Financiado por la Unión Europea NextGenerationEU
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D
GOBIERNO DE ESPAÑA

SUCCESS-6G: VERIFY

WP3 Deliverable E8

Joint source-channel coding for efficient computation in V2X systems

Project Title:	SUCCESS-6G: VERIFY

Title of Deliverable:	Joint source-channel coding for efficient computation in V2X systems
Status-Version:	Final-1.0
Delivery Date:	31/03/2025

Contributors:	Jesus Gomez, Adriano Pastore (CTTC)
Lead editor::	Jesus Gomez (CTTC)
Reviewers:	Miquel Payaro, Charalampos Kalalas (CTTC)

Keywords:	lossy source coding, coded distributed computing, multivariate coded computing schemes

Document Revision History

Version	Date	Description
v0.1	10 February 2025	Initial content addded
v0.2	21 February 2025	Main content addded
v0.3	10 March 2025	Revision and additional content added
v1.0	31 March 2025	Approved form of Deliverable E8

Acknowledgement
The research conducted by SUCCESS-6G - TSI-063000-2021-39/40/41 receives funding from the Ministerio de Asuntos Económicos y Transformación Digital and the European Union-NextGenerationEU under the framework of the "Plan de Recuperación, Transformación y Resiliencia" and the "Mecanismo de Recuperación y Resiliencia".

[1] http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

Executive summary

This deliverable addresses the need for efficient computation in vehicular-to-everything (V2X) systems by exposing the potential application of two innovative technological solutions: *i*) source-channel coding to enable resource-efficient computation over the wireless medium and *ii*) coded computing to allow efficient distributed computation under faulty computation nodes.

In the case of computation over the wireless medium, we consider distributed decision-making scenarios, where independent measurements from multiple network nodes, such as connected vehicles, are sent to a central node for aggregation, often to compute a function like the average of vehicle velocities. Traditional methods involve encoding each measurement separately and requiring coordinated channel access to decode and aggregate the data. A more efficient approach, known as over-the-air computation, leverages the superposition property of wireless channels, enabling simultaneous transmission of encoded signals and direct computation of their sum. This document introduces the concept of compute-forward schemes, which use structured codes to facilitate this approach. However, practical challenges such as synchronization limitations are addressed through lossy source coding, where measurements are compressed and transmitted with reduced precision. The edge node receives both digital compressed messages and an analogue combination of the measurements, enabling the reconstruction of a desired linear combination with tolerable distortion.

In the case of coded distributed computing, we consider the situation when we require distributing a computation task into many nodes in order to process large amounts of data with stringent latency requirements, as is the interest in V2X communications. For this setting, we consider the design of multivariate polynomial coding schemes which have recently been proved useful to trade-off between computation speed and communication costs. We first formulate the computation latency-communication trade-off in terms of the computation complexity and communication overheads required by coded computing approaches as compared to a single server uncoded computing system. Then, we propose two novel multivariate coded computing schemes supporting arbitrary matrix partitions. The proposed schemes are shown to improve the studied trade-off as compared to univariate schemes.

Table of Contents

1 Source-Channel coding for computation over the wireless medium

1.1 Introduction

In distributed decision-making scenarios involving multiple network nodes, such as connected vehicles, it becomes essential to gather independent (though occasionally correlated) measurements from these nodes. These measurements are transmitted over a wireless link to a central node for processing. Often, the central node is only concerned with a specific aggregate function of the data, such as the mean value or another function derived from the individual signals. For example, in vehicular networks, a roadside edge node might need to calculate the average position, speed, or acceleration of a group of vehicles to enhance traffic coordination, optimize flow, or support platooning.

A straightforward approach would involve each vehicle independently encoding its measurement (e.g., its velocity vector components) using error-correcting codes for reliability. These encoded signals would then be transmitted over the wireless medium, and the receiving node would decode each one individually before computing the average of all velocity measurements. However, with multiple nodes transmitting simultaneously, this method demands a coordinated channel access strategy to distinguish the encoded signals. This traditional process is known to be inefficient.

In contrast, a more advanced technique, known as *over-the-air computation*, offers a smarter alternative. This method encodes the signals so they can be transmitted simultaneously using the same time-frequency resources. The core principle of over-the-air computation exploits the natural superposition property of the wireless medium, enabling the signals to be combined in the air to directly compute a sum. This synergistic approach eliminates the need for separate decoding and simplifies the aggregation process.

1.2 Source and channel coding for computation

Over-the-air computation has been studied traditionally in the context of *channel coding*. However, it also applies to the dual setting of *source coding*, as we shall explain below.

1.2.1 Channel coding

In the channel coding setting, as depicted in Figure 1, the goal is to have encoders map digital messages m_1 and m_2 to encoded blocks of symbols $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, which get transmitted over a noisy wireless channel, before being received and processed by a decoder which directly attempts to recover a linear combination $\boldsymbol{w} = a_1\boldsymbol{x}_1 + a_2\boldsymbol{x}_2$ of the encoded blocks (which translates back to some linear combination of the messages m_1 and m_2, since the codebooks are linear). This setting was proposed in [1] and exhaustively studied in follow-up publications, including [2, 3, 4].

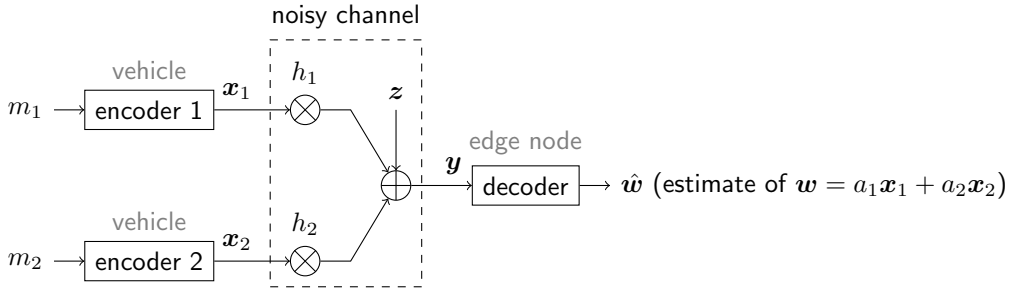

Figure 1: Coded computation over the air (the "compute–forward" problem)

1.2.2 Source coding

The compute-forward scheme briefly presented above suffers from some practical limitations when applied in the context of over-the-air computation. Notably, the synchronization of encoded signals must be very precise (on a frame and sub-symbol level). A more viable approach in the context of vehicular communications is that of lossy source coding (compression). In the source coding setting (Figure 2) that is a counterpart of the compute-forward setting from Figure 1, the measurements $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are encoded into digital, lossily compressed messages m_1 and m_2, as depicted in Figure 2 below. These messages are forwarded over error-corrected digital links with a limited rate (limited number of bits per time unit), which we assume to be error-free (their specific implementation is not part of our system modeling).

In addition to the digital, quantized and compressed measurements m_1 and m_2, the nodes may also transmit an uncoded, analogue version of their measurements $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ over a helper channel on a dedicated frequency band. The edge node thus receives the digital messages m_1 and m_2 and, ideally, an analogue noisy linear combination $\boldsymbol{y} = h_1\boldsymbol{x}_1 + h_2\boldsymbol{x}_2 + \boldsymbol{z}$ of the measurements $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. The combination of these three observations is then used to reconstruct a desired linear combination $\hat{\boldsymbol{w}}$ up to some tolerable distortion margin D.

In the source coding setting, as depicted in Figure 2, the goal is to have encoders compress some (possibly correlated) source observations $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ into digital messages m_1 and m_2, that get conveyed via a rate-limited uplink to the decoder, which attempts to directly reconstruct a signal $\hat{\boldsymbol{w}}$ that approximates a linear combination $a_1\boldsymbol{x}_1 + a_2\boldsymbol{x}_2$ of the source observations to within a target distortion D. Additionally, a variable $\boldsymbol{y}$ that carries some side information about the sources $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, can assist the decoder.

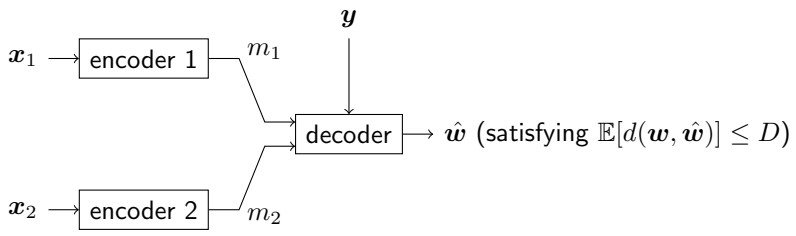

Figure 2: Distributed compression for sum recovery with side information (the "distributed lossy computation" problem)

1.2.3 Source and channel coding

Compute–forward schemes are based on the use of structured codes with an algebraic structure, such as lattice codes or linear codes. Also note that compute-forward techniques were developed in the context of relay communication [1], rather than distributed sensing. Hence, instead of measurement sources (correlated data sources), the compute–forward results were derived for the encoding of digital messages (bit messages). For a direct application of compute–forward results to our problem, we would need to lossily compress the analogue sources (denoted as $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ in Figure 2) into bit streams at the nodes (vehicles) before applying the appropriate error-correcting codes.

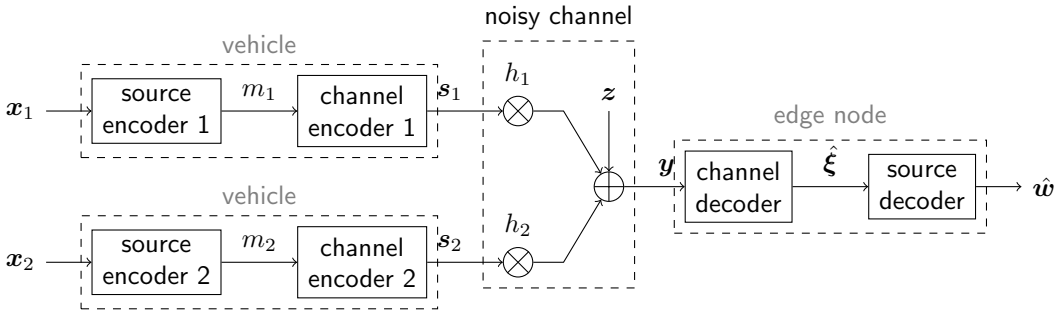

Figure 3: Combined (but separate) source and channel coding for over-the-air transmission

In the separate approach depicted in Figure 3, the vehicle nodes perform a source coding step, followed by a channel coding step. At the receiver side, the edge node performs these two steps in reverse: a channel decoding step, followed by a source decoding step. The former attempts to losslessly recover a linear combination $\boldsymbol{\xi} = a_1\boldsymbol{s}_1 + a_2\boldsymbol{s}_2$ of the additive codewords $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$.

A more efficient (but computationally more complex) alternative is to perform *joint* source-channel coding. In this variant, as shown in Figure 4, the processing blocks are merged in pairs, so as to yield the codewords. There is no requirement, in this setting, that source coding and channel coding be performed separately. Accordingly, there is no requirement that the decoder first recovers a linear combination $\boldsymbol{\xi}$ before computing the desired (lossy) linear combination of sources $\hat{\boldsymbol{w}}$.

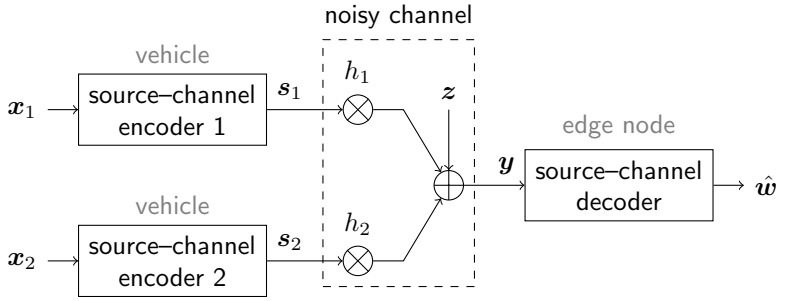

Figure 4: Joint source and channel coding for over-the-air transmission

The next section explains the theoretical foundations of distributed lossy computation (DLC) and presents an important theoretical result from our conference publication [5].

1.3 Theoretical foundations of over-the-air computing

We have considered the problems of CF and DLC, where the goal is to recover one or more linear combinations of the codewords (in case of CF) or the sources (in case of DLC) at the decoder. For certain configurations, it is known that codes with algebraic structure can outperform i.i.d. codebooks. For the special case of finite-alphabet sources, recent work [2, 3] has demonstrated how to incorporate joint typicality decoding alongside linear encoding and binning. Following the footsteps of [4] (for CF), our work uses a similar discretization approach to derive a dual rate region for the DLC setting, that covers both integer- and real-valued sources. As a case study, the rate region is evaluated for the Gaussian case. The resulting joint-typicality-based rate region recovers and generalizes the best-known rate region for this scenario, based on lattice encoding and sequential decoding.

1.3.1 Achievable rate region for CF

Consider the CF setting as depicted in Figure 1. In this problem, multiple messages m_1 through m_K are encoded and sent over a noisy uplink channel, and a receiver attempts to simultaneously (or sequentially) recover multiple linear combinations of the transmitted codewords in a reliable manner. The main result of [4] is the derivation of the following CF rate region:

$$\mathscr{R}_{\mathrm{CF}}(\mathbf{A}) = \bigcup_{\mathbf{B}} \bigcap_{M} \bigcup_{\mathcal{S}} \bigcap_{\mathcal{T}} \Big\{ (R_1, \ldots, R_K) \in \mathbb{R}_+^K : \sum_{k \in \mathcal{T}} R_k < \mathcal{H}([\boldsymbol{u}]_{\mathcal{T}}) - \mathcal{H}_{\mathbf{B}}(\boldsymbol{u}|Y) + \inf_{\boldsymbol{C} \in \mathscr{C}_{\mathbb{A}}(M)} \mathcal{H}_{\boldsymbol{CB}}(\boldsymbol{u}|Y) \Big\}. \quad (1)$$

Here, we assume some arbitrary joint probability distribution $P_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K,\boldsymbol{y}}$ that is compatible with the source distributions and the analogue helper channel $P_{\boldsymbol{y}|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K}$. In addition,

- $\mathbf{B} \in \mathbb{A}^{\ell \times K}, \ell \in \{L, \ldots, K\}$ generates a superlattice $\Lambda(\mathbf{B}) \supset \Lambda(\mathbf{A})$
- M iterates over all size-ℓ matroids except the full matroid
- $\mathcal{S}$ iterates over bases of the dual matroid M^*
- $\mathcal{T}$ iterates over bases of the matroid of $[\mathbf{B}]_{\mathcal{S}}$.

Since in its general form, this result is rather opaque, it is instructive to evaluate this rate region for a relevant special case. If we consider the special case of two users ($K = 2$) and a single equation to be reconstructed ($L = 1$), as well as Gaussian auxiliaries and an AWGN channel, then the rate region simplifies to the following expression:

$$R_{\mathrm{CF}}(\mathbf{a}) = \frac{1}{2} \log \left(\frac{1 + P\|\boldsymbol{h}\|^2}{\|\mathbf{a}\|^2 + P\big(\|\mathbf{a}\|^2\|\boldsymbol{h}\|^2 - (\mathbf{a}^{\mathsf{T}}\boldsymbol{h})^2\big)} \right)^+, \quad (2)$$

Here, we have set $\mathbf{A} = \mathbf{B} = \mathbf{a}^{\mathsf{T}} = [a_1 \ a_2]$ and $\boldsymbol{h}^{\mathsf{T}} = [h_1 \ h_2]$.

1.3.2 Achievable rate region for DLC

Now consider the dual problem in which multiple signal sources $\boldsymbol{x}_1$ through $\boldsymbol{x}_K$ are encoded in a distributed manner into rate-limited representations, and a receiver attempts to simultaneously (or sequentially) recover multiple linear combinations of these sources from said representations in a reliable manner, while at the same time satisfying a distortion constraint for each desired linear combination:

$$\mathbb{E}[d(W_\ell, \hat{W}_\ell)] \leq D_\ell, \quad \text{for } \ell = 1, \ldots, L. \tag{3}$$

Our main result in [5] consists in a rate region characterization, i.e., a direct formula that is an analogue to (1) and that describes a set of rate tuples and an associated coding scheme at which the DLC system can operate reliably, in the sense of arbitrarily low probability of infringing the distortion constraints. Specifically, the rate region is a function of the weight matrix $\boldsymbol{A}$, defined as the following set of points:

$$\mathscr{R}(\mathbf{A}) = \bigcup_{\mathbf{B}} \bigcap_{M} \bigcup_{\mathcal{S}} \bigcap_{\mathcal{T}} \Big\{ (R_1, \ldots, R_K) \in \mathbb{R}_+^K : \sum_{k \in \mathcal{T}} R_k > -\mathcal{H}([\boldsymbol{u}]_{\mathcal{T}} | [\boldsymbol{x}]_{\mathcal{T}}) + \mathcal{H}_{\mathbf{B}}(\boldsymbol{u}|Y) - \inf_{\boldsymbol{C} \in \mathscr{C}_{\mathbb{A}}(M)} \mathcal{H}_{\boldsymbol{CB}}(\boldsymbol{u}|Y) \Big\}.$$

Here, we assume some arbitrary joint probability distribution $P_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K,\boldsymbol{y}}$ that is compatible with the source distributions and the analog helper channel $P_{\boldsymbol{y}|\boldsymbol{x}_1,\ldots,\boldsymbol{x}_K}$. In addition, the set operations follow the same constraints as in the dual CF expression (1), i.e.,

- $\mathbf{B} \in \mathbb{A}^{\ell \times K}, \ell \in \{L, \ldots, K\}$ generates a superlattice $\Lambda(\mathbf{B}) \supset \Lambda(\mathbf{A})$
- M iterates over all size-ℓ matroids except the full matroid
- $\mathcal{S}$ iterates over bases of the dual matroid M^*
- $\mathcal{T}$ iterates over bases of the matroid of $[\mathbf{B}]_{\mathcal{S}}$.

Our main result consists in a proof that the above defined set $\mathscr{R}(\mathbf{A})$ is a set of achievable rates for the messages $m_1, \ldots, m_L$. Once again, it is instructive to evaluate this rate region for a relevant special case. If we consider the special case of two users ($K = 2$) and a single equation to be reconstructed ($L = 1$), then the rate region, expressed as a function of a target quadratic distortion $\mathbb{E}[(W - \hat{W})^2]$, boils down to the following formula:

$$R_{\mathrm{DLC}}(D) = \frac{1}{2} \log \left(\frac{\|\mathbf{a}\|^2 + P\big(\|\mathbf{a}\|^2 \|\boldsymbol{h}\|^2 - \rho(\mathbf{a}^\mathsf{T}\boldsymbol{h})^2\big)}{(1 + P\|\boldsymbol{h}\|^2)(1 - \rho)} \right), \tag{4}$$

where

$$\rho = \left(1 - \frac{D}{P\|\boldsymbol{h}\|^2}\right)^+ \tag{5}$$

and where $\mathbf{A} = \mathbf{B} = \mathbf{a}^\mathsf{T} = [a_1 \ a_2]$, $\boldsymbol{h}^\mathsf{T} = [h_1 \ h_2]$ and $\mathbb{E}[\boldsymbol{x}\boldsymbol{x}^\dagger]P\mathsf{I}$. Unsurprisingly, this formula is very reminiscent of the already well-known rate formula of the two-user compute-forward problem (2). In a sense, Equation (5) can be considered a *dual*, source-coding counterpart to the classic compute-forward result.

1.4 Conclusion

The presented work [5] lays the theoretical foundations for efficient distributed lossy computation (DLC) schemes by exhaustively characterizing a unified rate region that encompasses (and sometimes surpasses) previously known results. Together with earlier work on CF achievable rate regions, these results lay the groundwork for studying the joint source-channel coding problem for computation and elucidating fundamental tradeoffs between analog and digital channel resource allocation in the context of vehicular communication, as well as for the development of practical coded computation coding schemes.

2 Communication-Efficient Fast Distributed Computation

Efficient computation solutions are essential for V2X systems due to the need for real-time, high-speed communication and decision-making. These systems are designed to improve traffic safety, enhance vehicle autonomy, and optimize traffic management by enabling vehicles to communicate with each other, infrastructure, and various devices. In such dynamic and critical environments, timely data processing is crucial. Delays in computation could lead to accidents, inefficient traffic flow, or missed opportunities for safety interventions.

V2X systems need to process large volumes of data generated by vehicles' sensors, cameras, and communication devices. However, the limited processing power and storage capacity of vehicles or roadside units can hinder the ability to perform these complex computations in real time. Environmental factors, like signal interference or loss caused by urban structures or weather, further complicate the situation by impacting data transmission quality and computational accuracy.

As more and more data needs to be processed with smaller latency, it becomes essential to split the job into multiple subtasks and execute them on multiple servers in parallel. These servers might be locally available or otherwise remotely available. In any case, we shall consider clusters of computation nodes, i.e., servers including small, low-end, and unreliable computational nodes which are severely affected by *"system noise"*, i.e., faulty behaviors due to computation or memory bottlenecks, load imbalance, resource contention, hardware issues, etc [6]. As a result, task completion times of individual workers become largely unpredictable, and the slowest workers dominate the overall computation time. This is referred to in the literature as the *straggler problem*. The code computing framework studied in this work can play a crucial role in enhancing computations in V2X systems.

2.1 Introduction

Coded computing solutions are available for quite general family of polynomial functions. However, more efficient algorithms can be found by focusing on a particular important set of functions. In this work, we focus on Matrix-Matrix multiplication. Matrix-matrix multiplication is a key operation in machine learning, however handling large datasets, as is usually the case in V2X applications, on a single local computation unit within a reasonable time frame, is usually unfeasible. To address this, distributed computing across multiple servers is required, either locally or more frequently outsourcing the computations to remote servers. However, modern servers often face "system noise," which causes unreliable task completion times and results in the "straggler problem," where slow workers dominate computation time.

To tackle this issue, we adopt the coded computing framework [7, 8, 9, 10], which uses maximum distance separable (MDS) codes. Unlike traditional methods that repeat tasks, coded computing allows delayed tasks to be replaced, resulting in faster completion times. Univariate polynomial codes, in particular, offer efficient decoding while balancing computational complexity and communication resources. For further details see, [10], and [11]. Follow-up works extended the results to secure matrix-matrix multiplications [12, 13], matrix chain multiplications[14], among others.

Table 1: Computation complexity and upload/download communications overheads for different coded distributed computing methods

Method*	R_{th}	δ	$\delta_{u,0}$	$\delta_{u,1}$	δ_d
epc	$p_1p_2p_0 + p_1 - 1$	$\frac{p_1-1}{p_0p_1p_2}$	$p_2 - 1 + p_2\delta$	$p_0 - 1 + p_0\delta$	$p_1 - 1 + p_1\delta$
Bi0	$p_0p_2p_1 + p_0(p_1 - 1)$	$\frac{p_1-1}{p_0p_1}$	δ	$p_0 - 1 + p_0\delta$	$p_1 - 1 + p_1\delta$
Bi2	$p_0p_1p_2 + p_2(p_1 - 1)$	$\frac{p_1-1}{p_1p_2}$	$p_2 - 1 + p_2\delta$	δ	$p_1 - 1 + p_1\delta$
Tri	$p_0p_2p_1 + p_0p_2(p_1 - 1)$	$\frac{p_1-1}{p_1}$	δ	δ	$p_1 - 1 + p_1\delta$

*See following subsections for the definition of these abbreviations.

Building on prior works [15, 16, 17, 18, 19], recent research has extended these concepts to secure matrix multiplication and matrix-chain multiplications. Some approaches also assign multiple subtasks to each worker, enabling the exploitation of partial work done by stragglers. However, univariate codes suffer from inefficiencies in upload communication costs. To address this, bivariate polynomial codes were introduced, but they do not support generalized partition schemes. In this work, we extend bivariate codes to accommodate generalized matrix partitions and introduce two novel multivariate schemes that improve the trade-off between computation complexity and communication overhead.

2.2 System Model and Problem Formulation

We consider the problem of outsourcing the task of multiplying two large matrices $M_0 \in \mathbb{F}^{r_0 \times r_1}$ and $M_1 \in \mathbb{F}^{r_1 \times r_2}$ from a centralized master node to a more computationally powerful entity with N parallel computation nodes/workers, see Figure 5. Matrix M_0 is partitioned into p_0 partitions vertically and p_1 partitions horizontally. Similarly, matrix M_1 is partitioned into p_1 partitions vertically, and p_2 partitions horizontally

$$M_i \;=\; \begin{bmatrix} M_i^{0,0} & \cdots & M_i^{0,p_{i+1}-1} \\ \vdots & \ddots & \vdots \\ M_i^{p_i-1,0} & \cdots & M_i^{p_i-1,p_{i+1}-1} \end{bmatrix}$$

for $i \in \{0, 1\}$. The result of the multiplication of M_0 and M_1, as a function of its matrix blocks, can be written as

$$M = M_0M_1 \;=\; \begin{bmatrix} M^{0,0} & \cdots & M^{0,p_2-1} \\ \vdots & \ddots & \vdots \\ M^{p_0-1,0} & \cdots & M^{p_0-1,p_2-1} \end{bmatrix} \tag{6}$$

with $M^{n_0,n_2} = \sum_{n_1=0}^{p_1-1} M^{n_0,n_1}M_1^{n_1,n_2}$.

Although coded computing is an efficient method to achieve lower *computation latencies*, it incurs *communication* and *computation complexity overheads*.

We define the *computation complexity overhead*, δ, as the increment in computation complexity, i.e., number of element-wise multiplications, required by coded computing C^{CC}, relative to the computation complexity in a single server, C^{SS}. That is, $C^{\mathsf{CC}} = C^{\mathsf{SS}}(1 + \delta)$.

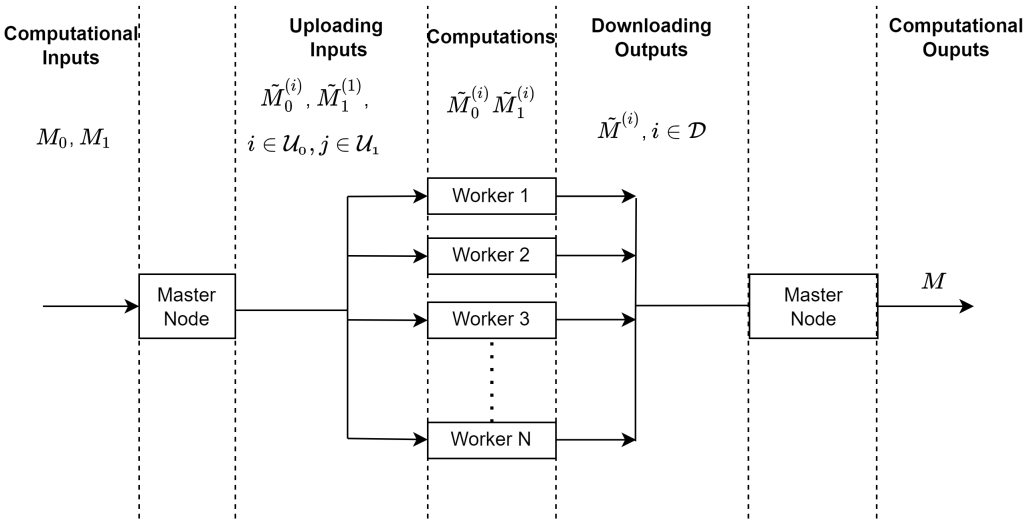

Figure 5: Distributed computational system.

The *download communication overhead*, δ_d, is similarly defined as the increment of the cost of communicating the block matrix products from the workers to the master in coded computing, C_d^{CC}, relative to the cost of communicating the original product result from a single server, C_d^{SS}. That is $C_d^{\mathsf{CC}} = C_d^{\mathsf{SS}}(1+\delta_d)$.

The *upload communication overhead* is similarly defined as the increment in the cost of communicating the coded block matrices from the master to the workers, relative to the cost of communicating the original matrices M_0 and M_1 to a single server. That is $C_{U,0}^{\mathsf{CC}} = C_{U,0}^{\mathsf{SS}}(1+\delta_{u,0})$, and $C_{U,1}^{\mathsf{CC}} = C_{U,1}^{\mathsf{SS}}(1+\delta_{u,1})$

2.3 Univariate Polynomial Codes

As a reference, we first provide the computation complexity and communication overheads for a more general version of the univariate polynomial coding schemes considered in the literature, e.g., entangled polynomial codes presented in [11, 20] and [10]. The original coding scheme was described assuming that only one coded block matrix product is assigned to each worker. Here, we describe its direct extension to support multiple coded computations per worker that are executed sequentially. This extension provides more flexibility to the system design, as now the number of matrix partitions is not limited by the number of parallel workers in the system.

With entangled polynomial codes (epc), the block matrices $M_0^{(b_0,b_1)}$ and $M_1^{(b_1,b_2)}$ are encoded with the polynomials

$$\begin{aligned}\tilde{M}_0^{\mathsf{epc}}(x) &= \sum_{b_0=0}^{p_0-1}\sum_{b_1=0}^{p_1-1} M_0^{(b_0,b_1)} x^{p_1 p_2 b_0 + b_1} \\ \tilde{M}_1^{\mathsf{epc}}(x) &= \sum_{b_2=0}^{p_2-1}\sum_{b_1=0}^{p_1-1} M_1^{(p_1-1-b_1,b_2)} x^{p_1 b_2 + b_1}.\end{aligned}$$

Observe that within the coefficients of the univariate product polynomial $\tilde{M}^{\mathsf{epc}}(x) =$

$\tilde{M}_0^{\mathsf{epc}}(x)\tilde{M}_1^{\mathsf{epc}}(x)$, we can find the matrix blocks of the target matrix product. In particular, $M^{n_0,n_2} = \sum_{n_1=0}^{p_1-1} M_0^{n_0,n_1} M_1^{n_1,n_2}$ is given by the coefficient of the monomial $x^{p_1p_2n_0+p_1n_2+p_1-1}$ of $\tilde{M}^{\mathsf{epc}}(x)$. Moreover, the product polynomial has degree $p_1p_2p_0+p_1-2$ and thus, via polynomial interpolation we can recover its coefficients from $R_{th}^{\mathsf{epc}} = p_1p_2p_0+p_1-1$ evaluations of $M(x)$.

2.4 Multivariate Coded Computing

In [21], a bivariate polynomial coding scheme was presented supporting partial computation at workers and improving the upload communication costs, as compared to univariate coding schemes for the case $p_1 = 1$. In contrast to univariate polynomial codes of degree d, for which, any set of $d+1$ distinct evaluation points guarantee decodability, for multi-variate polynomials, there exists only a few known sets of multi-variate evaluation points for which decodability can be guaranteed. With an independent random choice of the evaluation points we can achieve almost decodability, that is, decodability with probability increasing with the size of the operation field. The interested reader is referred to [22] for details on multivariate polynomial interpolation and to [21, 23] for its application to coded computing. However, as we will see here, to achieve low upload communications overheads, the evaluation set must have structure. For simplicity, in this work we restrict the analysis to the multivariate Cartesian product evaluation set. The Cartesian product set guarantees decodability and provides the lowest possible upload communication overheads. However, it may require large storage capacity at the workers in offline upload communication settings, i.e., if all the subtask inputs are uploaded before any computation starts at workers.

For the tri-variate polynomial coding scheme, the block matrix inputs are encoded with

$$\begin{aligned}\tilde{M}_0^{\mathsf{Tri}}(x,y) &= \sum_{b_0=0}^{p_0-1}\sum_{b_1=0}^{p_1-1} M_0^{(b_0,b_1)} x^{b_0} y^{b_1} \\ \tilde{M}_1^{\mathsf{Tri}}(y,z) &= \sum_{b_2=0}^{p_2-1}\sum_{b_1=0}^{p_1-1} M_1^{(p_1-1-b_1,b_2)} y^{b_1} z^{b_2}.\end{aligned}$$

Observe that within the coefficients of the tri-variate product polynomial $\tilde{M}^{\mathsf{Tri}}(x,y,z) = \tilde{M}_0^{\mathsf{Tri}}(x,y)\tilde{M}_1^{\mathsf{Tri}}(y,z)$, we can find the matrix blocks of the desired matrix product. Specifically, M^{n_0,n_2} is given by the coefficient of the monomial $x^{n_0}y^{p_1-1}z^{n_2}$ in $\tilde{M}^{\mathsf{Tri}}(x,y,z)$. Moreover, the product polynomial has degree p_0-1 in x, p_2-1 in z, and $2p_1-2$ in y. Thus, via multivariate polynomial interpolation it is potentially possible to recover its coefficients from $R_{th}^{\mathsf{Tri}} = p_0p_2(2p_1-1)$ evaluations of $\tilde{M}^{\mathsf{Tri}}(x,y,z)$. To obtain these evaluations, we consider the Cartesian product set, $\mathcal{X}\times\mathcal{Y}\times\mathcal{Z}$, with $|\mathcal{X}| = p_0$, $|\mathcal{Y}| = 2p_1-1$ and $|\mathcal{Z}| = p_2$. Then, the server broadcasts $\tilde{M}_0^{\mathsf{Tri}}(x,y)$ for each $(x,y)\in\mathcal{X}\times\mathcal{Y}$, and thus $R_0^{\mathsf{Tri}} = (2p_1-1)p_0 = \frac{R_{th}^{\mathsf{Tri}}}{p_2}$, and $\tilde{M}_1^{\mathsf{Tri}}(y,z)$ for each $(y,z)\in\mathcal{Y}\times\mathcal{Z}$, and thus $R_1^{\mathsf{Tri}} = (2p_1-1)p_2 = \frac{R_{th}^{\mathsf{Tri}}}{p_0}$. Then workers, coordinated by the master can compute one-by-one all the products $\tilde{M}^{\mathsf{Tri}}(x,y,z)$.

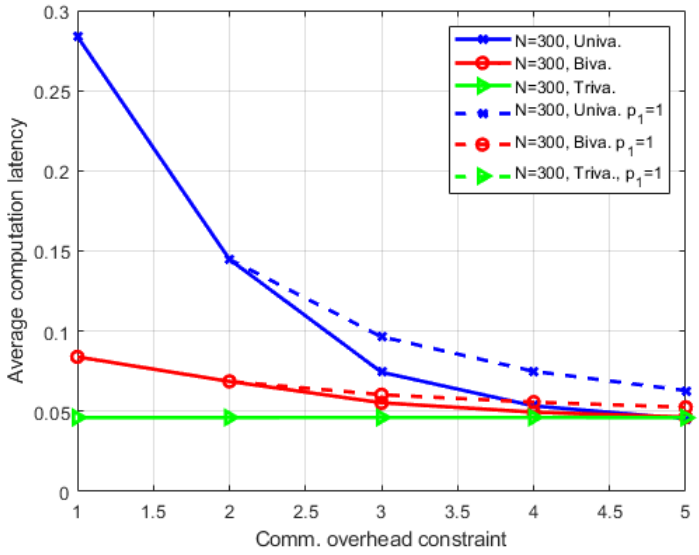

Figure 6: Average computation latency as a function of the communication overhead constraint for the different coding schemes.

2.5 Numerical Results

We are interested in characterizing the trade-off between the average computation latency and the upload/download communication overhead. For that, we search for the partition scheme (p_0, p_1, p_2) that minimizes the average computation latency $T(p_0, p_1, p_2)$ while guaranteeing bounded upload/download communication overheads. That is,

$$\min_{p_0,p_1,p_2} \quad T(p_0, p_1, p_2) \tag{7}$$

s.t. $\delta_{u,0} \leq \hat{\delta}_{u,0}$, $\delta_{u,1} \leq \hat{\delta}_{u,1}$, and $\delta_d \leq \hat{\delta}_d$, where $\hat{\delta}_{u,0}$, $\hat{\delta}_{u,1}$ and $\hat{\delta}_d$ are fixed system constraints on upload and download costs. We solve the problem via Monte Carlo simulations. To model the subtask completion time at workers, we consider the shifted exponential model, which is a common model employed in distributed computing [7, 24]. Suppose that the completion of the full task in a single server has an average completion time given by $T_{\text{full}} = T_0 + T_E$ where T_0 is a constant and T_E is exponentially distributed random variable with parameter λ. Then, by partitioning the full task into K subtasks, the cumulative distribution function of the completion time of each individual subtask T_i is

$$\mathsf{F}_i(t) \quad = \quad P(T_i \leq t) = 1 - \exp\left(-\lambda K \left(t - \frac{T_0}{K}\right)\right),$$

and thus, T_i is the sum of a constant $T_{0,i} = \frac{T_0}{K}$ plus an exponential random variable with parameter $\lambda_i = \lambda K$.

We search over all partition schemes (p_0, p_1, p_2) that satisfy all the communication overhead constraints. Here we consider the particular case when $\hat{\delta}_{u,0} = \hat{\delta}_{u,1} = \hat{\delta}_d = \hat{\delta}_{u,d}$. To reduce the search space, we further limit the maximum partition levels to satisfy $p_0 \leq \hat{p}_0$ and $p_2 \leq \hat{p}_2$, while p_1 is left unbounded. As an illustrative case, we choose $\frac{1}{\lambda} = 10$ and $T_0 = \frac{1}{10\lambda}$, $N = 300$

workers and $\hat{p}_0 = \hat{p}_2 = 10$. In Figure 6, we show the average computation latency as a function of the communication overhead constraint for the three coded computing schemes. As a reference, we also show the results obtained by forcing $p_1 = 1$ (dotted lines). We can observe that the tri-variate scheme obtains the best performance in this situation, while the univariate scheme is heavily penalized at low communication overheads.

2.6 Conclusion

We formulated the computation latency versus communication/complexity overheads trade off analysis in distributed coded computing systems. We presented a novel multivariate polynomial coding schemes supporting arbitrary matrix partitions, which, compared to univariate polynomial codes, require lower upload communication overheads at the cost of increasing the computation complexity.

3 Final remarks

This deliverable explored two advanced technological solutions to enhance computation efficiency in V2X systems: source-channel coding for resource-efficient wireless computation and coded computing for effective distributed computation under faulty conditions.

For wireless computation, we introduced over-the-air computation and compute-forward schemes, leveraging the superposition property of wireless channels for simultaneous transmission and computation of encoded signals. Practical challenges, such as synchronization, are addressed with lossy source coding, enabling the transmission of compressed measurements with reduced precision while still supporting accurate aggregation. In distributed coded computing, we analyzed the trade-off between computation latency and communication/complexity overheads. The proposed multivariate polynomial coding schemes, supporting arbitrary matrix partitions, offer significant improvements over univariate polynomial codes, requiring lower upload communication overheads at the cost of increased computation complexity. Together, these contributions provide a comprehensive framework for efficient distributed computation in V2X systems, improving both scalability and performance in distributed environments.

References

[1] B. Nazer and M. Gastpar, "Compute-and-forward: Harnessing interference through structured codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6463–6486, 2011.

[2] S. H. Lim, C. Feng, A. Pastore, B. Nazer, and M. Gastpar, "A joint typicality approach to compute–forward," *IEEE Transactions on Information Theory*, vol. 64, no. 12, pp. 7657–7685, 2018.

[3] ——, "Compute-forward for dmcs: Simultaneous decoding of multiple combinations," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6242–6255, 2020.

[4] A. Pastore, S. H. Lim, C. Feng, B. Nazer, and M. Gastpar, "A unified discretization approach to compute–forward: From discrete to continuous inputs," *IEEE Transactions on Information Theory*, vol. 69, no. 1, pp. 1–46, 2023.

[5] ——, "Distributed lossy computation with structured codes: From discrete to continuous sources," in *2023 IEEE International Symposium on Information Theory (ISIT)*, 2023, pp. 1681–1686.

[6] J. Dean and L. A. Barroso, "The tail at scale," *Communications of the ACM*, vol. 56, pp. 74–80, 2013. [Online]. Available: http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext

[7] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, and K. Ramchandran, "Speeding up distributed machine learning using codes," *IEEE Transactions on Information Theory*, vol. 64, no. 3, pp. 1514–1529, 2017.

[8] K. Lee, C. Suh, and K. Ramchandran, "High-dimensional coded matrix multiplication," in *2017 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2017, pp. 2418–2422.

[9] Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, "Polynomial codes: An optimal design for high-dimensional coded matrix multiplication," in *Proc. Int'l Conf. on Neural Information Processing Systems*, 2017, pp. 4406–4416.

[10] ——, "Straggler mitigation in distributed matrix multiplication: Fundamental limits and optimal coding," *IEEE Transactions on Information Theory*, vol. 66, no. 3, pp. 1920–1933, 2020.

[11] S. Dutta, M. Fahim, F. Haddadpour, H. Jeong, V. Cadambe, and P. Grover, "On the optimal recovery threshold of coded matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 1, pp. 278–301, 2019.

[12] W.-T. Chang and R. Tandon, "On the capacity of secure distributed matrix multiplication," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[13] R. G. L. D'Oliveira, S. El Rouayheb, and D. Karpuk, "Gasp codes for secure distributed matrix multiplication," *IEEE Transactions on Information Theory*, vol. 66, no. 7, pp. 4038–4050, 2020.

[14] X. Fan, A. Saldivia, P. Soto, and J. Li, "Coded matrix chain multiplication," in *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*, 2021, pp. 1–6.

[15] M. M. Amiri and D. Gündüz, "Computation scheduling for distributed machine learning with straggling workers," *IEEE Transactions on Signal Processing*, vol. 67, no. 24, pp. 6270–6284, 2019.

[16] L. Song, L. Tang, and Y. Wu, "An optimal coded matrix multiplication scheme for leveraging partial stragglers," in *2023 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2023, pp. 1741–1744.

[17] S. Kiani, N. Ferdinand, and S. C. Draper, "Exploitation of stragglers in coded computation," in *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2018, pp. 1988–1992.

[18] E. Ozfatura, S. Ulukus, and D. Gündüz, "Straggler-aware distributed learning: Communication–computation latency trade-off," *Entropy*, vol. 22, no. 5, p. 544, 2020.

[19] A. Reisizadeh, S. Prakash, R. Pedarsani, and A. S. Avestimehr, "Coded computation over heterogeneous clusters," *IEEE Transactions on Information Theory*, vol. 65, no. 7, pp. 4227–4242, 2019.

[20] S. Dutta, Z. Bai, H. Jeong, T. M. Low, and P. Grover, "A unified coded deep neural network training strategy based on generalized polydot codes for matrix multiplication," *arXiv preprint arXiv:1811.10751*, 2019.

[21] B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate polynomial coding for efficient distributed matrix multiplication," *IEEE Journal on Selected Areas in Information Theory*, vol. 2, no. 3, pp. 814–829, 2021.

[22] R. A. Lorentz, *Multivariate Birkhoff Interpolation*. Springer, 2006.

[23] B. Hasırcıoğlu, J. Gómez-Vilardebó, and D. Gündüz, "Bivariate polynomial codes for secure distributed matrix multiplication," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 3, pp. 955–967, 2022.

[24] G. Liang and U. C. Kozat, "Tofec: Achieving optimal throughput-delay trade-off of cloud storage using erasure codes," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 826–834.