Improved Capacity Outer Bound for Private Quadratic Monomial Computation

Improved Capacity Outer Bound for Private Quadratic Monomial Computation

Karen M. Dæhli, Sarah A. Obead, Hsuan-Yin Lin, and Eirik Rosnes
Simula UiB, N–5006 Bergen, Norway
Emails: kamadaehli@gmail.com, {sarah, lin, eirikrosnes}@simula.no

***Abstract*—In private computation, a user wishes to retrieve a function evaluation of messages stored on a set of databases without revealing the function's identity to the databases. Obead *et al.* introduced a capacity outer bound for private nonlinear computation, dependent on the order of the candidate functions. Focusing on private *quadratic monomial* computation, we propose three methods for ordering candidate functions: a graph edge-coloring method, a graph-distance method, and an entropy-based greedy method. We confirm, via an exhaustive search, that all three methods yield an optimal ordering for $f < 6$ messages. For $6 \leq f \leq 12$ messages, we numerically evaluate the performance of the proposed methods compared with a directed random search. For almost all scenarios considered, the entropy-based greedy method gives the smallest gap to the best-found ordering.**

I. Introduction

Private computation (PC) [1] is a generalization of the renowned private information retrieval (PIR) problem that aims at addressing privacy concerns in distributed computing services. For example, in distributed machine learning, many common classification, dimensionality reduction, and linear regression algorithms operate on the inner products of the data samples rather than the individual data samples. In PC, the user wants to privately download a function evaluation of the messages stored across a set of databases, i.e., without leaking any information to the databases (in an information-theoretic manner) on the identity of the desired function evaluation.

To measure the efficiency of a PC protocol, the PC rate, defined as the ratio between the (smallest) size of the function evaluation and the number of downloaded symbols, is typically considered. The maximum PC rate is referred to as the PC capacity, and it is known for the case of linear function evaluations, referred to as private linear computation (PLC), from noncolluding replicated and coded databases [1]–[3]. Private polynomial computation (PPC) was first considered by Karpuk in [4], and later in [5]–[8]. The capacity of PPC is still unknown, and there is generally a substantial gap between the best achievable rate and the best-known capacity outer bound. Private inner product retrieval from noncolluding replicated databases was considered in [9], while the general case of PC for nonlinear function evaluations was considered in [10] where an outer bound on the capacity was first introduced. It was noted in [10] that the value of the PC capacity outer bound depends on how the candidate functions are ordered.[1] To the best of our knowledge, an optimal order for the outer bound has not yet been considered in the open literature.

In this work, inspired by private inner product retrieval [9], we focus on the case of private quadratic nonparallel monomial computation (PQNMC) and propose three methods for finding a *good* ordering of the μ candidate functions. In PQNMC, the set of candidate functions is the set of all quadratic *nonparallel* monomials of f messages where each message symbol is chosen from a size-q finite field $\mathbb{F}_q$, thus, $\mu = {}^{f(f-1)}/_{2}$. Given that graphs present a strong framework for illustrating the interdependence among random variables, offering insights into the dependency structure of the candidate function set, we propose two graph-based methods. The first is an edge-coloring method we name *(enhanced) edge-coloring* ((E-)EC) and the second is a graph-distance method we name *longest-distance first* (LDF). Then, we compare the resulting PC capacity outer bound with the one found by our third method: an *entropy-based greedy* (EBG) algorithm.

For $f < 6$ messages chosen from $\mathbb{F}_2$, we verify through an exhaustive search that the proposed methods output *optimal* orders. However, we note that the orders are not unique and are finite field-dependent, which illustrates the difficulty of finding a general ordering of quadratic nonparallel monomials that will optimize the outer bound of the PQNMC capacity. Moreover, for larger numbers of messages, an exhaustive search quickly becomes infeasible even over $\mathbb{F}_2$. As a result, we opt for a *directed* random search to numerically analyze the performance of the proposed methods. Accordingly, we note that for $6 \leq f \leq 10$ and $f = 12$ messages, the EBG algorithm outperforms the proposed graph-based methods with the smallest gap to the best-found ordering. Nevertheless, the significance of the graph-based methods arises as a relatively low-complexity alternative to the EBG method as the complexity of computing the entropies needed for the EBG algorithm grows exponentially with the number of candidate monomials μ with base equal to the size q of the underlying finite field $\mathbb{F}_q$. Finally, although we consider PQNMC in this work, we note that the results may also have independent interest beyond PC.

II. Preliminaries

A. Notation

We denote by $\mathbb{N}$ the set of all positive integers and $[a] \triangleq \{1, 2, \ldots, a\}$ for $a \in \mathbb{N}$. A random variable is denoted by a capital Roman letter, e.g., X, while its realization is denoted by the corresponding small Roman letter, e.g., x.

[1] Due to the inherent relation between PC and PIR, a similar observation was made in [11] for PIR with dependent messages, i.e., dependent PIR (DPIR).

Vectors are boldfaced, e.g., $\boldsymbol{X}$ denotes a random vector, and $\boldsymbol{x}$ denotes a deterministic vector. Sets are denoted by calligraphic uppercase letters, e.g., $\mathcal{X}$. Concatenation of vectors $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_a$ is represented by $(\boldsymbol{x}_1 \mid \cdots \mid \boldsymbol{x}_a)$. Furthermore, some constants and functions are depicted by Greek letters or a special font, e.g., X. The entropy of X is represented by $\mathsf{H}(X)$. A degree g monomial $\boldsymbol{z}^{\boldsymbol{i}}$ in f variables $z_1, \ldots, z_f$ over a finite field $\mathbb{F}_q$ is written as $\boldsymbol{z}^{\boldsymbol{i}} = z_1^{i_1} \cdots z_f^{i_f}$, where $\boldsymbol{i} \triangleq (i_1, \ldots, i_f) \in (\{0\} \cup \mathbb{N})^f$ is the exponent vector with $\sum_{j=1}^{f} i_j = g$ and $i_j \leq q-1$ for all $j \in [f]$.[2] A simple undirected graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$ is denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

B. Problem Statement

We consider the PQNMC problem, which is formally described as follows.[3] Consider a distributed storage system (DSS) consisting of n noncolluding databases, each storing a replica of f independent messages. The messages are denoted by $\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(f)}$ and each message $\boldsymbol{W}^{(m)} = \bigl(W_1^{(m)}, \ldots, W_{\beta\mathsf{L}}^{(m)}\bigr)$, $m \in [f]$, is a length-$\beta\mathsf{L}$ vector with independent and identically distributed symbols that are chosen uniformly at random from the field $\mathbb{F}_q$ for some $\beta, \mathsf{L} \in \mathbb{N}$.[4] Hence, we have

$$\mathsf{H}\bigl(\boldsymbol{W}^{(m)}\bigr) = \beta\mathsf{L}, \forall m \in [f],$$
$$\mathsf{H}\bigl(\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(f)}\bigr) = f\beta\mathsf{L} \quad \text{(in q-ary units)}.$$

In PQNMC, a user wishes to privately compute exactly one quadratic nonparallel monomial $X_i^{(k,\ell)} \triangleq W_i^{(k)} W_i^{(\ell)}$, $\forall i \in [\beta\mathsf{L}]$, for some $k, \ell \in [f]$, $k < \ell$, out of $\mu \triangleq \binom{f}{2}$ *candidate* quadratic nonparallel monomials. For convenience, we denote by $\mathcal{T} \triangleq \{(k,\ell)\colon k, \ell \in [f], k < \ell\}$ the set of all ordered 2-tuples, where $|\mathcal{T}| = \mu$. In q-ary units, we have

$$\mathsf{H}\bigl(\boldsymbol{X}^{(k,\ell)}\bigr) = \beta\mathsf{L}\,\mathsf{H}\bigl(X^{(k,\ell)}\bigr), \forall (k,\ell) \in \mathcal{T},$$
$$\mathsf{H}\bigl(\boldsymbol{X}^{(1,2)}, \ldots, \boldsymbol{X}^{(f-1,f)}\bigr) = \beta\mathsf{L}\,\mathsf{H}\bigl(X^{(1,2)}, \ldots, X^{(f-1,f)}\bigr),$$

for *prototype* random variables $X^{(k,\ell)}$.

The user privately selects an index (k,ℓ) and wishes to compute the (k,ℓ)-th quadratic nonparallel monomial while keeping the requested index (k,ℓ) private from each database. In order to retrieve the desired function $\boldsymbol{X}^{(k,\ell)}$, $(k,\ell) \in \mathcal{T}$, from the DSS, the user sends a random query to the j-th database for all $j \in [n]$. The user generates the queries without any prior knowledge of the realizations of the stored messages, and they are independent of the candidate quadratic nonparallel monomials. In response to the received query, the j-th database sends an answer back to the user.

To measure the efficiency of a PC protocol, we consider the required number of downloaded q-ary symbols for retrieving the $\beta\mathsf{L}$ q-ary symbols of the desired function evaluation.

Definition 1 (PQNMC Rate and Capacity): The rate of a PQNMC protocol, denoted by R, is defined as the ratio between the *smallest* desired monomial size $\beta\mathsf{L}\,\mathsf{H}_{\mathsf{min}}$ and the total required download cost D, i.e.,

$$\mathsf{R} \triangleq \frac{\beta\mathsf{L}\,\mathsf{H}_{\mathsf{min}}}{\mathsf{D}},$$

where $\mathsf{H}_{\mathsf{min}} \triangleq \min_{(k,\ell)\in\mathcal{T}} \mathsf{H}\bigl(X^{(k,\ell)}\bigr)$. The PQNMC *capacity*, denoted by $\mathsf{C}_{\mathsf{PQNMC}}$, is the maximum achievable PQNMC rate over all possible PQNMC protocols.

Note that for quadratic nonparallel monomials, we have $\mathsf{H}_{\mathsf{max}} \triangleq \max_{(k,\ell)\in\mathcal{T}} \mathsf{H}\bigl(X^{(k,\ell)}\bigr) = \mathsf{H}_{\mathsf{min}}$. Accordingly, every quadratic nonparallel monomial carries the same amount of information and following the terminology of DPIR [11] we denote the PQNMC problem as *balanced*.

Let $\mathcal{P}(\mathcal{T})$ be the set of all permutations on the set $\mathcal{T}$, and denote an ordered set $\mathcal{S} \triangleq (\boldsymbol{s}_1, \ldots, \boldsymbol{s}_\mu) \in \mathcal{P}(\mathcal{T})$. Using the same approach as [10], it can be shown that the PQNMC capacity is bounded from above by $\mathsf{C}_{\mathsf{PQNMC}} \leq \overline{\mathsf{C}}(\mathcal{S})$, where

$$\overline{\mathsf{C}}(\mathcal{S}) \triangleq \frac{n^{\mu}\,\mathsf{H}_{\mathsf{min}}}{\sum_{v=1}^{\mu} n^{\mu-v+1}\,\mathsf{H}\bigl(X^{(\boldsymbol{s}_v)} \,\big|\, X^{(\boldsymbol{s}_1)}, \ldots, X^{(\boldsymbol{s}_{v-1})}\bigr)}. \tag{1}$$

The goal of this work is to determine the best (lowest) outer bound to the PQNMC capacity $\mathsf{C}_{\mathsf{PQNMC}}$, among all the possible orders of the quadratic nonparallel monomials for a given number of messages f, i.e., we are interested in obtaining the best-ordered set that achieves $\min_{\mathcal{S}\in\mathcal{P}(\mathcal{T})} \overline{\mathsf{C}}(\mathcal{S})$.

Remark 1: We have observed by exhaustive search for $f \leq 5$ that the set of optimal orderings (the ones that minimize the capacity outer bound in (1)) is independent of $n \geq 2$, which suggests that one can choose $n = 2$ for finding an optimal order for the capacity outer bound $\overline{\mathsf{C}}(\mathcal{S})$.

C. Edge-Coloring and Matching

Quadratic nonparallel monomials in f variables can be represented by a simple undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V} = [f]$ where the vertices k and ℓ correspond to the messages $W^{(k)}$ and $W^{(\ell)}$ (for *prototype* random variables $W^{(k)}$ and $W^{(\ell)}$), respectively, and the edge $(k,\ell) \in \mathcal{E}$ represents the nonparallel monomial $X^{(k,\ell)}$. Thus, we have $\mu = |\mathcal{E}|$ and the set of all quadratic nonparallel monomials in f variables are represented by a *complete* graph $\mathcal{K}_f$.

Definition 2 (Distance): Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, the distance between two vertices $u, v \in \mathcal{V}$, denoted by $d(u,v)$, is the length of the shortest path connecting them, measured in number of edges.

Definition 3 (Matching): Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a matching $\mathcal{M}$ in $\mathcal{G}$ is a set of pairwise nonadjacent edges, i.e., a set of edges where no two edges share common vertices. When $|\mathcal{V}|$ is an even number, a *perfect matching* is a matching that includes all vertices of the graph, and when $|\mathcal{V}|$ is an odd number, a *near-perfect matching* is a matching that includes $|\mathcal{V}| - 1$ vertices of the graph.

Definition 4 (Edge-Coloring [13, Ch. 17]): A *proper* edge-coloring of a graph is an assignment of colors to the edges

[2] For nonvanishing polynomials, following the Combinatorial Nullstellensatz theorem [12, Thm. 1.2], the degree of every variable in a multivariate polynomial must be strictly smaller than the finite field size.

[3] A monomial $m(\boldsymbol{z})$ is said to be *parallel* if it can be raised by another monomial to a positive integer power, i.e., $m(\boldsymbol{z}) = (\boldsymbol{z}^{\boldsymbol{i}})^d$ for some $d \in \mathbb{N}$ and $\boldsymbol{i} \in (\{0\} \cup \mathbb{N})^f$.

[4] For consistency, we use the notation required for the achievable rate in [10, Thm. 2] where asymptotically $\mathsf{L} \to \infty$ but β is fixed.

such that the edges incident to a vertex have distinct colors. A graph $\mathcal{G}$ is said to be κ-edge colorable if $\mathcal{G}$ has a proper edge-coloring with κ colors. The *chromatic index* of a graph $\mathcal{G}$, denoted by $\chi'(\mathcal{G})$, is the minimum number of colors required to properly edge-color $\mathcal{G}$.

The definition of proper edge-coloring of a graph $\mathcal{G}$ implies that the κ-edge coloring of a graph partitions the graph edge set $\mathcal{E}$ into κ (near) perfect matchings $\mathcal{M}_1, \dots, \mathcal{M}_\kappa$ such that $\mathcal{E} = \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_\kappa$, and the sets $\mathcal{M}_1, \dots, \mathcal{M}_\kappa$ are known as the color sets of edges. For complete graphs of f vertices, denoted by $\mathcal{K}_f$, it is known [14, Thm. 1] that

$$\chi'(\mathcal{K}_f) = \begin{cases} f-1 & \text{if } f \text{ is even,} \\ f & \text{if } f \text{ is odd.} \end{cases}$$

Remark 2: Let $\mathscr{M}$ be the set of all (near) perfect matchings of a complete graph $\mathcal{K}_f$. Let $\mathcal{S}[1:\eta] = (\boldsymbol{s}_1, \dots, \boldsymbol{s}_\eta)$ be the first η elements of the ordered set $\mathcal{S}$, where $\eta \triangleq {}^{f(f-1)}/_{2\chi'(\mathcal{K}_f)}$ is the number of edges within a complete graph matching. For $\mathcal{S}$ to be an *optimal* order of quadratic nonparallel monomials of $f > 3$ variables, we conjecture that $\mathcal{S}[1:\eta]$ must constitute a (near) perfect matching of $\mathcal{K}_f$, i.e., $\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_\eta\} \in \mathscr{M}$.

III. Algorithms to Determine a Good Order

In this section, we propose three methods for finding a *good* order, one based on edge-coloring, one based on graph distance, and one entropy-based greedy algorithm.

Remark 3: Computing the capacity bound of (1) for a given order entails the computation of μ conditional entropies, resulting in a computational complexity of order $\mathcal{O}(q^\mu)$. Taking that into account, it can be seen that performing an exhaustive search over all possible orders to optimize the PQNMC capacity outer bound would intuitively require a complexity of order $\mathcal{O}(\mu! \times q^\mu)$.

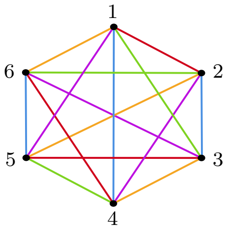
A. Edge-Coloring

The key idea is to first find a proper edge-coloring of the complete graph $\mathcal{K}_f$. Then, build an order based on grouping the edges according to their color, i.e., first take the edges corresponding to one of the colors, then the edges corresponding to another color, etc., until all colors have been considered. The time complexity of finding an order based on simple edge-coloring follows from the time complexity of finding a proper edge-coloring of $\mathcal{K}_f$. We follow the procedure in [15, App. A] which runs in polynomial time, i.e., of order $\mathcal{O}(\mu)$.

We first give an example to illustrate how edge-coloring can give us a good order $\mathcal{S}_{\text{EC}}$ for the capacity outer bound.

Example 1: For $f = 6$ messages, a proper 5-edge coloring of $\mathcal{K}_6$ is as follows:

Purple	$(1,5)(2,4)(3,6)$
Yellow	$(1,6)(2,5)(3,4)$
Red	$(1,2)(3,5)(4,6)$
Green	$(1,3)(2,6)(4,5)$
Blue	$(1,4)(2,3)(5,6)$



The resulting order is

$$\mathcal{S}_{\text{EC}} = \big((1,5),(2,4),(3,6),(1,6),(2,5),(3,4),(1,2),\\ (3,5),(4,6),(1,3),(2,6),(4,5),(1,4),(2,3),(5,6)\big),$$

and for $n = 2$ and $q = 2$, the capacity outer bound is $\overline{\mathsf{C}}(\mathcal{S}_{\text{EC}}) = 0.5198943946817$.

The permutation of the colors affects the value of $\overline{\mathsf{C}}$, which indicates that edge-coloring by itself does not guarantee finding an optimal order. Thus, a search over color permutations for the capacity bound, i.e., over $(\chi'(\mathcal{K}_f) - 1)!$ permutations (the first η edges corresponding to a single color can be fixed; see Remark 2), can potentially improve it in exchange for added complexity. We refer to this improved method as enhanced edge-coloring (E-EC), and its complexity is of order $\mathcal{O}(\mu + (\chi'(\mathcal{K}_f) - 1)! \times q^\mu)$. For example, if we reorder the colors in Example 1 as (purple, yellow, blue, red, green), we obtain the order

$$\mathcal{S}_{\text{E-EC}} = \big((1,5),(2,4),(3,6),(1,6),(2,5),(3,4),(1,4),\\ (2,3),(5,6),(1,2),(3,5),(4,6),(1,3),(2,6),(4,5)\big),$$

which results, for $n = 2$ and $q = 2$, in the improved capacity outer bound $\overline{\mathsf{C}}(\mathcal{S}_{\text{E-EC}}) = 0.5198121367672$.

We have observed that even within a set of edges of a given color, the permutation of the edges also affects the value of $\overline{\mathsf{C}}$. For instance, considering the blue color in Example 1, the values of $\overline{\mathsf{C}}$ between the orders $\{(1,4),(2,3),(5,6)\}$ and $\{(2,3),(1,4),(5,6)\}$ are different. However, when searching for the best order within the edges of every color, the computational complexity becomes $\mathcal{O}(\mu + ((\eta!)^{\chi'(\mathcal{K}_f)}) \times q^\mu)$, which renders finding the best order quickly infeasible. Here, we are presenting the (E-)EC solution as a low-complexity solution for finding a *good* order. Thus, we opt out of optimizing the (E-)EC solution any further, and we simply order the edges (k,ℓ) within each color set according to the lexicographical order on $[f] \times [f]$.

The E-EC method is briefly summarized as Algorithm 1.

B. Longest-Distance First

The goal of LDF is to minimize dependency among the first selected monomials within an order. Thus, the intuition behind the LDF method also follows from graph matching. However, unlike in the (E-)EC method, we do not restrict ourselves to (near) perfect matchings of the complete graph $\mathcal{K}_f$. Here, we follow the convention that if two vertices belong to different connected components, then the distance is defined as infinite [13], i.e., there is no path connecting the two vertices. The LDF method is summarized with the following sequential steps (further details and a pseudo-code can be found in [15, App. B]).

Start with the null graph $\mathcal{G} = \mathcal{N}_f$, i.e., a graph with $\mathcal{V} = [f]$ and $\mathcal{E} = \emptyset$. Then, adhere to the following steps, adding an edge (u,v) to $\mathcal{G}$ and partial order $\mathcal{S}_{\text{LDF}}$ with each step repetition.

1) Repeatedly add an edge not adjacent to any other edge.
2) Repeatedly add an edge that connects two vertices with the longest distance and lowest degree in the graph, until a length-f cycle is formed.

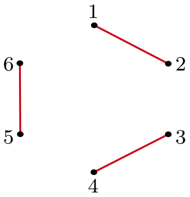
Algorithm 1: Searching for a good order for $\overline{\mathrm{C}}$ based on edge-coloring (E-EC)

```
Input  : $f$, $q$, $n$
Output: A good order of edges $\mathcal{S}_{\text{E-EC}}$
$\mathcal{E}_1, \dots \mathcal{E}_{\chi'(\mathcal{K}_f)} \leftarrow$ color sets of edges with edges ordered
    lexicographically on $[f] \times [f]$
$\mathcal{S}_{\text{E-EC}} \leftarrow \mathcal{E}_1$, $i \leftarrow 1$
$\mathcal{E}_{\text{c}} \leftarrow$ a permutation of the remaining color sets of edges
$\mathcal{S}_{\text{E-EC}} \leftarrow (\mathcal{E}_{\text{E-EC}} \mid \mathcal{E}_{\text{c}})$
Compute $\mathrm{C}_{\text{E-EC–best}} = \overline{\mathrm{C}}(\mathcal{S}_{\text{E-EC}})$
while $i \leq (\chi'(\mathcal{K}_f) - 1)!$ do
    $i \leftarrow i + 1$
    $\mathcal{E}_{\text{c}} \leftarrow$ next permutation of the remaining color sets of
      edges
    if $\overline{\mathrm{C}}(\mathcal{E}_{\text{E-EC}} \mid \mathcal{E}_{\text{c}}) < \mathrm{C}_{\text{E-EC–best}}$ then
        $\mathcal{S}_{\text{E-EC}} \leftarrow (\mathcal{E}_{\text{E-EC}} \mid \mathcal{E}_{\text{c}})$, $\mathrm{C}_{\text{E-EC-best}} \leftarrow \overline{\mathrm{C}}(\mathcal{S}_{\text{E-EC}})$
    end
end
return $\mathcal{S}_{\text{E-EC}}$
```

3) Repeatedly add an edge that connects two vertices with the lexicographically smallest numbers of induced length-l cycles for $3 \leq l \leq f$, until $\mathcal{G}$ is complete, i.e., $\mathcal{G} = \mathcal{K}_f$.

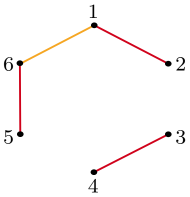
As for the (E-)EC method, we elaborate on the LDF algorithm with an illustrative example.

Example 2: For $f = 6$, first, add an edge that is not adjacent to any other edge. For example $(1, 2)$, then $(3, 4)$. As a result, one remaining edge can be added following Step 1), which is $(5, 6)$. Note that $\mathcal{E} = \{(1, 2), (3, 4), (5, 6)\}$ constitute a perfect matching of $\mathcal{K}_6$, adhering to Remark 2. The corresponding graph $\mathcal{G}$ and graph distances are illustrated in the following figure, where $v \in \mathcal{V}$ denotes the vertex label.



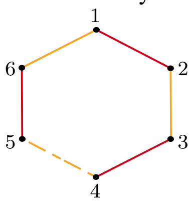
2	1				
3	∞	∞			
4	∞	∞	1		
5	∞	∞	∞	∞	
6	∞	∞	∞	∞	1
v	1	2	3	4	5

Next, to add an edge that connects two vertices with the longest distance and lowest degree in the graph, we select, for example, the edge $(1, 6)$ with $d(1, 6) = \infty$ and both vertices of degree 1. As a result, we have $d(2, 5) = 3$ and $d(2, 6) = d(1, 5) = 2$, as depicted in the following figure:



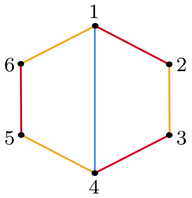
2	1				
3	∞	∞			
4	∞	∞	1		
5	2	3	∞	∞	
6	1	2	∞	∞	1
v	1	2	3	4	5

Next, to repeat Step 2), we can select from any available edge (u, v) with vertices of degree 1 and $d(u, v) = \infty$, i.e., $u, v \in \{2, 3, 4, 5\}$. For example, select the edge $(2, 3)$, then $(4, 5)$ forming a length-f cycle as illustrated in the left-hand side (l.h.s.) of the figure below where a dashed line indicates the order of adding. Now, for Step 3), we count the cycles induced from adding each of the remaining edges as seen in the following table, where $\boldsymbol{o} = (o_1, \dots, o_i, \dots, o_{f-2})$ and o_i is the number of cycles of length $i + 2$.



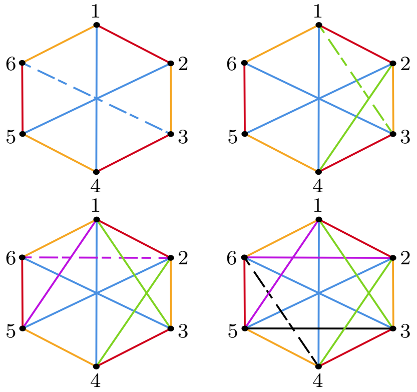
(u, v)	$\boldsymbol{o}$
$(1, 4), (2, 5), (3, 6)$	$(0, 2, 0, 1)$
$(1, 3), (1, 5), (2, 4)$ $(2, 6), (3, 5), (4, 6)$	$(1, 0, 1, 1)$

Each of the edges $(1, 4)$, $(2, 5)$, and $(3, 6)$ induces the smallest number of cycles, lexicographically, thus we select one of these edges. Let that edge be $(1, 4)$. The corresponding graph $\mathcal{G}$ is illustrated in the l.h.s of the following figure:



(u, v)	$\boldsymbol{o}$
$(2, 5), (3, 6)$	$(0, 5, 0, 2)$
$(2, 6), (3, 5)$	$(1, 2, 3, 1)$
$(1, 3), (1, 5)$ $(2, 4), (4, 6)$	$(2, 2, 1, 1)$

By repeating Step 3), the choices for the following edges remain the same. As can be seen from the above table, edges $(2, 5)$ and $(3, 6)$ induce the same number of cycles in $\mathcal{G}$. Thus, we select for example $(2, 5)$. The remaining edges follow from repeating Step 3) and are added to the order and $\mathcal{G}$ as illustrated in the following left-to-right top-to-bottom order:



At the end of the LDF procedure, we obtain the order

$$\mathcal{S}_{\text{LDF}} = \big((1, 2), (3, 4), (5, 6), (1, 6), (2, 3), (4, 5), (1, 4), (2, 5), (3, 6), (2, 4), (1, 3), (1, 5), (2, 6), (3, 5), (4, 6)\big),$$

and for $n = 2$ and $q = 2$, the capacity outer bound is $\overline{\mathrm{C}}(\mathcal{S}_{\text{LDF}}) = 0.5197824997350$, which is strictly better than for the E-EC method. Interestingly, $\mathcal{S}_{\text{LDF}}$ corresponds to a proper edge-coloring, i.e., no two adjacent edges share the same color, but it does not correspond to an optimal edge-coloring.

C. Entropy-Based Greedy Method

The EBG method starts with an empty graph and sequentially adds edges in a greedy manner, i.e., the edge that minimizes the (partial) bound in (1), computed based on the new edge and the already added edges, is added at each step in the algorithm. In case of ties, one of the candidate edges is

TABLE I

COMPARISON BETWEEN PQNMC CAPACITY OUTER BOUNDS OBTAINED WITH THE (E-)EC ($\overline{\mathsf{C}}(\mathcal{S}_{(\text{E-})\underline{\text{EC}}})$), LDF ($\overline{\mathsf{C}}(\mathcal{S}_{\text{LDF}})$), AND EBG ($\overline{\mathsf{C}}(\mathcal{S}_{\text{EBG}})$) METHODS, AS WELL AS WITH THE BEST BOUND FOUND BY EXHAUSTIVE/DIRECTED RANDOM SEARCH ($\overline{\mathsf{C}}(\mathcal{S}_{\text{ES/RS}})$), FOR $n = 2$ DATABASES, AND FOR A FIELD SIZE OF $q = 2$. THE BEST BOUND FOR EACH NUMBER OF MESSAGES f IS MARKED IN BOLD. THE BEST-KNOWN ACHIEVABLE RATE R FROM [10, THM. 2] IS GIVEN AS WELL TO SHOW THE CAP TO THE CAPACITY OUTER BOUND.

f	5	6	7	8	9	10	11	12
$\overline{\mathsf{C}}(\mathcal{S}_{\text{EC}})$	0.5382035621102	0.5198943946817	0.5158988408975	0.5088200966114	0.5071434701312	0.5041602427037	0.5033789063480	0.5020207578041
$\overline{\mathsf{C}}(\mathcal{S}_{\text{E-EC}})$[5]	**0.5321513151313**	0.5198121367672	0.5130098344723	0.5085684044374	0.5058885273733	0.5039972538181	0.5028028499055	0.5019311781396
$\overline{\mathsf{C}}(\mathcal{S}_{\text{LDF}})$	**0.5321513151313**	**0.5197824997350**	**0.5129571653366**	0.5085546467521	0.5058724664437	0.5039960955809	**0.5027529132784**	0.5019069907637
$\overline{\mathsf{C}}(\mathcal{S}_{\text{EBG}})$	**0.5321513151313**	**0.5197824997350**	**0.5129571653366**	0.5085546463038	0.5058724626997	**0.5039958945996**	0.5027582097217	**0.5019068074415**
$\overline{\mathsf{C}}(\mathcal{S}_{\text{ES/RS}})$	**0.5321513151313**	**0.5197824997350**	**0.5129571653366**	**0.5085546398430**	**0.5058724411573**	0.5039961304091	0.5027529200313	0.5019070293099
R	0.5026676304668	0.5001371033940	0.5000032431709	0.5000000359051	0.5000000001891	0.5000000000005	0.5000000000000	0.5000000000000

selected at random. In particular, in the first step, an arbitrary edge is added to an initially empty graph. Then, in the second step the bound in (1) is computed with $\mu = 2$ based on the previously added edge and a new edge selected among the possible remaining edges. The new edge that minimizes the computed partial bound is then selected and added to the graph. In this manner, an edge is added to the graph in each step of the algorithm and a monomial order is constructed. The complexity of the EBG method is of order $\mathcal{O}({}^{\mu(\mu+1)}/_{2} \times q^{\mu})$. Finally, note that the order returned by the EBG method is independent of the value of $n \in \mathbb{N}$, including $n = 1$, while Remark 1 requires $n \geq 2$. Hence, $n = 1$ can be used for better numerical stability when conducting the EBG search.

IV. DISCUSSION AND RESULTS

In Table I, we compare the results from the proposed (E-)EC, LDF, and EBG methods for $5 \leq f \leq 12$ messages, for $n = 2$ databases, and for a field size of $q = 2$ with those of an exhaustive search (for $f = 5$) and a directed random search (for $6 \leq f \leq 12$). The directed random search is done by first fixing at least the first f edges according to edge-coloring (corresponding to two or three colors) and then conducting a random search among the remaining orders. As can be seen from the table, the LDF and EBG methods and the exhaustive/directed random search yield the same capacity outer bound for $f \leq 7$ messages, while for $f = 8$ and $f = 9$ messages a directed random search gives slightly better results (in the 8-th digit). (E-)EC gives the same bound as LDF for $f = 4$ messages, while for $f \geq 6$, E-EC performs worse compared to the LDF and EBG methods. The EC method performs in general slightly worse compared to the E-EC method, but has the lowest computational complexity. Interestingly, for $f = 11$, the LDF method outperforms all other methods. The best-known achievable rate R from [10, Thm. 2] is given in the last row of Table I to show the cap to the capacity outer bound. As a final remark, we note that for larger q (results not included here), the gap between the bounds produced by the LDF and EBG methods increases, which can be attributed to the fact that the proposed simple undirected graph model captures less of the dependencies for larger q.

[5] Due to high computational complexity, we fix two color sets of edges and then search over the remaining color permutations for $f = 11$ and $f = 12$.

V. CONCLUSION

We proposed two graph-based methods and one EBG algorithm to optimize the order of quadratic monomials in an outer bound for the PQNMC capacity. For $f < 6$ messages, all three methods minimize the bound, while for $6 \leq f \leq 12$ the results were compared with those of a directed random search. For almost all examined cases, the EBG algorithm yields the smallest gap to the best-found monomial ordering.

REFERENCES

[1] H. Sun and S. A. Jafar, "The capacity of private computation," *IEEE Trans. Inf. Theory*, vol. 65, no. 6, pp. 3880–3897, Jun. 2019.

[2] S. A. Obead and J. Kliewer, "Achievable rate of private function retrieval from MDS coded databases," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 17–22, 2018, pp. 2117–2121.

[3] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Private linear computation for noncolluding coded databases," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 3, pp. 847–861, Mar. 2022.

[4] D. Karpuk, "Private computation of systematically encoded data with colluding servers," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Vail, CO, USA, Jun. 17–22, 2018, pp. 2112–2116.

[5] N. Raviv and D. A. Karpuk, "Private polynomial computation from Lagrange encoding," *IEEE Trans. Inf. Forens. Secur.*, vol. 15, pp. 553–563, 2020.

[6] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "Private polynomial function computation for noncolluding coded databases," *IEEE Trans. Inf. Forens. Secur.*, vol. 17, pp. 1800–1813, 2022.

[7] Y. Yakimenka, H.-Y. Lin, and E. Rosnes, "On the capacity of private monomial computation," in *Proc. Int. Zurich Sem. Inf. Commun. (IZS)*, Zurich, Switzerland, Feb. 26–28, 2020, pp. 31–35.

[8] J. Zhu, Q. Yan, X. Tang, and S. Li, "Symmetric private polynomial computation from Lagrange encoding," *IEEE Trans. Inf. Theory*, vol. 68, no. 4, pp. 2704–2718, Apr. 2022.

[9] M. H. Mousavi, M. A. Maddah-Ali, and M. Mirmohseni, "Private inner product retrieval for distributed machine learning," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 7–12, 2019, pp. 355–359.

[10] S. A. Obead, H.-Y. Lin, E. Rosnes, and J. Kliewer, "On the capacity of private nonlinear computation for replicated databases," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Visby, Sweden, Aug. 25–28, 2019.

[11] Z. Chen, Z. Wang, and S. A. Jafar, "The asymptotic capacity of private search," *IEEE Trans. Inf. Theory*, vol. 66, no. 8, pp. 4709–4721, Aug. 2020.

[12] N. Alon, "Combinatorial nullstellensatz," *Combinatorics, Probability Comput.*, vol. 8, no. 1–2, pp. 7–29, Jan. 1999.

[13] J. A. Bondy and U. S. R. Murty, *Graph Theory*. London, U.K.: Springer, 2008.

[14] M. Behzad, G. Chartrand, and J. K. Cooper, Jr., "The colour numbers of complete graphs," *J. London Math. Soc.*, vol. s1-42, no. 1, pp. 226–228, 1967.

[15] K. M. Dæhli, S. A. Obead, H.-Y. Lin, and E. Rosnes, "Improved capacity outer bound for private quadratic monomial computation," Jan. 2024. [Online]. Available: https://arxiv.org/abs/2401.06125