Introducing FedPref: Federated Learning Across Heterogeneous Multi-objective Preferences

Maria Hartmann[a,*], **Grégoire Danoy**[a,b] **and Pascal Bouvry**[b]

[a]SnT, University of Luxembourg, Luxembourg
[b]FSTM/DCS, University of Luxembourg, Luxembourg

Abstract. Multi-objective problems occur in all aspects of life; knowing how to solve them is crucial for accurate modelling of the real world. Rapid progress is being made in adapting traditional machine learning paradigms to the multi-objective use case, but so far few works address the specific challenges of distributed multi-objective learning. Federated Learning is a distributed machine learning paradigm introduced to tackle problems where training data originates in distribution and cannot be shared. With recent advances in hardware and model capabilities, Federated Learning (FL) is finding ever more widespread application to problems of increasing complexity, from deployment on edge devices to the tuning of large language models. However, heterogeneity caused by differences between participants remains a fundamental challenge in application. Existing work has largely focused on mitigating two major types of heterogeneity: data and device heterogeneity. Yet as the use of FL evolves, other types of heterogeneity become relevant. In this work, we consider one such emerging heterogeneity challenge: the preference-heterogeneous setting, where each participant has multiple objectives, and heterogeneity is induced by different preferences over these objectives. We propose FedPref, the first Personalised Federated Learning algorithm designed for this setting, and empirically demonstrate that our approach yields significantly improved average client performance and adaptability compared to other heterogeneity-mitigating algorithms across different preference distributions.

1 Introduction

Many real-world problems inherently involve the consideration of multiple objectives, from choosing a commuting route to work that maximises speed, but also minimises financial cost and environmental impact, to designing mechanical components for spacecraft that both minimise weight and maximise durability. When attempting to model such problems for computation, capturing this complexity is essential to producing meaningful solutions. As a result, multi-objective optimisation (MOO) problems have been studied for decades; more recently, the need for multi-objective approaches is also increasingly being recognised in the machine learning community. Recent research efforts include e.g. multi-objective variants of Reinforcement Learning (MORL) and Neural Networks [8]. The extension of multi-objective learning (MOL) to distributed settings, however, has received less attention to date. Federated Learning (FL), in particular, is rapidly gaining relevance, enabling joint learning on distributed platforms under constraints that would previously have prevented collaboration. Adapting FL to MOL problems could allow the solving of more complex problems in distribution. In this work, we present FedPref, a first adaptive algorithm designed to perform Federated Learning in a common multi-objective setting: multiple distributed participants solving the same multi-objective learning problem with mutually different preferences over these objectives.

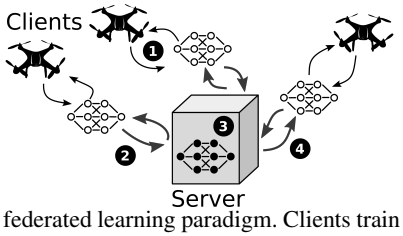

Figure 1: The federated learning paradigm. Clients train local models (1); these are periodically transmitted to a server (2), where they are aggregated into a global model (3), to be returned to the clients (4).

The FL paradigm was developed to enable collaborative machine learning in use cases where training data originates in distribution, and the distributed datasets cannot be shared. Reasons precluding the sharing of data involve concerns such as privacy, e.g. for models trained with personal preferences of smartphone users; confidentiality, such as for medical datasets collected by different healthcare providers; or simply technological constraints that limit the volume of data that can be transmitted, such as on drones or small spacecraft, where a limited communication budget is available. In such settings, FL allows participants to exploit the data available in distribution without violating these data separation constraints. The fundamental idea – illustrated in Figure 1 – is to shift the model training to the distributed participants, with each participant training a separate local model only on the locally available dataset. Only the resulting local models are shared periodically with a central server or directly with other participants, where multiple such models are aggregated to enhance accuracy. Different aggregation strategies are possible; a classical method is the weighted averaging of model weights. This federated approach permits participants to implicitly share information contained in their training data without exposing the data itself. Federated Learning has been shown to work very well in settings where participants' knowledge and capabilities are well-matched, yielding model accuracies matching those of models trained on the equivalent centralised datasets. However, in other settings FL is known to struggle – heterogeneity between clients, in particular, remains a well-known major challenge in FL, complicating the aggregation of distributed models. Differences between par-

* Corresponding Author. Emails: {firstname.lastname}@uni.lu.

ticipants, such as imbalanced data distributions or different hardware capabilities, lead to variances in the development of local models during training. This causes difficulties in converging to an accurate global model. State-of-the-art approaches for handling federated heterogeneity have largely focused on solving the two variants mentioned previously (known in the literature as data and device heterogeneity, respectively); yet other challenging types of heterogeneity can occur in real-world settings.

In this work, concerned with solving multi-objective learning problems in federation, we identify *preference heterogeneity* as a new challenge. This describes the setting where all participants solve the same MOL problem, but assign different importance preferences to each objective. The challenge is to differentiate between clients that can benefit from joint model training, i.e. clients whose objective functions are compatible at a given training stage, and clients whose objectives conflict. We propose FedPref, a first adaptive algorithm designed for such scenarios, where clients train under highly heterogeneous input. Our contributions can be summarised as follows:

- We describe and formalise a new type of heterogeneity problem that occurs naturally in the federated setting.
- We propose FedPref, a new algorithm to efficiently perform personalised Federated Learning in this setting, adaptively aggregating similar models based on a modified version of the cosine similarity metric. This algorithm does not require any knowledge about client preferences to function.
- We demonstrate the successful performance of our algorithm compared to several baselines on a range of MORL benchmark problems, and show that existing algorithms designed for data heterogeneity do not easily transfer to the preference-heterogeneous setting without loss of performance.
- We provide extensive additional validation experiments, studying the performance of the different components of our algorithm and the impact of different parameter choices.

2 Background and related work

Federated Learning was first proposed by McMahan et al. [10] as a way of training a single global model on a distributed dataset. In recent years, however, another variant has emerged in Personalised Federated Learning (PFL). Instead of training a global model that generalises over all clients' datasets, the aim of PFL is to find an individual – that is, personalised – model for each client, optimised to best fit that client's data specifically. With careful aggregation, each client in a PFL system can use the collaboration with the others to its own advantage. PFL is often deployed in heterogeneous settings, where finding a generalised global model satisfying all clients is much harder than obtaining individual solutions. In our work, we also choose a personalised learning approach to allow each client to find an individual fit for its personal multi-objective preferences.

Of the two recent works [15][6] that have addressed the federated multi-objective setting, neither has deployed a PFL strategy. The former work allows for preference-heterogeneous clients, similar to our work, but nevertheless trains only a single model across all clients, obtaining a pareto-stationary solution. In our work, we train a personalised model for each client, permitting clients to find a model tailored to their local preferences. This approach also allows us to strategically group clients by the compatibility of their learned models, enabling a more effective exploration of the search space.
The latter work considers a scenario where clients do not have fixed personalised preferences. In our work, we assume – arguably more realistically – that each client does have preferences describing the individual importance of each objective to the client, and that the server has no knowledge of or control over these preferences.

Beyond these two works, no other works in the literature – to the best of our knowledge – currently address this continuous objective-heterogeneous setting. However, several related problems have been studied previously: the problem of Multi-Task FL (MTFL) concerns a setting where each client has a single task, and different clients may have different tasks. This is an edge case of our problem, recoverable from the general formulation (where each task corresponds to one objective) by assigning a preference of zero to all but one objective. Several works in the literature address MTFL[2][5]. Most of these, e.g. [5], focus on attempting to cluster clients that solve the same task, excluding others from aggregation. Our work, in contrast, aims to adaptively aggregate clients with different tasks (preferences) while their training process is compatible, and separated them only if their training conflicts. In a similar sense, Cai et al. [2] propose to perform weighted aggregation across clients with different single tasks, based on a model similarity metric. However, their approach includes only weighted aggregation, with no clustering mechanism to permanently separate clients once they diverge.

Finally, our problem is also related to the other types of heterogeneity problems where the FL algorithm must account for differences between clients, such as data heterogeneity or hardware heterogeneity. Particularly in the case of data heterogeneity, client models also tend to develop in different directions, making the comparison with our problem setting an interesting one. Many varied approaches have been proposed to address this problem [16]; the ones most relevant here are those that do not rely on specific knowledge of the heterogeneity. One of the first such approaches was the FedProx framework [9], which relies on regularisation to encourage model adaptation, discouraging clients from diverging too far from the global model. Another approach proposes training personalised models for each client by clustering clients recursively based on direct knowledge about the underlying data distributions [17] or based on model similarity [12]. The latter work, proposing the Clustered FL [12] (CFL) algorithm, is of particular interest here. It deals with settings where the underlying data distributions known to participants are not fully compatible, leading to conflicts in the training of a joint model. To solve this, the idea of CFL is to train clients together in a classical federation until the global model converges to a stationary point, allowing clients to learn from each other until mutual conflicts stall the training process. Then clients are permanently separated into clusters based on the similarity of model gradients in the stationary point. Our multi-objective preference-heterogeneous setting is related to the data-incongruity problem tackled by CFL, in that we expect clients with preferences for conflicting objectives to also produce incompatible models during training. However, we expect the heterogeneity of clients to be more complex, given the number of potential objectives and different preference distributions. Therefore, we take inspiration from the clustering strategy of CFL for our approach, but additionally introduce the idea of personalising learning inside each cluster. Our aim is to allow a higher degree of individual exploration for clients at an earlier stage in the training, without cutting off cooperation earlier than necessary.

3 The FedPref algorithm

3.1 Problem formulation

We want to perform personalised Federated Learning across n clients, each of which has a learning problem with the same m dis-

tinct objectives $f_1, \cdots, f_m$. There is no general importance order assigned between objectives, but each client has a personal fixed preference weight vector across all objectives. Following a classical approach in multi-objective optimisation and multi-objective machine learning, we map this multi-objective problem to a single-objective problem in order to solve it, so that all clients learn a linear combination of these same objectives, with the preference weights assigned as scalars. Then client i is learning to optimise the objective function

$$f^i(\theta) = f(\vec{w^i}, \theta) = \vec{w^i}\vec{f}(\theta) = \sum_j^m w_j^i f_j(\theta), \tag{1}$$

where $\vec{w^i} = (w_1^i, \ldots, w_m^i)^T$ is the preference distribution assigned to this client. The preference distribution of each client is unknown to all other participants, including the federated server – we note that we can assume without loss of generality that all single-objective components f_j are known to all clients. Each client i trains a *personalised* model θ_i using its personal preference weights.

3.2 Concept sketch and definitions

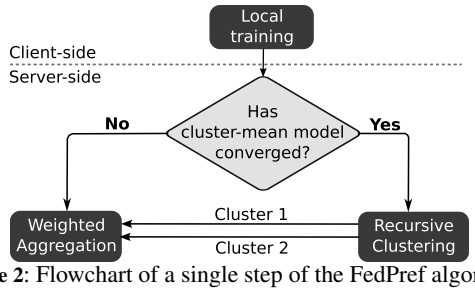

Figure 2: Flowchart of a single step of the FedPref algorithm.

The fundamental concept of the FedPref algorithm is to combine a recursive clustering mechanism, similar to CFL [12], and an adaptive weighted aggregation scheme, both based on a model similarity metric. The underlying idea behind this combination is to enable effective grouping and aggregation of clients whose preferences are compatible during the learning process (provided by the clustering component), while also maintaining the flexibility of training a personalised model for each client using weighted aggregation.

Figure 2 provides an illustration of the interaction between these components, also detailed below. Initially, all participating clients are grouped together in a single cluster. During every aggregation step, a personalised model is computed for each client, using adaptive weights computed based on mutual model similarity with all other clients in the cluster. The mean model of all clients in the cluster serves as an indicator of the success of the intra-cluster collaboration: the mean model converges if either all clients converge, or if the gradients of personalised client models start developing in conflicting directions. In this case, we perform a recursive clustering step, splitting the current cluster in two based on the same model similarity metric that is used for the weighted aggregation. The learning process is then continued in the same manner inside the new clusters.
Before discussing the functionality of each component in detail in the following sections, we here briefly discuss the modified similarity metric that underpins both components. The similarity metric in aggregation round t is computed on the basis of model updates

$$\Delta\theta_i = \theta_i - \bar{\theta}_C^{t-1}, \tag{2}$$

where $\bar{\theta}_C^{t-1}$ is the cluster-mean model obtained after the previous aggregation step. Using these gradients, we define the similarity of two models θ_i and θ_j on the basis of the standard cosine similarity:

$$cossim(\vec{u}, \vec{v}) = \frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\| \cdot \|\vec{v}\|}. \tag{3}$$

Here it suffices to consider the properties of the plain cosine similarity – note that the metric $sim(\cdot, \cdot)$ used in our algorithm is a modification thereof, described in full detail in the appendix. The fundamental idea of both metrics is to describe the geometric relation between two vectors, i.e. model gradients, in terms of a scalar in the range of $[-1, +1]$. Two collinear vectors (parallel and pointing in the same direction) have a cosine similarity of $+1$, two orthogonal vectors have a score of 0, and two vectors that are parallel, but pointing in opposite directions have a cosine similarity of -1. We consider the gradient updates of two models following a local learning step to be more compatible if their cosine similarity is higher, that is, the gradients are pointing in more similar directions on the loss surface.

3.3 Weighted aggregation

The weighted aggregation – formalised in Algorithm 1 – is carried out by the server for each separate cluster. Each weighted aggregation phase begins with computing the similarity matrix of the model gradients of all clients contained in the cluster. Recall that the similarity metric (defined in Equation 5) returns a value between -1 and $+1$, representing the lowest and highest possible similarity, respectively. These values are then clipped to a minimum lower similarity bound s_{min} – given to the algorithm as a parameter during initialisation – and subsequently normalised to the range $[0, 1]$ (see line 5 in Algorithm 1). This step enforces a minimum similarity required for aggregation, excluding all clients whose similarity to another client is lower than the given threshold from aggregation with that client. Following this precomputing of similarity values, the actual personalised aggregation takes place: to compute the new personalised model for each client, the similarity values, normalised once more so that the sum of weights adds up to one, are used to compute the weighted average of all client models – see lines 9 and 10 in Algorithm 1. This aggregation is repeated for each client inside the cluster; the resulting personalised models are returned to the respective clients.

Algorithm 1 Weighted aggregation

1: C list of c clients in cluster
2: $\mathcal{W} \leftarrow (0)^{c \times c}$ ▷ *Init aggregation-weight matrix*
3: **for** $i \in C$ **do**
4: **for** $j \in C$ **do**
5: $w_{ij} \leftarrow (sim(\Delta\theta_i, \Delta\theta_j) - s_{min})/(1 - s_{min})$
6: **end for**
7: **end for**
8: **for** $i \in C$ **do**
9: $\hat{w}_i \leftarrow w_i/|w_i|$
10: $\theta_i \leftarrow \sum_{c \in C} \hat{w}_{ic}\theta_c$
11: **end for**
12: **return** $(\theta_c | c \in C)$

3.4 Recursive clustering

The clustering procedure is performed whenever a cluster is found to have converged during an aggregation round. The exact convergence criterion is discussed in Section 3.5. The purpose of this procedure is to separate the clients contained in the cluster into two new sub-clusters in such a way that clients whose models are developing similarly are grouped together to continue learning from each other,

and clients that are no longer compatible are separated. This bipartitioning is based on the same similarity metric as the weighted aggregation. In principle, different clustering algorithms could be suited to performing the clustering itself; in this work, we choose to use spectral clustering [3], as it tends to produce well-balanced clusters, performs well for low numbers of clusters, and an implementation is readily available in common libraries. Clustering is performed no more than once per cluster per aggregation round.

Algorithm 2 Clustering

Require: $\|\Delta\bar{\theta}_C\| \leq \varepsilon$ ▷ *Cluster-avg model change* $\leq \varepsilon$
C list of c clients in cluster
$\mathcal{S} \leftarrow (0)^{c\times c}$ ▷ *Init similarity matrix*
for $i \in C$ **do**
  for $j \in C$ **do**
    $\Delta\theta_i, \Delta\theta_j \leftarrow \bar{\theta}_C^{t-1} - \theta_i, \bar{\theta}_C^{t-1} - \theta_j$
    $\mathcal{S}_{ij} \leftarrow sim(\Delta\theta_i, \Delta\theta_j)$
  end for
end for
$C_1, C_2 \leftarrow SpectralClustering(C, \mathcal{S}, 2)$ ▷ *Bipartition* $\mathcal{C}$
return C_1, C_2

3.5 Full algorithm

The complete FedPref algorithm combines the weighted aggregation and clustering components, as detailed in Algorithm 3 and conceptually in Figure 2. In every round, all local models are trained for a fixed number of steps. Once all models for a given cluster C have been reported to the server, the aggregation phase begins. As a first step, the clustering criterion is checked: the difference of the cluster-mean model $\Delta\bar{\theta}_C$ of the most recent local updates to the cluster-mean model $\bar{\theta}_C^{t-1}$ following the latest aggregation round is computed (see lines $7-8$ in Algorithm 3). If the magnitude of this change is less than a given convergence threshold ϵ, we assume that the models of clients inside the cluster are diverging. We therefore trigger the clustering process to bipartition the current cluster C into two new clusters C_1 and C_2. We then carry out weighted aggregation according to Algorithm 1 on the new clusters, before updating the server-side record of current clusters.

If the clustering criterion is not met, aggregation continues in the preexisting cluster: weighted aggregation is carried out in this cluster, and client-membership of this cluster is recorded unchanged.

In one full server-side aggregation step, this procedure is executed for every cluster, with personal aggregated models returned to the clients of each cluster after aggregation has concluded. The algorithm terminates after T such aggregation rounds. Note that even if clients are still part of a larger cluster after $T-1$ aggregation rounds, no aggregation is performed after the final local training round, to allow clients a degree of local fine-tuning (see line 23 in the algorithm).

4 Evaluation

4.1 Implementation and setup

Faced with a lack of standard benchmarking problems for federated multi-objective learning, we choose to use a number of multi-objective reinforcement learning (MORL) environments as our validation problems. These represent an intuitive class of multi-objective problems with varying characteristics and complexity, are extensions of classical RL baselines, and are implemented in a well-documented set of Python libraries [4] that aids reproducibility. We run our experiments on three such MORL environments: Deep-Sea Treasure [13]

Algorithm 3 FedPref-Server

1: $\mathcal{C} \leftarrow \{[1, \ldots, n]\}$ ▷ *Initial cluster*
2: $\theta_1^0, \ldots, \theta_n^0 \leftarrow$ Initialise client models
3: **for** $t \in 1, \ldots, T-1$ **do**
4: $\theta_1', \ldots, \theta_n' \leftarrow$ Train local models
5: **for** $C \in \mathcal{C}$ **do**
6: $\mathcal{C}_{temp} \leftarrow \{\}$
7: $\bar{\theta}_C^{t-1} \leftarrow 1/|C| \sum_{c\in C} \theta_c^{t-1}$
8: $\Delta\bar{\theta}_C \leftarrow \bar{\theta}_C^{t-1} - 1/|C| \sum_{c\in C} \theta_c'$
9: **if** $\|\Delta\bar{\theta}_C\| \leq \varepsilon$ **then** ▷ *Cluster converged*
10: $C_1, C_2 \leftarrow Clustering(\bar{\theta}_C^{t-1}, [\theta_c' | c \in C])$
11: $\Theta'_{C_1}, \Theta'_{C_2} \leftarrow \{\theta_c^t | c \in C_1\}, \{\theta_c^t | c \in C_2\}$
12: $\Theta^t_{C_1} \leftarrow WeightedAggregation(\bar{\theta}_{C_1}^{t-1}, \Theta'_{C_1})$
13: $\Theta^t_{C_2} \leftarrow WeightedAggregation(\bar{\theta}_{C_2}^{t-1}, \Theta'_{C_2})$
14: $\mathcal{C}_{temp} \leftarrow \mathcal{C}_{temp} \cup \{C_1, C_2\}$
15: **else**
16: $\Theta'_C \leftarrow \{\theta_c' | c \in C\}$
17: $\Theta^t_C \leftarrow WeightedAggregation(\bar{\theta}_C^{t-1}, \Theta'_C)$
18: $\mathcal{C}_{temp} \leftarrow \mathcal{C}_{temp} \cup \{\mathcal{C}\}$
19: **end if**
20: **end for**
21: $\mathcal{C} \leftarrow \mathcal{C}_{temp}$
22: **end for**
23: $\theta_1', \ldots, \theta_n' \leftarrow$ Train local models

(DST), Deterministic Minecart [1] (DMC) and the multi-objective extension (MO-LL) of OpenAI's Lunar Lander gym environment, using a classical DQN algorithm [11] to solve the scalarised RL problem on each client. The DQN algorithm is chosen based on its prior adaptation for the federated setting [7].

These environments represent multi-objective problems with different characteristics: the DST environment is relatively small and has a finite number of optimal solutions. The MO-LL environment is more complex and has a large number of optimal or near-optimal solutions closely aligned in the solution space. Conversely, the DMC environment has a sparse reward space, leading to a very low number of optimal solutions, which are mutually distant in the solution space. Fig. 3 presents a composite of the results obtained for different objectives across the three environments, for an illustration of these distinctions. These differences imply different challenges for federated aggregation, allowing us to analyse the suitability of FedPref to each type of problem. We measure the success of each algorithm as the mean reward obtained across all federated clients, with each client's reward scalarised according to its preference weights. The results have been made available on an interactive online platform[1]; the code of our implementation is available in a git repository[2].

Figure 3: Sample illustrations of solution spaces of different environments, separated by objectives. Left to right: Set of all solutions obtained experimentally for MO-LL, DMC and DST environments. For MO-LL and DMC, results have dimension 4 and 3, respectively, and are here represented as projections into a coordinate plane.

[1] https://wandb.ai/fed-mo/mofl-d/reports/Validating-FedPref--Vmlldzo5MTA1Mjg5?accessToken=7b1jrext64hye560zwq93uefgf1k3zfd2moqn290ys1sml2pbqfo3po5yuxv4fk9

[2] https://gitlab.com/maria.hartmann/FedPref

4.2 Comparison to baselines

4.2.1 Experimental setup

We compare our FedPref algorithm both to the classical baselines and to several state-of-the-art algorithms developed to deal with other types of heterogeneity. As baselines, we run the same local learning algorithms with no communication between clients (no-communication) and the classical federated averaging (FedAvg) algorithm, averaging the models of all clients while disregarding heterogeneity. To the best of our knowledge, no previous algorithms targeting this type of heterogeneity have been proposed in the literature; we therefore validate our approach against three algorithms from related fields that appear most relevant to our setting: FedProx [9], Many-Task Federated Learning [2] (MaTFL) and Clustered Federated Learning [12] (CFL). FedProx is a classical approach to the heterogeneity problem, commonly used as a baseline in data-heterogeneous settings. The underlying strategy appears intuitively to have the potential to transfer to the preference-heterogeneous setting, so we choose to retain this baseline. The MaTFL and CFL algorithms are chosen for the similarity of their approaches with the weighted aggregation and the clustering component of our algorithm, respectively; they also represent the two fields of Multi-Task FL and data-heterogeneous FL that we identified earlier in this work as most closely related to our problem setting. We tune the hyperparameters for all algorithms via an initial grid search on a set of preferences sampled from a Dirichlet distribution. For each algorithm and environment, we select the best-performing hyperparameter configuration from this search. The details of this parameter search and the local configurations of clients for each RL problem are reported in the supplementary material. Following the parameter tuning, we run all algorithms with client preference weights generated according to three different distributions: sampled from a Dirichlet distribution, sampled from a Gaussian distribution or weights generated to be equally spaced in the weight simplex.

4.2.2 Analysis

We report the numerical results obtained for all algorithms and distributions in Table 1. In the remainder of this section, we will discuss and contrast these results separately by preference distribution, from the most "extreme" preference differences between clients – the equidistant distribution – over the Dirichlet distribution to the Gaussian distribution, where client preferences are most similar.

Equidistant preferences. Under the equidistant distribution of preference weights across clients, we observe that FedPref outperforms all other algorithms quite significantly on two out of three environments. For the MO-LL environment, clients participating in FedPref obtain a mean scalarised reward of 29.470, far ahead of the second-highest result of 18.489 on the same environment. Indeed, the latter result is not accomplished by any federated algorithm, but by the baseline of non-communicating clients, with the remaining federated algorithms achieving much lower scores down to the lowest mean result of -94.059, returned by the CFL algorithm. Results for DST follow a similar pattern, while for the DMC environment no federated algorithm outperforms the result of the non-federated baseline. These results underscore the difficulty of this distribution – equidistant preference weights likely represent the most "extreme" scenario among our experiments, where individual client objectives have the greatest mutual differences. In general, we would expect this to also map to greater differences in the models that match the preferences of each client, resulting in an advantage for those algorithms training personalised models. Our results, reported in rows 4-6 of Table 1, appear to support this, as both FedProx and FedAvg perform notably worse in this scenario than for the other two types of preference distributions. This pattern persists across all experimental environments. Somewhat more surprisingly, we also observe a poor performance by CFL for many environments in this setting - further investigation shows that the greedy clustering algorithm defined for CFL [12] tends to yield highly unbalanced clusters in our experiments. As clusters in CFL train a single global model, this leads to less personalised models – a disadvantage for this type of preference distribution.

Indeed, in this scenario it becomes particularly important for any personalised algorithm to accurately judge the compatibility of models, and to separate non-compatible models. This appears to be a strength of our algorithm: FedPref not only outperforms all others by a significant margin in two out of three environments; in the third environment (DMC) none of the algorithms tested here perform better than the non-federated baseline. It appears likely that the high sparsity of the reward space, combined with the greater difference in client objectives, makes it difficult to group clients for aggregation.

Uniformly-sampled preferences. Under the Dirichlet distribution, our results, reported in row 1-3 of Table 1, show FedPref outperforming all other algorithms on all three experimental environments. Compared to the results obtained under the equidistant preference distribution, some of the gains of the FedPref algorithm over those compared, though still existent, are less drastic, particularly on the dense solution space of the MO-LL environment: In this case, e.g. the FedProx algorithm yields a mean client reward of 31.195, relatively close to the top result of 32.220 achieved by the FedPref algorithm. However, the difference remains larger for the DST environment, likely due to its discrete solution set: Here, FedPref obtains a mean scalarised client reward of 4.409, still followed by the no-communication baseline with a mean reward of -0.426. The ranking of algorithmic results is similar on the DMC environment, though less decisive. It appears that the lower density of (optimal) solutions available in the latter two environments, combined with the intermediate objective heterogeneity of this setting, continues to present a difficult challenge to the federated algorithms from the literature.

Gaussian-sampled preferences. Finally, in the setting where weight preferences are drawn from a Gaussian distribution, different algorithms achieve the top scores for each environment (see results in row 7-9 of Table 1): in the MO-LL environment, the CFL algorithm obtains a mean scalarised client reward of 37.726, slightly higher than the second-highest score of 36.434, returned by the FedPref algorithm. On the DMC environment, the FedAvg algorithm gives the highest score of -2.466, again followed by the FedPref algorithm with a score of -2.527. Results on the DST environment remain dominated by the FedPref algorithm, with no other federated algorithm outperforming the non-federated baseline.

Under this distribution, clients are more likely to have more similar preferences, potentially supporting more similar models. In this case, plain (equally-weighted) aggregation appears to do well, with the CFL algorithm delivering the best performance on the MO-LL environment. The two non-PFL algorithms also perform notably better under this preference distribution than in the other two settings - in fact, in this case the plain FedAvg algorithm outperforms all others in the DMC environment. While this result may be owing to the sparse solution space of the problem, with the federated clients jointly converging on a single local optimum, it remains part of a wider trend.

Conclusion. Following the detailed analysis of results by preference distribution, we conclude with a number of general observations. Firstly, FedPref yields the greatest improvement over the compared

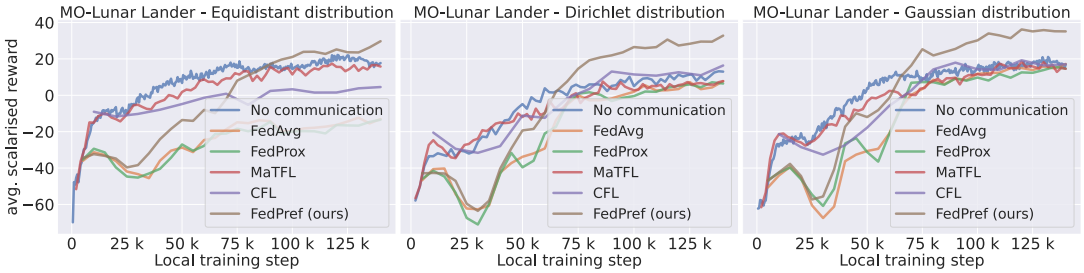

Figure 4: Mean scalarised reward obtained by different algorithms on the MO-LL environment, compared across preference distributions.

approaches in settings with medium-to-high preference heterogeneity – represented here by the Dirichlet and equidistant distributions – and on problems with medium-to-high solution density, such as the MO-LL and DST environments in our experiments. Secondly, we note that whenever our algorithm does not deliver the best performance, it is outperformed by only one other, and never twice by the same algorithm. FedPref is also notably more adaptable to different types of preference distributions than the other algorithms compared here. This indicates that FedPref is a good overall choice in the general case, where the distribution of preference weights of the characteristics of the learning problem may be unknown.

4.3 Ablation study

We perform an ablation study of the FedPref algorithm, comparing its performance with that of its individual components, i.e. performing only weighted aggregation or only the clustering strategy, respectively. This and the following validation experiments are limited to preferences generated from a Dirichlet distribution, representing the "medium-heterogeneous" setting in our comprehensive experiments in the previous section. The results, reported in Table 2, show two different outcomes for the three different types of problems we study.

Results. For the DMC environment with its very sparse reward space, we observe that both the clustering and the weighted-aggregation component perform better individually than combined – the clustering-only component achieves the highest average scalarised client reward of -1.526, whereas the combined components yield a mean reward of -2.423. In this case, it is likely that the individual clients' preferences ultimately lead to very different optimal models, with less benefit obtained from cooperation between different models. This hypothesis is also supported when comparing the results for the FedProx and FedAvg algorithm, discussed in Section 4.2.2, Table 1, for this environment. The approach behind these two algorithms forces a high level of collaboration between the clients, and does not lead to high overall results for this environment when preferences are sufficiently different. Of the two individual components, the clustering component likely succeeds more quickly in separating very different models, with the cluster-mean model converging more definitively. The weighted-aggregation component, achieving here the median result of -2.204, might allow for a more extreme divergence more quickly, leading to more effective similarity weight assignment than under the combined components. Nevertheless, it should be noted that all three variants succeed in outperforming the compared approaches under the Dirichlet distribution – see Table 1. In contrast, we observe significantly improved results for the combined components over each component individually for both the MO-LL and DST environments. We note that in both cases, the weighted aggregation performs quite badly in isolation – yielding mean scalarised rewards of 17.697 and -31.243, respectively – but in combination with the clustering strategy leads to a notable improvement over both isolated strategies. For the MO-LL environment, this is roughly an 82% increase over the weighted aggregation component, and an 18% increase over the clustering component; for the DST environment, the improvements are similarly notable.

In both of these cases, the poor performance of the weighted aggregation component in isolation is likely related to the choice of a low minimum-similarity threshold for both environments - without the clustering mechanism to separate clients by similarity, this could lead to the forced aggregation of incompatible clients. This effect might be less pronounced for the DMC environment, both because clients in this environment appear to separate very quickly into high mutual dissimilarity, and because the initial hyperparameter search for this environment led to a higher minimum-similarity threshold in the first place, additionally supporting the quick separation of clients.

4.4 Impact of similarity bound

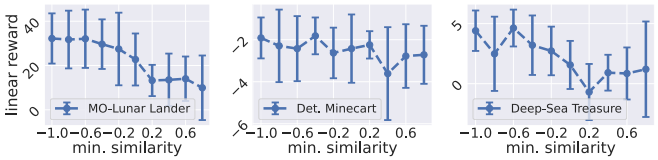

Figure 5: Impact of the min-similarity threshold on mean client reward. Left to right: results for MO-LL, DMC and DST environments.

We study the performance impact of a hyperparameter that is integral to our algorithm: the lower similarity bound used in computing aggregation weights. The full numerical results for this study may be found in the supplementary material, along with a further sensitivity analysis of our similarity metric. The results for the min-similarity threshold (see Figure 5) show commonalities across all three environments, suggesting that thresholds lower than 0 are remarkably beneficial to the learning outcome: the relative improvement in mean client reward between a threshold of 0 and the optimal discovered value ranges from 21.1% for the DMC environment with threshold -1 to a full 296.8% improvement for the DST environment with threshold -0.6. Though counter-intuitive at first glance, given the geometric interpretation of cosine similarity, this outcome is quite reasonable in the context of our algorithm. Firstly, we note that the purpose of our algorithm's clustering strategy is to cluster those clients together that can benefit from collaboration. Improved results for a lower min-similarity threshold indicate that this grouping is successful, as even relatively dissimilar clients inside the same cluster improve with collaboration. Secondly, the fact that clients train personalised, i.e. different, models means that some dissimilarity is induced by definition of the metric, through the choice of the cluster-mean model as a reference point in computing the cosine similarity.

Table 1: Experimental results comparing our algorithm to MaTFL, CFL, FedProx, FedAvg and individual learning without cooperation.

		No comm.	FedAvg	FedProx	CFL	MaTFL	FedPref (ours)
Dirichlet	MO-Lunar Lander	$14.318\ \sigma 13.34$	$30.516\ \sigma 14.74$	$31.195\ \sigma 16.90$	$31.177\ \sigma 17.85$	$7.817\ \sigma 9.89$	**32.220** $\sigma 11.34$
	Det. Minecart	$-2.524\ \sigma 0.86$	$-3.203\ \sigma 3.40$	$-3.345\ \sigma 3.57$	$-2.759\ \sigma 0.87$	$-4.388\ \sigma 2.17$	**-2.423** $\sigma 1.63$
	Deep-Sea Treasure	$-0.426\ \sigma 1.81$	$-16.502\ \sigma 24.69$	$-11.046\ \sigma 22.14$	$-12.895\ \sigma 21.77$	$-6.325\ \sigma 3.63$	**4.409** $\sigma 1.68$
Equidistant	MO-Lunar Lander	$18.489\ \sigma 10.49$	$-55.641\ \sigma 46.91$	$-73.980\ \sigma 45.33$	$-94.059\ \sigma 30.82$	$11.992\ \sigma 6.53$	**29.470** $\sigma 3.50$
	Det. Minecart	**-1.696** $\sigma 1.52$	$-6.771\ \sigma 0.25$	$-6.753\ \sigma 0.22$	$-2.736\ \sigma 0.48$	$-4.033\ \sigma 1.19$	$-2.213\ \sigma 1.71$
	Deep-Sea Treasure	$0.685\ \sigma 1.89$	$-17.871\ \sigma 26.23$	$-23.230\ \sigma 26.77$	$-34.623\ \sigma 23.51$	$-6.159\ \sigma 2.76$	**2.780** $\sigma 2.54$
Gaussian	MO-Lunar Lander	$15.326\ \sigma 13.42$	$31.532\ \sigma 13.12$	$32.745\ \sigma 12.99$	**37.726** $\sigma 11.49$	$6.480\ \sigma 9.41$	$36.434\ \sigma 7.34$
	Det. Minecart	$-3.607\ \sigma 2.22$	**-2.466** $\sigma 3.27$	$-2.941\ \sigma 3.08$	$-4.004\ \sigma 1.80$	$-5.038\ \sigma 1.12$	$-2.527\ \sigma 1.78$
	Deep-Sea Treasure	$1.133\ \sigma 0.93$	$-13.317\ \sigma 25.63$	$-2.066\ \sigma 16.01$	$-24.819\ \sigma 27.55$	$-6.243\ \sigma 3.35$	**2.868** $\sigma 2.65$

Table 2: Experimental results comparing the individual and combined components of the FedPref algorithm.

	MO-LL	DMC	DST
Clustering only	$27.265\ \sigma 13.9$	**-1.526** $\sigma 0.8$	$0.9475\ \sigma 3.8$
Weighted agg. only	$17.697\ \sigma 9.8$	$-2.204\ \sigma 0.9$	$-31.243\ \sigma 8.5$
FedPref	**32.22** $\sigma 12.0$	$-2.423\ \sigma 1.7$	**4.409** $\sigma 1.8$

4.5 Validation of clustering strategy

We validate the clustering strategy on all three environments by running FedPref on artificially constructed configurations where multiple clients share the same preferences. We construct two types of configurations: one where preferences are distributed among equal numbers of clients (4 distinct preference weights, each held by 5 clients), and one where the number of clients varies by preference (4 distinct preference weights, held by $2, 3, 6$ and 9 clients, respectively). We observe how well the clustering algorithm groups similar clients and how the similarity between clients develops during training. Due to scope constraints, we present only one such configuration here; the remaining results are shown in the supplementary material.

Figure 6 shows client similarities at selected steps of the training process on the MO-LL environment under the balanced preference distribution. Note that for ease of visualisation, clients with the same preferences are grouped together by index. We see in the visualisation that varying preference similarities are already reflected in the model similarity computed by our metric. At the earliest visualised stage (after five aggregation steps; left-most image in Fig. 6), all clients are still grouped together in a single cluster. Nevertheless, the weighted aggregation strategy gives individual models the freedom to develop separately, yet also appears to be successful in encouraging the weighted aggregation of clients with the same objectives.

In the second image, at an intermediate stage of the training process, a split into multiple clusters has occurred. The grouping of clients with the same preferences is preserved across experimental runs, but multiple groups of such clients continue to collaborate at this training stage, with different groups clustered together in different experimental runs. In the instance visualised here, clients $1-5$ and $11-15$ are all contained in the same cluster. In the final image, close to the end of the training phase, we observe that the similarity of the models obtained by clients with the same preferences is very high, while the similarity to other models appears lower than before. This indicates that these clients have been separated into individual clusters, and that the personalised models within these clusters are converging.

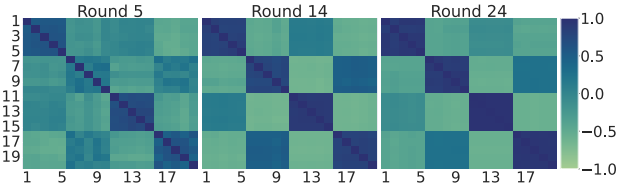

Figure 6: Mutual client similarity at different stages during a single experimental run on the MO-LL environment. Left to right: client similarities after aggregation round $5, 14$ and 24 of 28, respectively.

5 Conclusion, limitations, and outlook

In this work, we have discussed an approach for solving multi-objective learning problems in a federation of distributed participants. In particular, we have identified preference heterogeneity, a novel type of heterogeneity problem that arises naturally in many real-world scenarios, as a key challenge to be overcome by federated algorithms. This occurs when clients solve multi-objective problems, with each client assigning different preferences to each objective. To tackle this problem, we have proposed FedPref, a new algorithm to perform Federated Learning in this setting. FedPref is based on a combination of recursive clustering and weighted aggregation, both using model similarity. This algorithm preserves the privacy of clients with respect to training data and preferences. We have validated FedPref on multiple varied problems and preference distributions, comparing it to classical benchmarks as well as other heterogeneity-mitigating algorithms from the state of the art. We have shown that our algorithm outperforms the alternatives in many cases, and represents a reliable choice in all others. Further experiments were carried out to study the characteristics of the algorithm.

As this work presents an initial solution tailored to the objective-heterogeneous setting, several challenges inherent to the federated setting remain to be addressed in future work. This includes e.g. scenarios dealing with additional types of heterogeneity, such as data or hardware heterogeneity, combined with the preference heterogeneity discussed here. In principle, we expect that the model similarity-based design of our algorithm could adapt without change to a setting that includes data heterogeneity; solving device heterogeneity might require the integration of additional strategies dedicated to this purpose. Such strategies already exist in the literature; their integration into the cluster-aggregation step of FedPref appears quite feasible.

Finally, we note that, beyond the client-level performance analysis we have carried out here, a system-level analysis of the solutions obtained by different algorithms would be of interest for this setting. In multi-objective domains, the ability to find a diverse array of trade-off solutions for different objective preferences is often important; this likely extends to the federated setting.

Acknowledgements

This work is partially funded by the joint research programme UL/SnT–ILNAS on Technical Standardisation for Trustworthy ICT, Aerospace, and Construction. The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [14] – see https://hpc.uni.lu.

References

[1] A. Abels, D. M. Roijers, T. Lenaerts, A. Nowé, and D. Steckelmacher. Dynamic weights in multi-objective deep reinforcement learning, 2018. URL https://arxiv.org/abs/1809.07803.

[2] R. Cai, X. Chen, S. Liu, J. Srinivasa, M. Lee, R. Kompella, and Z. Wang. Many-task federated learning: A new problem setting and a simple baseline. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2023. doi: 10.1109/cvprw59228.2023.00532. URL http://dx.doi.org/10.1109/CVPRW59228.2023.00532.

[3] A. Damle, V. Minden, and L. Ying. Simple, direct and efficient multi-way spectral clustering. *Information and Inference: A Journal of the IMA*, 8(1):181–203, 06 2018. ISSN 2049-8772. doi: 10.1093/imaiai/iay008. URL https://doi.org/10.1093/imaiai/iay008.

[4] F. Felten, L. N. Alegre, A. Nowé, A. L. C. Bazzan, E. G. Talbi, G. Danoy, and B. C. da. Silva. A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*, 2023.

[5] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran. An efficient framework for clustered federated learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19586–19597. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/e32cc80bf07915058ce90722ee17bb71-Paper.pdf.

[6] M. Hartmann, G. Danoy, M. Alswaitti, and P. Bouvry. MOFL/d: A federated multi-objective learning framework with decomposition. In *International Workshop on Federated Learning in the Age of Foundation Models in Conjunction with NeurIPS 2023*, 2023. URL https://openreview.net/forum?id=Pj6BPHZy56.

[7] H. Jin, Y. Peng, W. Yang, S. Wang, and Z. Zhang. Federated reinforcement learning with environment heterogeneity. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, pages 18–37. PMLR, 2022. URL https://proceedings.mlr.press/v151/jin22a.html. ISSN: 2640-3498.

[8] S. Li, F. Wan, H. Shu, T. Jiang, D. Zhao, and J. Zeng. Monn: A multi-objective neural network for predicting compound-protein interactions and affinities. *Cell Systems*, 10(4):308–322.e11, Apr. 2020. ISSN 2405-4712. doi: 10.1016/j.cels.2020.03.002. URL http://dx.doi.org/10.1016/j.cels.2020.03.002.

[9] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks, 2018. URL https://arxiv.org/abs/1812.06127.

[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas. Communication-Efficient Learning of Deep Networks from Decentralized Data. In A. Singh and J. Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1273–1282. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/mcmahan17a.html.

[11] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 1476-4687. doi: 10.1038/nature14236. URL http://dx.doi.org/10.1038/nature14236.

[12] F. Sattler, K.-R. Müller, and W. Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 32:3710–3722, 2019. URL https://api.semanticscholar.org/CorpusID:203736521.

[13] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1):51–80, 2011. ISSN 1573-0565. doi: 10.1007/s10994-010-5232-5. URL https://doi.org/10.1007/s10994-010-5232-5.

[14] S. Varrette, H. Cartiaux, S. Peter, E. Kieffer, T. Valette, and A. Olloh. Management of an Academic HPC & Research Computing Facility: The ULHPC Experience 2.0. In *Proc. of the 6th ACM High Performance Computing and Cluster Technologies Conf. (HPCCT 2022)*, Fuzhou, China, 2022. Association for Computing Machinery (ACM). ISBN 978-1-4503-9664-6.

[15] H. Yang, Z. Liu, J. Liu, C. Dong, and M. Momma. Federated multi-objective learning. In A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 39602–39625. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/7cb2c2a8d35576c00078b6591ec26a7d-Paper-Conference.pdf.

[16] M. Ye, X. Fang, B. Du, P. C. Yuen, and D. Tao. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Comput. Surv.*, 56(3), oct 2023. ISSN 0360-0300. doi: 10.1145/3625558. URL https://doi.org/10.1145/3625558.

[17] Y. Yeganeh, A. Farshad, J. Boschmann, R. Gaus, M. Frantzen, and N. Navab. *FedAP: Adaptive Personalization in Federated Learning for Non-IID Data*, page 17–27. Springer Nature Switzerland, 2022. ISBN 9783031185236. doi: 10.1007/978-3-031-18523-6_2. URL http://dx.doi.org/10.1007/978-3-031-18523-6_2.

6 Appendix

This document contains supplementary material for the paper entitled *Introducing FedPref: Federated Learning Across Heterogeneous Preferences*. Section 6.2 contains detailed information on the set-up and configuration of experiments presented in the main paper, including the design and results of the hyperparameter search carried out for all algorithms; parameters used for the local training of client models, and computing resources required for the making of this paper. Section 6.3 shows additional results for the validation experiments that exceeded the scope of the main paper.

6.1 Details of the algorithm

6.1.1 Modified similarity metric

We formally introduce the modified similarity metric that underpins both components. The similarity metric in aggregation round t is computed on the basis of model updates

$$\Delta\theta_i = \theta_i - \bar{\theta}_C^{t-1}, \tag{4}$$

where $\bar{\theta}_C^{t-1}$ is the cluster-mean model obtained after the previous aggregation step. Using these gradients, we define the similarity metric $sim(\cdot,\cdot)$ of two models θ_i and θ_j as

$$sim(\Delta\theta_i, \Delta\theta_j) = \frac{1}{L}\sum_{\ell}^{L} cossim(topR(\Delta\theta_i^{\ell}), topR(\Delta\theta_j^{\ell})), \tag{5}$$

where $\Delta\theta_i^{\ell}$ is the ℓ-th layer of the neural network update $\Delta\theta_i$ and L is the total number of layers per model. The $topR$ operator is a variant of $topk$, where k is determined by the dimension of the input vector and a ratio $R \in (0, 1]$. $TopR$ maps a vector $\vec{v}$ to a vector of the same dimension where the top $k = \lceil dim(\vec{v}) \cdot R \rceil$ elements of $\vec{v}$ (in absolute terms) are retained and the remaining elements set to zero. So for $topR(\vec{v}) = \vec{u}$, we have

$$u_i = \begin{cases} v_i, & \text{if } |v_i| \text{ in top } \lceil R \cdot dim(\vec{v}) \rceil \text{ absolute elements of } \vec{v}. \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

The cosine similarity $cossim(\cdot,\cdot)$ is defined in the standard way:

$$cossim(\vec{u}, \vec{v}) = \frac{\langle \vec{u}, \vec{v} \rangle}{\|\vec{u}\| \cdot \|\vec{v}\|}. \tag{7}$$

We choose to use this modified metric instead of the more common direct applications of cosine similarity for two main reasons:

- We hope to mitigate the "curse of dimensionality" that makes this metric increasingly meaningless for larger vector dimensions.
- Selecting the subset of the largest weights for each layer allows us to compare the most impactful, or "important" aspects of the models. This could lead to more meaningful decisions about which models to aggregate together.

Our evaluation experiments contained in this appendix show that, compared to the pure cosine similarity metric without weight selection, the use of this metric does indeed lead to improved results in our validation experiments.

6.2 Details of experiment configurations

6.2.1 Hyperparameter tuning

We perform an initial hyperparameter search for all algorithms and environments, with all evaluated parameter values listed in Table 4. Each configuration was run five times on 20 clients and five different randomly-generated preference distributions. The five preference distributions remained fixed across all hyperparameter configurations to promote the comparability of results. The metric used to assess performance was the mean linearised reward obtained by the clients using their personalised preference weights. The parameter values selected as a result of the hyperparameter tuning are given in Table 3. These hyperparameters remain fixed to the same selection in our supplementary validation experiments, i.e. the ablation study and sensitivity analyses.

6.2.2 MORL environment parameters

Where available, the hyperparameters for the three MORL environments used in our experiments were obtained from published benchmark configurations. Where no such configurations were available, they were obtained by manual tuning. All modified parameters are reported below, in Tables 5, 6 and 7. Parameters that are not listed can be assumed to be set to the default setting, as implemented in the DQN algorithm of the stable-baselines3 package.

6.2.3 Computing resources

The number of experiments presented in this paper amounts to 2025 individual experimental runs. This corresponds to a total runtime of approximately 2280 hours on a single node of the computing cluster available to us.

6.3 Supplementary experimental results

In this section, we include supplementary numerical results and plots that exceeded the scope of the main paper.

6.3.1 Main validation experiments

6.3.2 Impact of topR parameter and similarity bound

This section lists numerical experimental results for the parameter sensitivity analysis carried out in the main paper; these same results are presented there in visual form, with some numbers quoted. We refer the reader to the relevant section in the main paper for the analysis and discussion of these results. This section also contains an additional analysis and discussion of the impact of the topR parameter, used in computing the client similarity metric.

Table 9 contains the results for the sensitivity analysis of the $topR$ parameter; Table 8 shows the results for the analysis of the minimum similarity threshold in aggregation. All experiments were carried out with 10 different random seeds, for 20 federated clients per run, on preference weights drawn from a Dirichlet distribution.

The parameter R describes the proportion of each model layer to be used by our metric in calculating similarity (see Equations 6 and 5, respectively, for the definitions of the $topR$ operator and our similarity metric). The minimum similarity bound is used during the weighted aggregation step to include only models exceeding a given similarity value in the aggregation.

Results for the $topR$ parameter, shown in Figure 8, indicate that this

Table 3: Complete list of parameter configurations tested during hyperparameter tuning.

		MO-Lunar Lander	Det. Minecart	Deep-Sea Treasure	Comment
No comm.	-	-	-	-	No federated parameters.
FedAvg	Number local iterations	$(2, 5, 10) \cdot 10^3$	$(2, 5, 10) \cdot 10^3$	$(5, 10, 15) \cdot 10^2$	
FedProx	Number local iterations	$2000, 5000, 10000$	$2000, 5000, 10000$	$500, 1000, 1500$	
	Proximal term μ	$0.01, 0.1, 1$	$0.01, 0.1, 1$	$0.01, 0.1, 1$	Based on recommendations in [9]
CFL	Number local iterations	$2000, 5000, 10000$	$2000, 5000, 10000$	$500, 1000, 1500$	
	Clustering threshold	$2.5, 5, 7.5$	$2, 3, 5$	$2.5, 5, 7.5$	Based on max. observed gradient magnitude[a]
	Patience	$1, 2$	$1, 2$	$1, 2$	Rounds below threshold before clustering triggered[b]
MaTFL	Number local iterations	$2000, 5000, 10000$	$2000, 5000, 10000$	$500, 1000, 1500$	
	Number voting clients k	$5, 8, 10$	$5, 8, 10$	$5, 8, 10$	Adequate range according to [2]
Ours	Number local iterations	$2000, 5000, 10000$	$2000, 5000, 10000$	$500, 1000, 1500$	
	Clustering threshold	$2.5, 5, 7.5$	$2, 3, 5$	$2.5, 5, 7.5$	Same as for CFL
	Patience	$1, 2$	$1, 2$	$1, 2$	Same as for CFL
	Minimum similarity	$-1, 0$	$-1, 0$	$-1, 0$	Used in computing aggregation weights[c]

[a] As suggested in [12].
[b] Introduced by us to handle slow initial gradient ramp-up.
[c] See Section 3.3 in the main paper for explanation.

Table 4: Parameter configurations selected for each algorithm following hyperparameter tuning.

		MO-Lunar Lander	Det. Minecart	Deep-Sea Treasure
No comm.	-	-	-	-
FedAvg	Number local iterations	5000	5000	500
FedProx	Number local iterations	5000	5000	500
	Proximal term μ	1	1	1
CFL	Number local iterations	10000	2000	1000
	Clustering threshold	5	3	5
	Patience	2	2	2
MaTFL	Number local iterations	2000	5000	1000
	Number voting clients k	10	10	10
Ours	Number local iterations	5000	5000	500
	Clustering threshold	5	3	5
	Patience	1	2	2
	Minimum similarity	-1	0	-1

Parameter name	Value
env	mo-lunar-lander-v2
policy	MlpPolicy
learning_rate	0.00063
batch_size	64
buffer_size	50000
learning_starts	0
gamma	0.99
target_update_interval	250
train_freq	4
gradient_steps	-1
exploration_fraction	0.12
exploration_final_eps	0.1
net_arch	[256, 256]

Table 5: Set of parameters used for the local training of the MO-Lunar Lander environment.

Parameter name	Value
env	minecart-deterministic-v0
policy	MlpPolicy
learning_rate	0.0002
batch_size	64
buffer_size	50000
learning_starts	50000
gamma	0.99
target_update_interval	750
train_freq	32
gradient_steps	32
exploration_fraction	0.8
exploration_final_eps	0.05
net_arch	[256, 256]

Table 6: Set of parameters used for the local training of the Deterministic Minecart environment.

parameter should be tuned carefully, as both the general trend and the optimal value appear highly sensitive to the type of learning problem. The results nevertheless support the use of this modified similarity metric: the use of a well-tuned $topR$ parameter is shown to improve performance significantly compared to the standard metric that is recovered with $R = 0$ in two out of three studied environments. For the MO-LL environment, the highest mean scalarised client reward of 33.18 is obtained for $R = 0.2$, representing an improvement of approximately 7.6% over the result of 30.84 for $R = 0$. For the DST environment, the improvement is even greater: from 2.52 for $R = 0$ to 3.76 for $R = 0.8$, an increase of roughly 49%.

6.3.3 Clustering validation

In this section, we show and briefly discuss additional results of the clustering validation experiments.

MO-Lunar Lander. Figure 9 shows the similarity of clients at three training stages during training in the MO-LL environment on an unbalanced preference distribution. Three groups with distinct similarity are clearly recognisable from the earliest stages of the training process; these correspond to the sets of clients that have been assigned the same preferences, with two such sets evidently grouped together. Later stages show the gradual separation of the different sets, likely through the clustering process. However, the two client sets that showed a high similarity from the beginning (both contained in the largest, top-left block in the figure) appear to remain in the same cluster until the end of the training process, never being separated. This could indicate either that the two different preference weights

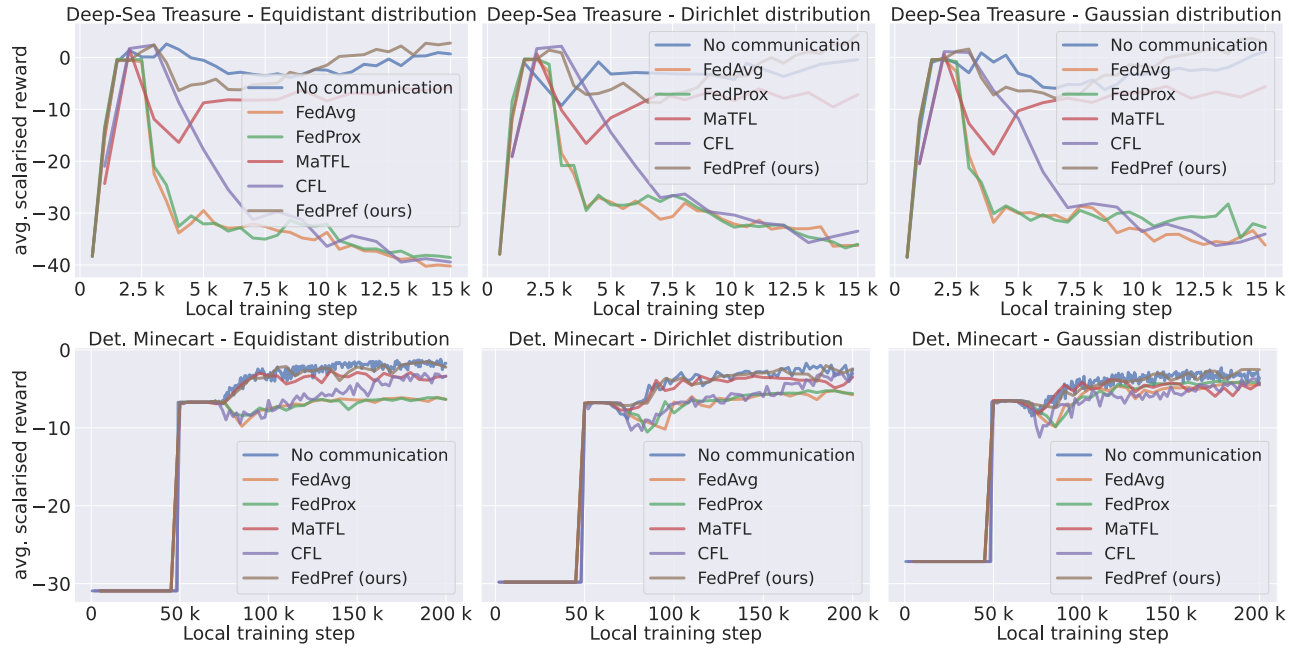

Figure 7: Mean scalarised reward obtained by different algorithms on the Deep-Sea Treasure (top) and Deterministic Minecart (bottom) environments, compared across preference distributions.

Parameter name	Value
env	deep-sea-treasure-v0
policy	MlpPolicy
learning_rate	0.004
batch_size	128
buffer_size	10000
learning_starts	1000
gamma	0.98
target_update_interval	600
train_freq	16
gradient_steps	8
exploration_fraction	0.2
exploration_final_eps	0.07
net_arch	$[256, 256]$

Table 7: Set of parameters used for the local training of the Deep-Sea Treasure environment.

Figure 8: Impact of the choice of $topR$ parameter on average reward obtained by clients. Left to right: results for MO-Lunar Lander, Deterministic Minecart and Deep-Sea Treasure environments.

assigned to the two sets are naturally compatible during the training process, or that the FedPref algorithm might sometimes struggle to fully separate incompatible sets of clients before they converge to a local optimum. The latter could also be a consequence of the imbalanced distribution of potentially incompatible clients in this case; perhaps a small number of incompatible clients is 'dominated' by the remaining large number of compatible clients in the same cluster.

Det. Minecart. Figure 10 and Figure 11 show client similarities during training on the Det. Minecart environment with the balanced and unbalanced distribution of preferences, respectively. These results also illustrate the challenges of this environment that were discussed in the main part of the paper: the sparse reward space

Threshold	MO-LL	DMC	DST
-1.0	**32.22** σ11.3	$-1.91\sigma1.0$	$4.41\sigma1.7$
-0.8	$31.74\sigma11.3$	$-2.29\sigma1.0$	$2.49\sigma1.7$
-0.6	$32.09\sigma13.0$	$-2.42\sigma1.7$	**4.63** $\sigma3.1$
-0.4	$29.65\sigma13.2$	**-1.82** $\sigma1.5$	$3.20\sigma1.5$
-0.2	$27.47\sigma11.2$	$-2.63\sigma0.9$	$2.73\sigma2.5$
0.0	$22.73\sigma16.4$	$-2.42\sigma1.2$	$1.56\sigma2.0$
0.2	$13.24\sigma11.7$	$-2.24\sigma1.6$	$-0.69\sigma2.0$
0.4	$13.53\sigma7.1$	$-3.61\sigma0.7$	$0.92\sigma2.3$
0.6	$14.03\sigma11.9$	$-2.78\sigma2.2$	$0.85\sigma1.5$
0.8	$9.95\sigma10.0$	$-2.72\sigma1.5$	$1.20\sigma2.2$

Table 8: Numerical results for minimum-similarity sensitivity analysis visualised in the main part of the paper. All configurations were run with 10 different random seeds across 20 clients per run.

appears to make it difficult to reliably discover client similarities during the clustering process. We observe in both figures that clients never reach high levels of similarity as seen in the results of the MO-LL environment; it is likely that this also impedes the clustering process, leading to a suboptimal grouping into clusters. However, some successful collaboration appears to take place, as evidences by the darker-coloured patches in the middle and right images in both figures. This matches our experimental conclusions in the main paper, that the FedPref algorithm does accomplish some useful collaboration leading to improvement of client results, but highly sparse solution spaces remain a challenge.

Deep-Sea Treasure. Sample results for the development of client

$topR$	MO-LL	DMC	DST
0.2	**33.18** σ11.1	-2.58 $\sigma1.0$	2.24 $\sigma3.0$
0.4	30.04 σ11.1	-3.18 $\sigma1.0$	1.89 $\sigma3.0$
0.6	30.42 σ13.3	**-2.44** $\sigma1.8$	1.35 $\sigma3.8$
0.8	31.17 σ11.7	-2.65 $\sigma1.3$	**3.76** $\sigma4.0$
1.0	30.84 σ10.4	-2.49 $\sigma1.2$	2.52 $\sigma2.8$

Table 9: Numerical results for $topR$ sensitivity analysis visualised in the main part of the paper. All configurations were run with 10 different random seeds across 20 clients per run.

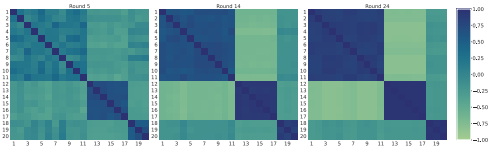

Figure 9: Mutual client similarity at different stages during a single experimental run on the MO-LL environment, with unbalanced preference distribution. Left to right: client similarities after aggregation round 5, 14 and 24 of 28, respectively.

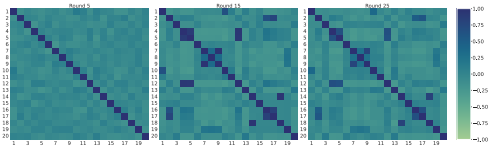

Figure 10: Mutual client similarity at different stages during a single experimental run on the DMC environment, with balanced preference distribution. Left to right: client similarities after aggregation round 5, 15 and 25 of 38, respectively.

similarity during training on the Deep-Sea Treasure environment with balanced and unbalanced preference assignment are shown in Figure 12 and Figure 13, respectively. In both figures, we observe that a grouping of clients becomes visible quite early in the learning process. Though this grouping is not perfect, it does largely correspond to those sets of clients that have been assigned the same preference. The flaws in the grouping process likely spring from an early clustering step, where preference similarities were not fully reflected in the respective model gradients.

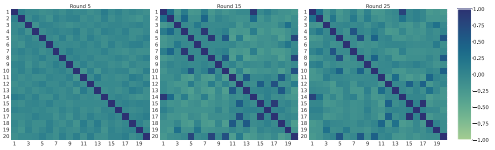

Figure 11: Mutual client similarity at different stages during a single experimental run on the DMC environment, with unbalanced preference distribution. Left to right: client similarities after aggregation round 5, 15 and 25 of 38, respectively.

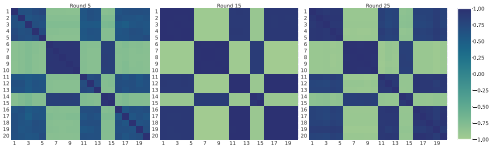

Figure 12: Mutual client similarity at different stages during a single experimental run on the DST environment, with balanced preference distribution. Left to right: client similarities after aggregation round 5, 15 and 25 of 28, respectively.

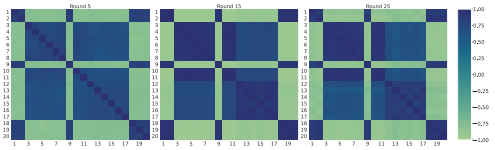

Figure 13: Mutual client similarity at different stages during a single experimental run on the DST environment, with unbalanced preference distribution. Left to right: client similarities after aggregation round 5, 15 and 25 of 28, respectively.