Energy-Efficient Edge-Facilitated Wireless Collaborative Computing using Map-Reduce

Antoine Paris, Hamed Mirghasemi, Ivan Stupia and Luc Vandendorpe
ICTEAM/ELEN/CoSy, UCLouvain, Louvain-la-Neuve, Belgium
Email: {antoine.paris, seyed.mirghasemi, ivan.stupia, luc.vandendorpe}@uclouvain.be

***Abstract*—In this work, a heterogeneous set of wireless devices sharing a common access point collaborates to perform a set of tasks. Using the Map-Reduce distributed computing framework, the tasks are optimally distributed amongst the nodes with the objective of minimizing the total energy consumption of the nodes while satisfying a latency constraint. The derived optimal collaborative-computing scheme takes into account both the computing capabilities of the nodes and the strength of their communication links. Numerical simulations illustrate the benefits of the proposed optimal collaborative-computing scheme over a blind collaborative-computing scheme and the non-collaborative scenario, both in term of energy savings and achievable latency. The proposed optimal scheme also exhibits the interesting feature of allowing to trade energy for latency, and vice versa.**

***Index Terms*—wireless collaborative computing, distributed computing, Map-Reduce, energy-efficiency, fog computing.**

I. Introduction

We consider a set of K nodes, indexed by the letter $k \in [K]$, sharing a common access point (AP), base station (BS) or gateway in the context of low-power wide-area networks (LPWAN). A node can be any device able to wirelessly communicate with the AP and perform local computations. Under a given latency constraint τ, each node k wants to compute a certain function $\phi(d_k, w)$ where $d_k \in [0,1]^D$ is some D-bit local information available to node k (e.g., sensed information or local state) and $w \in [0,1]^L$ is a L-bit file with $L \gg D$ bits (e.g., a dataset) that might, for instance, be cached at the AP [1]. In the context of smart cities or smart buildings, w could be the result of the aggregation over space and time of information sensed from the environment through a network of sensors (e.g., traffic density or temperature) whereas the nodes could be actuators having some local state d_k that periodically need to perform some latency-sensitive computations to decide whether to take some actions (e.g., smart traffic lights or smart thermostats). Other applications include fog computing, mobile crowd-sensing or wireless distributed systems.

Owing to the unacceptable delay of mobile cloud computing (MCC), and in the absence of a mobile edge computing (MEC) server nearby, the computing and storage capabilities of wireless devices are limited. It might thus be the case, for example, that w is too large to fit in the memory of a single node, or that the nodes are not individually powerful enough to satisfy the latency constraint. To overcome those limitations, a collaborative-computing scheme based on the Map-Reduce distributed computing framework [2] is proposed. This distributed computing model involves local computations at the nodes and communication between the nodes via the AP (i.e., the edge of the network is *facilitating* the communication between the nodes). In some applications, one could also deliberately avoid the use of a third-party owned MCC or MEC for privacy reasons.

The problem setup and distributed computing model used in this work essentially follows [3], with the exceptions that we consider the set of nodes to be *heterogeneous* in term of computing capabilities and channel strengths and add an explicit latency constraint. Prior works on wireless distributed computing using Map-Reduce, e.g., [3]–[6], mainly focus on *coded distributed computing* (CDC) and study the trade-off between the computation and communication loads incurred by the collaboration. Motivated by the fact that wireless devices are often limited in energy and that most computing tasks are accompanied by a latency constraint, this work shifts focus towards optimizing the collaborative-computing scheme to minimize the total energy consumption of the nodes, while satisfying the latency constraint. To our knowledge, this work is the first to incorporate energy-efficiency considerations in a Map-Reduce based wireless collaborative-computing scheme.

Throughout this paper, we assume that there is some central entity (e.g., the AP) having perfect knowledge of the channel state information (CSI) and computing capabilities of all the nodes that coordinates the collaboration.

Section II starts by describing in details the collaborative computing model and the energy and time consumption models for both local computation and communication between the nodes. Next, Sec. III formulates the problem as an optimization problem that turns out to be convex and to have a semi-closed form solution, given in Sec. IV. Section V then benchmarks the performances of the optimal collaborative-computing scheme against a blind collaborative-computing scheme and the non-collaborative scenario through numerical experiments. Finally, Sec. VI discusses the results obtained in this work and opportunities for future research.

II. System model

This section details the collaborative computing model used in this work, namely Map-Reduce, and quantifies the time and energy consumed by each phase of the collaboration.

A. Collaborative computing model

The tasks are shared between the K nodes according to the Map-Reduce framework [2]. First, we assume that the file w

can be arbitrarily divided in K smaller files w_k (one for each node) of size $l_k \in \mathbb{R}_{\geq 0}$ bits[1] such that $w_k \cap w_l = \emptyset$ for all $k \neq l$ and $w = \bigcup_{k=1}^{K} w_k$. We neglect the time and energy needed to transmit w_k from the AP to node k, for all $k \in [K]$.

To make collaboration between the nodes possible, we also assume that the local data $\{d_k\}_{k=1}^{K}$ were shared between all the nodes through the AP in a prior phase that we neglect in this work because D is assumed to be relatively small.

During the first phase of the Map-Reduce framework, namely the *Map phase*, each node k computes intermediate values

$$v_{k,l} = g_k(d_l, w_k), \qquad l \in [K]$$

where $g_k : [0,1]^D \times [0,1]^{l_k} \to [0,1]^{(l_k/L)T}$ is the *Map function* executed at node k. The size (in bits) of the intermediate values produced at node k is assumed to be proportional to l_k. Each node k thus computes intermediate values for all the other nodes (i.e., $v_{k,l}$ for all $l \neq k$) and for itself (i.e., $v_{k,k}$) using the part w_k of w received from the AP.

Next, the nodes exchange intermediate values with each other in the so-called *Shuffle phase*. In the absence of a reliable model for the energy consumption of coding operations and for the sake of tractability in order to lay out the foundations of the proposed collaborative-computing framework, coded Shuffling as discussed in [3]–[6] is not considered in this work. In this simplified Shuffle phase, each node k transmits the intermediate values $v_{k,l} = g_k(d_l, w_k)$ to node l via the AP, for all $l \neq k$. Node k thus needs to transmit $(K-1)(l_k/L)T$ bits of intermediate values to the AP.

Finally, during the *Reduce phase*, each node l combines the T bits of intermediate values $\{v_{k,l} = g_k(d_l, w_k)\}_{k=1}^{K}$ as

$$\phi(d_l, w) = h(g_1(d_l, w_1), g_2(d_l, w_2), \ldots, g_K(d_l, w_K))$$

where $h : [0,1]^T \to [0,1]^O$ is the *Reduce function*. The Map-Reduce distributed computing model is illustrated in Fig. 1.

B. Local computing model

During the Map and the Reduce phases, the nodes have to perform some local computations. The local computing model used in this work follows [7]. For both the Map and Reduce phases, the number of CPU cycles required to process 1-bit of input data is noted C_k while the amount of energy consumed per CPU cycle is noted P_k. The amounts of energy consumed at node k during the Map and Reduce phases are thus given by

$$E_k^{\text{MAP}} = (KD + l_k)C_k P_k \quad \text{and} \quad E_k^{\text{RED}} = TC_k P_k, \qquad (1)$$

respectively. Next, letting F_k be the number of CPU cycles per second at node k, the amounts of time required for the Map and the Reduce phases are given by

$$t_k^{\text{MAP}} = (KD + l_k)C_k / F_k \quad \text{and} \quad t_k^{\text{RED}} = TC_k / F_k, \qquad (2)$$

respectively. One can already observe that we can control the energy and time consumed at node k by the Map phase through the variable l_k. At the opposite, we don't have any control on the energy and time consumed by the Reduce phase. As a consequence, and because the Map and the Shuffle phases must be over before the Reduce phase can start, the time remaining for the Map and the Shuffle phases is given by $\tau - \max_k\{t_k^{\text{RED}}\}$, i.e., the slowest node reduces the available time τ by the amount of time it needs for the Reduce phase.

[1]In practice, l_k should be an integer multiple of the size of the smallest possible division of w. In this work, we relax this practical consideration to avoid dealing with integer programming later on. Note that $l_k = 0$ is also possible, in which case node k does not participate to the collaboration.

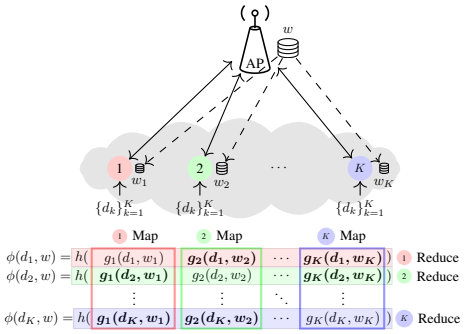


Fig. 1. Illustration of the the Map-Reduce distributed computing model. The computation of $\{\phi(d_k, w)\}_{k=1}^{K}$ is distributed between K nodes. During the Map phase, each node k computes intermediate values $\{g_k(d_l, w_k)\}_{l=1}^{K}$ (see framed columns). Next, during the Shuffle phase, the intermediate values in bold on the figure are transmitted via the AP to the nodes for which they have been computed. Finally, during the Reduce phase, each node l combines the intermediate values $\{g_k(d_l, w_k)\}_{k=1}^{K}$ to obtain $\phi(d_l, w)$ (see colored rows).

C. Communications from the nodes to the AP

During the Shuffle phase, nodes exchange intermediate values through the AP. This exchange thus involves both an uplink communication (nodes to AP) and a downlink communication (AP to nodes). In most applications however, it is reasonable to assume that the downlink rates are much larger than the uplink rates. For this reason, we neglect the time needed for the downlink communication in this work.

We assume that all the nodes can communicate in an orthogonal manner to the AP (e.g., through frequency division multiple access techniques). We also make the common assumption that the allowed latency τ is smaller than the channel coherence time. Let $h_k \in \mathbb{C}$ denote the wireless channel from node k to the AP, p_k the RF transmit power of node k, B the communication bandwidth, σ^2 the noise power at the AP in the bandwidth B, and Γ the SNR gap. The achievable uplink rate of node k is then given by

$$r_k(p_k) = B \log_2\left(1 + \tfrac{p_k |h_k|^2}{\Gamma \sigma^2}\right).$$

The time required by node k to transmit the $(K-1)(l_k/L)T$ bits of intermediate values to the AP is thus given by $t_k^{\text{SHU}} = \alpha l_k / r_k(p_k)$ where $\alpha = (K-1)T/L$ has been defined to ease notations. Then, inspired by [7], we define $f(x) = \sigma^2 \Gamma(2^{x/B} - 1)$, and write the energy consumed at node k to transmit the intermediate values as

$$E_k^{\text{SHU}} = p_k t_k^{\text{SHU}} = \tfrac{t_k^{\text{SHU}}}{|h_k|^2} f\left(\tfrac{\alpha l_k}{t_k^{\text{SHU}}}\right). \qquad (3)$$

Through the variables l_k and p_k (or, equivalently, t_k^{SHU}), we thus have control on the energy and time consumed at node k during the Shuffle phase.

III. Problem formulation

As mentioned in the introduction, the objective is to optimize the collaborative-computing scheme to minimize the total energy consumption of the nodes, while satisfying the latency constraint τ. This can be mathematically formulated as follows

$$\begin{aligned}
\underset{\{l_k\},\{t_k^{\text{SHU}}\}}{\text{minimize}} \quad & \textstyle\sum_{k=1}^K E_k^{\text{MAP}} + E_k^{\text{SHU}} + E_k^{\text{RED}} \\
\text{subject to} \quad & l_k, t_k^{\text{SHU}} \geq 0, \quad k \in [K] \\
& t_k^{\text{MAP}} + t_k^{\text{SHU}} \leq \tau - \max_k\{t_k^{\text{RED}}\}, \quad k \in [K] \quad (4) \\
& \textstyle\sum_{k=1}^K l_k = L. \quad (5)
\end{aligned}$$

Constraint (4) directly follows from the discussion at the end of Sec. II-B while (5) ensures that the partition $\{w_k\}_{k=1}^K$ of w fully covers w. Substituting Eqs. (1)-(3) in the above optimization problem and removing the constant terms from the objective function, we obtain

$$\begin{aligned}
\underset{\{l_k\},\{t_k^{\text{SHU}}\}}{\text{minimize}} \quad & \textstyle\sum_{k=1}^K l_k C_k P_k + \frac{t_k^{\text{SHU}}}{|h_k|^2} f\left(\frac{\alpha l_k}{t_k^{\text{SHU}}}\right) \quad (6) \\
\text{subject to} \quad & l_k, t_k^{\text{SHU}} \geq 0, \quad k \in [K] \\
& l_k \frac{C_k}{F_k} + t_k^{\text{SHU}} \leq \tau_k, \quad k \in [K] \quad (7) \\
& \textstyle\sum_{k=1}^K l_k = L
\end{aligned}$$

with τ_k, the *effective latency constraint* of node k, given by

$$\tau_k = \tau - T\max_k\{C_k/F_k\} - KDC_k/F_k. \quad (8)$$

This last optimization problem is very similar to the one formulated in [7] and is known to be convex [7, Lemma 1].

Next, one can observe that the objective function (6) is always decreasing with t_k^{SHU}. Indeed, for a fixed number of bits αl_k to transmit during the Shuffle phase, increasing the duration of the transmission t_k^{SHU} always decreases the energy consumption E_k^{SHU}. As a consequence, constraint (7) is always active at the optimum and can thus be turned into an equality constraint[2]. We can thus get rid of half of the optimization variables by substituting l_k by $\frac{F_k}{C_k}(\tau_k - t_k^{\text{SHU}})$. This leads to

$$\begin{aligned}
\underset{\{t_k^{\text{SHU}}\}}{\text{minimize}} \quad & \textstyle\sum_{k=1}^K (\tau_k - t_k^{\text{SHU}}) F_k P_k \\
& + \frac{t_k^{\text{SHU}}}{|h_k|^2} f\left(\alpha \frac{F_k}{C_k}\left(\frac{\tau_k}{t_k^{\text{SHU}}} - 1\right)\right) \\
\text{subject to} \quad & 0 \leq t_k^{\text{SHU}} \leq \tau_k, \quad k \in [K] \\
& \textstyle\sum_{k=1}^K \frac{F_k}{C_k}(\tau_k - t_k^{\text{SHU}}) = L. \quad (9)
\end{aligned}$$

IV. Optimal solution

We start by defining the partial Lagrangian as follows

$$\begin{aligned}
\mathcal{L}(\{t_k\}, \lambda) = \textstyle\sum_{k=1}^K (\tau_k - t_k) F_k P_k + \frac{t_k}{|h_k|^2} f\left(\alpha \frac{F_k}{C_k}\left(\frac{\tau_k}{t_k} - 1\right)\right) \\
+ \lambda\left(L - \textstyle\sum_{k=1}^K \frac{F_k}{C_k}(\tau_k - t_k)\right)
\end{aligned}$$

where t_k^{SHU} has been replaced by t_k to ease notations and with λ the Lagrange multiplier associated to (9). Then, applying the KKT conditions to the partial Lagrangian leads to

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial t_k}\Big|_* &= -F_k P_k + \tfrac{1}{|h_k|^2} f\left(\alpha \tfrac{F_k}{C_k}\left(\tfrac{\tau_k}{t_k^*} - 1\right)\right) \\
&\quad - \tfrac{\alpha}{|h_k|^2} \tfrac{F_k}{C_k} \tfrac{\tau_k}{t_k^*} f'\left(\alpha \tfrac{F_k}{C_k}\left(\tfrac{\tau_k}{t_k^*} - 1\right)\right) + \lambda^* \tfrac{F_k}{C_k} \\
&= -F_k P_k - \tfrac{\Gamma\sigma^2}{|h_k|^2} + \lambda^* \tfrac{F_k}{C_k} \\
&\quad + \tfrac{\Gamma\sigma^2}{|h_k|^2}\left(1 - \alpha \tfrac{\ln(2)}{B} \tfrac{F_k}{C_k} \tfrac{\tau_k}{t_k^*}\right) 2^{\frac{\alpha}{B}\frac{F_k}{C_k}\left(\frac{\tau_k}{t_k^*} - 1\right)} \\
&\begin{cases} > 0, & t_k^* = 0 \\ = 0, & t_k^* \in]0, \tau_k] \\ < 0, & t_k^* = \tau_k \Rightarrow l_k^* = 0, \end{cases}
\end{aligned}$$

with

$$\textstyle\sum_{k=1}^K \frac{F_k}{C_k}(\tau_k - t_k^*) = L.$$

The first case (i.e., > 0) can't happen as the objective goes to $+\infty$ when t_k goes to 0. The last case (i.e., < 0) tells us when a node does not participate in the Map and Shuffle phases (i.e., when $l_k^* = 0$). It can be re-written as

$$C_k P_k + \alpha \tfrac{\Gamma\sigma^2}{|h_k|^2} \tfrac{\ln(2)}{B} > \lambda^*. \quad (10)$$

The left-hand side of the inequality corresponds to the marginal energy consumption of node k per bit received, when node k hasn't received any bit yet, i.e., at $l_k = 0$. Indeed, the first term corresponds to the marginal energy consumption incurred by the Map phase while the second term corresponds to the marginal energy consumption incurred by the Shuffle phase. In other words, the left-hand side of (10) can be interpreted as the "price to start collaborating". If this price is greater than a threshold given by λ^*, then $l_k^* = 0$, meaning that node k does not participate to the Map and Shuffle phases[3]. Finally, solving the remaining case (i.e., $= 0$) for t_k^* leads to

$$t_k^* = \frac{\alpha \frac{\ln(2)}{B} \frac{F_k}{C_k} \times \tau_k}{W_0\left\{\frac{1}{e}\left(\frac{|h_k|^2}{\Gamma\sigma^2} \frac{F_k}{C_k}(\lambda^* - C_k P_k) - 1\right) e^{\alpha \frac{\ln(2)}{B} \frac{F_k}{C_k}}\right\} + 1} \quad (11)$$

where $W_0(\cdot)$ is the main branch of the Lambert function. The optimization problem can then be solved using a one-dimensional search for λ^*, as described in Algorithm 1.

Algorithm 1 Binary search for λ^*

1: $(\lambda_l, \lambda_h) = (\min_k\{C_k P_k + \alpha \frac{\Gamma\sigma^2}{|h_k^2|} \frac{\ln(2)}{B}\}$, a sufficiently large value$)$
2: $(L_l, L_h) = (\sum_k \frac{F_k}{C_k}(\tau_k - t_{k,l}^*), \sum_k \frac{F_k}{C_k}(\tau_k - t_{k,h}^*))$ where $t_{k,l}^*$ and $t_{k,h}^*$ are obtained using (11) with λ_l and λ_h, respectively.
3: **while** $L_l \neq L$ and $L_h \neq L$ **do**
4: $\quad L_m = \sum_k \frac{F_k}{C_k}(\tau_k - t_{k,m}^*)$ where $t_{k,m}^*$ is obtained using (11) with $\lambda_m = (\lambda_l + \lambda_h)/2$.
5: $\quad$ **if** $L_m > L$ **then**, $\lambda_h = \lambda_m$, compute L_h as in step 2.
6: $\quad$ **else if** $L_m < L$ **then**, $\lambda_l = \lambda_m$, compute L_l as in step 2.
7: $\quad$ **else** $\lambda^* = \lambda_m$.
8: **end while**

[2] In the particular case where $l_k = 0$, the value of t_k^{SHU} does not impact the objective function and imposing $t_k^{\text{SHU}} = \tau_k$ to make the constraint active is thus not an issue.

[3] Note that it still participates to the Reduce phase as it still needs to obtain $\phi(d_k, w)$.

V. Numerical results

In this section, the performances of the optimal collaborative-computing scheme are benchmarked against a blind collaborative-computing scheme and the non-collaborative scenario through numerical experiments[4]. The blind collaborative-computing scheme simply consists in uniformly distributing w between the K nodes, i.e., $l_k = L/K$, without taking into account their computing capabilities and the strength of their channel to the AP. Unless stated otherwise, the parameters used in the following numerical experiments are given in Table I [7].

TABLE I
Parameters used in the numerical experiments.

Parameter	Value	Units
C_k	$\overset{\text{i.i.d.}}{\sim}$ Unif($[500, 1500]$)	[CPU cycles/bit]
P_k	$\overset{\text{i.i.d.}}{\sim}$ Unif($[10, 200]$)	[pJ/CPU cycle]
F_k	$\overset{\text{i.i.d.}}{\sim}$ Unif($\{0.1, 0.2, \ldots, 1.0\}$)	[GHz]
h_k	$\overset{\text{i.i.d.}}{\sim}$ $\mathcal{CN}(0, 10^{-3})$ (Rayleigh fading)	/
B	15	[kHz]
σ^2	1	[nW]
Γ	1	/

A. Maximum computation load and outage probability

We start by looking at the maximum computation load (i.e., the maximum size of w) that can be processed by the different schemes under a given latency constraint. For both the optimal and the blind collaborative-computing schemes, the maximum computation load is achieved when τ_k, the effective latency, is entirely used to perform local computation, that is, when an infinite amount of energy is used for the Shuffling phase and $t_k^{\text{SHU}} \to 0$. The maximum computation load of the optimal and blind collaborative-computing schemes are thus given by

$$L_{\max}^{\text{opt}} = \sum_{k=1}^{K} \frac{F_k}{C_k} \tau_k$$

and

$$L_{\max}^{\text{blind}} = K \min_k \left\{ \frac{F_k}{C_k} \tau_k \right\},$$

respectively. For the case where the nodes do not collaborate (i.e., each node is working for itself only), the maximum computation load that can be processed in the allowed latency τ is given by

$$L_{\max}^{\text{solo}} = \min_k \left\{ \frac{F_k}{C_k} \left(\tau - D \frac{C_k}{F_k} \right) \right\}.$$

If we consider the computing capabilities of the nodes as random variables, $L_{\max}^*$ can also be considered as a random variable. Thus, for a given computation load L, one can define the outage probability P_{out}^* of the system as follows

$$P_{\text{out}}^* = \mathbb{P}[L_{\max}^* < L].$$

Figure 2 shows the empirical outage probability of the different schemes as a function of the allowed latency τ for several numbers of nodes K. This figure illustrates one of the advantage of the optimal scheme: for a given number of nodes K and a given allowed latency τ, this is the scheme with the highest probability of satisfying the latency constraint. Increasing the number of nodes is also more profitable with the optimal scheme than it is with the blind scheme. This is because the optimal scheme leverages *diversity* amongst the nodes, while the blind scheme, as suggested by its name, is blind to that diversity and considers all the nodes as being equals.

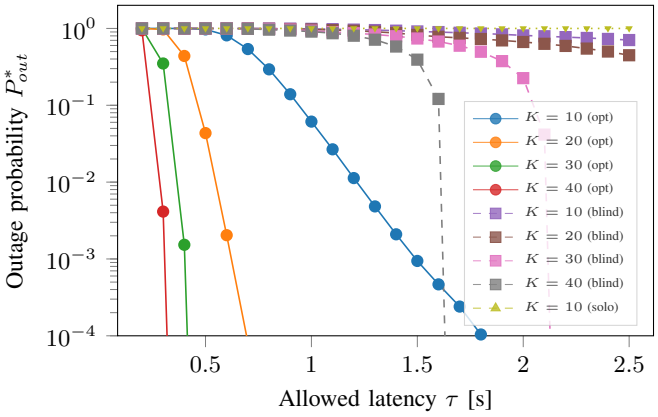


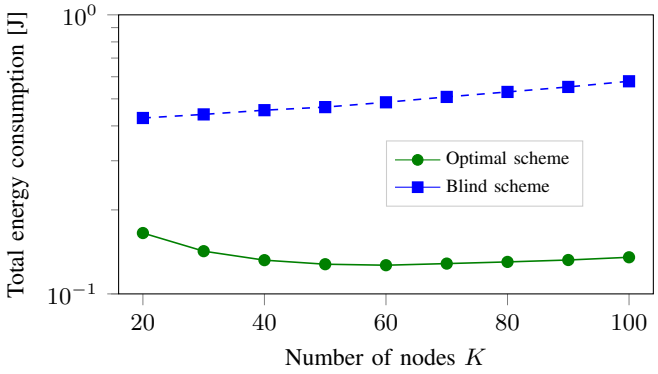
Fig. 2. Empirical outage probability for the optimal collaborative-computing scheme (opt), the blind collaborative-computing scheme (blind) and the non-collaborative scenario (solo). Each data point is the result of an average on 1M experiments with $L = 4$Mb, $D = 100$b and $T = 5$kb.

B. Energy consumption and energy-latency trade-off

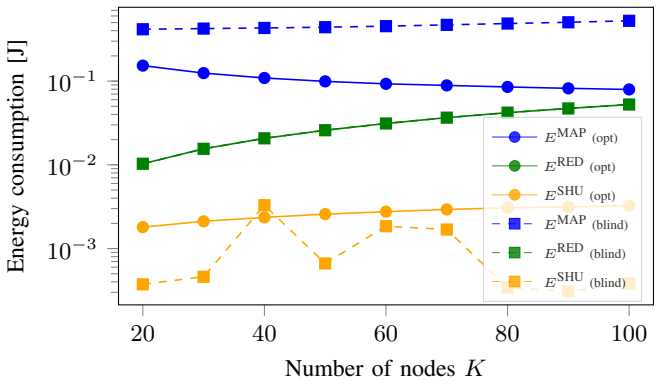
Figure 3a compares the total energy consumption of the nodes when using the optimal and the blind scheme. Each point on the figure is the result of an average over 10000 random feasible (for both the optimal and the blind schemes) instances of the problem, with $L = 4$Mb, $D = 100$b, $T = 5$kb and with the allowed latency τ set to 1 s to ensure feasibility by both schemes with relatively high probability. This figure shows that the optimal scheme consumes approximately four to five times less energy than the blind scheme. Note that the total energy consumed in the non-collaborative scenario can easily be shown to be roughly K times larger than the total energy consumed by the blind scheme. Next, Fig. 3b breaks down the total energy consumptions of both the optimal and the blind schemes into three components associated to the different phases of the collaboration, i.e., E^{MAP}, E^{SHU} and E^{RED}. First of all, this figure shows that most of the energy is consumed by the Map and Reduce phases[5]. Next, and at the opposite of the blind scheme, the optimal scheme is able to reduce E^{MAP} when K increases, again by leveraging diversity amongst the nodes. This explains the slow decrease of the total energy consumption with K visible on Fig. 3a. At some point however, the unavoidable energy consumption of the Reduce phase starts to grow faster than E^{MAP} decreases and the total energy consumption rises again. Finally, Fig. 3c depicts how the different energy components of the optimal scheme evolve with the allowed latency τ. In particular, this

[4]Source code available at https://github.com/anpar/EE-WCC-MapReduce.

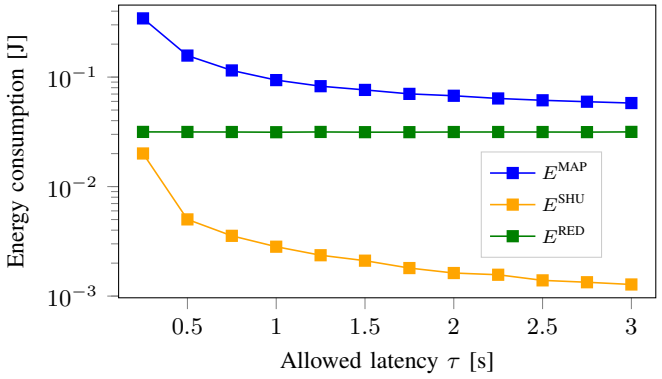
[5]Note that this result depends a lot on the value of the parameters presented in Table I and should thus not be interpreted as a general result.



(a) Comparison of the total energy consumed by the optimal and the blind scheme for $\tau = 1\,\mathrm{s}$.



(b) Breakdown of the energy consumed by the three phases of the collaboration, both for the optimal (opt) and the blind (blind) collaborative computing scheme, for $\tau = 1\,\mathrm{s}$. Note that the energy consumption for the Reduce phase is the same in both case.



(c) Evolution of the energy consumed by the three phases of the optimal collaborative-computing scheme, as a function of the latency, for $K = 60$.

Fig. 3. Energy consumption of the nodes when $L = 4\mathrm{Mb}$, $D = 100\mathrm{b}$ and $T = 5\mathrm{kb}$. Each point is the result of an average over 10000 feasible (for each scheme) instances of the problem.

figure shows that the optimal scheme is able to decrease the energy consumed by the Map phase when τ increases. This is again a benefit of the diversity amongst the nodes: increasing the allowed latency allows the optimal scheme to use slower but more energy-efficient nodes, hence decreasing the energy consumption.

VI. Discussion and future works

In this work, an energy-efficient wireless collaborative-computing scheme inspired by the Map-Reduce framework has been proposed. Numerical experiments highlighted the benefits of this scheme over a blind scheme and the non-collaborative scenario: lower achievable latency, reduced energy consumption and the ability to trade energy for latency and vice versa. Those benefits are obtained by leveraging the *diversity* of the nodes in term of computing capabilities and channel strength. Analytical results highlighting the benefits of diversity are however missing and their pursuit thus constitutes a first possible direction for future works.

A second obvious direction for future works might be to refine the optimization problem formulated in Sec. III to account for additional constraints, e.g., limited memory capacity, maximum RF transmit power or limited battery level.

Next, the models used in this work to quantify the time and energy consumed by the different phases of the collaboration are very simple and far from being realistic (see, for instance, [8]). Incorporating more realistic models in the proposed collaborative-computing scheme will thus certainly be a priority in future works.

In particular, the Shuffle phase considered in this work has been simplified. Indeed, as already mentioned, coded Shuffling [3]–[6] has not been considered, nodes were assumed to be able to communicate in an orthogonal manner in the uplink and the downlink was not considered. While convenient to lay out the foundations of this collaborative-computing framework, those simplifying assumptions will be reconsidered in future works.

Acknowledgment

AP is a Research Fellow of the F.R.S.-FNRS. This work was also supported by F.R.S.-FNRS under the EOS program (project 30452698, “MUlti-SErvice WIreless NETwork”).

References

[1] D. Liu, B. Chen, C. Yang, and A. F. Molisch, “Caching at the wireless edge: design aspects, challenges, and future directions,” *IEEE Communications Magazine*, vol. 54, no. 9, pp. 22–28, Sep. 2016.

[2] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” in *OSDI’04: Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, 2004, pp. 137–150.

[3] S. Li, Q. Yu, M. A. Maddah-Ali, and A. S. Avestimehr, “A scalable framework for wireless distributed computing,” *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 2643–2654, Oct 2017.

[4] M. Kiamari, C. Wang, and A. S. Avestimehr, “Coding for edge-facilitated wireless distributed computing with heterogeneous users,” in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, Oct 2017, pp. 536–540.

[5] S. Li, M. A. Maddah-Ali, Q. Yu, and A. S. Avestimehr, “A fundamental tradeoff between computation and communication in distributed computing,” *IEEE Transactions on Information Theory*, vol. 64, no. 1, pp. 109–128, Jan 2018.

[6] F. Li, J. Chen, and Z. Wang, “Wireless MapReduce Distributed Computing,” 2018. [Online]. Available: http://arxiv.org/abs/1802.00894

[7] C. You, K. Huang, H. Chae, and B. Kim, “Energy-efficient resource allocation for mobile-edge computation offloading,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1397–1411, March 2017.

[8] T. Bouguera, J.-F. Diouris, J.-J. Chaillout, R. Jaouadi, and G. Andrieux, “Energy consumption model for sensor nodes based on lora and lorawan,” *Sensors*, vol. 18, no. 7, p. 2104, 2018.