A Appendix

Code Availability: Our code will be made available upon acceptance.

The organization of the appendix is as follows:

1. Subsection A.1 provides the necessary background and preliminary information required for the proofs.
2. Subsection A.2 presents the proof of the optimality bound for the INTERACTIVE policy.
3. Subsection A.3 describes the details of the experimental setup.
4. Subsection A.4 contains additional results related to object detection experiments.
5. Subsection A.5 presents experimental analysis focusing on the diminishing return property.
6. Subsection A.6 reports the results of the experiments with non-submodular and non-monotone objectives.
7. Subsection A.7 includes the function evaluation and runtime complexity analysis of the algorithms.

A.1 Preliminaries

Here, we provide basic definitions of submodular functions, monotone functions, and matroids that we use in our proof.

Definition 2 (Submodular Function)**.** *A set function $f : 2^X \to \mathbb{R}$ is submodular if for all $A \subseteq B \subseteq X$ and $x \in X \setminus B$, we have*

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

Definition 3 (Monotone Function)**.** *A set function $f : 2^X \to \mathbb{R}$ is monotone if for all $A \subseteq B \subseteq X$, we have*

$$f(A) \leq f(B).$$

Based on the definitions of submodular and monotone functions, we can directly write the following:

Corollary 1. *Let f be a monotone submodular function. Then, f is also subadditive, and we have that:*

$$\forall A \subset X,\ B \subset X: \quad f(B) \leq f(A) + \sum_{x \in B \setminus A} f_A(x).$$

Here, we denote the marginal gain of adding an element to set A as $f_A(x) = f(A \cup \{x\}) - f(A)$, which is also monotone and submodular.

Proof. This is a standard result in submodularity. See Corollary 5 in [66]. □

Definition 4 (Matroid)**.** *A matroid is a pair $M = (E, I)$ where E is a finite set (called the ground set) and I is a nonempty set of subsets of E (called the set of independent sets) with the following conditions:*

1. $\forall B \in I: \quad A \subset B \Rightarrow A \in I.$
2. $\forall A, B \in I: \quad |A| < |B| \Rightarrow \exists x \in B \setminus A : A \cup \{x\} \in I.$

Definition 5 (Basis of a Matroid)**.** *A basis of a matroid is an independent set of the matroid which is not contained in any other independent set.*

Corollary 2. *If B_1 and B_2 are two bases of a matroid M, then there exists a bijection $\phi : B_1 \setminus B_2 \to B_2 \setminus B_1$ such that:*

$$\forall x \in B_1 \setminus B_2: \quad B_1 \cup \phi(\{x\}) \setminus \{x\} \in I.$$

Proof. This is a standard result in matroid theory. See Proposition 11 in [66]. □

Definition 6 (Partition Matroid)**.** *A partition matroid is a matroid where the ground set E is partitioned into l disjoint subsets $E_1, E_2, \ldots, E_l$ and the set of independent sets is defined as follows:*

$$I = \{B \subseteq E : |B \cap E_i| \leq k_i, \forall i \in \{1, 2, \ldots, l\}\}.$$

A.2 Proof of Optimality Bound for the Interactive Policy

First, we show that the robots' feasible action spaces $\mathcal{A}^r$ and the union set of all observed data X^r form a partition matroid.

Lemma 2 (Robots' feasible action space and set of all observed images form Partition Matroid)**.** *The robots' feasible action space $\mathcal{A}^r$ and the union of all observed data X^r form a partition matroid $M^r = (X^r, \mathcal{A}^r)$, where $X^r = \bigcup_{i=1}^{N_{\text{robot}}} X_i^r$ and $X_i^r \cap X_j^r = \emptyset \ \ \forall i \neq j$. The action space is defined as $\mathcal{A}^r = \{\bigcup_{i=1}^{N_{\text{robot}}} a_i^r : a_i^r \in \mathcal{A}_i^r \quad \forall i = 1, \ldots, N_{\text{robot}}\}$.*

Proof. The proof follows directly from the definition of action and observed data points. For all $a \in \mathcal{A}^r$, if $s \subset a$, then we know that for all j, $|s \cap X_j^r| \leq |a \cap X_j^r| \leq N_j^{\text{cache}}$ meaning $s \in \mathcal{A}^r$. And we know that for all $a, s \in \mathcal{A}^r$, if $|s| < |a|$, then there exists a subset $X_j^r \subset X^r$ such that $|s \cap X_j^r| < |a \cap X_j^r| \leq N_j^{\text{cache}}$ and $X_j^r \cap a \setminus s \neq \emptyset$. Then for any element $x \in X_j^r \cap a \setminus s$, it holds that $s \cup \{x\} \in \mathcal{A}^r$. □

Now we prove the main theorem of our paper. Our proof is similar to the proof given in [66], while our proof involves sequential optimization methods, such as the one shown in Alg. 4.

Theorem. *The algorithm given in Alg. 4 achieves a solution greater than $1/2$ of the optimal solution.*

Proof. Assume that a^{OPT} is the optimal solution for the problem 1. First, we show that a^{OPT} and a^I are bases of the matroid $(X^r, \mathcal{A}^r)$. Since we assume that the objective function f is monotone (Assmp. 2), it is trivial to see that $|a^{\text{OPT}} \cap X_i^r| = N_i^{\text{cache}}$ for all $i \in \{1, \ldots, N_{\text{robot}}\}$, making a^{OPT} a basis of matroid $(X^r, \mathcal{A}^r)$. a^I is a basis as well, since in Alg. 4 we construct it such that $|a^I \cap X_i^r| = N_i^{\text{cache}}$ for all i. For matroids, there exists a bijection $\phi : a^{\text{OPT}} \to a^I$, which maps the optimal solution to the solution of the INTERACTIVE policy. We can express these solution sets as follows:

$$a^{\text{OPT}} = \{x_{1,1}^{\text{OPT}}, x_{1,2}^{\text{OPT}}, \ldots, x_{N_{\text{robot}}, N_{N_{\text{robot}}}^{\text{cache}}}^{\text{OPT}}\} \quad \text{and} \quad a^I = \{x_{1,1}, x_{1,2}, \ldots, x_{N_{\text{robot}}, N_{N_{\text{robot}}}^{\text{cache}}}\}.$$

Here $x_{i,j} = \phi(x_{i,j}^{\text{OPT}})$ for all i, j. Let $a_{i,j}^I = \{x_{1,1}, \ldots, x_{i,j}\}$ and $a_{i,0}^I = \{x_{1,1}, \ldots, x_{i-1,N_{i-1}^{\text{cache}}}\}$ denote the sets of actions taken up to the i-th robot and the j-th cache and actions taken up to the i-th robot repectively. Then for $f_{\mathcal{D}_c^r}(x) = f(\mathcal{D}_c^r \cup \{x\}) - f(\mathcal{D}_c^r)$, we can write the following:

$$\begin{aligned}
f_{\mathcal{D}_c^r}(a^{\text{OPT}}) - f_{\mathcal{D}_c^r}(a^I) &\leq \sum_{i=1}^{N_{\text{robot}}} \sum_{j=1}^{N_i^{\text{cache}}} f_{\mathcal{D}_c^r \cup a^I}(x_{i,j}^{\text{OPT}}) \\
&\leq \sum_{i=1}^{N_{\text{robot}}} \sum_{j=1}^{N_i^{\text{cache}}} f_{\mathcal{D}_c^r \cup a_{i,j-1}^I}(x_{i,j}^{\text{OPT}}) \\
&\leq \sum_{i=1}^{N_{\text{robot}}} \sum_{j=1}^{N_i^{\text{cache}}} f_{\mathcal{D}_c^r \cup a_{i,j-1}^I}(x_{i,j}) \\
&= \sum_{i=1}^{N_{\text{robot}}} \sum_{j=1}^{N_i^{\text{cache}}} f_{\mathcal{D}_c^r}(a_{i,j}^I) - f_{\mathcal{D}_c^r}(a_{i,j-1}^I) = f_{\mathcal{D}_c^r}(a^I).
\end{aligned}$$

The first inequality is a result of Corollary 1, while the second inequality stems from the submodularity of the function $f_{\mathcal{D}_c^r}$. In the third inequality, we utilize the fact that in each iteration of Alg. 4, we select the element with the maximum marginal gain. Next, in the first equality, we use the fact that $a_{i.j-1}^I \cup \{x_{i,j}\} = a_{i,j}^I$. The last equality follows from the fact that for $f_{\mathcal{D}_c^r}(\emptyset) = 0$, the sum of the marginal gains equals to the value of $f_{\mathcal{D}_c^r}(a^I)$.

Therefore we have:

$$f_{\mathcal{D}_c^r}(a^I) \geq \frac{1}{2} f_{\mathcal{D}_c^r}(a^{\text{OPT}}).$$

This concludes the proof, showing that the INTERACTIVE policy achieves at least half of the value of the optimal solution. □

A.3 Experiments

To demonstrate the effectiveness of our proposed policy, we conducted simulations in scenarios involving multiple robots engaged in data collection from heterogeneous observation distributions. To create these heterogenous environments, we sampled incoming class distributions from the Dirichlet distribution, incorporating a skewness parameter denoted as α. This way, we ensure that environments have nonidentical incoming class distributions (non-i.i.d). Then, within each environment, we simulated robots that observed the same data points.

Initially, an initial dataset denoted as $\mathcal{D}_c^0$ was chosen from the training set to train the initial model $f_{\text{DNN}}(.;\theta_i^0)$. The initial dataset was generated with uniform class distribution, employing the Dirichlet distribution with a skewness parameter value of $\alpha = 5$. In each subsequent round, the vision model was retrained from the pre-trained vision model to ensure a fair evaluation of the performance using the selected training set. To prevent overfitting, when identical data points were selected from multiple devices, the redundant instances were filtered out, and only a single data point was added to the training set.

A.3.1 Embedding Functions:

To generate embeddings for the data points, we utilized multiple vision and language models depending on the datasets. Initially, we made use of the embeddings generated by the CLIP model [4], which is trained to create outputs in the same embedding space for both language and vision model inputs. However, we observed that when the embeddings generated by the CLIP model start to perform poorly on the datasets when there is a mismatch of the targets of the CLIP model with our classification output or the images are out-of-distribution for the CLIP model. For these datasets, we instead employ the embeddings generated by BADGE [33]. BADGE embeddings essentially correspond to the gradients of the final layer of the network with respect to the input.

A.3.2 Classification Experiments

In all classification experiments, we used the Adam optimizer with a learning rate of 0.001 with a batch size of 1000. Additionally, the learning rate scheduler is used with a decay rate of 0.99. We trained the DNNs in each round for 300 epochs. We did not apply any data augmentation to the training set. To ensure robustness, we conducted these experiments for 25 different seeds. Now, we provide additional explanations regarding the details and dataset-specific parameters used in the simulations.

MNIST: In our paper, we used the `MNIST` dataset to show the efficacy of our algorithm in a simple setting. The `MNIST` dataset is a collection of handwritten digits that contains 60,000 training images and 10,000 test images. Each image is a 28×28 grayscale image.

Simulation Parameters: In `MNIST` simulations, we used 5 heterogeneous environments, each containing 4 robots that observe identical samples, resulting in a total system of 20 robots. To create heterogenous incoming class distributions, we set the skewness parameter of the Dirichlet distribution to $\alpha = 1.3$. In each round, robots are observing 1000 data samples and collect $N^{\text{cache}} = 3$ data samples from their observations. We started with an initial dataset of size 16 and collected the data for 10 rounds. The final training dataset consists of 616 data points.

DNN and Embedding Function: We used a simple DNN with 6 layers, with 4 convolutional layers and 2 fully connected layers. Between each convolutional layer, we used the ReLU activation function and applied dropout with a probability of 0.3. To create embeddings, we utilized BADGE [33].

CIFAR-10: The `CIFAR-10` dataset consists of 60,000 $32 \times 32 \times 3$ RGB images with ten different classes. The dataset is split into training and testing datasets of size 50,000 and 10,000, respectively. The classes in the dataset are truck, ship, horse, frog, dog, deer, cat, bird, automobile, and airplane.

Simulation Parameters: In `CIFAR-10` simulations, we used 6 heterogeneous environments, each containing 4 robots that observe identical samples, resulting in a total system of 24 robots. To create heterogenous incoming class distributions, we set the skewness parameter of the Dirichlet

distribution to $\alpha = 1.6$. In each round, robots are observing 1000 data samples and collect $N^{\text{cache}} = 1$ data samples from observations. We started with an initial dataset of size 10 and collected the data for 10 rounds. The final training dataset consists of 250 data points.

DNN and Embedding Function: We leveraged a pre-trained ResNet-50 model [64] as the backbone for our vision model. To adapt it for our task, we replaced the final layer of the ResNet-50 with two fully connected layers, incorporating ReLU activation, and applied dropout with a probability of 0.3 to mitigate overfitting. Only these replaced layers were retrained, following the transfer learning approach. This strategy significantly reduces training time while mitigating overfitting risks. To create embeddings, we utilized embeddings created by the CLIP model [4].

Adverse-Weather Dataset The `Adverse-Weather` dataset comprises numerous RGB image sequences, each with dimensions of $720 \times 1280 \times 3$. These sequences were collected from moving vehicles around the University of Michigan campus, capturing diverse weather conditions. While most sequences exhibit dynamic scenes, some of them include static recordings. The dataset includes two metadata classes: weather and time of day. The weather class consists of labels such as rain, fog, snow, sleet, overcast, sunny, and cloudy. The time of day class includes labels for Sunset, Afternoon, and Dusk. By combining these weather and time of day labels, we established a total of 11 classes to train our model on. To avoid redundancy, we subsampled the images from the video sequences, selecting one image every 10 frames. Consequently, we constructed a dataset comprising 36,230 images, which we divided into a training set of 31,701 images and a test set of 4,529 images.

Simulation Parameters: In `Adverse-Weather` simulations, we used 6 heterogeneous environments, each containing 4 robots that observe identical samples, resulting in a total system of 24 robots. To create heterogenous incoming class distributions, we set the skewness parameter of the Dirichlet distribution to $\alpha = 1.2$. In each round, robots are observing 1000 data samples and collect $N^{\text{cache}} = 1$ data samples from observations. We started with an initial dataset of size 10 and collected the data for 10 rounds. The final training dataset consists of 250 data points.

DNN and Embedding Function: We adopted a pre-trained Vit-H14 model [63] as the backbone for our vision model. To adapt it to our specific task, we replaced the final layer of the Vit-H14 with two fully connected layers, employing ReLU activation, and incorporated dropout with a probability of 0.3 to mitigate overfitting. Only these replaced layers were subjected to retraining. This methodology significantly reduces training time while effectively preventing overfitting. To create embeddings, we have utilized BADGE [33].

DeepDrive Dataset: The `DeepDrive` data following the transfer learning approach, only encompasses 100,000 images taken from driving videos in diverse cities and weather conditions. It comprises 70,000 training images, 10,000 validation images, and 20,000 testing images. However, the testing images are not publicly accessible, so our analysis focused solely on the original training and validation datasets. The classification model was designed to predict weather labels, including rainy, snowy, clear, overcast, partly cloudy, and foggy. However, due to a limited number of foggy images (only 181), the foggy class was excluded from the simulations. Consequently, the classification model was trained on a subset of five classes.

Simulation Parameters: In `DeepDrive` simulations, we used 5 heterogeneous environments, each containing 6 robots that observe identical samples, resulting in a total system of 30 robots. To create heterogenous incoming class distributions, we set the skewness parameter of the Dirichlet distribution to $\alpha = 3.3$. In each round, robots observe 1000 data samples and collect $N^{\text{cache}} = 20$ data samples from observations. We started with an initial dataset of size 50 and collected the data for 20 rounds. The final training dataset consists of 12,050 data points.

DNN and Embedding Function: We used a pre-trained Vit-H14[63] model and replaced the final layer of the vision model with 2 fully connected layers with the ReLU activation layer and applied dropout with a probability of 0.3. We only retrained the replaced layers of the network. This approach, known as transfer learning, reduces training time and helps to prevent overfitting. To create embeddings, we utilized BADGE [33].

A.3.3 Object Detection Experiments

We ran object detection experiments on the DeepDrive dataset. As in the classification case, we used the training and validation images to train our model and report the final mean average precision

(mAP) scores. We used an SGD optimizer with a learning rate of 0.01 in this experiment. We trained the DNNs at the start and end of the data collection rounds. We applied all data augmentations detailed in [65].

Simulation Parameters: In object detection experiments, we used 5 heterogeneous environments, each containing 5 robots that observe identical samples, resulting in a total system of 25 robots. We set skewness parameter $\alpha = 1$. In each round, robots observe 1000 data samples and collect $N^{\text{cache}} = 50$ data samples from observations. We started with an initial dataset of size 5000 and collected the data for 20 rounds. The final training dataset consists of 30,000 data points.

DNN and Embedding Function: We used a pretrained YOLOv8-small model [65] and retrained all model weights in each training. Since we are dealing with the objects, we utilized the CLIP model [4] to create embeddings.

A.4 Object Detection Experiments Metrics

In the object detection experiments, we used several metrics to evaluate the performance of the policies. These metrics include mAP50, representing the mean of average precision at the intersection over union with a threshold of 50%; mAP50-95, denoting the mean of the average precision at the intersection over union for thresholds ranging from 50% to 95%; precision, and recall. We conducted the experiments with 25 different seeds and averaged the metrics across the seeds. We present results in Table 2. Our INTERACTIVE policy shows a similar performance to the CENTRALIZED policy while outperforming the DISTRIBUTED policy. In mAP50 results, the INTERACTIVE policy shows an improvement of 9.2%, whereas, in mAP50-95 results, the INTERACTIVE policy demonstrates a development of 6.1%. Furthermore, our INTERACTIVE policy surpasses the DISTRIBUTED policy in precision and recall results by 11.6% and 6.5%, respectively.

Method	mAP@50	mAP@50-95	Precision	Recall
INITIAL	33.2 ± 0.012	17.7 ± 0.007	51.5 ± 0.048	32.7 ± 0.01
DISTRIBUTED	37.2 ± 0.009	20.2 ± 0.006	56.3 ± 0.046	35.5 ± 0.008
CENTRALIZED	46.5 ± 0.003	26.3 ± 0.007	67.6 ± 0.007	42.4 ± 0.004
INTERACTIVE	46.4 ± 0.002	26.3 ± 0.002	67.9 ± 0.007	42.0 ± 0.003

Table 2: Additional metrics for object detection experiments

A.5 Diminishing Returns between accuracy and the percentage of training data

To support our claim about dataset quality being submodular in Assumption 3, and monotone 2. We conducted 5 experiments training classification models on four datasets of varying sizes. In Fig. 5, we show the accuracy of the classification models accuracy across the different percentages of the datasets. We can see in the general plot (Fig. 5-right) and zoomed-in version to smaller percentages ((Fig. 5-left) accuracy shows a diminishing returns property in terms of accuracy as the training dataset size increases. This indicates that the incremental improvement gained from adding new samples decreases with a larger dataset. Additionally, the accuracy of the models consistently increases as the dataset size grows, providing evidence for the monotonicity of dataset quality. These observations align with our assumption about dataset quality being submodular (Assmp. 3) and monotone (Assmp. 2).

A.6 Accuracy When Objective Function is Non-submodular, Non-monotone

In this section, we investigate the performance of our INTERACTIVE policy when the objective function f is not monotone or submodular through experiments. Instead of the submodular facility location function, we employed a variant called the maximin facility location function. This function aims to maximize the minimum distance between any two points in a set. Formally, we can define the maximin facility location function as follows:

$$f_{\text{maximin}}(S) = \max_{s_i \in S} \min_{s_j \in S \setminus \{s_i\}} \|s_i - s_j\|$$

where s_i is the i^{th} point in the set S and $\|s_i - s_j\|$ is a distance function. We used the same simulation parameters and models as we did with the facility location objective function. The accuracy results

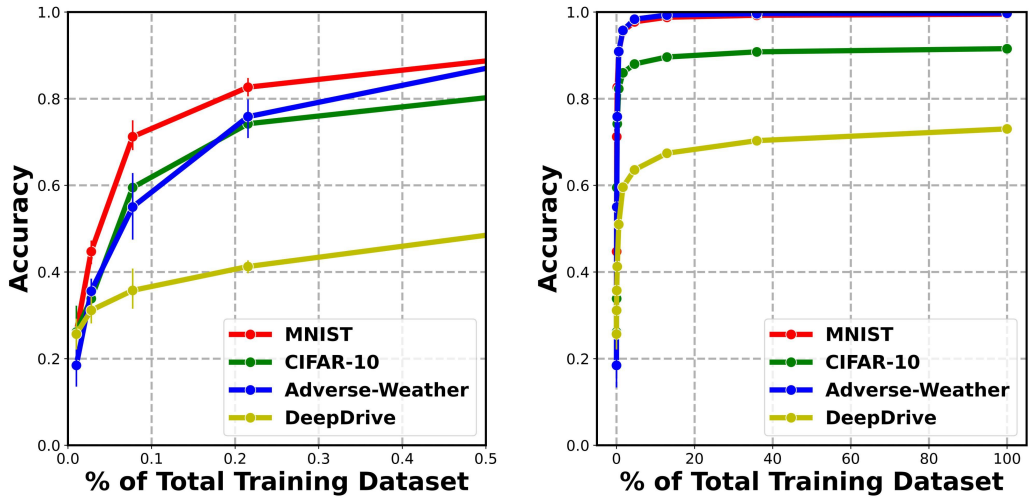

Figure 5: **Accuracy shows diminishing returns to training dataset size.** Each line represents the accuracy of the classification models on a different dataset. The x-axis represents the percentage of datasets used for training the model, while the y-axis represents accuracy. The left figure provides a closer view of the right figure, specifically focusing on training percentages ranging from 0 to 0.5. Across all datasets, we observe a consistent improvement in accuracy as the percentage of training data increases. However, as the dataset size grows, the slope of the curve gradually decreases, indicating diminishing returns where the incremental contributions of new data samples become smaller. This observation supports our assumption of dataset quality function being submodular and monotone.

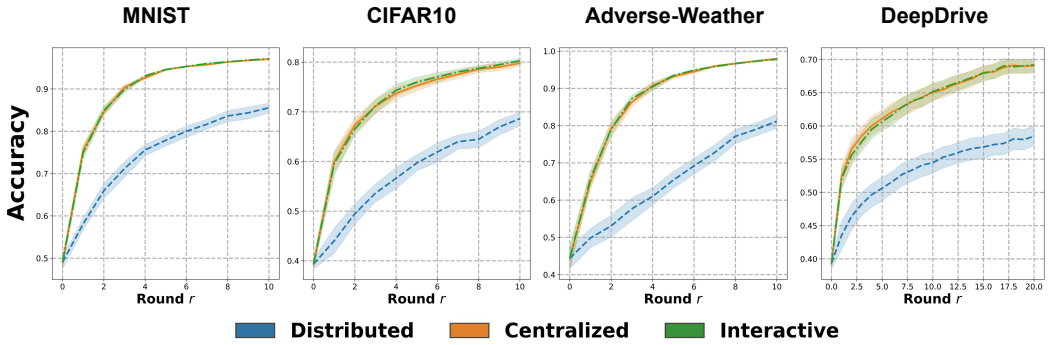

Figure 6: **Comparison of performance under non-monotone and non-submodular objective** We conducted the same experiments as in Fig. 3 but with a non-submodular and non-monotone objective function, described in A.6. Similar to Fig. 3, we observe that both the INTERACTIVE and CENTRALIZED policies outperform the DISTRIBUTED policy by a significant margin. This demonstrates the robustness and effectiveness of the INTERACTIVE and CENTRALIZED policies under non-submodular and non-monotone objectives.

of the simulations are presented in Fig. 6. Similar to the previous case, our INTERACTIVE policy exhibits equivalent performance to the CENTRALIZED policy and outperforms the DISTRIBUTED policy in all simulations.

A.7 Complexity Analysis

For all algorithms, suppose that there are N_{robot} robots, each robot i observes samples X_i^r in each data collection round, and in each round at most N_i^{cache} samples have to be chosen from samples X_i^r for each robot i.

It is a known result that the total number of function evaluations of the greedy algorithm that is used for submodular maximization is $O(|V|k)$, where V is the ground set for observations and k is the number of data points that need to be selected [51].

The* CENTRALIZED *Algorithm: In the CENTRALIZED algorithm, a central server carries out all the computation required in a data collection round to choose the samples that are going to be added to the cloud dataset. That is, the central server has access to all the observations made by the robots and has to select N_i^{cache} samples from the observation set X_i^r of each robot i. Therefore, our ground set is $\cup_{i=1}^{N_{\text{robot}}} X_i^r$, and the number of points that need to be selected is $\sum_{i=1}^{N_{\text{robot}}} N_i^{\text{cache}}$. Because the observed sets X_i^r of the robots are disjoint, the total number of elements in our ground set is $|\cup_{i=1}^{N_{\text{robot}}} X_i^r| = \sum_{i=1}^{N_{\text{robot}}} |X_i^r|$. This makes the computational complexity of the CENTRALIZED algorithm in terms of total number of function evaluations $O(\sum_{i=1}^{N_{\text{robot}}} |X_i^r| \times \sum_{i=1}^{N_{\text{robot}}} N_i^{\text{cache}})$.

The* DISTRIBUTED *Algorithm: In the DISTRIBUTED algorithm, all robots carry out the data selection process themselves. Therefore, for each robot i, the ground set is X_i^r and the number of points that have to be selected is N_i^{cache}. Summing the number of function evaluations over all robots, the total complexity of the DISTRIBUTED algorithm is $O(\sum_{i=1}^{N_{\text{robot}}} |X_i^r| \times N_i^{\text{cache}})$.

The* INTERACTIVE *Algorithm: In our INTERACTIVE algorithm, just like the DISTRIBUTED algorithm, the robots select data points in the confines of their observation sets. Thus, the ground sets and the number of points to be selected from each ground set is again X_i^r and N_i^{cache}, respectively. Summing again over all robots, the total complexity of our INTERACTIVE algorithm is $O(\sum_{i=1}^{N_{\text{robot}}} |X_i^r| \times N_i^{\text{cache}})$, which is the same as the DISTRIBUTED algorithm.

Number of Message Exchanges: Since our problem is distributed in nature, we should also analyze the total number of messages passed between robots. The CENTRALIZED algorithm requires $\sum_{i=1}^{N_{\text{robot}}} N_i^{\text{cache}}$ iterations over all robots, resulting in total of $O(N_{\text{robot}} \sum_{i=1}^{N_{\text{robot}}} N_i^{\text{cache}})$ message exchanges. On the other hand, our INTERACTIVE policy requires only one iteration over all robots, leading to a significantly lower number of message exchanges, specifically $O(N_{\text{robot}})$. Lastly, since the DISTRIBUTED policy is executed without any interaction between robots, there are no message exchanges involved.