information

Article

Locally Centralized Execution for Less Redundant Computation in Multi-Agent Cooperation

Yidong Bai *,† and Toshiharu Sugawara *,†

Department of Computer Science and Engineering, Waseda University, Tokyo 169-8555, Japan
* Correspondence: ydbai@moegi.waseda.jp (Y.B.); sugawara@waseda.jp (T.S.)
† These authors contributed equally to this work.

Abstract: Decentralized execution is a widely used framework in multi-agent reinforcement learning. However, it has a well-known but neglected shortcoming, redundant computation, that is, the same/similar computation is performed redundantly in different agents owing to their overlapping observations. This study proposes a novel method, the *locally centralized team transformer* (LCTT), to address this problem. This method first proposes a *locally centralized execution* framework that autonomously determines some agents as *leaders* that generate *instructions* and other agents as *workers* to act according to the received instructions without running their policy networks. For the LCTT, we subsequently propose the *team-transformer* (T-Trans) structure, which enables leaders to generate targeted instructions for each worker, and the *leadership shift*, which enables agents to determine those that should instruct or be instructed by others. The experimental results demonstrated that the proposed method significantly reduces redundant computations without decreasing rewards and achieves faster learning convergence.

Keywords: cooperation; multi-agent deep reinforcement learning; redundant computation

check for updates

Citation: Bai, Y.; Sugawara, T. Locally Centralized Execution for Less Redundant Computation in Multi-Agent Cooperation. *Information* **2024**, *15*, 279. https://doi.org/10.3390/info15050279

Academic Editors: Katsuhide Fujita

Received: 19 April 2024
Revised: 7 May 2024
Accepted: 9 May 2024
Published: 14 May 2024

1. Introduction

Multi-agent reinforcement learning (MARL) has achieved impressive success in several domains, including autonomous driving [1–5], sensor networks [6], and game playing [7,8]. *Centralized training and centralized execution* (CTCE) [9,10] and *centralized training and decentralized execution* (CTDE) [11,12] are two primary frameworks of MARL. In CTCE, the information of all agents is processed in a centralized network, and the network determines the joint actions to control all agents. However, owing to the positive and negative interference between the actions of the agents and the exponential size of the joint action space, CTCE can only be used in limited applications. In contrast, in CTDE, agents learn policies in a centralized manner to avoid non-stationarity while making decisions based only on their local observations. In this sense, the CTDE framework seems to be more appropriate for applications of multi-agent systems (MASs) [13–18].

However, certain shortcomings of the CTDE framework have been neglected for a long time. Specifically, *redundant computation*, which implies that the same or similar computation is performed redundantly by different agents, is a well-known traditional problem in MASs [19] and comes from overlapping observations, leading to similar associated processes in several agents. This problem widely exists in applications such as distributed robot systems [20], distributed sensor networks [19,21] and distributed air traffic control [22], where there is a natural spatial distribution of information and where the agents have many overlapping observations. As shown in Figure 1a, the observations of the decentralized executing agents contain numerous overlapping regions. Information on agents and other entities, such as obstacles and tasks, in the overlapping regions is processed by multiple agents to determine their actions, which superficially appears as a waste of computational resources, particularly for cooperative tasks with communications among agents. By contrast, such redundant computations do not appear in CTCE because their observations are aggregated before they are fed into the centralized network (Figure 1b).

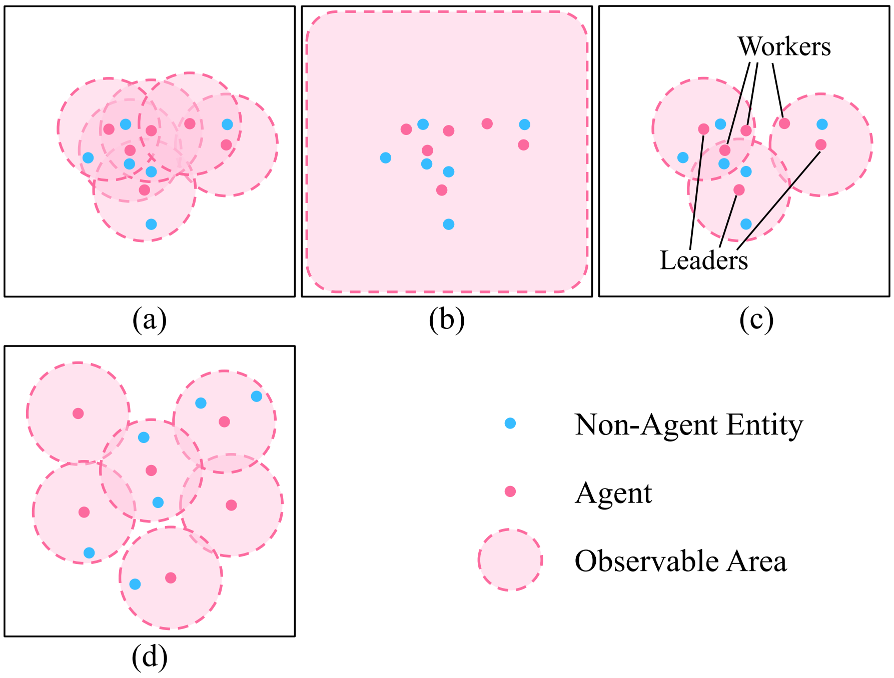


Figure 1. MARL environments. (**a**) Decentralized execution: agents observe and process their local information. (**b**) Centralized execution: agents process aggregated global information. (**c**) Locally centralized execution: agents perform less repeated observations and processes than those in (**a**). (**d**) Dispersed decentralized execution: agents with sparse observations.

A proper reduction in overlapping observations can significantly eliminate the waste of computational resources without lowering the performance of an MAS. Therefore, we propose an execution framework between centralized and decentralized executions, referred to as the *locally centralized execution* (LCE) framework, after centralized training (CT), as shown in Figure 1c. By making some agents *leaders* that make decisions for themselves and for other agents, called *workers*, within their observable regions, workers are freed from observing their surroundings and determining their actions. The LCE framework does not rely on aggregated observations by all agents but on the local observations of leaders; thus, it alleviates redundant computations and reduces the cost of determining joint actions from a combinatorial but much smaller joint action space.

Another noteworthy feature of the LCE framework is that it does not compete with existing approaches that reduce computational expenses (e.g., pruning inputs [23] and designing more concise network structures) because it is an extension of another independent dimension. Thus, it can be easily integrated with them. Based on LCE, we further demonstrate a *centralized training and locally centralized execution* (CTLCE) framework and propose a *locally centralized team transformer* (LCTT) to generate targeted messages to other agents. To quantitatively describe the redundant computation of an MAS algorithm, we also define the *redundant observation ratio* R_{dd} (≥ 1), which can be used to accurately compare the computational expenses of methods using deep networks with a similar parameter number. To the best of our knowledge, this study is the *first* approach to address redundant computation issues in the context of *multi-agent deep reinforcement learning*.

The proposed LCTT method comprises LCE, *team transformer* (T-Trans), and *leadership shift* (LS). First, LCE is a collaboration regime in which agents are dynamically divided into *leader–worker* groups according to the direction of the associated instruction messages. Second, T-Trans applies an attention-like mechanism [24] to enable leaders to provide targeted instructions to each worker. Finally, the *leadership Q-value* whose value is used for each agent to decide its suitability as a leader, is introduced because we must solve the following problem for agent grouping: *how to determine which agent is suitable to be the leader*. Thereafter, we propose LS, which allows a leader to designate the observable workers that should be leaders in the next time step based on leadership Q-values. Thus, agents continuously hand over their leadership throughout the episodes, ensuring that the most suitable agents lead the group.

We implemented the proposed LCTT method based on QMIX [25], trained it in the CTLCE framework, and experimentally evaluated it in *level-based foraging* (LBF) [7,26] and *cooperative navigation* (CN) environments. The experimental results in LBF demonstrated that the proposed method obtained comparable rewards with faster convergence while reducing redundant computation by comparing the baselines, which are QMIX without LCTT and *multi-agent incentive communication* (MAIC) [7]. Through ablation studies, we further confirmed that LS can reduce redundant computation and mitigate the reward decrease with the number of leaders and that the policy learned through our method can provide effective instructions for both leaders and workers. The experimental results in CN also clearly show the behavioral patterns of leader agents, in that they tend to stay in specific areas to instruct workers to ensure that the agent team can achieve more rewards and smaller redundant observation ratios.

The contributions of this study are four-fold: First, we analyze the redundant computation problem, which has been neglected for a long time, and propose a metric, theredundant observation ratio, to describe the degree of redundant computation. Second, we introduce the CTLCE framework to reduce redundant computation and enhance multi-agent cooperation. Thereafter, we provide an LCE-based method, LCTT, and demonstrate its effectiveness in reducing redundant computation. Finally, we conduct experiments in two typical multi-agent cooperation tasks, LBF and CN, to show that the proposed framework, CTLCE, can eliminate redundant actions. Because the proposed method can be easily integrated with existing methods to improve efficiency, we believe that it can be applicable to broader areas.

2. Related Work

2.1. Centralized Training and Decentralized Execution

A simple approach to tackling multi-agent learning problems is treating the learning problem as a single-agent case and learning from a centralized network to determine joint actions to control the behaviors of all agents. This scheme is typically known as CTCE [10,27]. A centralized network can comprehensively consider the information of all the agents and achieve remarkable performance [8,28]. However, the joint action space increases exponentially with the number of agents, thereby limiting the application of CTCE methods to environments with a large number of agents. Another scheme is DTDE [27]. DTDE methods directly apply reinforcement learning to each agent and independently train their policies [29]. Thus, DTDE avoids the combinatorial joint-action space.

However, from the perspective of individual agents, the environment is non-stationary, owing to the dynamics of other agents; that is, they are also learning and may change their activities. Thus, DTDE methods are typically confronted with convergence problems, making it difficult to achieve state-of-the-art performance. CTDE methods [12,25] learn a shared policy for all agents in a centralized manner and enable agents to act based on local observations in a decentralized manner. Therefore, the CTDE scheme avoids non-stationarity and alleviates the combinatorial joint action space problem. However, decentralized execution (DE) methods (i.e., DTDE and CTDE) have the disadvantage of redundant computations, as mentioned. We proposed LCE to reduce redundant computations and demonstrated a CTLCE framework, which can be considered a scheme between CTCE and CTDE.

2.2. Multi-Agent Communication

Effective communication is crucial to facilitate cooperation and has been applied in CTCT, DTDE, and CTDE methods. CTCE methods involve communication layers for sharing information among agents. Sukhbaatar et al. [30] collected and averaged the hidden states of agents and broadcast them to all the agents. Peng et al. [28] used recurrent neural networks to process agent observations sequentially. Han et al. [10] used fully convolutional networks to provide a large receptive field for agents to communicate. DTDE and CTDE agents exchange messages with others during the execution phase for communication. Jaques et al. [31] used influence rewards to encourage each agent to generate messages that affected the behaviors of others. Ding et al. [32] enabled agents to learn adaptive peer-to-peer communication. Yuan et al. [7] allowed agents to generate incentive messages to directly bias the value functions of others. In these DE methods, the behavior of an agent is determined by its observations, policies, and messages received from others. When the DE agents generate messages and actions, redundant computations are likely to occur because of their overlapping observations and the accompanying similar reasoning processes performed simultaneously.

In contrast, we enable dynamically determined leader agents to generate instructions for workers in their local area. The instructions can be considered a special message because the worker receiving them will act without considering their surroundings. In this manner, redundant computations in the system will be drastically reduced. We have already reported this method from the same perspective [33], but its formulation was not complete, and the experimentation was limited. Therefore, we described and discussed the proposed method in more detail. We also conducted extensive experiments, including easier and harder problems, to investigate how leaders cover other agents and the effectiveness of the proposed method in reducing redundant computations.

3. Background and Problem

3.1. Cooperative Decentralized Partially Observable Markov Decision Process with Communication

We considered a fully cooperative *decentralized partially observable Markov decision process* (Dec-POMDP) augmented with communication, similar to several previous methods [6,7,32,34]. Let $\mathcal{N} = \{1, \dots, n\}$ be the set of n agents, $\mathcal{S}$ be the set of global states, and let $\mathcal{A}$ be the finite set of actions executed by $\forall i \in \mathcal{N}$. We also introduce a discrete time step $t \geq 0$, and for simplicity, the subscript t is omitted in this paper. At each time step t, agent $i \in \mathcal{N}$ obtains a local observation $o_i = O(s, i)$, where $s \in \mathcal{S}$ and $O(\cdot)$ denote partial observation functions. Based on o_i, i may generate a message m_i and send it to other observed agents. Simultaneously, i may receive messages from other agents, $\boldsymbol{m_{*,i}} = (m_{1,i}, \dots, m_{n,i})$, where $m_{j,i}$ is the message sent from agent j to i. If no message arrives from j, $m_{j,i} = \textit{null}$. We also assume $m_{i,i} = \textit{null}$. Then, according to o_i and $\boldsymbol{m_{*,i}}$, i selects action $a_i \in \mathcal{A}$ based on policy $\pi_i(a_i | o_i, \boldsymbol{m_{*,i}})$. After i executes a_i, it receives reward $r_i(s, a_i)$ from the environment and the state s changes to the next state. The objective of i is to maximize the expected discounted cumulative reward $R_i = \mathbb{E}[\sum_t \gamma^t r_i(s, a_i)]$ by optimizing π_i.

Q-learning [35,36] is the primary approach in reinforcement learning that determines the optimal policy of each agent. Q-learning methods construct action–value functions $Q(o_i, a_i)$ to enable agents to appropriately determine their actions. DQN [37,38] represents the action–value functions with a deep neural network $Q(o_i, a_i | \theta)$ parameterized by θ. The parameters are learned by minimizing the loss as follows:

$$L^Q(\theta) = \mathbb{E}_{o_i, a_i, r_i, o_i'}[(r_i + \gamma \max_{a_i'} \bar{Q}(o_i', a_i' | \bar{\theta}) - Q(o_i, a_i | \theta))^2], \tag{1}$$

where o_i' and a_i' are the observation and action at the next time step. To generate stable target values for the training of the Q network, a target network $\bar{Q}$ is used, whose parameters $\bar{\theta}$ are periodically updated with θ. Policy π_i is then learned using the DQN.

3.2. Level-Based Foraging (LBF) Environment

We will evaluate the proposed method in the CN [12] and LBF [26] environments. Because LBF is slightly complicated, we will explain it briefly (the CN environment used in the experiments in this study will be explained in Section 5.4). In an LBF environment, each agent must learn to cooperate with other agents to collect and load food. We selected LBF because agents acting alone are often unlikely to be rewarded, and overlapping observations seem to be a prerequisite for agent coordination/cooperation to earn rewards. Thus, the learning convergence in instances of LBF clearly indicates that the proposed method can perform coordination and cooperation using different approaches from those of existing methods, thereby reducing overlapping observations and eliminating redundant associated computations.

At the beginning of the game, β (> 0) foods $\{f_1, \ldots, f_\beta\}$ and n agents spawn at random locations. Any food f_μ has an association level $lv_f(f_\mu) > 0$, which indicates the difficulty in obtaining food f_μ. Agent i also has its associated level, $lv_a(i)$, indicating the ability of i to obtain food. Figure 2 presents a snapshot of the LBF environment. At each time step t, the agents obtain local observations, communicate with each other, and execute actions selected from $\mathcal{A}$. The action space $\mathcal{A}$ comprises six actions: movement in four directions, a "none" action that means doing nothing, and a "loading" action with which the agent attempts to load the food f_μ on the adjacent node. Then, only when the sum of the levels of the agents attempting to collect f_μ simultaneously is greater than or equal to the level of f_μ, that is, $\sum_{i \in \mathcal{N}_{f_\mu}} lv_a(i) \geq lv_f(f_\mu)$, can they collect, divide, and load it, where $\mathcal{N}_{f_\mu}$ denotes the set of agents that performed "loading" f_μ simultaneously at its neighbor. Hence, if $lv_a(i) < lv_f(f_\mu)$, agent i cannot collect f_μ alone, and its attempt fails. For instance, agent j in Figure 2 will fail if it attempts to collect food at a level of five; thus, it must wait for other agents (e.g., i and k) to come up to the food.

When agents successfully load f_μ, they earn their rewards by dividing $lv_f(f_\mu)$ by the ratio of their levels, that is, for $\forall i \in \mathcal{N}_{f_\mu}$:

$$r_i(s, a_i) = lv_f(f_\mu) \cdot \frac{lv_a(i)}{\sum_{\forall j \in \mathcal{N}_{f_\mu}} lv_a(j)}. \quad (2)$$

This also implies that the reward for food is identical to the associated food level. Agent i attempts to earn more *individual rewards* R_i from the environment, as well as more *team rewards* $R = \sum_{i \in \mathcal{N}} R_i$. Therefore, although the agents are cooperative in terms of maximizing R, this may not be the case for R_i if an extra agent redundantly joins the team to collect food.

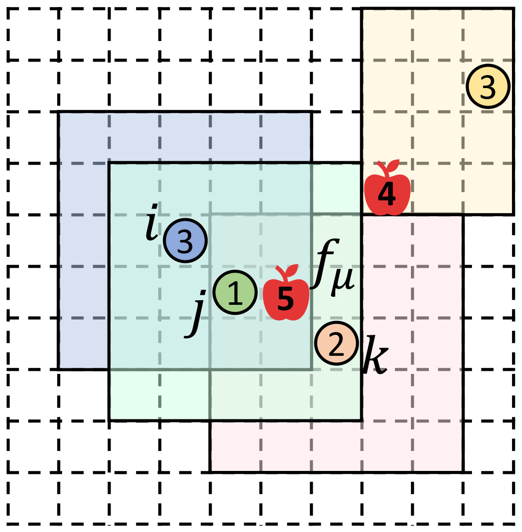


Figure 2. Example snapshot of LBF. The numbers on the agents and foods indicate their levels. The observable area of each agent is indicated by the color that is a lighter shade of that of the agent. Apples indicate foods whose levels are shown as the numbers inside apples.

3.3. Problem Description

In Figure 2, agents i, j, and k observe food f_μ and communicate and cooperate to load it. However, the information in the observations of i, which includes information on i, j, and f_μ, is also included in those of j. Similarly, the information in the observations of k is included in those of the j observation. Overlapping information is processed in all networks of i, j, and k, which is a waste of computational resources. In particular, when all the agents share the same network parameters in the CTDE framework and communication is allowed, this problem is likely to be pronounced.

Therefore, rather than having i, j, and k make decisions separately, it is better to let j, using j's observation and network, instruct i and k to act coherently by allowing communication, instead of running the networks in agents i and k. We believe that this can reduce the computational cost and improve the learning efficiency of the entire MAS. This study proposes a framework in which agents autonomously decide which agents should instruct and be instructed by others and how.

4. Proposed Method

4.1. Redundant Observation Ratio

The amount of computational resources occupied by redundant computation is related to the degree of overlap in the agents' observations. The more overlapping observations in the environment, the more resources are occupied due to redundant calculations. To grasp the degree of redundant computation, we introduce a metric *redundant observation ratio* to describe *the number of times the information of each entity is processed by agents on average*. We define the redundant observation ratio R_{dd} for DE in MASs as follows:

$$R_{dd} = \frac{\sum_{i\in\mathcal{N}} U_i}{\sum_{e\in E}\delta(e)}, \tag{3}$$

where $E = \{e_1, \dots, e_{|E|}\}$ denotes the set of entities (including agents, obstacles, and targets) in the environment, and the positive integer U_i is the number of entities observed by agent i (we assume that any agent can recognize (and observe) itself). We set $\delta(e) = 1$ if entity $e \in E$ is observed by at least one agent; otherwise, $\delta(e) = 0$. Thus, R_{dd} reflects the degree of overlapping observations and associated redundant computations. Typically, $R_{dd} \geq 1$. For instance, R_{dd} values in Figure 1a,c are 2.75 and 1.17, respectively. Moreover, when agents are dispersed, they have fewer overlapping observations and a smaller R_{dd}. Figure 1d shows a special case in which agents have no overlapping observations and their R_{dd} value is 1.0 For a centralized-executing MAS (Figure 1b), we define $R_{dd} \equiv 1$ because all the observations of agents are aggregated before they are fed into the centralized network.

4.2. Locally Centralized Execution (LCE)

To reduce redundant observations and computations, we enable l ($\leq n$) agents, referred to as leaders, to generate instructions for actions for both themselves and other $n - l$ agents, referred to as workers, such that the workers can act on instructions from leaders and be freed from observing their surroundings (so their observations in Equation (3) were excluded), and their actions were determined. The sets of leaders and workers are denoted by $\mathcal{L}$ and $\mathcal{W}$, respectively (where $\mathcal{L}, \mathcal{W} \subseteq \mathcal{N}$). The instruction $m_{i,k}$ from leader i to worker k is the Q value of k's possible actions, $m_{i,k} = Q_{i,k}(o_i, a_k|\theta)$ for observation o_i and $a_k \in \mathcal{A}$.

Figure 3 shows an example of the CTLCE framework, in which each agent can observe its adjacent two agents. Agents $1, 3$, and 5 are workers, and agents 2 and 4 are leaders. Agents 2 and 4 can instruct the actions of its adjacent workers. A more detailed pseudocode for LCE in one episode is shown in Algorithm 1.

When leaders observe each other, they also generate instructions for themselves and other observed leaders. When a worker/leader agent j receives more than one instruction from multiple leaders, it calculates the average value of these instructions as the Q values. We normally set $0 < l < n$ for an LCE framework; LCE with $l = 0$ corresponds to a DE without communication, and that with $l = n$ corresponds to a DE with dense communication among agents. In practice, selecting an appropriate value for l through experiments and experience is necessary.

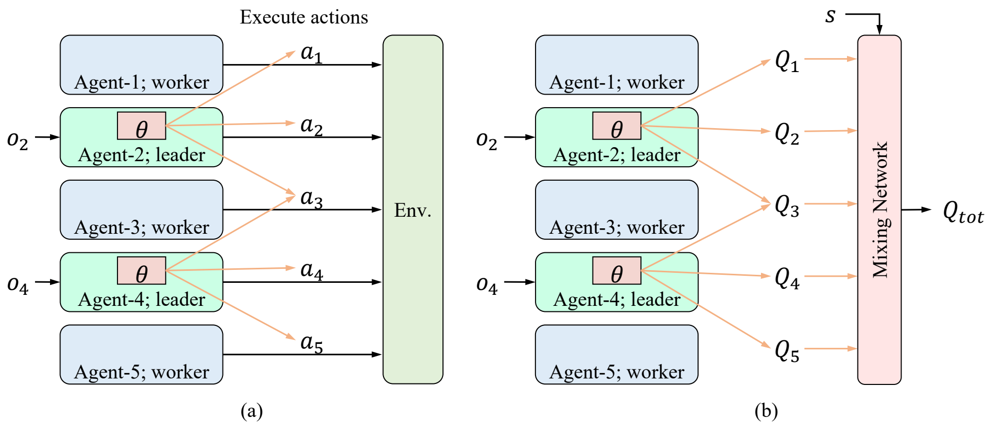


Figure 3. Example structure of the CTLCE framework, where θ is the policy network, s is the global state, and Q_{tot} is the global Q-values. Agents 1, 3 and 5 are workers, and agents 2 and 4 are leaders. (**a**) The LCE phase. The actions of agents 1, 3, and 5 are instructed by their adjacent leaders. (**b**) The CT phase. The policy can be trained similarly to the standard QMIX.

Algorithm 1 Locally Centralized Execution

```
1: Initially, we randomly select $l$ agents as leaders.
2: Other $(n - l)$ agents are workers.
3:
4: for $t = 0, \dots, T$ do
5:     parallel for all leaders $i \in \mathcal{L}$ do
6:         Obtain local observation $o_i$.
7:         parallel for observed agents $j$ (including $i$) do
8:             Generate an instruction $m_{i,j}$
9:             Send $m_{i,j}$ to agent $j$
10:        end parallel for
11:        Leadership Shift: Appoint an observed agent to be the leader at the $t + 1$.
12:    end parallel for
13:    parallel for all worker $k \in \mathcal{W}$ do
14:        if $k$ does not receive any instruction then
15:            Acquire local observation $o_k$.
16:            Determine an instruction for itself, $m_{k,k}$.
17:        end if
18:    end parallel for
19:    Calculate average values of instructions as Q-values. for all agents in $\mathcal{N}$
20:    Agents act based on the Q-values.
21: end for
```

Note that all agents have the same network, even though the workers almost do not run their networks. When a worker is not observed by any leader and receives no instructions, it observes its surroundings and runs its network to generate Q-values for itself (line 13∼18 in Algorithm 1). These workers that can determine their own actions are called *positive workers*; the experiments show that positive workers are essential for the LCE method to achieve high rewards and low redundant observation ratios (as described in Section 5.2). Moreover, positive workers provide the proposed method with the ability to resist communication obstruction. When applying our method in practice, the instructions may be interfered with or blocked and cannot be received by the workers. Then, the workers can determine their actions by themselves. Positive workers can make the performance of the proposed method not worse than those of the DE methods. The negative impact of communication obstruction on the proposed algorithm is similar to that of an excessively small l; that is, workers experience difficulty receiving instructions from leaders and have to frequently make decisions on their own. Therefore, in practical applications where communication quality cannot be guaranteed, we can manually set the number of leaders to be slightly larger to alleviate and offset the adverse effects of communication obstruction. Details of generating instructions (lines 8 and 16 in Algorithm 1) and LS (line 11 in Algorithm 1) are described in Sections 4.3 and 4.4 using Figure 4 and Algorithm 2.

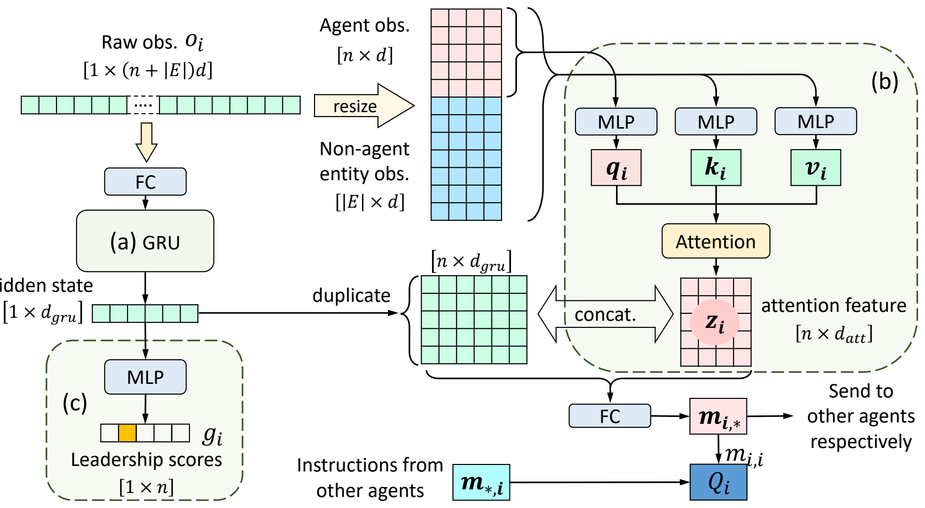


Figure 4. Structure of T-Trans, where d is the dimension of observation of each entity; d_{gru} and d_{att} are dimensions of features of GRU and attention layers. $\boldsymbol{q_i}$ is the query matrix, and its size is $n \times d_{att}$. $\boldsymbol{k_i}$ and $\boldsymbol{v_i}$ are key and value matrices, respectively, and their sizes are $(n+|E|) \times d_{att}$. (**a**) The GRU cell to extract temporal information. (**b**) The attention-like module represents the association between entities. (**c**) The classifier to select a leader.

Algorithm 2 Leadership Shift (LS)

```
1: Set the leadership scores $g_i = (g_{i,1}, \dots, g_{i,n})$.
2: Select agent $l$ that is the best suited to be the leader based on $l = \mathrm{argmax}_j g_{i,j}$.
3:
4: if agent $l$ is not appointed to be a leader at $t+1$ then
5:     Send a signal to $l$ to appoint it to be a leader at $t+1$.
6:     Appoint agent $i$ to be a worker at $t+1$.
7: else
8:     Appoint agent $i$ to be a leader at $t+1$.
9: end if
```

4.3. Team Transformer (T-Trans)

In some conventional studies, teammate modeling [6,7] is used to enable agents to generate targeted messages for a specific teammate. However, this technique is not applicable to our leaders, because the behaviors of the workers are not (always) determined by their own policies. Therefore, we proposed T-Trans for leaders to produce targeted messages for each of their teammates. The proposed T-Trans used in the CTLCE framework with LS, which is described below, is called the *locally centralized T-Trans* (LCTT).

The structure of T-Trans is shown in Figure 4. First, we represent the observation of i as a tuple of the respective information of entities, $o_i = (o_{i,1}, \dots, o_{i,i}, \dots, o_{i,|E|})$, and $o_{i,j} = \emptyset$ if entity $e_j \in E$ is not in the observable area of i. The observations are thereafter fed into a *gate recurrent unit* (GRU) [39] cell (Figure 4a) to extract temporal information, and an attention-like module [24] (Figure 4b), which represents the association between entities. The features of each agent are concatenated with the temporal features and finally processed by an action head to generate the Q values. A classifier (Figure 4c) is used to select the most suitable agent as the leader in the next time step.

In the attention module (Figure 4b), the attention feature is calculated as follows:

$$\boldsymbol{z_i} = \text{softmax}(\frac{\boldsymbol{q_i k_i^T}}{\sqrt{d_{att}}})\boldsymbol{v_i}, \tag{4}$$

where $\boldsymbol{k_i^T}$ denotes the transpose of $\boldsymbol{k_i}$ and $\boldsymbol{z_i}$ contains a teammate-specific representation of the agents. $z_{i,j} \in \boldsymbol{z_i}$ can be interpreted as follows: *i imagines that if i was j, j would observe* $z_{i,j}$. Subsequently, $\boldsymbol{z_i}$ is concatenated with the duplicated hidden states. Then, the result is converted into teammate-specific instructions by a fully connected layer (FC-layer) action head.

4.4. Leadership Shift (LS)

Initially, l (> 0) agents were randomly designated as leaders in each episode (line 3 in Algorithm 1). Thereafter, we designed an LS to allow agents to automatically hand over leadership to ensure that instructions for workers are provided by suitable agents. The leader agents use a multi-layer perceptron (MLP) (Figure 4c) to produce the hidden states and generate the *leadership scores*, $g_i = (g_{i,1}, \ldots, g_{i,n})$, where $g_{i,j}$ represents i's consideration of the degree of j's appropriateness as a leader. Then, according to the scores, the leader appoints an observed agent as a leader at the next time step. The pseudocode of LS for agent i is presented in Algorithm 2.

The *leadership Q-value* to construct pseudo-labels for g_i is introduced to effectively train the leadership scores g_i. First, we calculate the leadership Q-value for every agent $k \in \mathcal{N}$ as follows:

$$\bar{Q}_k^L(o_k, o_j, |\bar{\theta}) = \sum_j \bar{Q}_j(o_j, \text{argmax}_{a_j} \bar{Q}_k(o_k, a_j|\bar{\theta})|\bar{\theta}), \tag{5}$$

where j is an agent observed by k. Note that o_j is required to calculate $\bar{Q}_k^L$ only in CT. The proposed method does not rely on obtaining observations of other agents during LCE. Subsequently, the pseudo-labels for g_i, expressed by g_i^*, are calculated as follows:

$$g_i^* = \text{ONE\_HOT}(g_{i,1}^*, \ldots, g_{i,n}^*), \tag{6}$$

where ONE_HOT is a onehot function. $g_{i,k}^* = \bar{Q}_k^L$ if k is in the observable area of i; otherwise $g_{i,k}^* = -\infty$. Subsequently, g_i is learned by minimizing the cross-entropy loss as follows:

$$L_i^g(\theta) = H(g_i'^*) + D_{KL}(g_i'^*||g_i), \tag{7}$$

where $H(g_i'^*)$ is the entropy of $g_i'^*$ and $D_{KL}(g_i'^*||g_i)$ is the Kullback–Leibler divergence of $g_i'^*$ from g_i. Note that $g_i'^*$ is calculated using the observations at $t+1$ from the target network with $\bar{\theta}$, whereas g_i is calculated using the observations of i at t from the network with θ.

Finally, the Q-values of the agents are input into a *mixing network* [25], and the overall learning objective of all the parameters is

$$L(\theta) = \sum_{i=1}^{n} L_i^Q(\theta) + \lambda \sum_{i=1}^{n} L_i^g(\theta), \tag{8}$$

where $L_i^Q(\theta)$ is the standard DQN loss function (Equation (1)) and λ is a weight hyperparameter. The overall framework of the proposed LCTT is shown in Figure 5.

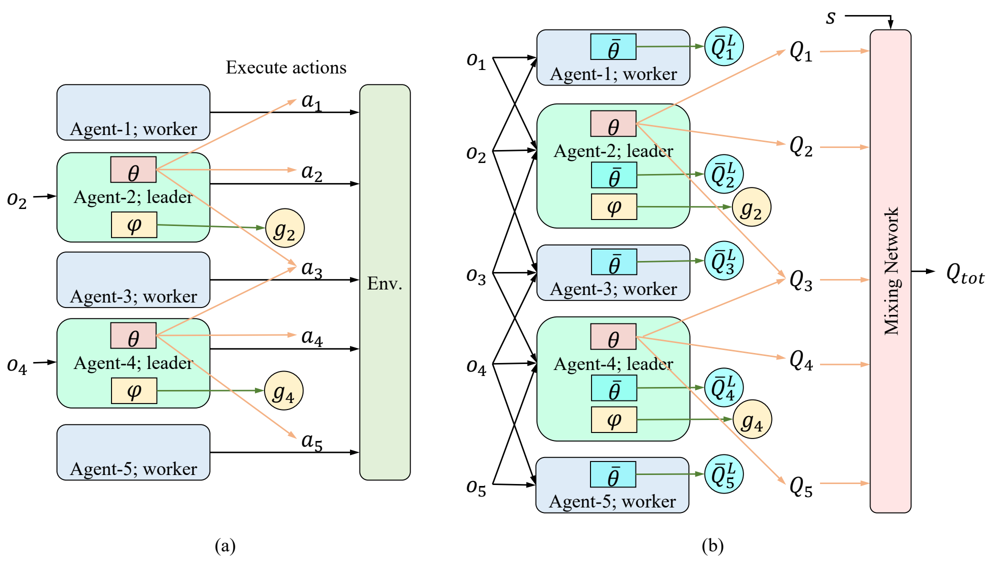


Figure 5. Example structure of the CTLCE framework in which ϕ is the network parameter for generating leadership scores. Agents 1, 3, 5 are workers, and 2 and 4 are leaders. (**a**) The LCE phase. Agents 2, 4 can instruct the actions of their adjacent workers and appoint leaders at the next time step. (**b**) The CT phase. Note that observations from other agents are only required when calculating the leadership Q-value $\bar{Q}^L_*$.

5. Experiments

5.1. Comparison to Baselines

We evaluated the proposed method, LCTT, by comparing it to the baselines, MAIC [7] and QMIX [25], in the LBF environment, as shown in Figure 2, where $n = 4$, $\beta = 2$ and the size of the grid world is 10×10. The level of any agent i is a random integer $1 \leq lv_a(i) \leq 5$. The level of food f_μ is also a random integer $1 \leq lv_f(\mu) < \sum_{i \in \mathcal{N}} lv_a(i)$. All the agents have observable areas of size 5×5. The experimental environment was the same as that of the official repository of MAIC. The network architecture we used to conduct experiments in LBF is illustrated in Figure 4. Raw observation is processed by a GRU cell (Figure 4a) with a dimension of 64 to extract historical information. The attention-like module (Figure 4b) uses three MLPs, each containing two FC layers with 32 units, to represent the features of entities. The classifier (Figure 4c) has two fully connected layers with 64 units to calculate leadership scores. We selecteed $\lambda = 0.01$ for the weight of the leadership loss. The entire architecture is trained end-to-end by an RMSProp optimizer with parameters including the learning rate of 5×10^5, $\alpha = 0.99$, and RMSProp $\epsilon = 1 \times 10^5$. The number of model parameters of QMIX, MAIC, and LCTT in Section 5.1 are 27, 260, 48, 796, and 45, 802, respectively.

The experimental results are shown in Figure 6, in which LCTT-lL indicates the LCTT with l leaders. Therefore, LCTT-1L indicates that one leader instructs others although the leader may change over time. LCTT-0L correspond to DE, whereas in LCTT-4L, all agents attempted to instruct other agents. We also conducted an experiments in CN environment to understand the behaviors of leader agents. We explain the experimental setting of CN below.

Figure 6a illustrates the mean rewards of LCTT and baselines over timestep in the test phase. All methods achieved similar rewards after convergence; however, LCTT with two leaders, referred to as LCTT-2L, showed the fastest convergence. MAIC allows each agent to generate incentive messages and directly bias the value functions of other agents. MAIC can be considered a special case of LCTT in which $l = n = 4$; that is, all agents are leaders and can mutually instruct others. LCTT-4L converged faster than MAIC probably because we used T-Trans rather than teammate modeling to generate messages, and T-Trans does not require explicit modeling or an additional loss functions. QMIX is a frequently used learning method based on CTDE and is the basis of LCTT and MAIC, in which agents make decisions based only on their own observations without communication. QMIX can be considered as a special case of LCTT in which $l = 0$; that is, all agents are workers. Thus, they determine their actions individually, without relying on T-Trans. Figure 6a shows that LCTT-0L converged faster than QMIX, indicating that T-Trans can learn better than naive MLP networks. We omitted the curves of LCTT-1L and LCTT-3L for the readability of Figure 6. They both converge faster than MAIC but slower than LCTT-2L.

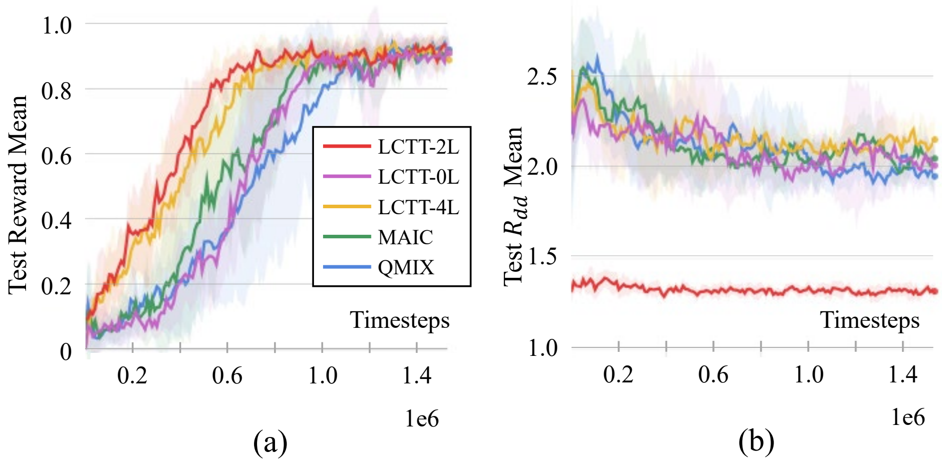


Figure 6. Experimental results of LCTT and baselines in the LBF environment. (**a**) Mean rewards. (**b**) Mean redundant observation ratios. The curves in (**a**,**b**) show mean performance over five random seeds, and light-colored areas show standard deviation.

The mean values of the redundant observation ratio, R_{dd}, are shown in Figure 6b. The redundant observation ratio of LCTT-2L is obviously lower than that of the other methods, which indicates that the two leaders of LCTT-2L learned to instruct workers to cooperate, rather than letting them make their own decisions. In LCTT-4L, LCTT-0L, MAIC, and QMIX, the redundant observation ratios are larger than those of LCTT-2L, because all four types of agents must make their own decisions independently. The R_{dd} after convergence for MAIC and LCTT-2L are around 2.1 and 1.3, respectively. Therefore, we believe that our method saves about 38.1% of the computational resources. $((2.1 - 1.3)/2.1 \approx 38.1\%)$. In addition, at the beginning of the learning process, their redundant observation ratios increased temporarily, implying that agents realize that they usually cannot load food alone and must gather to load jointly. Their redundant observation ratios then gradually decreased to a relatively stable range, probably because they became aware that they required proper dispersion to explore the environment to find foods. In contrast, the redundant observation ratio of LCTT-2L was maintained at a low level owing to LCE, although it decreased slightly over time. Specifically, while agents gathered, the redundant observation ratio would not increase because the worker entered the instruction region of leaders and stopped observing; while agents dispersed, the redundant observation ratio would not decrease because the worker left the instruction region of leaders and started observing.

Moreover, the proposed method requires a small bandwidth because instruction messages are unidirectional from the leaders to the workers instead of bidirectional between agents like existing methods. Table 1 lists the number of messages per time step in an episode. The values are calculated over 1000 test episodes. As the number of leaders decreases, that of messages decreases significantly, resulting in smaller communication bandwidth and costs. Considering convergence (Figure 6a), redundant observation ratios (Figure 6b), and communication costs (Table 1), we believe that LCTT-2L achieves the best performance.

Table 1. Mean number of messages per time step in LBF.

MAIC	LCTT-4L	LCTT-3L	LCTT-2L	LCTT-1L
5.071 ± 0.223	5.288 ± 0.414	4.106 ± 0.252	3.338 ± 0.203	1.528 ± 0.075

5.2. Ablation Study and the Effect of the Number of Leaders

We conducted ablation studies to evaluate the contributions of LS and the decisions of worker agents for themselves. We also analyzed the effect of the number of leaders on the overall performance, i.e., individual and team rewards. First, by omitting LS, l agents are randomly designated as leaders initially in each episode, and they do not hand over the leadership; therefore, we refer to them as *fixed leaders*, and this setting is denoted by "Fl," such as F2. In contrast, in the normal LCTT, l leaders can hand over their leadership by LS, and this setting is denoted by "Sl." Second, in LCTT with Fl or Sl settings, workers out of leaders' instructing regions will determine their actions themselves (lines 13∼18 in Algorithm 1); these workers are called *positive workers*, and this setting is denoted by "$+$" such as F$l+$ and S$l+$. Meanwhile, if the workers out of leaders' instructing regions do not decide their action autonomously and take random actions, they are called *negative workers* and the experimental setting in which all workers are negative is denoted by "$-$" such as F$l-$. Therefore, LCTT-0L, LCTT-2L, and LCTT-4L in Figure 6a can also be concisely denoted by S0+ (= F0+), S2+ and S4+ (= F4$\pm$).

Figure 7 shows the mean rewards and redundant observation ratios of QMIX, MAIC, and ablation versions of LCTT with different numbers of leaders after convergence. According to the numbers of leaders, these 16 algorithms can be classified into five groups (0 leaders, 1 leader, and 3 workers). QMIX, F0+, F4+, and MAIC achieved similar performance with large rewards and high redundant observation ratios. In contrast, the LCTT with $0 < l < n$ can obviously reduce redundant observation ratios. Comparing the respective algorithms of the groups with 1, 2, and 3 leaders, generally, fewer leaders lead to lower redundant observation ratios but also result in smaller rewards; for instance, F1+, F2+, and F3+ achieved rewards of 0.873, 0.894, and 0.921, with redundant observation ratios of 1.30, 1.35, and 1.64, respectively. We believe that S2+ shows the best performance; it achieved relatively low redundant observation ratios without causing a reduction in rewards.

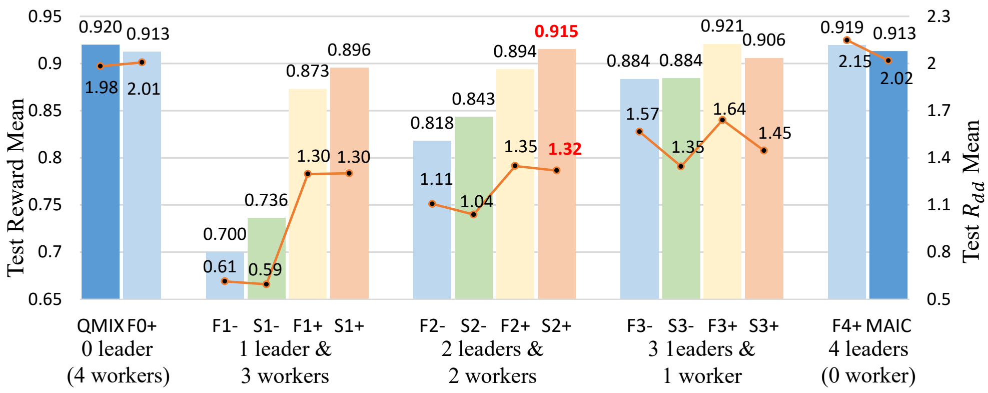


Figure 7. Mean rewards (bars) and redundant observation ratios (lines). All values are averaged from 1600 test episodes among five random seeds.

Moreover, S$l+$ (S$l-$) achieves larger rewards than F$l+$ (F$l-$), indicating that LSs can help agents find suitable ones to generate effective instructions for their workers. The special case is that S3+ achieves smaller rewards than F3+, probably because, when most agents are leaders, LS brings limited benefits but also causes certain learning difficulties. Additionally, S$l+$ (S$l-$) achieved lower redundant observation ratios than F$l+$ (F$l-$), indicating that LS tends to find agents that can observe more entities as leaders. Furthermore, S$l+$ (F$l+$) achieves larger rewards and higher redundant observation ratios than S$l-$ (F$l-$), indicating that the policy learned in the CTCLE framework can not only generate effective instructions for other agents but also generate effective actions for worker agents themselves.

5.3. Experimental Results on the Difficult LBF Setting

We evaluated LCTT in a more difficult mode of LBF. In this case, the environment only contains food of a specific level such that all agents must cooperate to successfully load the food, i.e., $lv_f(f_\mu) = \sum_{i\in\mathcal{N}} lv_a(i)$. This environment has the same size and number of agents and foods as those shown in Figure 2. We refer to this setting as LBF-hard.

Figure 8a shows the reward of LCTT and baselines. Some random seeds of QMIX failed to converge; therefore, QMIX achieved low mean rewards and large variance. MAIC and versions of LCTT achieved similar rewards. Figure 8b shows the redundant observation ratios of LCTT and baselines. As the number of leaders decreases, the redundancy decreases significantly, indicating that LCTT leaders can effectively instruct workers in this difficult game. In this particular experiment, as all agents need to cooperate to load food, a centralized regime to control other agents would be better. In this situation, LCE functions like CE, and our experiment shows that agents in LCE can achieve cooperative behaviors by appropriately selecting leaders that directly instruct other agents in their observable regions.

We also evaluate the proposed method in environments with larger sizes and more entities; the results are provided in Appendix A.

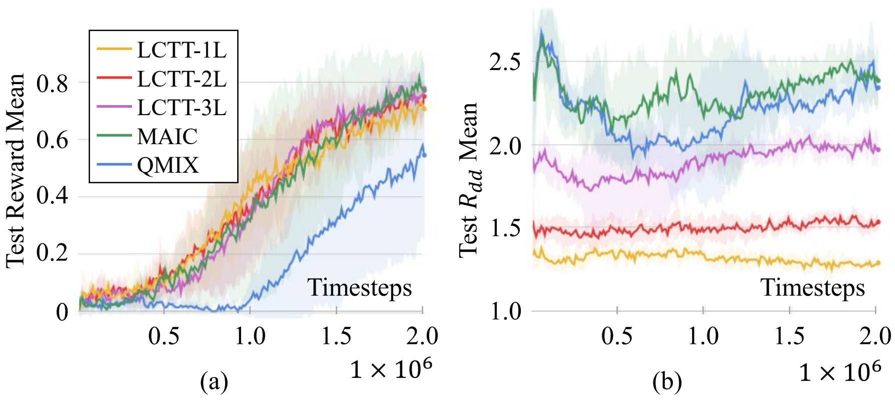


Figure 8. Mean rewards and redundant observation ratios on LBF-hard.

5.4. Experimental Results on Cooperative Navigation

To show the behaviors of leader agents more clearly, we conducted experiments in a CN environment [12], as shown in Figure 9a. Initially, in each episode, 16 agents and 16 landmarks appear randomly in the square area $\mathcal{M}$ ($-1 < x < 1$ and $-1 < y < 1$) of a two-dimensional space. Agent $i \in \mathcal{N}$ can observe landmarks and other agents within its visible range within a radius of 0.5. Throughout the task, agents move around, attempting to cover landmarks within their view while avoiding collisions with other agents. Note that the agents may move out of $\mathcal{M}$. All the agents shared a negative team reward, represented by the sum of the negative distances between each landmark and its nearest agents. The team receives a negative reward of -1 upon each collision. Agents obtained no positive rewards, and rewards closer to 0 indicate better behavior. Figure 10 presents the mean rewards and redundant observation ratios per time step of LCTT-2L, LCTT-4L, LCTT-9L, and QMIX baseline. All the three LCTT models achieved higher rewards and smaller redundant observation ratios than the QMIX baseline model. This figure shows that LCTT-2L achieved the largest rewards (Figure 10a) and LCTT-4L achieved the smallest redundant observation ratios.

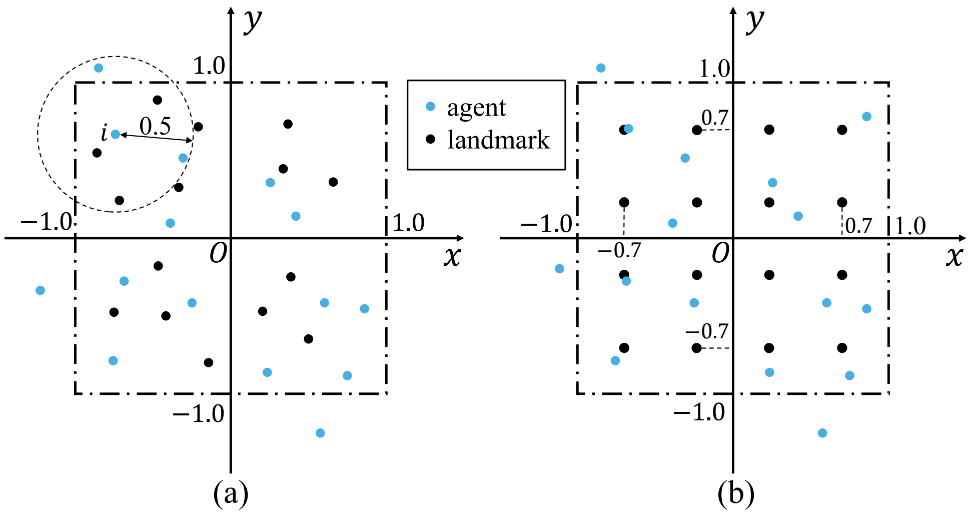


Figure 9. (**a**) A CN environment. Agents cover landmarks without collision. Agents have visible ranges with radius of 0.5. (**b**) A manually set CN environment for analyzing the behaviors of leader agents. Landmarks are arranged in a square array.

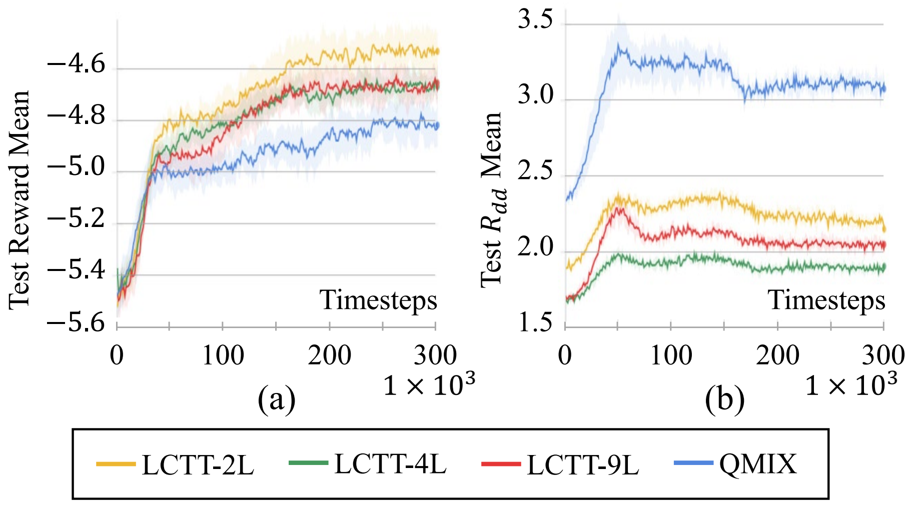
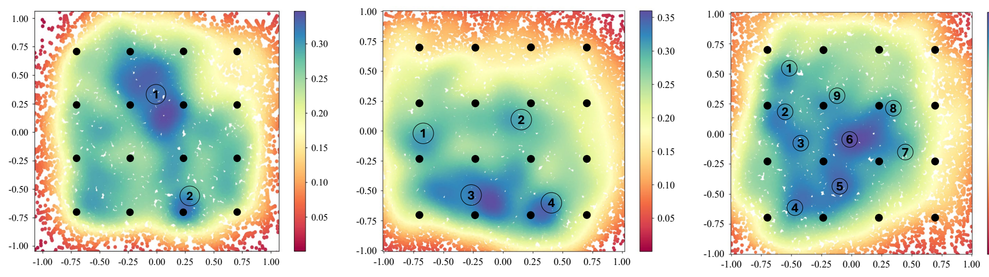


Figure 10. Mean rewards and redundant observation ratios on CN.

To understand this phenomenon in CN, we further analyzed the behaviors of leader agents by counting their location distribution in a manually set environment in which all landmarks were arranged in a square array, as shown in Figure 9b. Thereafter, we loaded the models that are trained in the previous environment (Figures 9a and 10) and tested their performance in this environment (Figure 9b). We tested models of LCTT-2L, LCTT-4L, and LCTT-9L, and recorded the locations of the leaders in 1000 episodes. These locations are shown in Figure 11 as heat maps, where the color temperature indicates the frequency of staying locations on the map. In Figure 11a, the two leaders of LCTT-2L tend to remain in two main areas. We believe that leaders in these two areas can provide instructions to a larger number of workers, which will help the team obtain greater rewards and achieve smaller redundant observation ratios. Similarly, in Figure 11b,c the leaders of LCTT-4L and LCTT-9L tended to be distributed across four and nine areas, respectively. We believe that this behavioral pattern of the leaders is the reason why our method can achieve more rewards than the QMIX baseline.

In contrast, LCTT-2L achieved the largest rewards (Figure 10a) while LCTT-4L achieved the smallest redundant observation ratios (Figure 10b). In LCTT-4L, Figure 11b shows that as R_{dd} is the smallest, its four leaders could just barely cover other workers and the square environment, $\mathcal{M}$. However, it also showed that four leaders were not enough to appropriately determine the instructions of all workers. Meanwhile, in LCTT-2L, two leaders could not cover the entire environment; thus, some worker agents determined their actions individually. This mixed regime with two leaders and a few independent worker agents achieved locally centralized control with distributed autonomous decisions and was appropriate for obtaining higher rewards, although this regime led to the higher redundant observation ratio. The redundant observation ratio of LCTT-9L is slightly lower than that of LCTT-2L; however, like LCTT-4L, LCTT-9L was still sufficient to cover the entire environment. Therefore, some workers near the boundary of visible area of leaders waited for the instructions, although they might be able to autonomously determine better actions using the information on the other side of the leaders.



(**a**) (**b**) (**c**)

Figure 11. Heat maps of locations of LCTT leaders. (**a**) Leader locations of LCTT-2L. (**b**) Leader locations of LCTT-4L. (**c**) Leader locations of LCTT-9L.

This discussion also suggests another issue for CTLCE, particularly LS. Figure 10 indicates that CTLCE decides that the *key* location in CN is the center area, because leaders were selected based on the number of visible entities. Therefore, agents located near the boundary were unlikely to be selected as leaders. This selection is effective when the number of leaders is not as large; however, even with more agents, the current CTLCE does not fully utilize them. Covering the observable area in a mutually complementary manner is advisable. This issue must be our future research topic.

6. Conclusions

This study focused on the redundant computation problem of multi-agent systems, a well-known but neglected traditional problem. We proposed the redundant observation ratio metric to quantitatively describe the degree of redundant computations. Thereafter, we demonstrated that the redundancy can be reduced by enabling the agents to generate instructions. Therefore, we introduced the LCTT framework in which each agent determines which agents should instruct other agents and be instructed by leaders, and how. The results of the experiments using LBF showed that the proposed method significantly eliminates redundant computation while simultaneously ensuring that it does not decrease the rewards, enhancing the efficiency of the learning process. Through an ablation study, we confirmed that LS can reduce redundant computation and mitigate the reward decrease with the number of leaders. Additionally, we confirmed that the policy learned through CTLCE can provide effective instructions for other agents and each agent itself. Furthermore, in the CN environment, we demonstrated the behavioral pattern of leaders who prefer to stay in *key* locations where they can instruct more workers, leading to higher rewards and smaller redundant observation ratios.

The proposed method can significantly reduce the redundant observation ratios; thus, during the execution stage, a large computational cost of the MAS can be saved. Note that the proposed method does not compete with existing approaches that reduce computational costs because it is an extension of another independent dimension; thus, it can be easily integrated with them. We believe that further discussions on reducing overlapping observation and redundant computation will be of great interest, such as the issue discussed in Section 5.4. We believe that flexibly integrating this study with studies on multi-agent communications, for instance, developing communication between leader agents, can achieve some remarkable results.

Author Contributions: Conceptualization, Y.B. and T.S.; methodology, Y.B. and T.S.; software, Y.B.; validation, Y.B.; formal analysis, Y.B. and T.S.; investigation, Y.B. and T.S.; resources, T.S.; writing—original draft preparation, Y.B.; writing—review and editing, T.S.; visualization, Y.B.; supervision, T.S.; project administration, T.S.; funding acquisition, T.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by JST KAKENHI, Grant Number 20H04245, Waseda University Grant for research base creation, Grant Number 2024C-117, and SPRING, Grant Number JPMJSP2128.

Data Availability Statement: Data for reproducing the results in this paper will be made available in a public repository after acceptance.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Experimental Results in a Large Environment

We evaluate the scalability of the proposed method to a larger number of entities in three larger LBF environments. The first one, whose size is 15×15, contains eight agents and four foods, and this setting is denoted by LBF15. The second one also contains eight agents and four foods, but its size is 20×20, denoted by LBF20. The last one has the same number of entities and environment size as LBF20, but the observable areas of agents are 7×7, instead of 5×5. We refer to this setting as LBF20/7. Therefore, the LBF setting in Figures 2 and 6 is denoted by LBF10. Note that LBF15 has a similar entity density to LBF10, i.e., $(8+4)/15 \times 15 \approx (4+2)/10 \times 10$. In contrast, LBF20 provides a sparser and more capacious environment.

Figure A1 shows the rewards and redundant observation ratios of LCTT and baselines. Figure A1a shows that in LBF15, LCTT achieved similar rewards to baselines with a lower redundant observation ratio, indicating that the LCTT can scale up to eight agents. Note that in Figure A1b, LCTT-6L, LCTT-4L, and LCTT-2L achieved lower redundant observation ratios with a smaller number of leaders; however, LCTT-1L achieved higher redundant observation ratios than LCTT-4L and LCTT-2L. This is because one leader cannot instruct the behaviors of several (seven) workers with its small indicating region; the workers must frequently leave the instructing region of the leader to explore and observe the environment.

However, in LBF20, LCTT cannot achieve significantly lower redundant observation ratios with fewer leaders. Figure A2b shows the redundant observation ratios of LCTT and baselines in LBF20. As we reduce the number of leaders, the redundant observation ratios slightly decrease instead of decreasing significantly similar to that in Figure 6b. This is because first, in the LBF20 environment, entities are sparsely distributed, and agents of baseline methods also have fewer overlapping observations and lower redundant observation ratios (approximately 1.5 after convergence) than those in LBF10 and LBF15 (approximately 2.0). Second, even with our designated leaders, workers must leave the indicating regions to explore and observe the environment. Figure A2a shows that LCTT achieved fewer rewards and slower convergence than baselines, probably because the leaders tend to maintain the workers not far away from leaders to instruct their behaviors, which limits the agents from fully exploring the environment.

We further evaluate LCTT and baselines in LBF20/7 to verify our speculation on the reason why LCTT degrades the performance in LBF20. In LBF20/7, LCTT achieved similar rewards to baselines with lower redundant observation ratios because of the larger observable/instructing area than LBF20, by which LCTT leaders can instruct workers both to spread out to explore the environment and to gather to loading foods. This indicates that our speculation for LBF20 is reasonable.

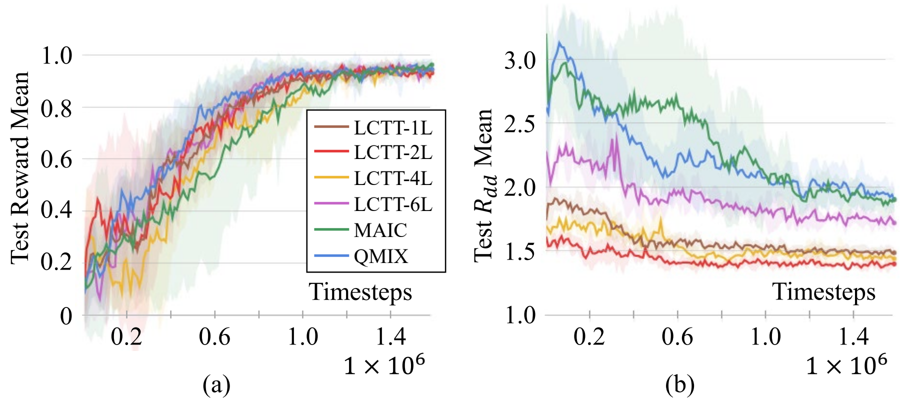


Figure A1. Mean rewards and redundant observation ratios on LBF15.

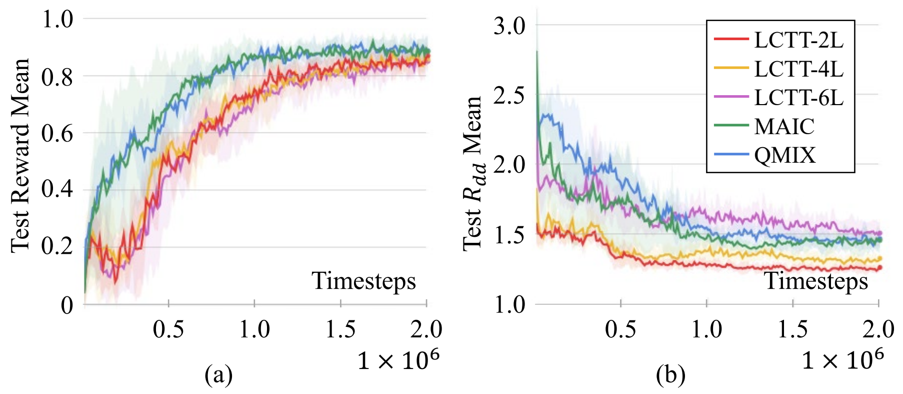


Figure A2. Mean rewards and redundant observation ratios on LBF20.

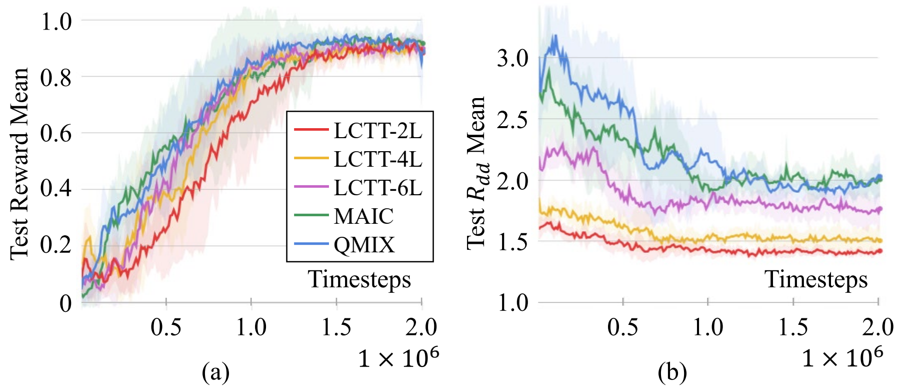


Figure A3. Mean rewards (**a**) and redundant observation ratios (**b**) on LBF20/7.

References

1. Yu, C.; Wang, X.; Xu, X.; Zhang, M.; Ge, H.; Ren, J.; Sun, L.; Chen, B.; Tan, G. Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 735–748. [CrossRef]
2. Wachi, A. Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving. *arXiv* **2019**, arXiv:1903.10654.
3. Bhalla, S.; Ganapathi Subramanian, S.; Crowley, M. Deep multi agent reinforcement learning for autonomous driving. In Proceedings of the Canadian Conference on Artificial Intelligence, Ottawa, ON, Canada, 13–15 May 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 67–78.
4. Palanisamy, P. Multi-agent connected autonomous driving using deep reinforcement learning. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020 IEEE: Amsterdam, The Netherlands, 2020; pp. 1–7.
5. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv* **2016**, arXiv:1610.03295

6. Wang, Y.; fangwei zhong.; Xu, J.; Wang, Y. ToM2C: Target-oriented Multi-agent Communication and Cooperation with Theory of Mind. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
7. Yuan, L.; Wang, J.; Zhang, F.; Wang, C.; Zhang, Z.; Yu, Y.; Zhang, C. Multi-agent incentive communication via decentralized teammate modeling. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual Event, 22 February–1 March 2022; Volume 36, pp. 9466–9474.
8. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with large scale deep reinforcement learning. *arXiv* **2019**, arXiv:1912.06680.
9. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Proceedings of the Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, 8–12 May 2017; Revised Selected Papers 16; Springer: Berlin/Heidelberg, Germany, 2017; pp. 66–83.
10. Han, L.; Sun, P.; Du, Y.; Xiong, J.; Wang, Q.; Sun, X.; Liu, H.; Zhang, T. Grid-wise control for multi-agent reinforcement learning in video game AI. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 2576–2585.
11. Kraemer, L.; Banerjee, B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* **2016**, *190*, 82–94. [CrossRef]
12. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *arXiv* **2017**, arXiv:1706.02275.
13. Wooldridge, M. *An Introduction to Multiagent Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
14. Weiss, G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*; MIT Press Cambridge, MA, USA, 1999.
15. Shoham, Y.; Leyton-Brown, K. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*; Cambridge University Press: Cambridge, UK, 2008.
16. Ferber, J.; Weiss, G. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*; Addison-Wesley Reading: Boston, MA, USA, 1999; Volume 1.
17. Yang, Y.; Wang, J. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv* **2020**, arXiv:2011.00583.
18. Sugawara, T. A cooperative LAN diagnostic and observation expert system. In Proceedings of the 1990 Ninth Annual International Phoenix Conference on Computers and Communications, Scottsdale, AZ, USA, 21–23 March 1990; IEEE Computer Society: Washington, DC, USA, 1990; pp. 667–668.
19. Durfee, E.H.; Lesser, V.R.; Corkill, D.D. Coherent cooperation among communicating problem solvers. *IEEE Trans. Comput.* **1987**, *100*, 1275–1291. [CrossRef]
20. Krnjaic, A.; Steleac, R.D.; Thomas, J.D.; Papoudakis, G.; Schäfer, L.; To AW, K.; Lao, K.-H.; Cubuktepe, M.; Haley, M.; Börsting, P.; et al. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. *arXiv* **2022**, arXiv:2212.11498.
21. Xu J.; Zhong F.; Wang Y. Learning multi-agent coordination for enhancing target coverage in directional sensor networks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10053–10064.
22. Cammarata S.; McArthur D.; Steeb R. Strategies of cooperation in distributed problem solving. *Readings in Distributed Artificial Intelligence*; Elsevier: Amsterdam, The Netherlands, 1988; pp. 102–105.
23. Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of PPO in cooperative, multi-agent games. *arXiv* **2021**, arXiv:2103.01955.
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *arXiv* **2017**, arXiv:1706.03762.
25. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 7234–7284.
26. Papoudakis, G.; Christianos, F.; Schäfer, L.; Albrecht, S.V. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv* **2020**, arXiv:2006.07869.
27. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: a survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [CrossRef]
28. Peng, P.; Wen, Y.; Yang, Y.; Yuan, Q.; Tang, Z.; Long, H.; Wang, J. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv* **2017**, arXiv:1703.10069.
29. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference On Machine Learning, Amherst, MA, USA, 27–29 June 1993; pp. 330–337.
30. Sukhbaatar, S.; Fergus, R. Learning multiagent communication with backpropagation. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 2252–2260.
31. Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P.; Strouse, D.; Leibo, J.Z.; De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In Proceedings of the International Conference on Machine Learning (PMLR 2019), Beach, CA, USA, 9–15 June 2019; pp. 3040–3049.
32. Ding, Z.; Huang, T.; Lu, Z. Learning individually inferred communication for multi-agent cooperation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22069–22079.
33. Bai, Y.; Sugawara, T. Reducing Redundant Computation in Multi-Agent Coordination through Locally Centralized Execution. *arXiv* **2024**, arXiv:2404.13096.

34. Jiang, J.; Lu, Z. Learning attentional communication for multi-agent cooperation. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 7265–7275.
35. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
36. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, Cambridge, UK, 1989.
37. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
38. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602
39. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.