Generosity Pays Off: A Game-Theoretic Study of Cooperation in Decentralized Learning

Generosity Pays Off: A Game-Theoretic Study of Cooperation in Decentralized Learning

G. Di Giacomo[†], F. Malandrino[‡,*], C. F. Chiasserini[†,‡,*]
[†]Politecnico di Torino, Italy; [‡]CNR-IEIIT, Italy; [*]CNIT, Italy

***Abstract*—Decentralized learning, a paradigm enabling the training of Machine Learning (ML) models using multiple nodes, is gaining momentum, as it (i) improves data privacy and (ii) permits to leverage the computational capabilities of a wide set of nodes, thus being an excellent fit for the support of edge intelligence applications. However, such nodes, like users' smartphones or vehicles, cannot be *forced* to participate in the learning process, and incentivizing them to do so is one of the foremost challenges of decentralized learning. To address this issue, we propose GENIAL – a game-theoretic approach, based upon generous games, to promote cooperation among user nodes for training or fine-tuning ML models. By allowing such nodes to be (moderately) *generous*, i.e., to contribute to decentralized training processes more often than what would be convenient for them in the short term, GENIAL leads to a Nash equilibrium where all nodes cooperate. Importantly, such equilibrium is also proven to converge to the Pareto optimal operating point that ensures a fair treatment to all nodes. Our theoretical findings are supported by numerical experiments, which further underline the effectiveness, and the benefits for rational nodes, of being generous in decentralized training.**

***Index Terms*—Decentralized learning, game theory, incentive mechanism, node cooperation**

I. Introduction

Machine Learning (ML) and, more specifically, Deep Learning models are the state-of-the-art methods used across many different domains. Traditionally, the training of such models has been performed in a centralized manner; however, that requires significant quantities of data and large computational resources, besides posing privacy concerns [1]. On the other hand, the opposite approach where each node independently trains its model with its own data, may result in poor learning performance [2].

To cope with these issues, the decentralized learning paradigm [3], [4] has emerged as a powerful alternative. Contrary to centralized learning, the decentralized learning paradigm leverages the computational power of the participating nodes, which locally train the model by using their own data, thus ensuring data locality and privacy [4], and then exchange such model till convergence is reached.

Such a learning paradigm is especially relevant in the case where nodes at the edge and/or far-edge of the network have to be involved for the training or the fine-tuning of ML models, owing to the diversity of the data they own and the need to keep such data local. However, in many real-world scenarios, edge and far-edge nodes belong to different entities or are user devices, hence, they are under no obligation to participate in the learning and will do so only if advantageous to them. In game-theoretic terms, they are *rational*, and a proper incentive mechanism must be in place to make them participate [5].

To address this issue, we propose an algorithm called GENerous Incentive Algorithm for Learning (GENIAL), which is based on the classic Generous Tit-for-Tat strategy [6], [7] (GTFT). Under GTFT strategies, nodes start by being willing to cooperate (hence, "generous"), and then keep doing so only if they observe that other nodes cooperate as well (hence, applying "tit-for-tat"). We show that GENIAL yields a Nash equilibrium, i.e., a profile strategy where no player gains some profit by unilaterally modifying its strategy. Furthermore, we prove that such an equilibrium is Pareto optimal and fair, i.e., it is not possible to make a player better off without negatively affecting another one.

Intuitively, reaching a Pareto-optimal Nash equilibrium means that the system operates in an efficient (hence, desirable) manner, and such a result is reached *in spite* of the fact that all nodes solely pursue their own interest. GENIAL is able to account for the most relevant aspects of ML training, most importantly, that nodes may belong to different *classes*; the class of a node dictates the complexity of the models each node needs to train, its target learning quality, and its available computational resources. It follows that decisions about whether or not to cooperate may have different consequences, i.e., impact on the nodes' resource consumption, each time they are made.

The rest of the paper is organized as follows. Sec. II introduces the system model, while Sec. III formulates the problem of cooperative decentralized learning as a maximization problem of the payoff received by the nodes. Further, Sec. III derives the Pareto-optimal operating points. Sec. IV proposes the GENIAL algorithm as incentive mechanism, and it shows that GENIAL attains the nodes' Pareto-optimal operating points, which also correspond to a Nash Equilibrium. The performance of GENIAL is presented in Sec. V. Finally, Sec. VI discusses some related work and highlights the novelty of our study, while. Sec. VII draws our conclusions.

II. System Model

We consider a learning orchestrator at the edge of the network, which assists a cluster $\mathcal{N}$ of N nodes with the training or the fine-tuning of ML models. For simplicity, the cluster topology is assumed to be fully connected. Further, each node belongs to one of K classes, with the number of nodes in class k being denoted with N_k. All nodes in class k are associated with an average and a maximum computing

capability constraint, denoted respectively by $\bar{\chi}_k$ and $\chi_k^{\max}$; without loss of generality, we assume: $\bar{\chi}_1<\bar{\chi}_2<\ldots<\bar{\chi}_K$ and $\chi_1^{\max}<\chi_2^{\max}<\ldots<\chi_K^{\max}$. Also, each class k is associated with a set of models $\mathcal{M}_k$ that the nodes in the class may need to train or update: the complexity of the models in $\mathcal{M}_k$ is denoted with m_k. The quantities $\bar{\chi}_k$ and $\bar{\chi}_k^{\max}$ are measured in millions of floating point operations per second (MFLOPS), while the complexity m_k indicates the number of floating point operations (FLOPs) to process one sample during training, e.g., in the case of Deep Neural Networks (DNNs), perform the forward and backward passes.

A node asking for help is referred to as *requester*, while a node accepting to help train a model is called *learner*. Whenever a training process is concluded, the orchestrator selects the next node to start a new training process among the candidate requesters with probability $1/N$, so that only one training session at the time can be performed.

We denote by $h+1$ the minimum number of nodes needed to train the model at hand, where one of such nodes is the requester itself. Fixing *a priori* the value of h, we can compute the minimum number of training samples d_k that each node should use to train a model of complexity m_k, such that the target learning quality is achieved. More specifically, in, e.g., [8], it is empirically found that the generalization error (i.e., the test loss value) exhibits a power-law improvement with respect to the training set size. The power-law parameters, however, depend on the specific DNN and dataset at hand; as also envisioned in [9], to estimate such parameters, one could run a small-scale profiling for a specific range of test losses. Thus, given the target test loss value (which is lower than the values used for the profiling), it is possible to derive the minimum number of required training samples D_k for a model of complexity m_k. Given D_k and the value of h, which are inputs to our problem, it is straightforward to compute the smallest number of required training samples d_k, i.e., $d_k=\lceil \frac{D_k}{h+1} \rceil$. For the sake of simplicity, we assume that each node $n\in\mathcal{N}$ owns a dataset $\mathcal{D}_n$ such that $|\mathcal{D}_n|\geq d_k$, with $k=1,\ldots,K$.

A requester will then ask $H\geq h$ other nodes in the cluster for help: indicating with $l\leq H$ the number of nodes that accept the request (hereinafter also referred to as *learners*), if $l<h$, the learning process fails and the requester has to wait a time T_w before starting generating another request. When instead at least h nodes agree to help, i.e., $h\leq l\leq H$, the training session can start: the set of l learners that accept the request and, thus, contribute to the training, remains the same till the training of the model is completed.

A training session is said to belong to type (k,j) when the requester belongs to class k and the minimum class of the set of the H nodes receiving the request is equal to j. The probability that a node accepts to cooperate in a training session of type (k,j) is denoted by π_{kj}. The dependence of acceptance probabilities upon the session type allows us to reproduce the reasoning of potential learners, which will also consider the expected amount of resources to devote to the process when deciding whether to accept.

When a training session begins, the requester sends the model to the l learners, which, in parallel and along with the requester, perform one learning epoch using their own local dataset. Nodes then exchange the updated models with each other, and combine newly-received models with their local one. A model training is terminated after a number of epochs e_k needed to achieve the target learning quality. Notice that such a number depends on the requester class k, hence, on the complexity m_k of the model to train, and can be determined *a priori* by using an approach similar to the one in [10] if the model is already characterized. Alternatively, it is possible to use methods that find convergence bounds, such as the one proposed in [11]. The time required to perform one epoch is equal to the processing time taken by the slowest learner; in general, the local processing time of a node depends on the number of processed samples, its computing capability, and the model complexity m_k. For the sake of simplicity, we do not take into account the CPU consumption due to communication, i.e., to the radio functions enabling the transmission of the models' parameters.

To properly describe and assess the performance of the learning process, we define the following metrics:

- $x_n^{kj}(t)$: number of requests generated by node n for a training session of type (k,j) accepted till time t;
- $X_n^{kj}(t)$: number of requests generated by node n for a training session of type (k,j) till time t;
- $y_n^{kj}(t)$: number of requests made to node n for a training session of type (k,j) accepted till time t;
- $Y_n^{kj}(t)$: number of requests made to node n for a training session of type (k,j) till time t;
- $\mu_n^{kj}(t)=\frac{x_n^{kj}(t)}{X_n^{kj}(t)}$, $\nu_n^{kj}(t)=\frac{y_n^{kj}(t)}{Y_n^{kj}(t)}$, which indicate the quantity of service, (resp.) received and given by n for type-(k,j) training sessions till time t.

Last, we define the Normalized Received Service of a generic requester n for training sessions of type (k,j) as $\text{NRS}_n^{kj}(t)=\lim_{t\to\infty}\mu_n^{kj}(t)$. Beyond its role in the mathematical formulation of the problem described below, this metric is very important to assess how willing nodes are to cooperate, hence, intuitively, how effective our cooperation scheme is.

III. Problem Formulation

As nodes are rational, each node aims to maximize its own long-term utility, that is its average *payoff* over all training sessions. Assuming that a node receives a payoff of 1 when its training request is successful, we define the utility of a node n of class k in a type (k,j) session as $U_n^{kj}=\lim_{t\to\infty}\frac{x_n^{kj}(t)}{S(t)}=P_n^{kj}\Pi_{kj}$, where $S(t)$ indicates the total number of sessions (accepted and rejected) till time t, P_n^{kj} denotes the probability that node n is a requester in a type (k,j) session, while Π_{kj} is the probability for a training session of type (k,j) to be accepted. The latter event occurs when the number l of accepting nodes is higher than the required minimum number h, i.e.,

$$\Pi_{kj}=P(l\geq h)=\sum_{\ell=h}^{H}\frac{H!}{\ell!(H-\ell)!}\pi_{kj}^{\ell}(1-\pi_{kj})^{H-\ell}, \quad (1)$$

where, as mentioned, we consider that all nodes accept to contribute to a training session of type (k,j) with the same probability π_{kj}. We will thus highlight below the dependency of the node's utility on π_{kj} by writing $U_n^{kj}(\pi_{kj})$.

Finally, denoting the mean computing expenditure of the generic node n with c_n, the objective of node n, belonging to class k and acting as a requester, is given by:

$$\max_{\pi_{kj}} \sum_j U_n^{kj}(\pi_{kj}), \quad \text{s.t. } c_n \leq \bar{\chi}_k, \forall n \in \mathcal{N}. \tag{2}$$

In the objective function above, the decision variables are the probabilities π_{kj} with which a node accepts to cooperate in a session of type (k,j). Note that, since all nodes are rational and are provided by the orchestrator with the same system information (number of nodes and their class), they will all end up computing the same optimal values for the π_{kj}'s.

A. Stationary case: closed-form solution

Given the problem above, we now assume stationary conditions in which all nodes consistently use the same π_{kj} values (such an assumption will then be removed in Sec. IV). In this case, we can analytically derive the π_{kj}'s, as follows. Let us write the mean duration of a session, T_s, as:

$$T_s = \sum_k \sum_j p_{kj} \Big[T_{kj} \Pi_{kj} + T_w (1 - \Pi_{kj}) \Big] \tag{3}$$

where the first and second terms on the right-hand side account for the case when a session is accepted and rejected, respectively. In particular, p_{kj} is the probability that an occurring session is of type (k,j), while T_{kj} is the training time for a session of type (k,j), i.e., $T_{kj} = \frac{m_k e_k d_k}{\chi^{\max}_{\min(k,j)}}$. In this latter expression, the term $m_k e_k d_k$ is the total number of FLOPs required to finish training the requester's model, while $\chi^{\max}_{\min(k,j)}$ is the smallest across the maximum computational capability of the nodes participating in the training session. The time required for transmitting the model parameters and for the models' aggregation is assumed to be negligible when compared to the computational time. Being T_s the mean session duration, it must hold: $T_s > 0$.

The mean computing expenditure for a node n of class k when the node is a requester and a learner is (resp.) given by:

$$c_n^r = \sum_{j=1}^{K} P_n^{kj} \frac{m_k e_k d_k}{T_s} \Pi_{kj} \quad \text{and} \tag{4}$$

$$c_n^l = \sum_{j=1}^{K} \sum_{i=1}^{k} q_{ji}^{(k)} \frac{m_j e_j d_j}{T_s} \pi_{ji} \Omega_{ji}, \quad \text{where} \tag{5}$$

- $q_{ji}^{(k)}$ the probability for a node of class k to receive a training request in a session of type (j,i);
- Ω_{ji} denotes the probability that out of the remaining $H-1$ nodes that receive the request at least $h-1$ accept.

Overall, the average computing expenditure for a node n is then given by: $c_n = c_n^r + c_n^l$. By considering the constraint of the node's computing capability in (2), we get K inequalities and K^2 unknown variables. To solve the system, we need to reduce the number of variables to K, which can be done by exploiting the nodes' *rationality*, as described below.

Rationality. Consider a system with only $N=2$ nodes belonging to the same class, i.e., $K=1$ and $N_1=N$. By rationality, each node must have the same NRS value. Indeed, having the same average computing capability constraint, one node cannot increase its NRS without reducing the NRS of the other node; the latter, however, would not find such an operating point acceptable. Therefore, the NRSs must be equal for both nodes. This can be extended to the case where all $N>2$ nodes belong to the same class: also in this scenario, by rationality, each node must have the same value of NRS. In this case, it is straightforward to derive the value of π_{11} that maximizes the objective function while satisfying the computing capabilities constraint.

Next, consider a system with $N_1=n_1$, $N_2=1$, $H=h=1$, and that the quantity of received service is the same for all requesters when the learner belongs to class 1, i.e.,

$$m_1 e_1 d_1 \Pi_{11} = m_2 e_2 d_2 \Pi_{21} \,. \tag{6}$$

Further, by rationality, the quantity of service received by class 1 nodes from the node in class 2 must be equal to the quantity of service received by the latter from the nodes in class 1. Indeed, the class-2 node has no interest in being more generous to the nodes of class 1 than what the latter ones are to the class-2 node, since the latter will not receive a higher service share anyway. Thus, we have:

$$m_1 e_1 d_1 \ \Pi_{12} = m_2 e_2 d_2 \ \Pi_{21} \,. \tag{7}$$

From (6) and (7), it derives that: $\Pi_{11}=\Pi_{12}$. Notably, in this simple scenario where $H=h=1$, from (1) one can derive that $\Pi_{kj}=\pi_{kj}$ for all possible session types (k,j).

We now extend (6) and (7) to the most general case, i.e., with $K \geq 1$, $H \geq 1$ and $h \geq 1$. In this case, we have:

(i) the values of Π_{kj} are such that $\Pi_{kk}=\Pi_{kj}, 1 \leq k \leq j \leq K$, which implies

$$\pi_{kk} = \pi_{kj}, \quad 1 \leq k < j \leq K; \tag{8}$$

(ii) when a node receives a training request for a session of type (j,k), the rational values of Π_{jk} are such that

$$m_k e_k d_k \ \Pi_{kj} = m_j e_j d_j \ \Pi_{jk}, \quad 1 \leq k < j \leq K, \tag{9}$$

from which one can compute the ratio between π_{kj} and π_{jk}, $1 \leq k < j \leq K$.

Thus, given (8) and (9), and solving the aforementioned system of K inequalities by using (1) and (3)-(5), one can compute the π_{kj} values that maximize (2) and allow obtaining the Pareto optimal NRS values, i.e., values such that no node can improve its NRS without decreasing another node's NRS, while meeting the constraint on the mean computing expenditure. Importantly, (8) and (9) guarantee, in their respective cases, equal service shares received by all nodes; hence the Pareto optimal π_{kj} values are fair.

IV. GENIAL: Algorithm and Properties

In the previous section, we have derived the π_{kj}'s under the assumption that all nodes consistently use such values. However, since the nodes are rational, they cannot be relied upon to do so; intuitively, nodes will try to exploit their peers' naivety by first accepting the cooperation of other nodes, and then refusing to reciprocate. It follows that *no* stationary acceptance policy would be feasible; therefore, to effectively foster cooperation among nodes, a *behavioral* strategy is required, whereby nodes decide their actions based upon past history. Specifically, we propose a novel game-theoretic approach called GENerous Incentive Algorithm for Learning (GENIAL), which is based on the GTFT strategy and allows attaining the derived π_{kj}'s.

Consider a node n receiving at time t a request from a node of class k, that the smallest class among the H nodes that have received the request is j; thus the learning session is of type (k,j). To accept or deny the request, n applies the GENIAL algorithm, that is:

- if $\nu_n^{kj}(t)>\pi_{kj}\Omega_{kj}$ or $\mu_n^{ik}(t)<\frac{\Pi_{ik}}{\pi_{kj}\Omega_{kj}}\nu_n^{kj}(t)-\gamma$ reject;
- else accept.

According to this strategy, n rejects the requests if at least one of the following two conditions holds:

- $\nu_n^{kj}(t)>\pi_{kj}\Omega_{kj}$, i.e., in sessions of type (k,j) node n has performed more training than what it should have;
- $\mu_n^{ik}(t)<\frac{\Pi_{ik}}{\pi_{kj}\Omega_{kj}}\nu_n^{kj}(t)-\gamma$, with $\gamma>0$, i.e., the amount of help received by node n in type-(i,k) sessions is lower than the amount of help it has given in type-(k,j) sessions, normalized by the term $\frac{\Pi_{ik}}{\pi_{kj}\Omega_{kj}}$, which is the value to which the ratio $\frac{\mu_n^{ik}(t)}{\nu_n^{kj}(t)}$ should converge. Since γ is a small positive value, nodes are a little generous by agreeing to train for others even if they have not received as much help as they should have.

We now prove that GENIAL yields a Nash Equilibrium, first in the simple case where all nodes belong to the same class and only one learner is necessary for training (i.e., $K=H=h=1$).

Theorem 1. *Consider a system of N nodes, all belonging to class 1, with mean and maximum computing capability constraint (resp.) $\bar{\chi}_1$ and $\chi_1^{\max}$. Models have complexity m_1, which implies that d_1 samples and e_1 epochs are required to reach the target learning quality. Also, $H=h=1$, and if the node receiving the training request accepts, the time required for the training is T_{11}, otherwise a time T_w is waited. Then:*

1) *if all nodes, except for node n, employ GENIAL, then* $\limsup_{t\to\infty}\mu_n^{11}(t)\leq\pi_{11}=\frac{\bar{\chi}_1 N}{2}\frac{T_w}{m_1 d_1 e_1-\frac{\bar{\chi}_1 N}{2}(T_{11}-T_w)}$;
2) *if all nodes apply GENIAL, then* $\lim_{t\to\infty}\mu_n^{11}(t)=\pi_{11}=\frac{\bar{\chi}_1 N}{2}\frac{T_w}{m_1 d_1 e_1-\frac{\bar{\chi}_1 N}{2}(T_{11}-T_w)}$, $\forall n\in\mathcal{N}$.

Proof: Please see the Appendix available at [12]. ■

Next, we move to the multi-class case.

Theorem 2. *Consider a system of N nodes and K classes, $H=h=1$, and N_k nodes in class k. Also, the computing capabilities constraints are as follows: $\bar{\chi}_1<\bar{\chi}_2<\ldots<\bar{\chi}_K$ and $\chi_1^{\max}<\chi_2^{\max}<\ldots<\chi_K^{\max}$. Then:*

1) *if all nodes, except for node n, employ GENIAL, then* $\limsup_{t\to\infty}\mu_n^{kj}(t)\leq\pi_{kj}$;
2) *if all nodes apply GENIAL, then* $\lim_{t\to\infty}\mu_n^{kj}(t)=\pi_{kj}$, $\forall n\in\mathcal{N}$ *and* $k,j=1,\ldots,K$.

Proof: Please see the Appendix available at [12]. ■

From Theorems 2 and 3, it is easy to show, by using randomizing arguments, that GENIAL constitutes a Nash Equilibrium and allows converging to the fair Pareto optimal operating point also when considering more than one node receiving a training request ($H>1$), and more than one learner ($h>1$).

Theorem 3. *Consider a system with N nodes, K classes, $H>1$, $h>1$ and N_k nodes in class k, $k=1,\ldots,K$. As for the computing capabilities constraints, assume that $\bar{\chi}_1<\bar{\chi}_2<\ldots<\bar{\chi}_K$ and $\chi_1^{\max}<\chi_2^{\max}<\ldots<\chi_K^{\max}$. Then:*

1) *if all nodes, except for node n, use GENIAL,* $\limsup_{t\to\infty}\mu_n^{kj}(t)\leq\Pi_{kj}$;
2) *if all nodes apply GENIAL, then* $\lim_{t\to\infty}\mu_n^{kj}(t)=\Pi_{kj}$, $\forall n\in\mathcal{N}$ *and* $k,j=1,\ldots,K$.

Proof: Please see the Appendix available at [12]. ■

V. Numerical Results

In this section, we experimentally analyze the behavior of the nodes when applying GENIAL, as well as the effects of GENIAL on the social utility of the system.

TABLE I
Small-scale scenario: additional settings

	Class 1	Class 2	Class 3
Dataset size	3×10^3	5×10^3	7×10^3
Model complexity [FLOPs]	2×10^3	3×10^3	4×10^3
Epochs required	40	55	70
$\chi^{\max}$ [MFLOPS]	45	60	75
$\bar{\chi}$ [MFLOPS]	8.4	12	15

Small-scale scenario. We consider a set of $N=12$ nodes and $K=3$ classes, with four nodes in each class training the models in Tab. I. Also, we set $H=h=2$, i.e., the requester asks two nodes for help and both need to accept for the training to take place. All nodes employ the GENIAL algorithm, with T_w set to 1 s.

The Π_{kj} values (hence, the Pareto optimal NRSs) are obtained by solving the equations (3)–(5) jointly with (8), (9) and the K inequalities stating the computing capability constraints. Fig. 1 depicts such values (black markers), along with the temporal evolution of the requesters' NRS. Specifically, the plots refer to requesters belonging, respectively, to class 1 (left), 2 (center), and 3 (right). One can observe that, when $\gamma=0.01$ (solid lines), the NRS values converge to the fair Pareto optimal values, while they reach much smaller values for $\gamma=0$ (dotted lines). The results thus confirm that being moderately generous ($\gamma=0.01$) yields much better performance, even if the nodes are self interested.

We now consider the same scenario, but with one and two nodes per class, hence, a total of (resp.) 3 and 6 nodes, that

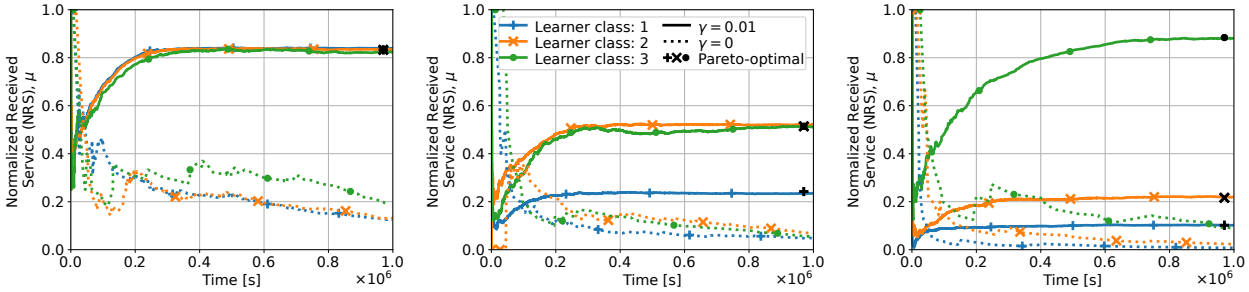

Fig. 1. Small-scale scenario: Temporal evolution of the NRS of a requester in class 1 (left), 2 (center), and 3 (right), with the fair Pareto optimal values indicated by the black markers. The NRSs converge to the target values only for $\gamma = 0.01$ (solid lines), while for $\gamma = 0$ (dotted lines), they converge to much lower values.

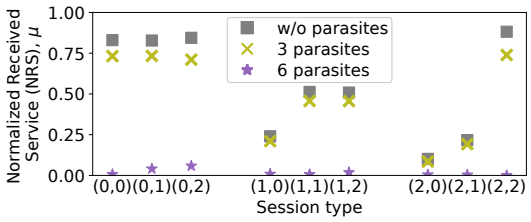

Fig. 2. Small-scale scenario: NRS values for each session type in three different cases: without parasites, and with 3 (25%) and 6 (50%) parasites that set $\gamma = -0.02$. The remaining nodes have γ=0.01.

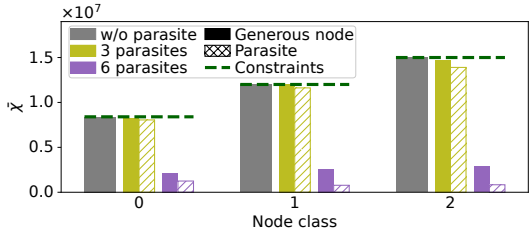

Fig. 3. Small-scale scenario: $\bar{\chi}$ for each class of nodes in three different cases: without parasites, and with 3 (25%) and 6 (50%) parasites that set $\gamma = -0.02$. The remaining nodes have γ=0.01. Dashed lines depict the $\bar{\chi}$ constraints for the respective classes of nodes.

use $\gamma = -0.02$ and, hence, are slightly selfish. Hereinafter we refer to such nodes as *parasite*. Fig. 2 presents the NRS values for each session type in such scenarios, and compares them to the case without parasite nodes. Notice how, with 3 parasite nodes out of 12 (i.e., 25% parasites), the NRSs are lower than in the case with all nodes being generous, and, with 6 parasites (i.e., 50% parasites), the performance degrades to such an extent that the NRSs approach 0. Importantly, this outcome underlines that, by using GENIAL, all nodes get lower NRSs if a selfish (i.e., non-collaborative) behavior is adopted; as a consequence, rational nodes have no incentive to follow an ungenerous behavior.

Fig. 3 sheds light on the resource usage for generous and parasite nodes. When no parasites are present (solid, grey bars), the nodes fully exploit the available resources, hence, the resource-usage constraint is met with an equal sign. This is what we can expect from a system that works well, adequately exploiting the available resources. On the contrary, in the scenario with 3 parasites (green), parasite nodes consume a slightly smaller amount of resources (checked bars), but, as shown by Fig. 2, at a cost of a lower NRS. With 6 parasites (purple), the computational expenditure drops dramatically for all nodes, as, essentially, very few models get trained.

The NRS values are an indicator of individual utility, as they refer to each single node. To analyze the social utility, i.e., the benefit for the set of nodes as a whole, we show in Tab. II the normalized number of accepted, and, hence, carried out, training sessions. The obtained social utility values are consistent with the results presented above: the scenario where all nodes are generous (all use γ=0.01) yields the larger social utility, confirming the importance of being generous also from the social utility point of view.

In conclusion, these results emphasize that to foster cooperation in a decentralized learning task and reach the Pareto optimal NRS values, all nodes have to employ GENIAL with $\gamma > 0$: as nodes are rational, they are motivated to adopt a generous behavior.

TABLE II
SOCIAL UTILITY IN DIFFERENT SCENARIOS

Scenario	Social utility
w/o parasites, γ=0.01	0.454
w/o parasites, γ=0	0.011
25% parasites	0.398
50% parasites	0.014

Larger-scale scenario. We now consider K=3 and N_1=12, N_2=18, and N_3=30, for a total of 60 nodes. Also, we set H=5, h=4, and T_w=1 s. All nodes employ GENIAL with γ=0.01; the additional settings for this scenario are

TABLE III
LARGER-SCALE SCENARIO: ADDITIONAL SETTINGS

	Class 1	Class 2	Class 3
Dataset size	2×10^3	2.5×10^3	3×10^3
Model complexity [FLOPs]	1×10^3	2×10^3	2.5×10^3
Epochs required	35	40	45
$\chi^{\max}$ [MFLOPS]	30	45	54
$\bar{\chi}$ [MFLOPS]	15.1	24.9	26.9

TABLE IV
LARGER-SCALE SCENARIO: NRSS AT CONVERGENCE, WITH PARETO OPTIMAL VALUES IN PARENTHESES

		Requester's class		
		1	2	3
Learner's class	1	0.63 (0.63)	0.22 (0.22)	0.13 (0.13)
	2	0.63 (0.63)	0.67 (0.67)	0.40 (0.40)
	3	0.63 (0.63)	0.67 (0.67)	0.97 (0.99)

reported in Tab. III. Tab. IV presents the obtained NRS values, which converge to the fair Pareto optimal values, shown in parentheses. This is consistent with the small-scale scenario we have focused earlier, and further highlights how GENIAL can work effectively even when the number of nodes grows.

VI. Related work

Our study leverages game theory to foster cooperation among the nodes participating in decentralized learning tasks. We draw on [7], which considers a wireless ad-hoc network where source nodes can communicate with their target destination only if intermediate nodes accept to relay the packets to be delivered. Here, we extended the game-theoretic framework in [7] to make it suitable for decentralized learning scenarios, where nodes are asked to collaborate with each other, by making available their computational resources to train ML models.

As for decentralized learning, [4] presents a comprehensive theoretical analysis of convergence in decentralized settings and introduces a unified framework for Decentralized Stochastic Gradient Descent (SGD) optimization, which generalizes earlier methods and is proven to achieve the optimal convergence rates for local SGD across different scenarios. [3] theoretically and experimentally demonstrates the convergence speedup of a decentralized optimization algorithm on networks with low bandwidth or high latency, when compared to a distributed scenario with a central node aggregating the updates computed by the training nodes. The latter case is generally referred to as distributed learning, whose most popular representative is Federated Learning (FL) [1]. Note that, also when FL is deployed in real scenarios, nodes must be encouraged to train by means of an incentive mechanism [13].

Incentive mechanisms can be based on economic or game-theoretic approaches [14]. In a similar vein, [15], considers a distributed learning scenario with rational nodes and a Fusion Center, i.e., the central coordinator, that receives the gradient updates from the nodes and fosters their cooperation by using a reward mechanism based on zero-determinant strategies. Such interactions are treated as repeated games. Similarly, [16] models the interactions between the FL coordinator and the clients by using a Stackelberg game.

The study in [17] introduces an incentive mechanism for cross-silo FL, specifically designed to account for participating organizations' heterogeneity and the concept of public goods. This mechanism achieves social welfare maximization, while ensuring individual rationality and budget balance. [18] models FL as a hedonic game, where players form coalitions based on a cost, given by the weighted sum of the players' errors. Therein the authors propose an algorithm to find an optimal arrangement of players and analyze the trade-off between stability and optimality using the Price of Anarchy.

VII. Conclusions

We considered a set of rational nodes that can cooperatively train or fine-tune ML models according to the decentralized learning paradigm. To incentivize the nodes to participate in the training, we envisioned a game-theoretic mechanism, GENIAL, that achieves the nodes' fair Pareto optimal operating points and constitutes a Nash equilibrium. Our numerical experiments support the theoretical findings, and indicate that the presence of parasite nodes seeking to improve their short-term benefit by deviating from the Nash equilibrium hurts the performance of *both* generous and parasite, thus discouraging a parasite behavior. Future work will also include results concerning the task of fine-tuning ML models, will address the presence of malicious as well as misbehaving nodes, and it will evaluate both the overhead due to the decentralized learning approach and the overall node's energy consumption.

Acknowledgment

This work was supported by the European Commission through Grant No. 101095890 (PREDICT-6G project).

References

[1] B. McMahan, E. Moore, D. Ramage *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *PMLR Artificial intelligence and statistics*, 2017.

[2] Q. Li, Z. Wen, Z. Wu *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[3] X. Lian, C. Zhang, H. Zhang *et al.*, "Can decentralized algorithms outperform centralized algorithms? A case study for decentralized parallel stochastic gradient descent," *Advances in Neural Information Processing Systems*, 2017.

[4] A. Koloskova, N. Loizou, S. Boreiri *et al.*, "A unified theory of decentralized SGD with changing topology and local updates," in *PMLR International Conference on Machine Learning*, 2020.

[5] J. Kang, Z. Xiong, D. Niyato *et al.*, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet of Things Journal*, 2019.

[6] M. A. Nowak and K. Sigmund, "Tit for tat in heterogeneous populations," *Nature*, 1992.

[7] V. Srinivasan, P. Nuggehalli, C. Chiasserini *et al.*, "An analytical approach to the study of cooperation in wireless ad hoc networks," *IEEE Transactions on Wireless Communications*, 2005.

[8] J. Hestness, S. Narang, N. Ardalani *et al.*, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[9] F. Malandrino, C. F. Chiasserini, N. Molner *et al.*, "Network support for high-performance distributed machine learning," *IEEE/ACM Transactions on Networking*, 2023.

[10] F. Malandrino, G. Di Giacomo, A. Karamzade *et al.*, "Matching DNN compression and cooperative training with resources and data availability," in *IEEE INFOCOM*, 2023.

[11] X. Li, K. Huang, W. Yang *et al.*, "On the convergence of fedavg on non-iid data," in *ICLR*, 2020.

[12] Available at https://github.com/Giuse1/generosity/, accessed: 2023-01-12.

[13] W. Y. B. Lim, N. C. Luong, D. T. Hoang *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[14] X. Tu, K. Zhu, N. C. Luong *et al.*, "Incentive mechanisms for federated learning: From economic and game theoretic perspective," *IEEE Transactions on Cognitive Communications and Networking*, 2022.

[15] A. B. Akbay and J. Zhang, "Distributed learning with strategic users: A repeated game approach," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.

[16] L. U. Khan, S. R. Pandey, N. H. Tran *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Communications Magazine*, 2020.

[17] M. Tang and V. W. Wong, "An incentive mechanism for cross-silo federated learning: A public goods perspective," in *IEEE INFOCOM*, 2021.

[18] K. Donahue and J. Kleinberg, "Optimality and stability in federated learning: A game-theoretic approach," *Advances in Neural Information Processing Systems*, 2021.