FUNCTIONAL GRAPHS OF POLYNOMIALS OVER
FINITE FIELDS

SERGEI V. KONYAGIN, FLORIAN LUCA, BERNARD MANS,
LUKE MATHIESON, AND IGOR E. SHPARLINSKI

Abstract. Given a function $f$ in a finite field $\mathbb{F}_q$ we define the
functional graph of $f$ as a directed graph on $q$ nodes labelled by
elements of $\mathbb{F}_q$ where there is an edge from $u$ to $v$ if and only
if $f(u) = v$. We obtain some theoretic estimates on the number
of non-isomorphic graphs generated by all polynomials of a given
degree. We then develop an algorithm to test the isomorphism of
quadratic polynomials that has linear memory and time complex-
ities. Furthermore we extend this isomorphism testing algorithm
to the general case of functional graphs, and prove that, while its
time complexity increases only slightly, its memory complexity re-
 mains linear. We exploit this algorithm to provide an upper bound
on the number of functional graphs corresponding to polynomials
of degree $d$ over $\mathbb{F}_q$. Finally we present some numerical results
and compare function graphs of quadratic polynomials with those
generated by random maps and pose interesting new problems.

1. Introduction

Let $\mathbb{F}_q$ be the finite field of $q$ elements. For a function $f : \mathbb{F}_q \to \mathbb{F}_q$
we define the functional graph of $f$ as a directed graph $\mathcal{G}_f$ on $q$ nodes
labelled by elements of $\mathbb{F}_q$ where there is an edge from $u$ to $v$ if and
only if $f(u) = v$.

Clearly each connected component of $\mathcal{G}_f$ contains one cycle (possible
of length 1 corresponding to a fixed point) with several trees attached
to some of the cycle nodes.

Here we are mostly interested in the graphs $\mathcal{G}_f$ associated with polynomi-
als $f \in \mathbb{F}_q[X]$ of given degree $d$.

Some of our motivation comes from the natural desire to better un-
derstand Pollard’s $\rho$-algorithm (see [6, Section 5.2.1]). We note that
although this algorithm has been used and explored for decades, there
is essentially only one theoretic result due to Bach [11]. In fact, even a

2010 Mathematics Subject Classification. 05C20, 05C85, 11T06, 11T24.

Key words and phrases. polynomial maps, functional graphs, finite fields, char-
acter sums, algorithms on trees.
heuristic model adequately describing this algorithm is not quite clear, as the model of random maps, analysed by Flajolet and Odlyzko in [8], does not take into account the restrictions on the number of preimages. The model analysed by MacFie and Panario in [19] approximates Pollard’s algorithm better but it perhaps still does not capture it in full. Polynomial maps can also be considered as building blocks for constructing hash functions. For these applications, it is important to understand the intrinsic randomness of such maps.

Further motivation to investigation of the graphs \( G_f \) comes from the theory of dynamical systems, as \( G_f \) fully encodes many of the dynamical characteristics of the map \( f \), such as the distribution of period (or cycle) and pre-period lengths.

In particular, we denote by \( N_d(q) \) the number of distinct (that is, non-isomorphic) graphs \( G_f \) generated by all polynomials \( f \in \mathbb{F}_q[X] \) of degree \( \deg f = d \). Trivially, we have

\[
N_d(q) \leq q^{d+1}.
\]

Here, we use some ideas of Bach and Bridy [2] together with some new ingredients to obtain nontrivial bounds on \( N_d(q) \).

We also design efficient algorithms to test the isomorphism of graphs \( G_f \) and \( G_g \) associated with two maps \( f \) and \( g \). Furthermore, we design an efficient algorithm that generates a unique label for each functional graph. We use these algorithms to design an efficient procedure to list all \( N_d(q) \) non-isomorphic graphs generated by all the polynomials \( f \in \mathbb{F}_q[X] \) of degree \( \deg f = d \).

We conclude by presenting some numerical results for functional graphs of quadratic polynomials. These results confirm that many (but not all, see below) basic characteristics of these graphs, except for the total number of inner nodes, resemble those generated by random maps, that have been analysed by [8]. A probabilistic model of the distribution of cycles for functional graphs generated by polynomials (and more generally, by rational functions) has been also developed and numerically verified in [3]. Here, besides cycle lengths, we also examine other characteristics of functional graphs generated by quadratic polynomials, such as the number of connected components and the distribution of their sizes, for example. Furthermore, these numerical results exhibit some interesting statistical properties of the graphs \( G_f \) for which either there is no model in the setting of random graphs, or they deviate, in a regular way, from such a model.

We also note that the periodic structure of functional graphs associated with monomial maps \( x \mapsto x^d \) over finite fields and rings has been extensively studied (see [5, 10, 17, 20, 25, 26, 28, 29] and references
therein). However, these graphs are expected to be very different from those associated with generic polynomials.

In characteristic zero, graphs generated by preperiodic points of a map \( \psi \) (that is, by points that lead to finite orbits under iterations of \( \psi \)), have also been studied, (see, for example, \([7, 9, 22, 23, 24]\)).

We note that throughout the paper all implied constants in “\( O \)” symbols are absolute.

2. Bounds on the number of distinct functional graphs of polynomials

2.1. Upper bound. To estimate \( N_d(q) \) from the above, we use an idea of Bach and Bridy \([2]\) which is based on the observation that for any polynomial automorphism \( \psi \) the composition map \( \psi^{-1} \circ f \circ \psi \) has the same functional graph as \( f \). So the idea is to show that for any \( d \) there exists a rather small set of polynomials \( F_d \) such that for any polynomial \( f \in \mathbb{F}_q[X] \) of degree \( d \) there is a polynomial automorphism \( \psi \) such that \( \psi \circ f \circ \psi^{-1} \in F_d \). Then we have \( N_d(q) \leq \#F_d \). To construct the set \( F_d \) we introduce a group of certain transformations on the set of polynomials and show that

- polynomials in each orbit generate isomorphic graphs;
- each orbit is sufficiently long;
- most of the orbits are of the size of the above group.

This approach has been used in \([2]\) for \( d = 2 \) and even \( q = 2^n \), in which case it is especially effective and leads to the bound

\[
N_2(2^n) = \exp \left( O \left( \frac{n}{\log \log n} \right) \right) = q^{O(1/\log \log \log q)}.
\]

For general pairs \((d, q)\) this approach loses some of its power but still leads to nontrivial results, uniformly in both \( d \) and \( q \).

**Theorem 1.** For any \( d \) and \( q \), we have

\[
N_d(q) = O(q^{d-1}).
\]

**Proof.** For \( \mu \in \mathbb{F}_q \) and \( \lambda \in \mathbb{F}_q^* \), we define the automorphism

\[
\varphi_{\lambda, \mu} : X \mapsto \lambda X + \mu
\]

with \( \varphi_{\lambda, \mu}^{-1} : X \mapsto \lambda^{-1}(X - \mu) \). We verify that for a polynomial

\[f(X) = \sum_{j=0}^{d} a_j X^j \in \mathbb{F}_q[X], \quad \deg f = d,\]
we have
\[
\varphi_{\lambda,\mu}^{-1} \circ f \circ \varphi_{\lambda,\mu}(X) = \lambda^{-1}(f(\lambda X + \mu) - \mu)
\]
\[
= \sum_{j=0}^{d} A_j(a_d, \ldots, a_j; \lambda, \mu)X^j,
\]
for some coefficient \(A_j(a_d, \ldots, a_j; \lambda, \mu) \in \mathbb{F}_q\), \(j = 0, \ldots, d\). For any polynomial
\[
F(X) = \sum_{j=0}^{d} A_jX^j \in \mathbb{F}_q[X], \quad \text{deg } F = d,
\]
there are at most \(d \cdot \gcd(d-1, q-1)\) pairs \((\lambda, \mu) \in \mathbb{F}_q^* \times \mathbb{F}_q\) with \(\varphi_{\lambda,\mu}^{-1} \circ f \circ \varphi_{\lambda,\mu} = F\). Indeed, there are at most \(\gcd(d-1, q-1)\) values of \(\lambda \in \mathbb{F}_q^*\) with
\[
A_d(a_d; \lambda, \mu) = \lambda^{d-1}a_d = A_d.
\]
After this, taking \(X = 0\), we get the equation
\[
\lambda^{-1}(f(\mu) - \mu) = F(0),
\]
which is a nontrivial polynomial equation of degree \(d\) and this defines \(\mu\) in no more than \(d\) ways.

Therefore, for any \(f\), the orbit
\[
\text{Orb } f = \{\varphi_{\lambda,\mu}^{-1} \circ f \circ \varphi_{\lambda,\mu} : (\lambda, \mu) \in \mathbb{F}_q^* \times \mathbb{F}_q\}
\]
is of size
\[
\#\text{Orb } f \geq \frac{q(q-1)}{d \cdot \gcd(d-1, q-1)}.
\]
In particular, we immediately obtain
\[
N_d(q) < d \cdot \gcd(d-1, q-1)q^{d-1} < d^2q^{d-1}.
\]
We now estimate the number \(E_d(q)\) of polynomials \(f \in \mathbb{F}_q[X]\) of degree \(\text{deg } f = d\) for which
\[
\#\text{Orb } f < q(q-1).
\]
Note that (5) implies that \(\varphi_{\lambda,\mu}^{-1} \circ f \circ \varphi_{\lambda,\mu}(X) = f(X)\) for some pair \((\lambda, \mu) \in \mathbb{F}_q^* \times \mathbb{F}_q \setminus \{(1, 0)\}\), or \(f \circ \varphi_{\lambda,\mu}(X) = \varphi_{\lambda,\mu} \circ f(X)\), which is equivalent to
\[
f(\lambda X + \mu) = \lambda f(X) + \mu.
\]
Comparing the coefficients at the corresponding powers of \(X\) we derive
\[
\lambda^{d-1} = 1.
\]
First, consider the pairs \((1, \mu)\) with \(\mu \neq 0\). Comparing the coefficients of \(X^{j-1}\) in \(f(X + \mu)\) and \(f(X) + \mu\) for every \(j = 1, \ldots, d\), we obtain that
\[
d a_d = 0.
\]
Thus, \(p \mid d\), where \(p\) is the characteristic of \(\mathbb{F}_q\). We also obtain relations of the form
\[
j a_j \mu = F_j(a_d, \ldots, a_{j+1}, \mu), \quad j = 0, \ldots, d,
\]
for some polynomials
\[
F_j \in \mathbb{F}_q[Z_d, \ldots, Z_{j+1}, V],
\]
where the case \(j = d\) we have \(F_d(V) = 0\), that corresponds (7). In particular, for every \(j = 0, \ldots, d\) with \(\gcd(j, p) = 1\), we see that \(a_j\) is uniquely defined by \(a_d, \ldots, a_{j+1}, \mu\). Since \(\mu\) takes at most \(q - 1\) values, and \(p \mid d\), we get that
\[
\#\{f \in \mathbb{F}_q[X] : \deg f = d, f(X + \mu) = f(X) + \mu, \text{ for some } \mu \in \mathbb{F}_q^*\} \leq \begin{cases} 0, & \text{if } \gcd(d, p) = 1, \\ (q - 1)q^{d/p+1}, & \text{if } \gcd(d, p) = p. \end{cases}
\]
Assume now that \(\lambda \neq 0, 1\). We see that for every \(j = 0, \ldots, d\) there are polynomials
\[
G_j \in \mathbb{F}_q[Z_d, \ldots, Z_{j+1}, U, V]
\]
such that
\[
a_j(\lambda^j - \lambda) = G_j(a_d, \ldots, a_{j+1}, \lambda, \mu).
\]
In particular
\[
G_j(Z_d, \ldots, Z_{j+1}, 1, V) = F_j(Z_d, \ldots, Z_{j+1}, V), \quad j = 0, \ldots, d.
\]
Since \(\lambda \neq 0, 1\), we see from (6) that for every \(j\) with \(\gcd(j - 1, d - 1) = 1\) we have \(\lambda^j \neq \lambda\) and thus \(a_j\) is uniquely defined via \(a_d, \ldots, a_{j+1}, \lambda, \mu\).

Since \(\lambda \neq 1\), satisfying (8), can take at most \(\gcd(q - 1, d - 1) - 1 < d - 1\) values, \(\mu\) can take at most \(q\) values, so together with (8) we obtain
\[
E_d(q) < \begin{cases} (d - 1)q^{d-\varphi(d-1)}, & \text{if } \gcd(d, p) = 1, \\ (d - 1)q^{d-\varphi(d-1)} + (d - 1)q^{d/p+1}, & \text{if } \gcd(d, p) = p, \end{cases}
\]
where \(\varphi(d)\) is the Euler function of \(d\).

It is also clear that the number of orbits of size \(\#\text{Orb } f = q(q - 1)\) is at most \(q^{d-1}\). Thus, we derive
\[
N_d(q) \leq q^{d-1} + E_d(q).
\]
Clearly we can assume that \(d \geq 8\) as otherwise the result follows from (4).
We now note that for \( d \geq 8 \) we have \( \varphi(d - 1) \geq 4 \) and also that \( d/p \leq d/2 < d - 3 \). Thus for \( d \geq 8 \) the number of incomplete orbits is at most \( 2q^{d-2} \). Using (9) and (10), we conclude the proof. \( \square \)

2.2. Lower bound. Here we give a lower bound on \( N_d(q) \) in the case of \( \gcd(d, q - 1) \geq 2 \). In particular, this bound shows that the bound of the strength of (1) does not hold for fields of odd characteristic.

The idea is based on the following observation. Let \( H_a \) be the functional graph of \( f_a(X) = X^d + a \in \mathbb{F}_q[X] \) with \( a \in \mathbb{F}_q^* \). We note that the node \( a \) is the only node with the in-degree 1. We now define the iterations of \( f_a \)

\[
f_a^{(0)}(X) = X \quad \text{and} \quad f_a^{(k)}(X) = f_a\left(f_a^{(k-1)}(X)\right), \quad k = 1, 2, \ldots.
\]

and consider the path of length \( J \)

\[
a = f_a^{(0)}(a) \rightarrow f_a(a) = f_a^{(1)}(a) \rightarrow \cdots \rightarrow f_a^{(J)}(a)
\]

originating from \( a \). Let

\[
e = \gcd(d, q - 1) \geq 2.
\]

Then each node of this path has \( e - 1 \) edges towards it from \( \gamma f_a^{(j)}(a) \), where \( \gamma \) runs through the elements of the set

\[
\Gamma_e^* = \Gamma_e \setminus \{1\},
\]

where

\[
\Gamma_e = \{\gamma \in \mathbb{F}_q : \gamma^e = 1\}.
\]

Finally, we observe that \( \gamma f_a^{(j)}(a) \) is an inner node if and only if the equation

\[
z^d + a = \gamma f_a^{(j)}(a)
\]

has a solution.

We now note that if two graphs \( H_a \) and \( H_b \) are isomorphic then, since \( a \) and \( b \) are unique nodes with the in-degree 1 in \( H_a \) and \( H_b \), respectively, the paths originating at \( a \) and \( b \), and their neighbours have to be isomorphic too.

For \( j = 1, 2, \ldots \), we define \( \eta_j(a) \) as the number of \( \gamma \in \Gamma_e^* \) for which \( \gamma f_a^{(j)}(a) - a \) is the \( e \)th power nonresidue. Thus, \( \eta_j(a) \) is the number of leaves amongst the nodes \( \gamma f_a^{(j)}(a), \gamma \in \Gamma_e^* \).

Therefore, for any \( J \), the number of distinct vectors

\[
(\eta_1(a), \ldots, \eta_J(a)), \quad a \in \mathbb{F}_q^*,
\]

gives a lower bound on \( N_d(p) \).

In order to estimate the number of distinct vectors (12) we need several technical statements.
Let us consider the sequences of polynomials
\[ F_0(X) = X \quad \text{and} \quad F_k(X) = F_{k-1}(X)^d + X, \quad k = 1, 2, \ldots, \]
and also
\[ G_{k,\gamma}(X) = \gamma F_k(X) - X. \]
We now investigate some arithmetic properties of polynomials \( G_{k,\gamma} \) which we present in larger generality than we actually need for our purposes.

**Lemma 2.** For any positive integers \( k \) and \( h \) and \( \gamma, \delta \in \Gamma_e \), we have
\[ G_{k+h,\gamma} \equiv G_{h,\gamma} \pmod{G_{k,\delta}}. \]

**Proof.** We fix \( \gamma, \delta \in \Gamma_e \) and prove the desired statement by induction on \( h = 1, 2, \ldots. \)

We note that for \( \delta \in \Gamma_e \) we have
\[ F_k^d = (\delta^{-1} (G_{k,\delta}(X) + X))^d = (G_{k,\delta}(X) + X)^d. \]
For \( k = 1 \) we have \( G_{1,\gamma} = \gamma X^d + (\gamma - 1)X \). Hence, using (13), we derive
\[ G_{k+1,\gamma} = \gamma (F_k^d + X) - X = \gamma (G_{k,\delta}(X) + X)^d + (\gamma - 1)X \equiv \gamma X^d + (\gamma - 1)X \equiv G_{1,\gamma} \pmod{G_{k,\delta}} \]
so the desired congruence holds for \( h = 1 \).

Now assume it also holds for \( h = \ell \). Then
\[ G_{k+\ell+1,\gamma} \equiv \gamma (G_{k+\ell,\gamma} + X)^d - X \equiv \gamma (G_{\ell,\gamma} + X)^d - X \equiv G_{\ell+1,\gamma} \pmod{G_{k,\delta}}, \]
which implies the desired result. \( \square \)

**Lemma 3.** For any positive integers \( k \) and \( m \) we have
\[ \gcd(G_{k,\gamma}, G_{m,\gamma}) = G_{\gcd(k,m),\gamma}. \]

**Proof.** If \( k = m \), then there is nothing to prove. Otherwise we note that for \( m > k \), Lemma 2 implies \( G_{m,\gamma} \equiv G_{m-k,\gamma} \pmod{G_{k,\gamma}} \). Thus
\[ \gcd(G_{k,\gamma}, G_{m,\gamma}) = \gcd(G_{k,\gamma}, G_{m-k,\gamma}), \]
which immediately implies the desired result. \( \square \)

Now, from Lemma 3 we derive that for \( d = 2 \) products of polynomials \( G_{j,\gamma} \) over distinct integers are not perfect squares.

As usual, we use \( \overline{\mathbb{F}}_q \) to denote the algebraic closure of \( \mathbb{F}_q \).
Lemma 4. For $d = e = 2$ and any set $J \subseteq \{1, \ldots, J\}$, we have

$$\prod_{j \in J} G_{j,-1} \neq P^2$$

for any polynomial $P \in \overline{\mathbb{F}}_q[X]$.

Proof. Assume that $m$ is the largest element of $J$. The cases $m = 1$ and $m = 2$ can be verified by direct calculations.

Now, we assume that $m \geq 3$. Put $S = \{2, \ldots, m-1\}$. It suffices to show that $G_{k,-1}(X)$ has a simple root which is not a root of $Q_{m-1}(X) = \prod_{j=2}^{m-1} G_{j,-1}(X)$.

By Lemma 3, the distinct roots of $\gcd(G_{m,-1}(X), Q_{m-1}(X))$ are to be found among the distinct roots of

$$\prod_{k \leq m-1} \gcd(G_{m,-1}(X), G_{k,-1}(X)) = \prod_{k \leq m-1} G_{\gcd(m,k)-1}(X),$$

and the distinct roots of this last polynomial are the same as the distinct roots of the polynomial

$$R_m(X) = \prod_{k | m, k < m} G_{k,-1}(X)$$

which is of degree

$$\deg R_m(X) = \sum_{k | m, k < m} 2^k \leq \sum_{1 \leq k \leq m/2} 2^k < 2^{\lfloor m/2 \rfloor + 1}.$$

We now estimate the number of multiple roots of $G_{m,-1}(X)$. Clearly,

$$G_{m,-1}(X) = - (F_{m-1}(X)^2 + 2X)$$

$$= - (2F_{m-1}(X)F_{m-1}(X)^{\prime} + 2) = - 2(F_{m-1}(X)F_m(X)^{\prime} + 1).$$

Observe that $G_{m,-1}(X)^{\prime}$ is not the zero polynomial because for $k \geq 2$, we have $F_k(X) \equiv X + X^2 \pmod{X^3}$, therefore

$$F_{m-1}(X)F_{m-1}(X)^{\prime} + 1 \equiv X(1 + 2X) + 1 \equiv 1 + X \pmod{X^2}.$$

Now, let $z \in \overline{\mathbb{F}}_q$ be a root of $G_{m,-1}(X)$ and of $G_{m,-1}(X)^{\prime}$. Then

$$F_{m-1}(z)F_{m-1}(z)^{\prime} = -1 \quad \text{and} \quad F_{m-1}(z)^2 + z = 0.$$

Hence,

$$-zF_{m-1}(z) = F_{m-1}(z)^2 F_{m-1}(z)^{\prime} = - F_{m-1}(z),$$
therefore $z$ is a root of $F_{m-1}(X) - XF'_{m-1}(X)$. As we have seen, we have

$$F_{m-1}(X) - XF'_{m-1}(X) \equiv X + X^2 - X(1 + 2X) \pmod{X^3}$$

$$\equiv -X^2 \pmod{X^3},$$

so it is not the zero polynomial. Hence, the number of its roots with their multiplicities does not exceed

$$\deg (F_{m-1}(X)F_{m-1}(X)' + 1) \leq 2^{m-1}.$$ 

So, if all roots of $G_{m-1}(X)$ are either multiple or roots of $Q_{m-1}(X)$, then

$$2^m \leq 2^{m-1} + 2^{|m/2|},$$

which is false for all $m \geq 3$. \qed

In the case when $d > 2$ we can study the arithmetic structure of the polynomials $G_{k,\gamma}$ by using the theorem of Mason \[21\] that gives a polynomial version of the $ABC$-conjecture (see also \[27\]).

For a polynomial $F \in \mathbb{F}_q[X]$ we use $\text{rad}(F)$ to denote the product of all monic irreducible divisors of $F$.

**Lemma 5.** Let $A, B, C$ be nonzero polynomials in $\mathbb{F}_q[X]$ satisfying $A + B + C = 0$ and $\gcd(A, B, C) = 1$. If $\deg A \geq \deg \text{rad}(ABC)$, then $A' = 0$.

We are now ready to prove our main technical statement that we use for $d \geq 3$ which asserts that some general products of polynomials $G_{k,\gamma}$ over distinct integers are not perfect $e$th powers.

**Lemma 6.** For $d \geq 3$, $e = \gcd(d, q - 1) \geq 2$, any $J \geq 3$ and any collection of integers

$$A = \{\alpha_{j,\gamma} \in \{0, \ldots, e - 1\} : 3 \leq j \leq J, \gamma \in \Gamma_e^*\},$$

not all equal to zero, we have

$$\prod_{j=1}^{J} \prod_{\delta \in \Gamma_e^*} G_{j,\gamma}^{\alpha_{j,\gamma}} \neq P^e$$

for any polynomial $P \in \mathbb{F}_q[X]$.

**Proof.** Clearly we observe that for $j = 1, 2, \ldots$ we have $X \mid G_{j,\gamma}$. Hence, setting

$$G_{j,\gamma}^* = G_{j,\gamma}/X, \quad j = 1, 2, \ldots, \gamma \in \Gamma_e,$$

we see that for distinct $\gamma, \delta \in \Gamma_e$, we have

$$\gcd(G_{j,\gamma}^*, G_{j,\delta}^*) = 1.$$  

(14)
Therefore, we see from Lemma 2 that for any positive integers \( k \) and \( h \) and \( \gamma \in \Gamma \), we have

\[
\deg \gcd \left( G_{k+h,\gamma}^*, \prod_{\delta \in \Gamma} G_{k,\delta} \right) = \deg \gcd \left( G_{k,\gamma}^*, \prod_{\delta \in \Gamma} G_{k,\delta}^* \right)
\]

\[
= \deg \gcd \left( G_{h,\gamma}^*, \prod_{\delta \in \Gamma} G_{k,\delta}^* \right) \leq d^h - 1.
\]

(15)

We now note that applying Lemma 5 with \( A = -G_{k,\gamma}^* \), \( B = \gamma F_{k-1}^d / X \) and \( C = \gamma - 1 \), and taking into account that \( X \mid F_{k-1} \), for \( k = 1, 2, \ldots \), we derive

\[
d^k - 1 < \deg \rad \left( G_{k,\gamma}^* F_{k-1}^d / X \right) = \deg \rad \left( G_{k,\gamma} F_{k-1} \right)
\]

\[
\leq \deg \rad \left( G_{k,\gamma} \right) + d^{k-1}.
\]

Thus,

\[
\rad \left( G_{k,\gamma} \right) \geq (d - 1)d^{k-1}, \quad k = 1, 2, \ldots; \gamma \in \Gamma_e.
\]

Denote

\[
Q_{J,A} = \prod_{j=3}^{J} \prod_{\gamma \in \Gamma_e^*} G_{j,\gamma}^\alpha_{j,\gamma},
\]

and assume that \( Q_{J,A} = P^e \) for some \( P \in \overline{F}_q[X] \). Let \( k \geq 3 \) be the largest \( j \in \{3, \ldots, J\} \) for which one of the integers \( \alpha_{j,\gamma}, \gamma \in \Gamma_e^* \) is positive.

\[
R_{J,A} = \prod_{\gamma \in \Gamma_e^*} G_{k,\gamma}^\alpha_{k,\gamma}.
\]

Denote \( \tilde{\alpha}_{k,\gamma} = e - \alpha_{k,\gamma} \) if \( \alpha_{k,\gamma} \neq 0 \) and \( \tilde{\alpha}_{k,\gamma} = 0 \) otherwise. Let

\[
\tilde{R}_{J,A} = \prod_{\gamma \in \Gamma_e^*} G_{k,\gamma}^{\tilde{\alpha}_{k,\gamma}}.
\]

Considering, if necessary, \( \tilde{R}_{J,A} \) we can always assume that

\[
\alpha = \min_{\gamma \in \Gamma_e^*} \{\alpha_{k,\gamma} : \alpha_{k,\gamma} > 0\} \leq e/2.
\]

We now fix some \( \delta \in \Gamma_e^* \) with \( \alpha_{k,\delta} = \alpha, 1 \leq \alpha \leq e/2 \). Recalling (14), we see that if a polynomial \( H \in \overline{F}_q[X] \) is such that \( HG_{k,\delta}^* \) is a perfect \( e \)th power, then

\[
\rad \left( G_{k,\delta}^* \right)^e \mid \rad \left( R_{J,A} \right)^e \mid H R_{J,A}.
\]

Therefore,

\[
\deg H \left( G_{k,\delta}^* \right)^{\alpha_{k,\delta}} \geq e \deg \rad \left( G_{k,\delta}^* \right) \geq e(d - 1)d^{k-1}.
\]
Hence,
\[ \deg H \geq c(d - 1)d^{k-1} - \alpha_{k,\delta} \deg G_{k,\gamma}^* > \frac{e}{2}(d - 2)d^{k-1}, \]
which yields the lower bound
(16) \[ \deg \gcd \left( G_{k,\delta}^* \alpha_{k,\delta} k-1 \prod_{j=3}^{k-1} G_{j,\gamma}^{\alpha_{j,\gamma}} \right) > \frac{e}{2}(d - 2)d^{k-1}. \]

We now write
\[
\deg \gcd \left( G_{k,\delta}^* \alpha_{k,\delta} k-1 \prod_{j=3}^{k-1} G_{j,\gamma}^{\alpha_{j,\gamma}} \right)
\leq \deg \gcd \left( G_{k,\delta}^* \alpha_{k,\delta} k \prod_{j=3}^{k-2} G_{j,\gamma}^{\alpha_{j,\gamma}} \right) + \sum_{j=3}^{k-2} \sum_{\gamma \in \Gamma^*_{e}} \alpha_{j,\gamma} \deg G_{j,\gamma}
\leq \deg \gcd \left( G_{k,\delta}^* \alpha_{k,\delta} k-1 \prod_{j=3}^{k-2} G_{j,\gamma}^{\alpha_{j,\gamma}} \right) + (e - 1)^2 \sum_{j=3}^{k-2} \deg G_{j,\gamma}.
\]
Using (15) (with \( h = 1 \)) we now derive
\[
\deg \gcd \left( G_{k,\delta}^* \alpha_{k,\delta} k-1 \prod_{j=3}^{k-2} G_{j,\gamma}^{\alpha_{j,\gamma}} \right) \leq (e - 1)(d - 1).
\]
Since
\[
\sum_{j=3}^{k-2} \deg G_{j,1} \leq \frac{d^{k-1} - 1}{d - 1}
\]
we obtain
(17) \[
\deg \gcd \left( G_{k,\delta}^* \alpha_{k,\delta} k-1 \prod_{j=3}^{k-2} G_{j,\gamma}^{\alpha_{j,\gamma}} \right)
\leq (e - 1)(d - 1) + (e - 1)^2 \frac{d^{k-1} - 1}{d - 1}.
\]

It is now easy to verify that (17) contradicts (16). The result now follows. \( \square \)

Let \( X_e \) be the group of all multiplicative characters of \( \mathbb{F}_q^* \) of order \( e \), that is, characters \( \chi \) with \( \chi^e = \chi_0 \), where \( \chi_0 \) is the principal character. We also define \( X_e^* = X_e \setminus \{ \chi_0 \} \).

We recall the following special case of the Weil bound of character sums (see [12, Theorem 11.23]).
Lemma 7. For any polynomial $Q(X) \in \mathbb{F}_q[X]$ with $Z$ distinct zeros in $\mathbb{F}_q$ and which is not a perfect $e$th power in the ring of polynomials over $\mathbb{F}_q$, and $\chi \in \mathcal{X}_e^*$, we have

$$\left| \sum_{a \in \mathbb{F}_q} \chi(Q(a)) \right| \leq Zq^{1/2}.$$  

We are now ready to establish a lower bound on $N_d(q)$:

Theorem 8. For any $d \geq 2$ and $e = \gcd(d, q - 1) \geq 2$, we have

$$N_d(q) \geq q^{\rho_{d,e} + o(1)}$$

as $q \to \infty$, where

$$\rho_{d,e} = \frac{1}{2(e - 1 + \log d/\log e)}.$$  

Proof. We define $J$ by the inequalities

$$(de^{e-1})^J \leq q^{1/2}/\log q < (de^{e-1})^{J+1}.$$  

For each $j = 3, \ldots, J$ and $\gamma \in \Gamma_*$ we choose a representative $\sigma_{j,\gamma}$ of the factor group $\mathbb{F}_q^*/\Gamma_e$ and consider the collection

$$\mathcal{S} = \{ \sigma_{j,\gamma} : j = 3, \ldots, J, \gamma \in \Gamma_* \}.$$  

Let $A(\sigma)$ denote the number of $a \in \mathbb{F}_q^*$ such that

$$\gamma f_a^{(j)}(a) - a \in \sigma_{j,\gamma} \Gamma_e, \quad j = 3, \ldots, J, \gamma \in \Gamma_*.$$  

Clearly if for any $\sigma$ as in the above we have

(18) \hspace{1cm} A(\sigma) > 0  

then the vector (12) takes all $e^{J-2}$ possible values and thus we have

$$N_d(q) \geq e^{J-2},$$

which implies the desired result. Furthermore, let $\chi$ be a primitive character of order $e$ (that is, a generator of $\mathcal{X}_e$). We can now express $A(\sigma)$ via the following character sums

$$A(\sigma) = \sum_{a \in \mathbb{F}_q^*} \frac{1}{e^{(e-1)(J-2)}} \prod_{j=3}^{J-1} \prod_{\gamma \in \Gamma_e} \sum_{\alpha_{j,\gamma} = 0}^{e-1} \chi^{\alpha_{j,\gamma}}(G_{j,\gamma}(a)/\sigma_{j,\gamma})$$

Expanding the product, and changing the order of summation we obtain $e^{(e-1)(J-2)}$ character sums parametrized by different choices of $\alpha_{j,\gamma} \in \{0, \ldots, e-1\}, j = 3, \ldots, J, \gamma \in \Gamma_*$.  

The term corresponding to the choice $\alpha_{j,\gamma} = 0$, $j = 3, \ldots, J$, and $\gamma \in \Gamma_*$, is essentially the sum of principal characters with a polynomial
of degree $O(d^j)$ and so it is equal $q + O(d^j)$ (the error term $O(d^j)$ accounts for the zeros of this polynomial).

For other terms, we see from and using Lemma 4 (if $d = 2$) and Lemma 6 (if $d \geq 3$), that Lemma 7 applies to each of them. Hence, we obtain

$$A(\sigma) = \frac{q}{e(e-1)(J-2)} + O(d^j q^{1/2}),$$

which implies (18) for a sufficiently large $q$ and the above choice of $J$. \hfill \Box

We remark that

$$\max_{d, e | d} \rho_{d, e} = \rho_{2, 2} = 1/4.$$

3. ISOMORPHISM TESTING OF FUNCTIONAL GRAPHS

3.1. Preliminaries. In this section, we give an isomorphism testing algorithm for quadratic polynomials that is linear (in time and memory). We also extend this isomorphism testing algorithm from quadratic polynomials to any arbitrary function with only a slight increase in the time complexity. We first introduce several graph related notations.

3.2. Notations and graph input size. For any function $f : \mathbb{F}_q \to \mathbb{F}_q$, the functional graph of $f$ is a directed graph $G_f$ on $n = q$ vertices labelled by elements of $\mathbb{F}_q$ and exactly $n = q$ directed edges (where there is an edge from $u$ to $v$ if and only if $f(u) = v$). Hence, the size of the input that should be considered for efficient isomorphism testing is linear in the size of an adjacency list (that is, $O(n \log n)$), rather than an adjacency matrix (that is, $O(n^2)$) In the following, we aim at providing an algorithm with complexity linear in the size of the input, for both memory and time complexities.

Given two functions $f$ and $h$, we denote the functional graph $G_f$ of $f$ as $G$ and the functional graph $G_h$ of $h$ as $H$. Given a functional graph $G$, we collect its connected components of the same size in the sets $C_i^G$ with $1 \leq i \leq s^G$, where $s^G$ is the total number of distinct sizes of components of $G$. For each set $C_i^G$ we denote the size of the components in the set by $k_i^G$ and the size of the set itself by $c_i^G = \#C_i^G$. Let

$$k^G = \max_{1 \leq i \leq s^G} k_i^G \quad \text{and} \quad c^G = \max_{1 \leq i \leq s^G} c_i^G.$$ 

When there is no ambiguity, we omit the superscript $G$. For convenience we denote the in-degree of a vertex $v$ as $d^- (v)$ and the corresponding in-neighbourhood as $N^- (v)$. Since the out-degree of any vertex is 1, each connected component $C$ in a functional graph has exactly one cycle
(which may be a self-loop), which we denote $\text{Cyc}(C)$. Each vertex $v$ on this cycle may subtend a tree where all edges are directed towards $v$, and we consider $v$ as the root of the tree.

3.3. **Isomorphism testing of functional graphs of quadratic polynomials.** We now present our meta-algorithm to test the isomorphism. It comprises three phases:

**Phase 1:** Given two functional graphs $G$ and $H$, we first identify the connected components in each graph, and the associated cycle and trees in each component.

**Phase 2:** For each component we produce a canonical encoding.

**Phase 3:** Finally we construct a prefix tree (formally a trie [16]), using the encodings of $G$, noting at each vertex of the trie the number of code strings that terminate at that vertex. Then for each encoded component of $H$ we match the code string against the trie, and decrement the counter at the appropriate trie vertex.

If all counters are zero after this is complete, the two graphs are isomorphic.

The first phase is achieved by combining a cycle detection algorithm and depth-first search, as laid out in Algorithm 1.

**Algorithm 1 Identification of Connected Components**

1: while unassigned vertices remain do
2:   Pick an unassigned vertex $v$.
3:   Perform Floyd’s cycle detection algorithm starting at $v$.
4:   for each cycle vertex $u$ do
5:       Perform a depth-first search on the tree attached at $u$.
6:   end for
7: end while

The cycle detection algorithm can be done in linear time and space (of the size of each connected component) with Floyd’s algorithm [15] using only two pointers. The depth-first search is a simple pre-order traversal of the tree and thus only requires linear time and space [14]. In total, the complexity of the first phase is thus linear in time and space with the size of the graph. Note that this phase is independent of the function $f$ (it has linear complexity for any function $f$), leading to the following lemma.

**Lemma 9.** For any functional graph $G$ of $n$ vertices, Algorithm [11] identifies all Connected Components and has linear time and memory complexities.
On the other hand, the second phase depends on the nature of the function. In this section, we focus on quadratic polynomials which provide an especially interesting case when considering the isomorphism of functional graphs. A \( k \)-ary tree is full if every non-leaf vertex has exactly \( k \) children. As a quadratic polynomial can have at most one repeated root, the functional graph is almost a full binary tree. This allows certain savings in building a canonical labelling of the graph. We note that if there is a repeated root, we can deal with the containing component specially by noting which vertex has one child, and adding a dummy second child, then in the two graphs under consideration the dummy vertices must be matched to each other in any isomorphism.

We recall that the number of different binary trees on \( n \) nodes is the \( n \)-th Catalan number. For large \( n \), this is about \( 4^n \). Thus, we need at least about \( 2n \) bits to encode such a tree. However, if the binary tree is full, there are only \( 2^n \) different trees, and only \( n \) bits are required. Our canonical labelling matches this bound and extends the labelling to include the cycle without using extra bits.

To produce the canonical labelling of a functional graph derived from a quadratic polynomial we employ Algorithms 2 and 3, where \( \varepsilon \) is the empty string, \( s_i \) is the string \( s \) after circular shift to the bit position \( i \), and \( \text{val}(s) \) is the interpretation of the string \( s \) as a number. In the description of the algorithms we denote string concatenation by \( \circ \).

\begin{algorithm}[H]
\caption{CanonicalLabelling}
\begin{algorithmic}
\Require component \( C \)
\State \( s := \varepsilon \)
\For {each vertex \( v \) in Cyc(\( C \))}
\State \( s := s \circ \text{LABEL}(v) \)
\EndFor
\State \( \text{max} := \text{val}(s_1) \)
\State \( \text{maxpos} := 1 \)
\For {\( i := 2 \) to \# Cyc(\( C \))}
\If {\( \text{val}(s_i) > \text{max} \)}
\State \( \text{max} := \text{val}(s_i) \)
\State \( \text{maxpos} := i \)
\EndIf
\EndFor
\State \text{return} \( s_{\text{maxpos}} \)
\end{algorithmic}
\end{algorithm}

Algorithm 2 is run on each component in turn and produces a canonical label for the component by applying Algorithm 3, that is, function \( \text{LABEL}(v) \), to each tree rooted on a vertex of the component’s cycle.
Algorithm 3 \text{LABEL}(v)

\begin{algorithm}
\begin{algorithmic}[1]
\Require vertex $v$.
\State \textbf{if} $d^-(v) = 0$ \textbf{then}
\State \hspace{1em} \Return “0”
\State \textbf{else}
\State \hspace{1em} \textbf{left} := \text{LABEL}(\text{left}(v))
\State \hspace{1em} \textbf{right} := \text{LABEL}(\text{right}(v))
\State \hspace{1em} \textbf{if} left < right \textbf{then}
\State \hspace{2em} \Return $1 \circ \text{right} \circ \text{left}$
\State \hspace{1em} \textbf{else}
\State \hspace{2em} \Return $1 \circ \text{left} \circ \text{right}$
\State \textbf{end if}
\State \textbf{end if}
\end{algorithmic}
\end{algorithm}

(Figure 1 gives an example), concatenating these labels in the order given by the cycle, then shifting circularly the concatenated label to begin with the cycle vertex that gives the greatest value. Note that if $t$ such vertices exist (that is, $t$ possible circular shifts leading to the greatest value), the component must have at least a $t$-fold symmetry of rotation around the cycle. Thus, this maximal orientation of the cycle is unique up to automorphism.

Algorithm 3 encodes a full, rooted, binary tree by assigning each vertex a single bit: 1 if the vertex is internal, 0 if it is a leaf. The label is then recursively built by concatenating the assigned bit of the current vertex $v$ to the lexicographically sorted labels of its left child, left$(v)$, and right child, right$(v)$. In effect this produces a traversal of the tree where we traverse higher weight subtrees first. As each vertex contributes one bit to the label, the total length of the label is $k$ bits for a component of size $k$ and thus $n$ bits for the entire graph.

Lemma 10. For any functional graph $G$ of a quadratic polynomial over $\mathbb{F}_q$ with $n = q$ vertices, Algorithms 2 and 3 build an $n$-bit size canonical labelling of $G$ and have linear time and memory complexities.

Proof. From the description of the traversal process in Algorithms 2 and 3 it is clear that each node $v$ in the tree is associated with a canonical coding \text{LABEL}(v) of size $|T_v|$ bits, where $T_v$ is the subtree with root $v$. All leaves are labelled with 0, and the canonical label of the whole tree $T_v$ has exactly $k = |T_v|$ bits. The overall memory requirement remains linear: both child labels can be discarded, on the fly, as a parent label is generated.
Figure 1. An example binary tree labelled with the canonical coding generated at each level by Algorithm 3.

The worst-case time complexity is slightly more involved, a (lexicographic) sorting is required at each internal node. More precisely, each internal node $v$ requires a number of (lexicographic) bit comparisons $\text{comp}(v)$ equal to the size of the smallest label among both children:

$$
\text{comp}(v) = \min(\vert \text{LABEL(left}(v) \vert, \vert \text{LABEL(right}(v) \vert)
$$

(19)

$$
= \min(\vert T_{\text{left}}(v) \vert, \vert T_{\text{right}}(v) \vert) \leq \left\lfloor \frac{(\vert T(v) \vert - 1)}{2} \right\rfloor.
$$

Hence, we see that the worst case for the number of bit comparisons occurs when each subtree is balanced, that is when the full binary tree is complete. Using this simple recurrence, it is easy to see that this leads to less than $n \log n$ bit comparisons for any binary tree of size $n$, which is linear in the size of the input and completes the proof. $\square$

Note that to finally test the isomorphism between two graphs $G$ and $H$, it remains to compare the canonical labellings of each connected components of each graph with one another (Phase 3). A general brute-force approach (by comparing canonical labellings of connected components pair-wise) could be ineffective (as shown in the next section). To keep it linear in the size of the input, the third phase builds a trie (or prefix tree) using the encodings of the functional graph $G$ by inserting the canonical labelling of each connected components, obtained after Phase 2, one after the other. Each node in the trie is also equipped with a counter initialised to zero and incremented each time the node represents the terminating node of a newly inserted canonical labelling of a connected component. It then suffices to check that each
canonical labelling of each connected component of $H$ is represented
in the trie, decrementing the respective counter each time there is a
match. The two functional graphs are isomorphic if there is no mis-
mismatch for all canonical labellings of $H$ (all counters are zero after all
components have been considered), and are otherwise.

**Lemma 11.** For any functional graph $G$ and $H$, each with an $n$-bit
canonical labelling, Phase 3 tests their isomorphism by comparing the
canonical labelling of $G$ and $H$ and has linear time and memory com-
plexities.

**Proof.** It is easy to see that the trie built for the functional graph $G$ has
at most $n$ nodes. This case is only possible if all canonical labellings of
connected components are disjoints (that is, generate disjoints branches
in the tree). As more canonical labels overlap, less nodes are created. If
the labels match, the respective counter (and its size) are incremented,
but the cost of increasing the counter remains lower than the cost of
creating a distinct branch in the trie. Thus, the overall size remains
$O(n)$ in memory space. It is also easy to see that creating the initial
trie with the canonical labels of $G$ takes $O(n)$ time and memory, and
the same cost occurs for matching all canonical labels of $H$ (and may
stop before if the two graphs are not isomorphic). $\square$

Again it is interesting to note that the complexity of Phase 3 does
not depend on the type of functional graph but depends solely on the
size of the canonical labelling.

Combining Lemmas 9, 10 and 11 we obtain the following theorem.

**Theorem 12.** For any functional graphs $G$ and $H$ of quadratic func-
tions with $n$ vertices, Phases 1, 2 and 3 combined provide an isomorphic
test that has linear time and memory complexities.

It is also interesting to note that the trie built in Phase 3 provides
a canonical representation of size $O(n)$ for any functional graph of size
$n$. We exploit this property to present an algorithm to enumerate all
functional graphs corresponding polynomials of degree $d$ over $\mathbb{F}_q$ in
Section 3.5.

3.4. **General functional graph isomorphism.** Before extending the
algorithms of Section 3.3 to arbitrary functions, we first prove a simple
upper bound for Functional Graph Isomorphism for arbitrary functions
using standard techniques.

**Theorem 13.** For any functional graphs $G$ and $H$ of arbitrary func-
tions with $n$ vertices, there is an isomorphism test using standard al-
gorithms with $O(c_* n)$ time complexity, where $c_* = \max\{c_*^G, c_*^H\}$.
**Proof.** The graph isomorphism problem can be solved in time linear in the number of vertices for connected planar graphs \[11\] and (rooted) trees \[13\]. A simple approach we could apply to functional graphs would be to run Algorithm 1 (that build each connected component) and then compare the connected components of the two graphs pairwise, using the appropriate algorithm as a subroutine (for components with a cycle, we can use the planar graph algorithm, for components with a self loop, we can use the rooted tree algorithm where we treat the vertex with the self loop as the root). This involves at most \(\binom{n}{2}\) comparisons and thus gives an \(O(n^2)\) algorithm overall.

Using the sizes of the various components, we can refine this analysis slightly. Given two functional graphs \(G\) and \(H\), if we have the isomorphism \(G \cong H\) then \(s^G = s^H\) and for all \(i \in [1, s^G]\) we also have \(c_i^G = c_i^H\) and \(k_i^G = k_i^H\). (If the graphs are isomorphic, \(c_a = c_a^G = c_a^H\).) On the other hand, if one of these pairs of values disagree then \(G \not\cong H\). Then, denoting these common values as \(s\) and \(c_i, k_i, 1 \leq i \leq s\), clearly for both graphs we have

\[
\sum_{i=1}^{s} c_i k_i = n,
\]

where \(n\) is the order of the graphs.

Clearly we only need to compare components in the same size class. This gives a running time proportional to:

\[
\sum_{i=1}^{s} c_i^2 k_i \leq c^*_s \sum_{i=1}^{s} c_i k_i = c^*_s n,
\]

where \(c^*_s = \max_{i=1, \ldots, s} c_i\).

If each size class \(C_i\) is bounded, then this naive algorithm is linear in the number of vertices. In the general case however it is likely there are numerous components of the same size \[8\] thus possibly leading to a worst-case bound of \(O(n^2)\) time. Fortunately even in this case, as we now show that we can still solve the isomorphism problem with linear memory complexity and by increasing slightly the cost of building the canonical labels.

The challenge is that, in the general case, we cannot assume that the trees associated with each component are full, nor necessarily have any particular bound on the number of children (note that polynomials of degree \(d\) do however have at most \(d\) children in the trees).

For the general case we replace Algorithm \[3\] with Algorithms \[11\] and \[15\] and replace the call to Label in Algorithm \[2\] with a call to LeftLabel with the root vertex of the tree.
Algorithm 4 \textsc{LeftLabel}

\textbf{Require:} vertex $v$
\begin{itemize}
  \item[1:] $\text{label}_v := \varepsilon$
  \item[2:] $\text{labelSet} := \emptyset$
  \item[3:] \textbf{if} $v$ has a left child \textbf{then}
  \item[4:] $\text{label}_v := \mathit{1} \circ \textsc{LeftLabel}(\text{left}(v))$
  \item[5:] \textbf{else}
  \item[6:] $\text{label}_v := 10$
  \item[7:] \textbf{end if}
  \item[8:] \textbf{if} $v$ has a right child \textbf{then}
  \item[9:] $\text{labelSet} := \textsc{RightLabel}(\text{right}(v))$
  \item[10:] \textbf{end if}
  \item[11:] \text{labelSet} := \text{labelSet} \cup \{ \text{label}_v \}$
  \item[12:] \textsc{Sort}(\text{labelSet})
  \item[13:] finalLabel := $\varepsilon$
  \item[14:] \textbf{for} $i := 1$ \textbf{to} \#labelSet \textbf{do}
  \item[15:] finalLabel := finalLabel $\circ$ labelSet[$i$]
  \item[16:] \textbf{end for}
  \item[17:] \textbf{return} finalLabel $\circ$ 0
\end{itemize}

Algorithm 5 \textsc{RightLabel}

\textbf{Require:} vertex $v$
\begin{itemize}
  \item[1:] $\text{label}_v := \varepsilon$
  \item[2:] $\text{labelSet} := \emptyset$
  \item[3:] \textbf{if} $v$ has a left child \textbf{then}
  \item[4:] $\text{label}_v := \mathit{1} \circ \textsc{LeftLabel}(\text{left}(v))$
  \item[5:] \textbf{else}
  \item[6:] $\text{label}_v := 10$
  \item[7:] \textbf{end if}
  \item[8:] \textbf{if} $v$ has a right child \textbf{then}
  \item[9:] $\text{labelSet} := \textsc{RightLabel}(\text{right}(v))$
  \item[10:] \textbf{end if}
  \item[11:] \textbf{return} labelSet $\cup \{ \text{label}_v \}$
\end{itemize}

That is, the second phase, in the general case, is achieved by Algorithms 2, 4 and 5, which take each component of the input graph(s), produce a canonical label by first labelling each tree rooted at a cycle vertex, concatenating these labels then shifting the label to obtain the maximum value. Ultimately, we consider these labels as bit strings with the final label of a component taking $2k + 1$ bits where $k$ is the number of vertices in the component. Then we can encode the graph
as a whole with less than $3n$ bits. To obtain this bound we represent the trees attached to the cycles with left-child-right-sibling binary trees (e.g. see Knuth[14] for binary representation of trees), in which the right child of a vertex is a sibling and the left child is the first child (we can take any ordering for our purposes).

The two tree labelling algorithms (\textsc{LeftLabel} and \textsc{RightLabel}) together produce the canonical labelling of the tree in several steps. First the tree is implicitly extended to a full binary tree by adding leaf vertices whenever a vertex is missing a child. Each internal vertex is labelled with “1” and each leaf with “0”. Each vertex extends is labeled by concatenating its label with the label of its left subtree, then adding this label to the set of labels received from its right subtree. If a vertex is a left child (that is, it is the first child of its parent in the normal representation), it sorts this set of labels, largest to smallest, concatenates them and passes this label to its parent (Figure 2 illustrates the process).

\textbf{Lemma 14.} The combined Algorithms 4 and 5 perform at most $O(k^2)$ bit comparisons and use linear memory space to build a canonical label of size $2k + 1$ bits for any component of size $k$.

\textit{Proof.} Transforming the arbitrary tree into the special full binary tree takes linear memory and time. The canonical label is again built on the fly by traversing the full binary tree. The main cost at each internal node is to lexicographically sort the labels of the descendants along
the right path of its left subtree. As the original function, and thus associated original tree, may have arbitrary degree $d_i$, each sort can cost $O(d_i \cdot k)$ bit comparisons (that is, the lexicographic sort of $d_i$ labels of size $2k + 1$ bits). However, for any component of size $k$ the number of bit comparisons is proportional to

$$\sum_{i=0}^{k} d_i \cdot k = k \cdot \sum_{i=0}^{k} d_i = k^2$$

which concludes the proof. \(\square\)

Combining the costs of labelling for all components, with the rest of the meta-algorithm, we obtain the following result for testing isomorphism.

**Theorem 15.** For any functional graphs $G$ and $H$ of arbitrary functions with $n$ vertices, there is an isomorphism test in $O(k^* \cdot n)$ bit comparisons and linear memory complexity, where $k^* = \max\{k_G^*, k_H^*\}$.

**Proof.** We need to label all components. Using the sizes of various classes of components in the graph, that is, Lemma 14, the overall running time is proportional to:

$$\sum_{i=1}^{s} c_i k_i^2 \leq k^* \sum_{i=1}^{s} c_i k_i = k^* n, \quad (20)$$

where

$$k^* = \max_{i=1, \ldots, s} |k_i|.$$ 

Combining this result with Lemmas 9 and 11 completes the proof. \(\square\)

It is interesting to note the trade-off between $c^*$, the maximum number of components of same size used in Theorem 13 and $k^*$, the largest component, used in Theorem 15 as it seemingly provides a choice among algorithms to test the isomorphism depending on related features of the graph. However, it should be emphasized that the comparison is not straightforward as the algorithm of Theorem 15 considers bit comparisons as the metric of the time cost, while Theorem 13 employs more involved algorithms.

We note that the bound (20) used in the proof of Theorem 15 together with Lemma 14 also lead to an upper bound on the size of the labelling of any functional graph.

**Corollary 16.** The meta algorithm used for isomorphism testing in Theorem 13 uses at most $O(k^* \cdot n)$ bit comparisons and linear memory space to build canonical labels of size of at most $3n$ bits that can be represented in a trie of size $O(n)$ for any functional graph of size $n$. 
Proof. As each connected component of \( k \) vertices contributes \( 2k + 1 \) bits to the final labelling of the graph of size \( n \), the total number of bits for representing all components is \( 2n + c^G \), where \( c^G \) is the total number of components of \( G \) (and thus \( c^G \) is at most \( n \)). Finally, using Phase 3, we can build a trie of at most \( 3n \) nodes to encode all canonical encodings. \( \square \)

3.5. Counting functional graphs. We now present an algorithm to enumerate all functional graphs corresponding to polynomials of degree \( d \) over \( \mathbb{F}_q \).

**Theorem 17.** For any \( d \) and \( q \), we can create a list of all \( N_d(q) \) distinct functional graphs generated by all polynomials \( f \in \mathbb{F}_q[X] \) of degree \( \deg f = d \) in \( O(d^2 q^d \log^2 q) \) arithmetic operations and comparisons of bit strings of length \( O(q^2) \).

Proof. Let \( e = \gcd(d, q - 1) \) and let \( \Omega = \omega_1, \ldots, \omega_e \) be a set of representatives of the factor group \( \mathbb{F}_q^\times / H_e \), where \( H_e \) is the group of \( e \)th powers in \( \mathbb{F}_q^\times \), that is \( H_q = \{ \eta : \eta^e = 1 \} \).

Let \( A_j(a_d, \ldots, a_j; \lambda, \mu) \in \mathbb{F}_q \), \( j = 0, \ldots, d \), be as in the proof of Theorem 1. In particular, for

\[
f(X) = \sum_{j=0}^{d} a_j X^j \in \mathbb{F}_q[X], \quad \deg f = d,
\]

we have

\[
A_d(a_d; \lambda, \mu) = \lambda^{d-1} a_d, \tag{21}
\]

\[
A_{d-1}(a_d, a_{d-1}; \lambda, \mu) = \lambda^{d-2} \mu a_d + \lambda^{d-2} a_{d-1}.
\]

We see from (21), that for any polynomial \( f \in \mathbb{F}_q[X] \) of degree \( \deg f = d \) we can find \( \lambda \in \mathbb{F}_q^\times \) such that \( A_d(a_d; \lambda, \mu) \in \Omega \). After this, we find \( \mu \in \mathbb{F}_q \) such that \( A_{d-1}(a_d, a_{d-1}; \lambda, \mu) = 0 \). Thus \( \text{Orb} f \), given by (21), always contains a polynomial \( F \) of the form \( F(X) = A_d X^d + g(X) \) where \( A_d \in \Omega \) and \( g(X) \in \mathbb{F}_q[X] \) is of degree \( \deg f = d - 2 \). Therefore, it is enough to examine the graphs \( G_F \) only for such \( eq^{d-2} < dq^{d-2} \) polynomials \( F \).

Given such a polynomial \( f \in \mathbb{F}_q[X] \) of degree \( \deg f = d \), we can construct the graph \( G_f \) in time \( O(dq^d \log^2 q) \) (see [4]). After this, by Corollary 16, for each graph, in time \( O(q^2) \) we compute its canonical label. Inserting these labels in an ordered list of length at most \( N_d(q) \) (or discarding if the label already in the list) gives an overall time of \( O(dq^d \log N_d(q) \log q) = O(d^2 q^d \log^2 q) \). \( \square \)
In particular, the running time of the algorithm of Theorem 17 is at most $d^2q^{d+2+o(1)}$.

4. Numerical results

4.1. Preliminaries. We note that the periodic structure of functional graphs has been extensively studied numerically (see, for example, [3]). These results indicate that “generic” polynomials lead to graphs with cycle lengths with the same distribution as of those associated with random maps (see [3, Section 5]).

It is not difficult to see that for an odd $q$, the functional graph of any quadratic polynomial over $\mathbb{F}_q$ has $(q-1)/2$ leaves. Indeed, for $f(X) = X^2 + a$ the node $a$ is always an inner node with in-degree 1 while other nodes are of in-degree 0 or 2. Thus there are $1 + (q-1)/2 = (q+1)/2$ inner nodes and $(q-1)/2$ leaves. On the other hand, the graph of a random map on $p$ nodes is expected to have $p/e \approx 0.3679p$ leaves. It is possible that there are some other structural distinctions. Motivated by this, we have studied numerically several other parameters of functional graph.

Our tests have been limited to quadratic polynomials in prime fields, which can be further limited to polynomials of the form $f(X) = X^2 + a$, $a \in \mathbb{F}_p$. Various properties of the corresponding function graphs $G_f$ have been tested for all $p$ polynomials of this form for the following sequences of primes:

- all odd primes up to 100 (mostly for the purpose of testing our algorithms, but this has also revealed an interesting property of $N_2(17)$);
- for the sequence of primes between 101 and 102407 where each prime is approximately twice the size of its predecessor;
- for the sequence of 30 consecutive primes between 204803 (which could also be viewed as the last element of the previous group) and 205171;
- for the sequence of 10 consecutive primes between 500009 and 500167;
- for the prime 1000003.

For these primes, we tested the number of distinct primes and also average and extreme values of several basic parameters of the graphs $G_f$.

Our numerical results revealed that some of these parameters are the same as those of random graphs, but some (besides the aforementioned number of inner nodes) deviate in a rather significant way. Motivated by our algorithms of Section 3 we have initiated the study of several
interesting parameters of graphs which apparently has never been discussed in the literature before this work.

We present some of our numerical results (limited to those that show some new and unexpected aspects in the statistics of the graphs $G_f$), only for the primes of the last two groups, that is, for the set of primes 
\{500009, 500029, 500041, 500057, 500069, 500083, 
 500107, 500111, 500113, 500119, 500153, 500167, 1000003\}.

4.2. Number of distinct graphs. For all tested primes we have $N_2(p) = p$ except for $p = 17$ in which case $N_2(17) = 16$. This indicates that most likely we have $N_2(p) = p$ for any odd prime $p$, except for $p = 17$. However, proving this may be difficult as the case of $p = 17$ shows that there is no intrinsic reasons for this to be true (besides the fact that, as $p$ grows, the probability for this to occur becomes smaller).

4.3. Cyclic points and the giant components. Our numerical tests show that the average values of
• the number of cyclic points,
• the size of the largest connected components,
behave like expected from random maps, which are predicted to be $\sqrt{\pi p/2}$, (see [8, Theorem 2 (ii)]) and $\gamma p$ where $\gamma = 0.75788\ldots$, (see [8, Theorem 8 (ii)]) respectively.

It is also interesting to investigate the extreme values. More precisely, let $c(f)$ be the number of cyclic points of $G_f$ and let
\[ C(p) = \max\{c(f) : f(X) = X^2 + a, a \in \mathbb{F}_q\}. \]
In all our tests, except for the primes $p = 5, 13, 17$, the value of $C(p)$ is achieved on the function graphs of polynomials $f_0(X) = X^2$ and $f_{-2}(X) = X^2 - 2$, for which
\[ c(f_0) = r + 1 \quad \text{and} \quad c(f_{-2}) = (r + s)/2, \]
where $r$ is the largest odd divisor of $p - 1$ and $s$ is the largest odd divisor of $p + 1$, see [28, Theorem 6 (b)] and [28, Corollary 18 (b)], respectively (note that in [28] the polynomials are considered as acting on $\mathbb{F}_p^\ast$). In particular, if $p \equiv 3 \pmod{4}$ then the function graph of $X^2$ has the largest possible number of $(p + 1)/2$ cyclic points. Hence, $C(p) = (p + 1)/2$, for $p \equiv 3 \pmod{4}$.

We also note that for any $p \geq 3$,
\[ (22) \quad C(p) \geq \max\{r + 1, (r + s)/2\} \geq (p + 3)/4. \]
Furthermore, if $f(X) = X^2 + a$ with $a \in \mathbb{F}_p^\ast$ then the number of cyclic points of $G_f$ is at most $3p/8 + O(1)$. Indeed, let $\mathcal{V}_f = \{f(x) : x \in \mathbb{F}_p\}$
be the value set of \( f \) (that is, the set of inner nodes of \( G_f \)). Clearly, \( v \in V_f \) if \( v - a \) is quadratic residue modulo \( p \). Since for the sums of Legendre symbols modulo \( p \) we have

\[
\left| \sum_{v \in \mathbb{F}_p} \frac{(v - a)(-v - a)}{p} \right| = 1
\]

(see [18, Theorem 5.48]), we see that there are \( p/4 + O(1) \) values of \( v \in \mathbb{F}_p \) with \( v, -v \in V_f \). However, because \( f(v) = f(-v) \), it is clear that only one value out of \( v \) and \( -v \) can be a cyclic point. Hence, the number of cyclic points in \( G_f \) for \( f(X) = X^2 + a \) with \( a \in \mathbb{F}_p^\ast \) is at most \( 3p/8 + O(1) \). In particular, we now see from (22) that

\[
C(p) = \frac{3p}{8} + O(1), \quad \text{for } p \equiv 5 \pmod{8}.
\]

The smallest number of cyclic points has achieved the value 2 for all tested primes except \( p = 3 \) and \( p = 7 \) (for which this is 1).

In Table 1, we provide some numerical data for the number of cyclic points taken over all polynomials except for the above two special polynomials. In particular, we give the results for

\[
C^\ast(p) = \max\{c(f) : f(X) = X^2 + a, \ a \in \mathbb{F}_q \setminus \{0, -2\}\}.
\]

| Prime  | Min | Max  | Average | Expected  |
|--------|-----|------|---------|-----------|
| 500009 | 2   | 3578 | 886.2239149 | 886.2349015 |
| 500029 | 2   | 3620 | 885.9897086 | 886.2526257 |
| 500041 | 2   | 3798 | 885.0688786 | 886.2632600 |
| 500057 | 2   | 3468 | 884.9626481 | 886.2774389 |
| 500069 | 2   | 3556 | 885.8313906 | 886.2880730 |
| 500083 | 2   | 3596 | 884.9700189 | 886.3004792 |
| 500107 | 2   | 3527 | 884.5065536 | 886.3217460 |
| 500111 | 2   | 3732 | 884.3407057 | 886.3252912 |
| 500113 | 2   | 3805 | 885.1602624 | 886.3270634 |
| 500119 | 2   | 3873 | 884.5585953 | 886.3323802 |
| 500153 | 2   | 3472 | 884.8337362 | 886.3625078 |
| 500167 | 2   | 3644 | 884.7563204 | 886.3749130 |
| 1000003| 2   | 5101 | 1252.451837 | 1253.316017 |

**Table 1.** Cyclic points of polynomials \( f(X) \neq X^2, X^2 - 2 \)

It is quite apparent from Table 1 (and from our results for smaller primes) that both the maximum values (that is, \( C^\ast(p) \)) and the average values behave regularly and, as we have mentioned, the average value
fits the model of a random map quite precisely. We have not attempted to explain the behaviour of \( C^*(p) \).

The size of the largest component achieved the largest possible value \( p \) in all tested cases (thus, for any \( p \) some quadratic polynomial generates a graph with just one connected component, see Table 2 below). On the other hand, the smallest achieved size of the largest component does not seem to have a regular behaviour or even monotonicity.

4.4. **Number of components.** On the other hand, the average number of connected components has exhibited a consistent (but slowly decreasing) bias of about 9.5\% over the predicted value \( 0.5 \log p \), see \[8, Theorem 2 (i)\].

For every tested prime, at least one graph \( G_f \) has just 1 component, while the largest number of components has been behaving quite chaotically in all tested ranges.

The above is illustrated in Table 2:

| Prime     | Min | Max | Average | Expected | Ratio       |
|-----------|-----|-----|---------|----------|-------------|
| 500009    | 1   | 135 | 7.19772 | 6.561190689 | 1.097014298 |
| 500029    | 1   | 631 | 7.20138 | 6.561210688 | 1.097568778 |
| 500041    | 1   | 58  | 7.19640 | 6.561222687 | 1.096807766 |
| 500057    | 1   | 139 | 7.19259 | 6.561238685 | 1.096224409 |
| 500069    | 1   | 48  | 7.19785 | 6.561250684 | 1.097024081 |
| 500083    | 1   | 56  | 7.19328 | 6.561264682 | 1.096325228 |
| 500107    | 1   | 129 | 7.19792 | 6.561288677 | 1.097028397 |
| 500111    | 1   | 104 | 7.19801 | 6.561292676 | 1.097044145 |
| 500113    | 1   | 160 | 7.19402 | 6.561294676 | 1.096432999 |
| 500119    | 1   | 81  | 7.19518 | 6.561300675 | 1.096608791 |
| 500153    | 1   | 143 | 7.19312 | 6.561334665 | 1.096289150 |
| 500167    | 1   | 77  | 7.19699 | 6.561348661 | 1.096876629 |
| 1000003   | 1   | 22  | 7.54330 | 6.907756779 | 1.092004285 |

Table 2. Numbers of connected components

4.5. **Most popular component size.** As we have mentioned, motivated by the complexity bounds of the algorithms of Section 3, we calculated the most popular size of the connected components of \( G_f \). Our results for large primes are given in Table 3. For all tested primes \( p \), the minimal value of the most common size is 1 or 2 (in fact, 2 becomes more common than 1 as \( p \) grows), while the largest value is \( p \), as in accordance with Table 2, for every \( p \) there is always connected a graph \( G_f \). The average value certainly shows a regular growth. However, there does not seem to be any results for this parameter for graphs of
random maps, so we have not been able to compare the graphs $G_f$ with such graphs. Our numerical results seem to suggest that the average of the most common size is proportional to $p^{1/2}$. However, we believe that more numerical experiments are needed before one can confidently formulate any conjectures.

| Prime    | Min | Max    | Average |
|----------|-----|--------|---------|
| 500009   | 1   | 500009 | 1689.24 |
| 500029   | 2   | 500029 | 1642.27 |
| 500041   | 2   | 500041 | 1604.86 |
| 500057   | 1   | 500057 | 1670.49 |
| 500069   | 2   | 500069 | 1638.32 |
| 500083   | 2   | 500083 | 1628.07 |
| 500107   | 2   | 500107 | 1635.19 |
| 500111   | 2   | 500111 | 1657.12 |
| 500113   | 2   | 500113 | 1655.44 |
| 500119   | 2   | 500119 | 1573.22 |
| 500153   | 2   | 500153 | 1690.84 |
| 500167   | 2   | 500167 | 1638.63 |
| 1000003  | 2   | 1000003| 2272.39 |

Table 3. Most common size of components

Furthermore, we also computed the number of components of the most popular size (see Table 4). Clearly, the minimal value has been 1 for all tested primes (as before, we appeal to Table 2 that shows that for every $p$ there is connected graph $G_f$). However, the largest multiplicity exhibits a surprising chaotic behavior.

The average value clearly converges to a certain constant. However, we made no attempt to conjecture the nature of this constant.

As above with the case of the most common size, this parameter has not been studied and there is no random map model to compare against our results.
5. Further Directions

It is certainly interesting to study multivariate analogues of our results, that is, to study graphs on $q^m$ vertices, generated by a system of $m$ polynomials in $m$ variables over $\mathbb{F}_q$.

Polynomial graphs over residue rings are also interesting and apparently totally unexplored objects of study. They may also exhibit some new and rather unexpected effects.

Finally, we pose an open question of obtaining reasonable approximations to the expected values of the quantities $k^G_*$ and $c^G_*$ for a graph associated with a random map.

Acknowledgement

The authors are very grateful to Domingo Gómez-Pérez for many useful comments.

The research of S. V. K. was partially supported by Russian Fund for Basic Research, Grant N. 11-01-00329, and Program Supporting Leading Scientific Schools, Grant Nsh-6003.2012.1; that of F. L. by Project PAPIIT IN104512, CONACyT Cooperación Bilateral 193539, CONACyT Cooperación Bilateral 163787 and a Marcos Moshinsky fellowship; that of B. M. by Australian Research Council Grant DP110104560; and that of I. E. S. by Australian Research Council Grant DP130100237 and Macquarie University Grant MQRDG1465020.
References

[1] E. Bach, ‘Toward a theory of Pollard’s rho method’, Inform. and Comp., 90 (1991), 139–155.
[2] E. Bach and A. Bridy, ‘On the number of distinct functional graphs of affine-linear transformations over finite fields’, Linear Algebra Appl., (to appear).
[3] R. L. Benedetto, D. Ghioca, B. Hutz, P. Kurlberg, T. Scanlon and T. J. Tucker, ‘Periods of rational maps modulo primes’, Math. Ann. 355 (2013), 637–660.
[4] R. Brent and P. Zimmerman, Modern computer arithmetic, Cambridge Univ. Press, 2010.
[5] W.-S. Chou and I. E. Shparlinski, ‘On the cycle structure of repeated exponentiation modulo a prime’, J. Number Theory, 107 (2004), 345–356.
[6] R. Crandall and C. Pomerance, Prime numbers: A computational perspective, 2nd edition, Springer-Verlag, New York, 2005.
[7] X. Faber, ‘Benedetto’s trick and existence of rational preperiodic structures for quadratic polynomials’, Preprint, 2013 (available from http://arxiv.org/abs/1305.0216).
[8] P. Flajolet and A. M. Odlzyko, ‘Random mapping statistics’, Lecture Notes in Comput. Sci., vol. 434, Springer-Verlag, Berlin, 1990, 329–354.
[9] E. V. Flynn, B. Poonen, and E. F. Schaefer, ‘Cycles of quadratic polynomials and rational points on a genus-2 curve’, Duke Math. J., 90 (1997), 435–463.
[10] J. B. Friedlander, C. Pomerance and I. E. Shparlinski, ‘Period of the power generator and small values of Carmichael’s function’, Math. Comp., 70 (2001), 1591–1605.
[11] J. E. Hopcroft and J. K. Wong, ‘Linear time algorithm for isomorphism of planar graphs (Preliminary Report)’, Proc. 6th Ann. ACM Symp. on Theory of Comp., 1974, 172–184.
[12] H. Iwaniec and E. Kowalski, Analytic number theory, Amer. Math. Soc., Providence, RI, 2004.
[13] P. J. Kelly, ‘A congruence theorem for trees’, Pacific J. Math., 7 (1957), 961–968.
[14] D. E. Knuth, The art of computer programming, vol. I: Fundamental algorithms, Addison-Wesley, 1968.
[15] D. E. Knuth, The art of computer programming, vol. II: Seminumerical algorithms, Addison-Wesley, 1969.
[16] D. E. Knuth, The art of computer programming, vol. III: Sorting and Searching, Addison-Wesley, 1973.
[17] P. Kurlberg and C. Pomerance, ‘On the period of the linear congruential and power generators, Acta Arith., 119 (2005), 149–169.
[18] R. Lidl and H. Niederreiter, Finite fields, Cambridge Univ. Press, Cambridge, 1997.
[19] A. MacFie and D. Panario, ‘Random mappings with restricted preimages, Lecture Notes in Comput. Sci., vol. 7533, Springer-Verlag, Berlin, 2012, 254–270.
[20] G. Martin and C. Pomerance, ‘The iterated Carmichael λ-function and the number of cycles of the power generator’, Acta Arith., 118 (2005), 305–335.
[21] R. C. Mason, Diophantine equations over functions fields, Cambridge, Cambridge Univ. Press, 1984.
[22] P. Morton, ‘Arithmetic properties of periodic points of quadratic maps. II’, *Acta Arith.*, 87 (1998), 89–102.
[23] P. Morton and J. H. Silverman, ‘Rational periodic points of rational functions’, *Internat. Math. Res. Notices*, 2 (1994), 97–110.
[24] B. Poonen, ‘The classification of rational preperiodic points of quadratic polynomials over $\mathbb{Q}$: a refined conjecture’, *Math. Zeit.*, 228 (1998), 11–29.
[25] M. Sha and S. Hu, ‘Monomial dynamical systems of dimension one over finite fields’, *Acta Arith.*, 148 (2011), 309–331.
[26] L. Somer and M. Krížek, ‘The structure of digraphs associated with the congruence $x^k \equiv y \pmod{n}$’ *Czechoslovak Math. J.*, 61 (2011), 337–358.
[27] N. Snyder, ‘An alternate proof of Mason’s theorem’, *Elemente Math.*, 55 (2000), 93–94.
[28] T. Vasiga and J. O. Shallit, ‘On the iteration of certain quadratic maps over $\text{GF}(p)$’, *Discr. Math.*, 277 (2004), 219–240.
[29] A. M. Zubkov and V. E. Tarakanov, ‘On the cycle structure of power maps in a residue ring’, Preprint, 2013.

**Steklov Mathematical Institute**, 8, Gubkin Street, Moscow, 119991, Russia  
*E-mail address*: konyagin@mi.ras.ru

**Fundación Marcos Moshinsky**, Circuito Exterior, C.U., Apdo. Postal 70-543, Mexico D.F. 04510, Mexico  
*E-mail address*: fluca@matmor.unam.mx

**Department of Computing, Macquarie University, Sydney, NSW 2109, Australia**  
*E-mail address*: bernard.mans@mq.edu.au

**Department of Computing, Macquarie University, Sydney, NSW 2109, Australia**  
*E-mail address*: luke.mathieson@mq.edu.au

**Department of Computing, Macquarie University, Sydney, NSW 2109, Australia**  
*E-mail address*: igor.shparlinski@mq.edu.au