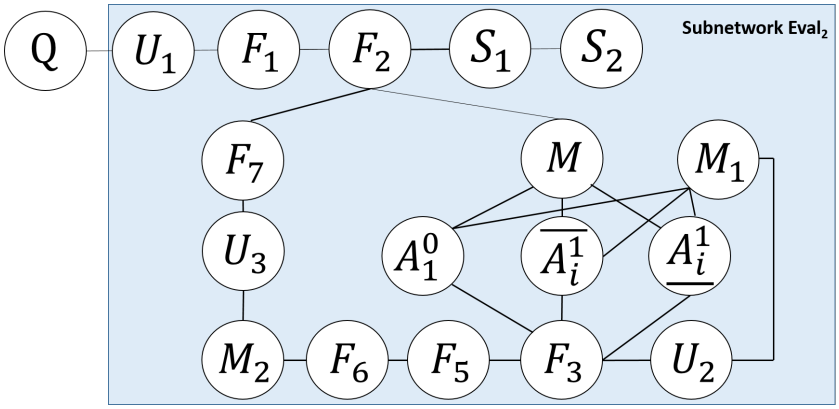

Figure 4. Subnetwork **Eval$_2$**.

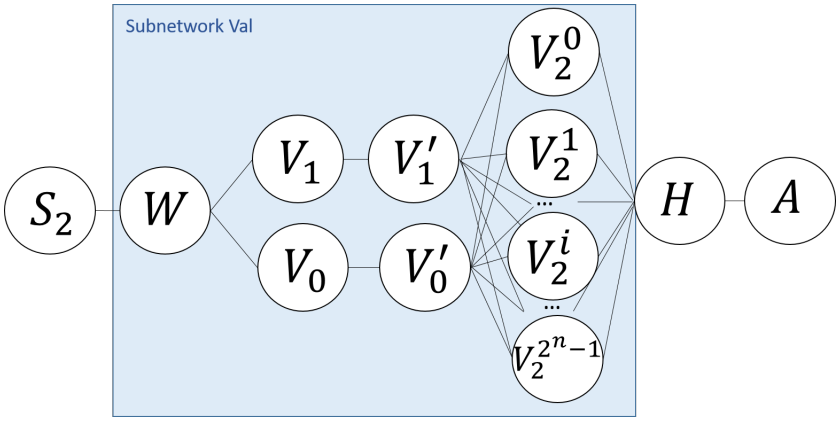

Figure 5. Subnetwork **Val**.

Table 2. Internals of the In node and nodes of the subnetwork **Gen**.

Node	M	α	β	S
In	$\{c_0 \to c_0'\}$	$[2n(n+1), 2n^2(n+1)]$ $\{-n(2^{(n+1)})\}$	$\{\downarrow\}$	$\{C \cup C''\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
$X_i^0, 1 \le i \le n$	$\{x_i \to \underline{x_i}, c_i \to c_i', c_0' \to c_0\}$	$[-(2^{(n+1)}+1), -1]$, $\{-n(2^{(n+1)})\}$	$\{+\}$	$\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
$X_i^1, 1 \le i \le n$	$\{x_i \to \overline{x_i}, c_i \to c_i', c_0' \to c_0\}$	$[-(2^{(n+1)}+1), -1]$, $\{-n(2^{(n+1)})\}$	$\{+\}$	$\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$

Table 3. Internals of the nodes of the subnetwork $\mathbf{Eval_1}$.

Node	M	α	β	S
G	{mask}	$[-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$	$\{\downarrow\}$	$\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$, $\{\{\blacklozenge\} \cup \{\$\}\}$
L	$\{\$ \to \blacksquare\}$	$[2n^2(n+2), \infty)$, $[-n^2(2^{(n+1)}), -n(2^{(n+1)})]$	$\{+\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$,
Q	{mask}	$[0, n(2n+1)]$, $[-n^2(2^{(n+1)}), -n(2^{(n+1)})]$	$\{\downarrow\}$	$\{\underline{X} \cup \underline{X'} \cup \overline{X} \cup \overline{X'} \cup \{1\} \cup \{0\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$,
$\underline{X_i^0}, 1 \le i \le n$	$\{\underline{x_i} \to \dot{x}_i\}$	$[-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$, $\{-1\}$	$\{\downarrow\}$	$\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$, $\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$
$\underline{X_i'^0}, 1 \le i \le n$	$\{\overline{p_i^{(k)}} \to 0, \underline{p_i^{(k)}} \to 0, \dot{x}_i \to \underline{x_i'}\}$	$[-(2^{(n+1)}+1), -1]$	$\{\downarrow\}$	$\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$, $\{\underline{X} \cup \underline{X} \cup \overline{X} \cup \overline{X'} \cup \{1\} \cup \{0\}\}$
$\overline{X_i^1}, 1 \le i \le n$	$\{\overline{p_i^{(k)}} \to \blacklozenge, \underline{p_i^{(k)}} \to \$', \overline{x_i} \to \dot{x}\}$	$[-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$, $\{2n^2(n+2)\}$	$\{\downarrow\}$	$\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$, $\{\{\blacklozenge\} \cup \{\$\}\}$
$\underline{X_i'^1}, 1 \le i \le n$	$\{\dot{x} \to \overline{x_i'}, \$' \to 0\}$	$[-(2^{(n+1)}+1), -1]$, $[-n^2(2^{(n+1)}), -n(2^{(n+1)})]$	$\{\downarrow\}$	$\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
$\overline{X_i'^1}, 1 \le i \le n$	$\{\dot{x} \to \overline{x_i'}, \blacklozenge \to 1\}$	$[2n^2(n+2), \infty)$, $[-n^2(2^{(n+1)}), -n(2^{(n+1)})]$	$\{\downarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$

Table 4. Internals of the nodes of the subnetwork $\mathbf{Eval_2}$.

Node	M	α	β	S
U_1	{unmask}	$[2n^2(n+2), \infty)$, $\{-2(n+1)(2^{(n+1)})\}$	$\{\downarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
F_1	$\{c_0' \to c_0'', \boxplus^k \to \boxtimes^k\}$ $1 \le k \le m$	$[-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$, $\{2n(n+1)\}$	$\{\downarrow\}$	$\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$ $\{C \cup C''\}$
F_2	$\{c_i'' \to c_{i+1}'', \blacksquare \to \$\}$ for all $0 \le i \le n$	$[2n^2(n+2), \infty)$, $\{-(n^2)(2^{(n+1)})\}$	$\{+\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
M	{mask}	$[2n^2(n+2), \infty)$, $\{-n(2^{(n+1)})\}$	$\{\rightarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
A_1^0	$\{\overline{p_{i,j}^{(k)}} \to 0, \underline{x_i} \to \$', \overline{x_i} \to \$'\}$ for all $1 \le i, j \le n$	$[2n^2(n+2), \infty)$, $\{-2\}$	$\{\uparrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$
$\overline{A_i^1}$	$\{\overline{p_{i,j}^{(k)}} \to 1, \overline{x_i} \to \$', \underline{x_i} \to \$\}$ for all $1 \le i, j \le n$	$[2n^2(n+2), \infty)$, $\{-2\}$	$\{\uparrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$
$\underline{A_i^1}$	$\{\overline{p_{i,j}^{(k)}} \to 0, \overline{x_i} \to \$, \underline{x_i} \to \$'\}$ for all $1 \le i, j \le n$	$[2n^2(n+2), \infty)$, $[-2, -1]$	$\{\uparrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$
F_3	$\{c_i' \to \ddot{c_i}, \$' \to \$\}$ for all $1 \le i \le n$	$[-(2^{(n+1)}+1), -1]$	$\{+\}$	$\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
U_2	{unmask}	$[2n^2(n+2), \infty)$, $\{-n(2^{(n+1)})\}$	$\{\rightarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
M_1	{mask}	$\{-4mn(n+1)(2^{(n+1)})\}$	$\{\leftarrow\}$	$\{\ddot{C} \cup B' \cup \{c_0^0\} \cup \{c_0^1\}\}$, $\{\{\blacklozenge\} \cup \{\$\}\}$
F_5	$\{\ddot{c_n} \to c_n', \boxtimes^i \to \boxplus^k\}$ $1 \le k \le m$	$\{-4mn(n+1)(2^{(n+1)})\}$	$\{\rightarrow\}$	$\{\ddot{C} \cup B' \cup \{c_0^0\} \cup \{c_0^1\}\}$, $\{\{\blacklozenge\} \cup \{\$\}\}$
F_6	$\{\$ \to \ddot{k}\}$	$[2n^2(n+2), \infty)$, $[-(n+2)(2^{(n+1)}), -3n(2^{(n+1)})]$	$\{\uparrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
M_2	{mask}	$[2n^2(n+2), \infty)$, $\{-3n(2^{(n+1)})\}$	$\{\uparrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
U_3	{unmask}	$[0, n(2n+1)]$, $\{2n(n+1)^2\}$	$\{\leftarrow\}$	$\{\underline{X} \cup \underline{X'} \cup \overline{X} \cup \overline{X'} \cup \{1\} \cup \{0\}\}$ $\{C \cup C''\}$
F_7	$\{\ddot{c_i} \to c_i'$, for all $1 \le i \le n\}$	$[2n^2(n+2), \infty)$, $\{-n(n+2)(2^{(n+1)})\}$	$\{\downarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
S_1	{unmask}	$[2n^2(n+2), \infty)$, $\{-(2n+1)n(2^{(n+1)})\}$	$\{\uparrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
S_2	{unmask}	$[2n^2(n+2), \infty)$, $\{2n(n+1)\}$	$\{\leftarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$ $\{C \cup C''\}$

Table 5. Internals of the nodes of the subnetwork **Val**.

Node	M	α	β	S
W	$\{\overline{x_i'} \to \ddot{x_i}, \underline{x_i'} \to \dot{x}$, $\ddot{k} \to \$\}$, $1 \le i \le n$	$[2n^2(n+2), \infty)$, $\{-n(n+1)(2^{(n+1)})\}$	$\{+\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
V_0	$\{\overline{p_0^{(k)}} \to 1, \underline{p_0^{(k)}} \to 0\}$ $1 \le k \le m$	$\{[-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$, $\{1, 3, 5, 7 \ldots . (2n+1)\}$	$\{+\}$	$\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$, $\{\underline{X} \cup \underline{X'} \cup \overline{X} \cup \overline{X'} \cup \{1\} \cup \{0\}\}$
V_1	$\{\overline{p_0^{(k)}} \to 1, \underline{p_0^{(k)}} \to 0\}$ $1 \le k \le m$	$\{[-2n(n+1)(2^{(n+1)}), -n(2^{(n+1)})]$, $\{0, 2, 4, \ldots 2n\}$	$\{+\}$	$\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$, $\{\underline{X} \cup \underline{X'} \cup \overline{X} \cup \overline{X'} \cup \{1\} \cup \{0\}\}$
V_0'	$\{c_0'' \to c_0^1, c_1' \to \ddot{c_1}$, $c_i' \to \$\}$, $2 \le i \le n$	$[2n^2(n+2), \infty)$, $\{-n(n+1)(2^{(n+1)})\}$	$\{\downarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
V_1'	$\{c_0'' \to c_0^0, c_1' \to \ddot{c_1}$, $c_i' \to \$\}$, $\forall\, 2 \le i \le n$	$[2n^2(n+2), \infty)$, $\{-n(n+1)(2^{(n+1)})\}$	$\{\downarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{C' \cup \{\blacksquare\} \cup \{\ddot{k}\} \cup B\}$
V_2^i, $0 \le i < 2^n$	$\{c_1' \to R_i\}$, $1 \le i < 2^n$	$[2n^2(n+2), \infty)$, $\{2^i\}$	$\{+\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$, $\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$
H	$\{\ddot{x_i} \to \overline{x_i}, \dot{x_i} \to \underline{x_i}\}$, $1 \le i \le n$	$[-(2^{(n+1)}+1), -1]$	$\{+\}$	$\{X \cup \dot{X} \cup \ddot{X} \cup \{\$'\}\}$, $\{\{\blacklozenge\} \cup \{\$\}\}$
A	$\{c_0^0 \to 0\}$	$[2n^2(n+2), \infty)$	$\{\downarrow\}$	$\{\{\blacklozenge\} \cup \{\$\}\}$

4.1. The Node in

By default, in the initial configuration, the node *In* holds the input picture coding the instance of the problem to solve. For our network Γ, the node *In* holds the input pictures representing each polynomial $P^{(1)}(\mathbf{x}), P^{(2)}(\mathbf{x}), \ldots, P^{(m)}(\mathbf{x})$. In addition, this node serves like a both receptor and transmitter of the pictures processed by subnetwork **Gen**. For this reason, this node only has a substitution rule that changes one control symbol by other one. In particular, while the subnetwork **Gen** is generating configurations, i.e., pictures with the corresponding symbols assigned to each indeterminate, these are allowed to pass to the $\mathbf{Eval_1}$ subnetwork by the *In* node for further processing. This node's internals are shown in Table 2.

4.2. Generating Subnetwork

The subnetwork **Gen** represents the *generating phase*, namely, the phase in charge of generating $m \cdot 2^n$ configurations for the initial pictures $\pi^1, \pi^2, \pi^3, \ldots \pi^m$ encoding an RMQE instance. In particular, this subnetwork replaces each x_i for a symbol $\overline{x_i}$ or $\underline{x_i}$, which represents the value 1 or 0 respectively, and hence generates 2^n configurations for each initial picture π^k. The graphical representation of the underlying subgraph for subnetwork **Gen** is depicted in the Figure 2.

According to the Figure 2, each indeterminate x_i is represented by two nodes, namely, X_i^0 and X_i^1. Concretely, the node X_i^0 replaces all occurrences of symbol x_i for the symbol $\underline{x_i}$ (which represents 0). On the other hand, the node X_i^1 replaces all occurrences of symbol x_i for one symbol $\overline{x_i}$ (which represents the value 1). After n processing steps, the subnetwork **Gen** will have modified all x_i symbols and generated 2^n configurations for each initial picture π^k. These configurations will be collected by the *In* node. The internals of each node $X_i^1, X_i^0, 1 \leq i \leq n$ in this subnetwork are defined in the Table 2.

4.3. Linear-Evaluating Subnetwork

The aim of this subnetwork is to evaluate AND operations for each pair of symbols $p_i^{(k)} x_i$ with $1 \leq i \leq n$, which are encoded at the row $n+1$ of each picture. The graph representing this subnetwork is depicted in the Figure 3.

This subnetwork starts off with the node G receiving pictures from the node In. The received pictures are then processed by the node G, by masking the two last rows (at indexes $(n+2)$ and $(n+3)$) from them. This makes each processed picture's last visible row be the row that contains the terms $p_i^{(k)} x_i$ (i.e., the $n-1$ row). This row is what will then be processed by other nodes in this subnetwork, by applying on it substitution rules with the β mode $\downarrow$.

The node L serves as a control node to keep the traceability of the substitution of each pair $p_i^{(k)} x_i$ done by the subnetwork X_i depicted in the Figure 3 (the internals of these boxes can be seen in the rectangle at left-upper corner). In a similar way as the subnetwork **Gen** operates, the nodes $\underline{X_i^0}$ and $\overline{X_i^1}$ replace all occurrences of symbols $\underline{x_i}$ and $\overline{x_i}$ for an intermediary symbol in the set $\dot{X}$. With them, this subnetwork is able to evaluate whether the result of the AND operation for the pair $p_i^{(k)} x_i$ is 0 or 1 through the evaluation sets defined in the nodes $\underline{X_i'^0}$ and $\overline{X_i'^1}$, respectively. In these nodes, the symbols belonging to the set P_L are substituted by the symbols either $\underline{x_i'}$ or $\overline{x_i'}$ according to the result from the AND operation (representing 0 and 1, respectively). Concretely, in the node $\overline{X_i^1}$, the symbols representing the $p_i^{(k)}$ values are substituted by trash symbols either \$′ or ♦. Subsequently, the nodes $\overline{X_i'^1}$ and $\underline{X_i^1}$ let pass only the pictures which evaluation represents 1 or 0, respectively. Then, they substitute the trash symbols for the specific result (1 or 0). Therefore, the result of the evaluation has been assigned to the place for the $p_i^{(k)}$ symbols at the picture. The processing for L and nodes represented in the boxes X_i is repeated until all terms $p_i^{(k)} x_i$ in the pictures have been substituted, with the valid resulting pictures leaving the node L and entering the node

Q. Finally, the node Q masks the recently evaluated row for all received pictures making only the first n rows visible, i.e., the rows that encode the terms the $p_{i,j}^{(k)} x_i x_j$. In this way, the pictures are prepared for the following phase (subnetwork **Eval$_2$**), which evaluates the quadratic terms. The internals of each node in this subnetwork Eval$_1$ are defined in the Table 3.

4.4. Quadratic-Evaluating Subnetwork

This subnetwork is responsible for evaluating each term $p_{i,j}^{(k)} x_i x_j$ in the pictures. The result from evaluating the AND operation between each one these three terms will be represented by the substitution of the $p_{i,j}^{(k)}$ symbols by a symbol 0 or 1 (depending of the result). Similarly at the subnetwork Eval$_1$, the symbols $\underline{x_i} \mid \overline{x_i}$ are replaced by the trash symbols \$ or \$′. The Figure 4 depicts the graph of this subnetwork. The internals of each node in this subnetwork are defined in the Table 4.

The processing of this subnetwork starts off with the node U_1 receiving all pictures from the node Q. Then, this node unmasks all invisible rows such that it makes the last rows visible. More specifically, the node U_1 unmasks the rows at indexes $(n+1)$,$(n+2)$ and $(n+3)$ in each received picture. After that, the processed pictures are allowed to move to the node F_1, which substitutes the control symbol c_0' and the symbol $\boxplus^k$ in each picture for the symbols c_0'' and $\boxtimes^k$, respectively. Note that this node only does this substitution once and it is only for controlling the internal processing.

After this substitution is carried out, the process for evaluating each row containing the terms $p_{i,j}^{(k)} x_i x_j$, $1 \leq i \leq j \leq n$ starts such that each term is evaluated one at time. The node F_2 is responsible for keeping track the evaluation of each row. Specifically, this node starts with the substitution of the symbol c_0'' for c_1'' in order to indicate that the row 1 will be evaluated subsequently. In addition, this node substitutes all symbols ■ for the trash symbol \$, because they are no longer needed.

The next time this node receive pictures from node F_7, it can substitute the symbol again and so on until all rows are evaluated. The node M is able to mask all columns from the rightmost column to the column containing the x_j (exclusive) symbol of the current $p_{i,j}^{(k)} x_i x_j$ to be evaluated (this is controlled by means of the c_i' symbol place at this same column). With this masking process, Γ makes visible only the current three terms $p_{i,j}^{(k)} x_i x_j$ at the first visible row. Subsequently, the operation AND is calculated for these three symbols in a similar way as it is processed for the subnetwork **Eval$_1$** (in the boxes X_i). More concretely, this evaluation is done by the nodes A_i^0, $\underline{A_i^1}$ and $\overline{A_i^1}$. Then, the node F_3 substitutes the c_i' for $\ddot{c}_i$ in order to identify that the current quadratic term was evaluated. In addition, this node substitutes the \$′ symbol for the original one. Note that the result of the AND operation is encoded at the place in which the symbol $p_{i,j}^{(k)}$ was placed. In order to carry out the evaluation for the rest of the $p_{i,j}^{(k)} x_i x_j$ terms in the current row, the nodes U_2 and M_1 unmask the rightmost three columns and mask the current three visible columns, respectively. M_1 is connected with the nodes A_i^0, $\underline{A_i^1}$ and $\overline{A_i^1}$ and it allows that the process for evaluating each $p_{i,j}^{(k)} x_i x_j$ term is repeated until all of them are evaluated. When it occurs, the pictures from node F_3 migrate to the node F_5 and it makes the other branch of the subnetwork will be processed.

Given that the last control symbol substituted was c_n' by $\dot{c}_n$, the pictures can enter to the node F_5. Then, this node returns the symbols $\dot{c}_n$ to their original one (c_n'). Similarly, this node substitutes the symbols $\boxtimes^k$ by their original ones. After that, the pictures migrate to the node F_6 and this node substitutes one symbol \$ for a $\ddot{k}$ in order to indicate that the current row was just evaluated. Then, the pictures enter to M_2 and the current row is masked. Subsequently, the node U_3 unmasks all masked columns to the left. The processing for this branch finishes when the node F_7 substitutes all $\ddot{c}$ symbols for their original ones in the set C'. With these substitutions, Γ is able to evaluate the next visible row and hence, the pictures migrate to the F_2 node in order to repeat the processing of the next row. When the pictures do not contain $p_{i,j}^{(k)} x_i x_j$ symbols, then they migrate from the node F_2 to the node S_1. Note that all pictures with the symbol $\ddot{k}$ are allowed into the

node S_1. Finally, the nodes S_1 and S_2 are responsible for unmasking all hidden rows and columns in order to promote them for the next phase.

4.5. Validation Subnetwork

The goal of this subnetwork is twofold. First, this subnetwork calculates the final value for each picture, e.g., the evaluation to the polynomial that the picture represents. More specifically, this subnetwork computes the resulting value from each $P^{(i)}$ (i.e., 0 or 1), for all $1 \leq i \leq m$, after performing XOR operations between the terms that compose the picture (summation of all them). This result will be stored as a value for the control symbol c_0 at the last row in the pictures. Second, this subnetwork groups all the pictures sharing the same values for the indeterminates $x_1, x_2, \ldots x_n$. This grouping will allow to obtain the solutions for the problem, that is, to find vectors $\hat{x}$ satisfying Equation (4). The values for each x_i remains represented by the symbols belonging to $\underline{X'}$ or $\overline{X'}$ placed into the row n (0 or 1, respectively). The Figure 5 depicts the graph of this subnetwork.

Concretely, the nodes V_1, V_0, V_1' and V_0' performs a XOR operation with the values placed in rows from one to n and replace the control symbol c_0 with the result obtained from performing the XOR operation, i.e., by substituting the symbol c_0 by other control symbol c_0^0 if the result of XOR operation is 0 or c_0^1 otherwise. The nodes $V_2^i, 0 \leq i < 2^n$, group all pictures belonging to each possible combination of n values assigned to the indeterminates. In particular, each configuration is mapped to an integer in the range $[0, 2^n - 1]$. Previously, the node W has substituted all symbols in the sets $\underline{X'}$ and $\overline{X'}$ for their corresponding symbols in the sets $\dot{X}$ and $\ddot{X}$, respectively. It is necessary in order to evaluate each configuration filtering each picture according to the value obtained by nodes $V_2^i, 0 \leq i < 2^n$. These nodes substitutes the control symbol c_1 for a control symbol $R_i, 0 \leq i \leq 2^n - 1$, where the index i represents the number of configuration. Finally, the node *Halt* substitutes all symbols belonging to the sets $\dot{X}$ and $\ddot{X}$ for the original ones and the node *Accept* collects all pictures representing the solution of the input instance. The internals of each node in this subnetwork are defined in the following Table 5.

4.6. Discussion

Lemma 1. *The* NPPES Γ *defined in Section 4 computes a solution for the RMQE problem.*

Proof of Lemma 1. At the beginning of the computation, Γ contains the set of pictures $\Pi = \{\pi^{(1)}, \pi^{(2)}, \ldots, \pi^{(m)}\}$ encoding the polynomials to be evaluated (all of them in the node In). Thus, the initial configuration is defined by $C_0(In) = \Pi$ and $C_0(x) = \emptyset$, $\forall\, x \in X_G \backslash \{In\}$. The next configuration of Γ consist of the initial pictures (with the c_0 symbol substituted by c_0') for all $X_i^0, X_i^1, 1 \leq i \leq n$ represented in the Figure 2. After n processing steps, the node In has collected $m \cdot 2^n$ pictures. Note that for each initial picture $\pi \in \Pi$, the subnetwork **Gen** generates a picture for each possible $n-$tuple of values to the indeterminates $x_1, x_2, \ldots x_n$ in π. Therefore, it generates $m \cdot 2^n$ pictures that are collected by the node In and are then sent out to the subnetwork $\mathbf{Eval_1}$, more concretely to the node G. This node masks the three last rows in all received pictures and then sends the pictures out to the node L, which includes one new trash symbol indicating the first step to evaluate the terms placed in the row n, namely $p_i^{(k)} x_i$, for all $1 \leq i \leq n, 1 \leq k \leq m$.

Note that the nodes $\underline{X_i^0}, \underline{X_i'^0}$ evaluate, in two processing steps, whether each term is 0. At the same processing steps, the $\overline{X_i^0}, \overline{X_i'^0}$ nodes evaluate whether each term is 1. These evaluations are done by applying simple substitution rules. After n processing steps, each picture will have all $p_i^{(k)} x_i$ terms evaluated and be collected by the node L. All pictures then migrate to the Q node, which masks the row containing the recently evaluated terms (last row) within a processing step. Then, the pictures are sent out to the subnetwork $\mathbf{Eval_2}$, more precisely to the node U_1. This node makes all masked rows in each

received picture visible through two processing steps. Note that F_1 is a control node which performs its task once, then it performs one processing step. F_2 controls the rows being evaluated by the subnetwork, starting from the first row by substituting one symbol c'_i and n symbols ■ using n processing steps. The nodes $M, M_1, F_3, U_2, A_1^0, \underline{A_i^1}, \overline{A_i^1}$ evaluate each term $p_{i,j}^{(k)} x_i x_j$ in each corresponding row of a picture, by making the three columns occupied by each term visible and then performing an AND operation with the corresponding values. When all terms are evaluated, the hidden columns are made visible again and a mechanism to indicate that this row was evaluated is used. Then, these nodes perform n times their respective steps.

The nodes F_5, F_6, F_7, M_2 and U_3 make the just-evaluated row invisible, mark the index of the evaluated row (substituting the respective symbol in C' and preparing the pictures for the evaluation of the next row). After the evaluation of the possible occurrences of the $p_{i,j}^{(k)} x_i x_j$ terms, the node S_1 and subsequently S_2 collect all evaluated pictures and return the masked rows to the visible state after perform n and $n-3$ processing steps, respectively. Then S_2 contains a set of pictures encoding each and every admissible evaluation of each polynomial in the Equation (3). However, at this stage, we need to organize all pictures by classes, where each groups pictures of each polynomial for one of the 2^n $n-$tuple of values that can be assigned to the indeterminates, and then determine which classes satisfy the Equation (4), if any (if a class has m pictures then the corresponding $n-$tuple is a solution).

The node W performs n processing steps in order to do all substitutions. V_0 and V_1 substitutes the independent term in the $n+1$ row of a picture for the corresponding symbol for all pictures at the same processing step. In addition, these nodes let can pass to the nodes V'_0 and V'_1 only the pictures with evaluation of all symbols 1 being wither odd or even, respectively. V'_0 and V'_1 nodes substitute control symbols at the last row in the same processing step. The nodes V_2^i group the pictures according to each possible $n-$tuple of values for the indeterminates that satisfy Equation (4), via marking the picture with an R_i symbol, where the index i represents the i_{th} $n-$tuple of values, at same processing step. The transition of H to A is immediate: H returns the symbols of the vector to the original ones and A puts the final result of the evaluation in the control symbol c_0^0. Each one of these two nodes perform one processing step. Finally, the node A collects the pictures containing the solution for the given instance. □

Therefore, as a consequence of this result, we obtained the following theorem:

Theorem 1. *The RMQE problem as it was defined in Section 3 can be solved by a* NPPES Γ *in time* $O(n^2)$.

Proof of Theorem 1. Following the identical conditions as we have mentioned before, the processing steps in Γ are:

- Node *In*: $n+1$ steps
- Subnetwork **Gen**: n steps performed by X_i^0,X_i^1, $1 \leq i \leq n$, nodes.
- Subnetwork $\mathbf{Eval_1}$: $3n+5$ steps performed as follows:
  - Node G: 2 steps
  - Node L: $(n+1)$ steps
  - Node Q: 2 steps
  - Nodes $\overline{X_i^0}$,$\overline{X_i^1}$, $1 \leq i \leq n$: n steps
  - Nodes $\underline{\overline{X_i'^0}}$, $\underline{X_i'^1}$, $\overline{X_i'^1}$, $1 \leq i \leq n$: n steps
- Subnetwork $\mathbf{Eval_2}$: $3n(n+3)$ steps performed as follows:
  - Node U_1: 2 steps.
  - Node F_1: 1 step.
  - Nodes: $F_2, F_3, F_4, F_5, F_6, M_2, F_7$: $7n$ steps (1 processing step performed by every node)
  - Nodes M_1, U_2: $6n$ steps ($3n$ processing steps performed by every node)

- Nodes M, $\underline{U_3}$: at most $n(n-3)$ steps for each of these nodes, then $2n(n-3)$
- Nodes A_1^0, A_i^1 and $\overline{A_i^1}$: at most n processing steps for each n row, then, n^2 steps
- Node S_1: n steps
- Node S_2: $n-3$ steps

- Subnetwork **Val**: $n+3$ steps performed as follows:
  - Node W: n processing steps
  - Nodes V_1, V_0, 1 step
  - Nodes V_1', V_0', 1 step
  - Nodes V_2^i, $1 \leq i \leq n$, 1 step
- Nodes H and A, 1 step every node, then, 2 steps.

Therefore, the total number of processing steps is not more than $3n^2 + 15n + 11$. Note that, if a given instance of the problem has k solutions, with $k \geq 0$, then Γ finds the k $n-$tuples of values for the indeterminates that are solutions for the given instance. Consequently, the overall time for solving an instance of the RMQE problem, as it is defined in Equation (4), is $O(n^2)$. □

We remark that the proposed network Γ seems to have a significant size in terms of number of nodes. This can be seen as an aspect to be improved; however, note that this construction allows us to control the exponential growth of the generated strings in any moment. In addition, we obtain a viable solution in an acceptable computation time given the complexity of the problem we are dealing with. Also, Γ sieves the correct pictures in an early identification step, minimizing the unnecessary copies. Additionally, note that other NBP models do not give such flexibility when the problem needs to meet integer constraints. Consequently, NPPES opens up an interesting perspective to fill this gap. Another important reason to support the solution we presented here is that, given the high maturity achieved by the ultra-scalable computing platforms, since this facilities deploying distributed algorithms in a reasonable way [16], our proposed solution might be deployed by using any hardware/software solution as those reported in [30], without requiring any modification (re-structuring of nodes, rules, mechanism to workload distribution between computational units or so on). That is to say, we potentially may put our solution to run on workers or parallel computing units in any scalable software or hardware platform without re-defining the proposed algorithm, because our distribution of nodes would be more suitable to reallocate parallel processing units within a real computing platform.

5. Conclusions and Final Remarks

We have introduced a solver for the RMQE problem that is based on NPPES and runs in quadratic time, suggesting that the NPPES model can be employed to cope with hard complex problems in which numerical evaluation has an essential role. In this context, the extended model is able to deal with a new variety of hard problems because of the model's new features, in particular its filtering strategy makes the selection of correct pictures easier compared to the previous model. Additionally, the model exhibits efficiency and uses up less resources than similar bio-inspired models facing similar hard problems. To the best of our knowledge, this is the first time in which an NBP model is used to propose a solver for the RMQE problem. Also this solver demonstrates the computational power that may be achieved by employing bio-inspired computational models, since existing non-bio-inspired solutions run in exponential time for specific n, m input values. Furthermore, our proposed solution for the RMQE problem opens an interesting opportunity to propose new solvers for these complex problems, bringing about real computing solutions with a reasonable computing time (which can be understood as "efficient time").

In previous works [14,31], it is stated that NPEP solutions need an adaptation for allowing the deployment on computational platforms for massive distributed processing. Moreover, this adaptation must preserve the solution's complexity in order to achieve practical solutions to real problems on big data

computing platforms (which is extensible to any solution using similar models computing pictures instead of strings). In this context, the proposed solution here does not require any adaptation to be deployed in massively parallel and distributed computing platforms, and therefore, we avoid the previously commented drawbacks when these algorithms are deployed in practical scenarios.

Our results additionally suggest that the NPPES model may be more simple than other models working on pictures or strings for tackling hard optimization problems with numerical evaluations. We thus think this model shows features worthy of further investigation, both from a theoretical approach and from a practical point of view. With respect to the latter, we think studying the limits of software and hardware implementations to attack complex problems with big data computing platforms deserves effort and attention. Additionally, researching into the suitability of extending this model to reinforce it for other related high-processing problems also is interesting. Finally, we shall return to these topics in a forthcoming work.

Author Contributions: Conceptualization, R.V.-P., S.G.-C. and K.P.J.; methodology, R.V.-P., S.G.-C. and K.P.J.; theoretical and practical validation, S.M.S.; formal analysis, R.V.-P., S.G.-C. and K.P.J.; investigation, R.V.-P., S.G.-C. and K.P.J.; writing–original draft preparation, R.V.-P., S.G.-C. and K.P.J.; writing–review and editing, R.V.-P., S.G.-C. and K.P.J.; All authors have read and agreed to the published version of the manuscript.

Funding: Ricardo Villanueva-Polanco was funded by Universidad del Norte (Grant 2019-029), and Karina Paola Jiménez was funded by Fundación Carolina (call 2014) and Universidad Simon Bolivar.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NBP	Network of Bio-inspired Models
NPEP	Networks of Polarized Evolutionary Processors
NPP	Networks of Picture Processors
RMQE	Random Multivariate Quadratic Equations
MQ	Multivariate Quadratic problem
NPPES	Network of Picture Processors with Evaluation Sets

References

1. Castellanos, J.; Martín-Vide, C.; Mitrana, V.; Sempere, J. Networks of evolutionary processors. *Acta Inform.* **2003**, *39*, 517–529. [CrossRef]
2. Manea, F.; Martin-Vide, C.; Mitrana, V. Accepting networks of splicing processors: Complexity results. *Theor. Comput. Sci.* **2007**, *371*, 72–82.
3. Campos, M.; Sempere, J. Accepting Networks of Genetic Processors are computationally complete. *Theor. Comput. Sci.* **2012**, *456*, 18–29. [CrossRef]
4. Bottoni, P.; Labella, A.; Manea, F.; Mitrana, V.; Sempere, J.M. Networks of Evolutionary Picture Processors with Filtered Connections. In *Unconventional Computation*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 70–84.
5. Manea, F.; Martín-Vide, C.; Mitrana, V. Accepting networks of evolutionary word and picture processors. A survey. *Math. Comput. Lang. Life Front. Math. Linguist. Lang. Theory World Sci.* **2010**, *2010*, 523–560.
6. Bottoni, P.; Labella, A.; Mitrana, V. Networks of Evolutionary Picture Processors. *Fundam. Informaticae* **2014**, *131*, 337–349. [CrossRef]
7. Bordihn, H.; Bottoni, P.; Labella, A.; Mitrana, V. Networks of picture processors as problem solvers. *Soft Comput.* **2017**, *21*, 5529–5541. [CrossRef]
8. Bordihn, H.; Bottoni, P.; Labella, A.; Mitrana, V. Solving 2D-Pattern Matching with Networks of Picture Processors. In *Theory and Practice of Natural Computing*; Dediu, A.H., Lozano, M., Martín-Vide, C., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 157–168.

9. Popescu, S. Solving 2-D Pattern Matching using Networks of Polarized Picture Processors with Circular Permutation. In Proceedings of the 2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), Timisoara, Romania, 21–24 September 2017; pp. 338–343.
10. Arroyo, F.; Gómez-Canaval, S.; Mitrana, V.; Sánchez, J. Networks of Picture Processors with Circular Permutation. *Proc. Rom. Acad. Ser.* **2019**, *20*, 311–319.
11. Popsecu, S. Networks of Polarized Evolutionary Picture Processors. *Rom. J. Inf. Sci. Technol.* **2015**, *18*, 3–17.
12. Alarcón, P.; Arroyo, F.; Mitrana, V. Networks of polarized evolutionary processors. *Inf. Sci.* **2014**, *265*, 189–197. [CrossRef]
13. Arroyo, F.; Gómez-Canaval, S.; Mitrana, V.; Popescu, S. Networks of Polarized Evolutionary Processors Are Computationally Complete. In *Language and Automata Theory and Applications*; Lecture Notes in Computer Science; Springer International Publishing: Berlin/Heidelberg, Germany, 2014; Volume 8370, pp. 101–112. [CrossRef]
14. Gómez-Canaval, S.; Jiménez, K.; Ortega, A.; Vakaruk, S., Towards Quantitative Networks of Polarized Evolutionary Processors: A Bio-Inspired Computational Model with Numerical Evaluations. In *Trends in Practical Applications of Scalable Multi-Agent Systems, the PAAMS Collection*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 251–259. [CrossRef]
15. Gómez-Canaval, S.; Mitrana, V.; Sánchez-Couso, J. Networks of Splicing Processors With Evaluation Sets as Optimization Problems Solvers. *Inf. Sci.* **2016**, *369*, 457–466. [CrossRef]
16. Arroyo, F.; Gómez-Canaval, S.; Jiménez, K.; Ortega, A. A Linear Time Solution for N-Queens Problem Using Generalized Networks of Evolutionary Polarized Processors. *Int. J. Found. Comput. Sci.* **2020**, *31*, 7–21. [CrossRef]
17. Faugere, J.C.; Horan, K.; Kahrobaei, D.; Kaplan, M.; Kashefi, E.; Perret, L. Fast Quantum Algorithm for Solving Multivariate Quadratic Equations. *Quantum Inf. Comput.* **2017**, *2017*, 1236–1252.
18. Ding, J. A New Proof of Work for Blockchain Based on Random Multivariate Quadratic Equations. In *Applied Cryptography and Network Security Workshops*; Zhou, J., Deng, R., Li, Z., Majumdar, S., Meng, W., Wang, L., Zhang, K., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 97–107.
19. Fernández-Caramès, T.M.; Fraga-Lamas, P. Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks. *IEEE Access* **2020**, *8*, 21091–21116. [CrossRef]
20. Thomae, E.; Wolf, C. Solving Underdetermined Systems of Multivariate Quadratic Equations Revisited. In *Public Key Cryptography–PKC 2012*; Fischlin, M., Buchmann, J., Manulis, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 156–171.
21. Courtois, N.; Klimov, A.; Patarin, J.; Shamir, A. Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations. In *Advances in Cryptology—EUROCRYPT 2000*; Preneel, B., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; pp. 392–407.
22. Chen, J.; Ning, J.; Ling, J.; Lau, T.S.C.; Wang, Y. A new encryption scheme for multivariate quadratic systems. *Theor. Comput. Sci.* **2020**, *809*, 372–383. [CrossRef]
23. Chen, M.; Hulsing, A.; Rijneveld, J.; Samardjiska, S.; Schwabe, P. MQDSS Specifications. Post-Quantum Cryptography Standardization. 2020. Available online: http://mqdss.org/files/mqdssVer2point1.pdf (accessed on 30 September 2020).
24. Garey, M.; Johnson, J. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: New York, NY, USA, 1990.
25. Ding, J.; Schmidt, D. Rainbow, a new Multivariable Polynomial Signature Scheme. In *Applied Cryptography and Network Security*; Ioannidis, J., Keromytis, A., Yung, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 164–175.
26. Beullens, W.; Preneel, B. Field Lifting for Smaller UOV Public Keys. In *Progress in Cryptology–INDOCRYPT 2017*; Patra, A., Smart, N.P., Eds.; Springer International Publishing: Cham, Switzerland, 2017; pp. 227–246.

27. Beullens, W.; Preneel, B.; Szepieniec, A.; Vercauteren, F. LUOV: Signature Scheme proposal for NIST PQC Project (Round 2 version). Post-Quantum Cryptography Standardization. 2020. Available online: https://github.com/WardBeullens/LUOV/blob/master/Supporting_Documentation/luov.pdf (accessed on 30 September 2020).
28. Chen, M.; Hülsing, A.; Rijneveld, J.; Samardjiska, S.; Schwabe, P. From 5-Pass MQ-Based Identification to MQ-Based Signatures. In *Advances in Cryptology–ASIACRYPT 2016*; Cheon, J.H., Takagi, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; pp. 135–165.
29. Ding, J.; Chen, M.S.; Petzoldt, A.; Schmidt, D.; Yang, B.Y.; Kannwischer, M.; Patarin, J. Rainbow-Algorithm Specification and Documentation. Post-Quantum Cryptography Standardization. 2020. Available online: https://www.pqcrainbow.org/ (accessed on 30 September 2020).
30. Gómez-Canaval, S.; Ordozgoiti, B.; Mozo, A., NPEPE: Massive Natural Computing Engine for Optimally Solving NP-complete Problems in Big Data Scenarios. In *New Trends in Databases and Information Systems*; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 207–217. [CrossRef]
31. Gómez-Canaval, S.; Sánchez-Couso, J.; Vinyals, M. Solving optimization problems by using networks of evolutionary processors with quantitative filtering. *J. Comput. Sci.* **2016**, *16*, 65–71. [CrossRef]