Scalable Lifelong Reinforcement Learning

Yusen Zhan

The School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99163, USA

Haitham Bou Ammar

Prowler i.o., Cambridge, United Kingdomn

Matthew E. Taylor*

The School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99163, USA

Abstract

Lifelong reinforcement learning provides a successful framework for agents to learn multiple consecutive tasks sequentially. Current methods, however, suffer from scalability issues when the agent has to solve a large number of tasks. In this paper, we remedy the above drawbacks and propose a novel scalable technique for lifelong reinforcement learning. We derive an algorithm which assumes the availability of multiple processing units and computes shared repositories and local policies using only local information exchange. We then show an improvement to reach a *linear convergence rate* compared to current lifelong policy search methods. Finally, we evaluate our technique on a set of benchmark dynamical systems and demonstrate learning speed-ups and reduced running times.

Keywords: Reinforcement Learning, Lifelong Learning, Distributed Optimization, Transfer Learning

*Corresponding author

Email addresses: yusen.zhan@wsu.edu (Yusen Zhan), haitham@prowler.io (Haitham Bou Ammar), taylorm@eecs.wsu.edu (Matthew E. Taylor)

1. Introduction

Reinforcement learning (RL) provides the ability to solve sequential decision-making problems with limited feedback. Applications with these characteristics range from robotics control [1] to personalized medicine [2, 3]. Though successful, typical RL methods require a substantial amount of experience before acquiring acceptable behavior. The cost of obtaining such experience, however, can be prohibitively expensive in terms of time and data [4]. Also, these costs only worsen when considering multiple tasks[1]. Transfer learning[2] and multi-task learning [7, 8] have been developed to remedy these problems by allowing agents to reuse knowledge from other tasks. Unfortunately, both these techniques suffer from scalability problems as the number of tasks considered grows large. In a recent attempt to target these issues, Bou Ammar *et al.* proposed PG-ELLA, an online hybrid of the above two paradigms—transfer and multi-task learning [9]. PG-ELLA decomposes a task's policy parameters into a shared latent repository $\boldsymbol{L}$ and task specific coefficients $\boldsymbol{s}_t$, one for each task. Consequently, it allows for knowledge transfer between tasks (using the latent repository $\boldsymbol{L}$) while scaling multi-task learning by streaming problems online. In the original paper, the authors show a closed form solution to the shared repository.

In a lifelong setting, the agent deals with a sequence of tasks, potentially leading to computational intractability when the agent observes and learns many tasks (e.g., in big data scenario, the agent may observe many tasks in its life). Unfortunately, PG-ELLA suffers when considering a large number of tasks or dimensions — determining the shared knowledge-base $\boldsymbol{L}$ becomes intractable — due to two inefficiencies. The first inefficiency is that computing the expansion's operating point amounts to solving a local RL problem described by the current task's observed trajectories. The second inefficiency reducing PG-ELLA's

[1] The mutiple tasks learning tried to learning a single polices or strategy to deal with a set of tasks. See [5] for details.

[2] The idea of transfer learning is to reuse previous trained information to speed up learning process. See [6] for details.

scalability arises when updating the shared repository $\boldsymbol{L}$.

In this paper, we remedy the above drawbacks of lifelong policy search and propose a scalable method capable of handling large number of tasks and high-dimensional policies, while ensuring meaningful parameters at each iteration of the algorithm (i.e., each step conveys more information than a single gradient step, as done in PG-ELLA). Our method assumes multiple processing units and derives a novel algorithm for lifelong policy search. Such an assumption leads us to better convergence rates and more accurate local policies. Using augmented Lagrangian methods, we propose two algorithms for computing the local operating point and the shared repository, while allowing for scalable solutions. We show that such updates can be computed locally by each processing unit in closed form under broad assumptions. We theoretically and empirically analyze the performance of these new techniques. On the theoretical side, we demonstrate superiority to PG-ELLA by proving *dimensionality-free linear convergence* in both determining local policies and updating the shared repository. On the empirical side, we show our algorithm outperforms existing algorithms on a set of 5 domains with 50 variations, including the control of a simulated helicopter. In a final set of experiments, we also demonstrate the improved generalization capabilities of this new method on unobserved tasks and report a decrease in learning time, relative to current techniques. The contributions of this paper can be summarized as: *i)* deriving a novel scalable algorithm for lifelong policy search, *ii)* acquiring linear convergence rate in both steps of the method, *iii)* demonstrating the effectiveness of our technique on 5 dynamical systems with 50 variations each, and *iv)* demonstrating learning speed-ups on unobserved tasks.

The rest of the paper is organized as follows: Section 2 introduces the background; Section 3 provides some related works; In section 4, we define our lifelong policy search problem; Section 5 proposes the scalable lifelong policy search method; In section 6, it shows the experimental results and section 7 summarizes the whole paper.

2. Reinforcement Learning

In reinforcement learning (RL) an agent must sequentially select actions to maximize its total expected pay-off. These problems are typically formalized as Markov decision processes (MDPs) $\langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{U} \subseteq \mathbb{R}^m$ denote the state and action spaces. $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X} \to [0, 1]$ represents the transition probability governing the dynamics of the system, $\mathcal{R} : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ is the reward function quantifying the performance of the agent and $\gamma \in [0, 1)$ is a discount factor specifying the degree to which rewards are discounted over time. At each time step m, the agent is in state $\boldsymbol{x}_m \in \mathcal{X}$ and must choose an action $\boldsymbol{u}_t \in \mathcal{U}$, transitioning it to a successor state $\boldsymbol{x}_{m+1} \sim p(\boldsymbol{x}_{m+1}|\boldsymbol{x}_m, \boldsymbol{u}_m)$ as given by $\mathcal{P}$ and yielding a reward $r_{m+1} = \mathcal{R}(\boldsymbol{x}_m, \boldsymbol{u}_m)$. A policy $\pi : \mathcal{X} \times \mathcal{U} \to [0, 1]$ is defined as a probability distribution over state-action pairs, where $\pi(\boldsymbol{u}_m|\boldsymbol{x}_m)$ denotes the probability of choosing action $\boldsymbol{u}_t$ is state $\boldsymbol{x}_m$.

Policy gradients [1, 4] are a class of reinforcement learning algorithms that have shown successes in solving complex robotic problems [1]. Such methods represent the policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{u}_m|\boldsymbol{x}_m)$ by an unknown vector of parameters $\boldsymbol{\theta} \in \mathbb{R}^d$. The goal is to determine the optimal parameters $\boldsymbol{\theta}^\star$ that maximize the expected average pay-off:

$$\mathcal{J}(\boldsymbol{\theta}) = \int_{\boldsymbol{\tau}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathfrak{R}(\boldsymbol{\tau}) d\boldsymbol{\tau},$$

where $\boldsymbol{\tau} = [\boldsymbol{x}_{0:T}, \boldsymbol{u}_{0:T}]$ denotes a trajectory over a possibly infinite horizon T. The probability of acquiring a trajectory, $p_{\boldsymbol{\theta}}(\boldsymbol{\tau})$, under the policy parameterization $\pi_{\boldsymbol{\theta}}(\cdot)$ and average per-time-step return $\mathfrak{R}(\boldsymbol{\tau})$ is given by:

$$p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) = p_0(\boldsymbol{x}_0) \prod_{m=0}^{M-1} p\left(\boldsymbol{x}_{m+1} \mid \boldsymbol{x}_m, \boldsymbol{u}_m\right) \pi_{\boldsymbol{\theta}}(\boldsymbol{u}_m \mid \boldsymbol{x}_m), \quad \mathfrak{R}(\boldsymbol{\tau}) = \frac{1}{M} \sum_{m=0}^{M} r_{m+1},$$

with an initial state distribution $p_0 : \mathcal{X} \to [0, 1]$. Policy gradient methods, such as episodic REINFORCE [10] and Natural Actor Critic [11, 12], typically employ a lower-bound on the expected return $\mathcal{J}(\boldsymbol{\theta})$ for fitting the unknown policy parameters $\boldsymbol{\theta}$. To achieve this, such algorithms generate trajectories using the current policy $\boldsymbol{\theta}$, and then compares performance with a new parameterization

$\tilde{\boldsymbol{\theta}}$. As detailed in [1], the lower bound on the expected return can be attained using Jensen's inequality and the concavity of the logarithm:

$$\log \mathcal{J}\left(\tilde{\boldsymbol{\theta}}\right) = \log \int_{\boldsymbol{\tau}} p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau}) \mathfrak{R}(\boldsymbol{\tau}) d\boldsymbol{\tau} \geq \int_{\boldsymbol{\tau}} p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathfrak{R}(\boldsymbol{\tau}) \log \frac{p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})}{p_{\boldsymbol{\theta}}(\boldsymbol{\tau})} d\boldsymbol{\tau} + \text{cnst.}$$
$$\propto -\text{KL}\left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau}) \mathfrak{R}(\boldsymbol{\tau}) || p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\right) = \mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right),$$

where $\text{KL}(p(\boldsymbol{\tau})||q(\boldsymbol{\tau})) = \int_{\boldsymbol{\tau}} p(\boldsymbol{\tau}) \log \frac{p(\boldsymbol{\tau})}{q(\boldsymbol{\tau})} d\boldsymbol{\tau}$. Thus, we can directly optimize the lower bound instead of the original objective.

3. Related Work

Transfer learning aims to improve the learning speed of an agent on a new target task by transferring knowledge learned from one or more previous source tasks [6]. However, most current transfer learning methods only consider the target tasks instead of all previously visited tasks. In contrast, multi-task learning (MTL) methods optimize performance over all tasks by training models over all possible tasks [7, 8], but are computationally expensive in lifelong learning setting since the agent must learn a large number of tasks over time [13, 14].

There are many works that consider parallelizing reinforcement learning. For example, Caarls and Schuitema apply parallel on-line temporal Temporal Difference (TD) learning to motor control [15]. Shixiang *et al.* proposed an asynchronous parallel method to deep reinforcement learning[16]. Yahya *et al.* introduce a distributed asynchronous guided policy search method to deep reinforcement learning [17]. Sergey *et al.* also employed Bregman ADMM, which is a centralized ADMM method, to guided policy search setting [18]. However, all of the above methods are based on standard reinforcement learning instead of lifelong learning setting.

PG-ELLA is a recent lifelong policy gradient RL algorithm that can efficiently learn multiple tasks consecutively while sharing knowledge between task policies to accelerate learning [9]. In fact, PG-ELLA, which is a centralized method, can be viewed as one agent or node case of the algorithm we propose in this paper. MTL for policy gradients has also been explored by Deisenroth

et al. [19] through customizing a single parameterized controller to individual tasks that differ only in the reward function. Another closely related work is on hierarchical Bayesian MTL [20], which can learn RL tasks consecutively, but unlike our approach, requires discrete states and actions. Snel and Whiteson's representation learning approach is also related, but limited to a potential function representation [21]. GO-MTL is also a multi-task learning algorithm that focuses on multi-task supervised learning [22]. All of these MTL methods suffer from scalability issues due to their dependence on centralized methods.

4. Lifelong Policy Search

We adopt the lifelong learning framework previously introduced in [9]. An agent has to master multiple RL tasks while supporting transfer to improve learning. Let $\mathcal{T}$ denote the set of tasks in which each element is an MDP. At any time, the learner may face any of the previously seen tasks — it must maximize its performance across all tasks. The goal is to learn optimal policies, $\pi^{\star}_{\boldsymbol{\theta}^{\star}_1}, \ldots, \pi^{\star}_{\boldsymbol{\theta}^{\star}_{|\mathcal{T}|}}$ for all tasks with corresponding parameters $\boldsymbol{\theta}^{\star}_1, \ldots, \boldsymbol{\theta}^{\star}_{|\mathcal{T}|}$, where $\boldsymbol{\theta}^{\star}_j \in \mathbb{R}^d$ parameterizes the policy for task j. For each task j, the learner samples a set of n_j trajectories $\mathbb{T}^{(j)} = \left\{ \boldsymbol{\tau}_j^{(1)}, \ldots, \boldsymbol{\tau}_j^{(n_j)} \right\}$ from the environment by predefined policies, and each trajectory has a length $M^{(j)}$. To transfer knowledge between tasks, we adopt the shared knowledge-base representation, where the policy parameters for each task are represented as a linear combination of a shared latent basis $\boldsymbol{L} \in \mathbb{R}^{d \times p}$ and coefficient vectors $\boldsymbol{s}_j \in \mathbb{R}^p$, i.e., $\boldsymbol{\theta_j} = \boldsymbol{L}\boldsymbol{s}_j$. Here, each column of $\boldsymbol{L}$ encodes a set of transferable knowledge that is tailored to suit the peculiarities of each task using $\boldsymbol{s}_j$. Such a decomposition has been widely used in the multi-task learning literature [9, 13, 22, 23]. Hence, the optimization problem for a total of t tasks can be written as:

$$\min_{\boldsymbol{L}, \boldsymbol{s}_1 : \boldsymbol{s}_t} \frac{1}{t} \sum_{j=1}^{t} \Big(- \mathcal{J}_j \left(\boldsymbol{L}\boldsymbol{s}_j \right) + \mu_1 \left\| \boldsymbol{s}_j \right\|_1 \Big) + \mu_2 \left\| \boldsymbol{L} \right\|_{\mathsf{F}}^2, \tag{1}$$

where $||\boldsymbol{v}||_1$ and $||\boldsymbol{Z}||_{\mathsf{F}}$ denote the L_1 norm of $\boldsymbol{v} \in \mathbb{R}^p$ and the Frobenius norm of $\boldsymbol{Z} \in \mathbb{R}^{d \times p}$, respectively. μ_1 and μ_2 denote regularization parameters, and

$\mathcal{J}_j(\boldsymbol{L}\boldsymbol{s}_j)$ is the total pay-off for a task j:

$$\mathcal{J}_j\left(\boldsymbol{L}\boldsymbol{s}_j\right) = \int_{\boldsymbol{\tau}^{(j)} \in \mathbb{T}^{(j)}} p^{(j)}_{\boldsymbol{L}\boldsymbol{s}_j}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) d\boldsymbol{\tau}^{(j)}.$$

Here, $p^{(j)}_{\boldsymbol{L}\boldsymbol{s}_j}\left(\boldsymbol{\tau}^{(j)}\right)$and $\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)$ denote the task-specific trajectory probability and expected total reward:

$$p^{(j)}_{\boldsymbol{L}\boldsymbol{s}_j}\left(\boldsymbol{\tau}^{(j)}\right) = p^{(j)}_0\left(\boldsymbol{x}^{(j)}_0\right) \prod_{m=0}^{M^{(j)}-1} p^{(j)}\left(\boldsymbol{x}^{(j)}_{m+1} \mid \boldsymbol{x}^{(j)}_m, \boldsymbol{u}^{(j)}_m\right) \pi^{(j)}_{\boldsymbol{L}\boldsymbol{s}_j}(\boldsymbol{u}^{(j)}_m \mid \boldsymbol{x}^{(j)}_m)$$

$$\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) = \frac{1}{M^{(j)}} \sum_{m=0}^{M^{(j)}} \mathcal{R}^{(j)}\left(\boldsymbol{x}^{(j)}_m, \boldsymbol{u}^{(j)}_m\right).$$

As noted in [9], Equation (1) describes batch multi-task learning as opposed to the online lifelong learning setting. This is due to the dependency of the above optimization problem on all trajectories from all tasks. To remedy this problem, the authors employed a local second-order Taylor expansion of the lower-bound to $\mathcal{J}_t(\boldsymbol{\theta}_t)$ around $\tilde{\boldsymbol{\theta}}^{\star}_t = \arg\min_{\tilde{\boldsymbol{\theta}}} -\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right)$, where $\mathcal{J}_{\mathcal{L},\boldsymbol{\theta}}\left(\tilde{\boldsymbol{\theta}}\right) = \mathrm{KL}\left(p_{\boldsymbol{\theta}}(\boldsymbol{\tau})\mathfrak{R}(\boldsymbol{\tau})||p_{\tilde{\boldsymbol{\theta}}}(\boldsymbol{\tau})\right)$, leading to:

$$\min_{\boldsymbol{L},\boldsymbol{s}_1:\boldsymbol{s}_t} \frac{1}{t} \sum_{j=1}^{t} \left(\left\| \tilde{\boldsymbol{\theta}}^{\star}_j - \boldsymbol{L}\boldsymbol{s}_j \right\|^2_{\boldsymbol{H}^{(j)}} + \mu_1 \left\| \boldsymbol{s}_j \right\|_1 \right) + \mu_2 \left\| \boldsymbol{L} \right\|^2_{\mathsf{F}}, \tag{2}$$

where j denotes task j, $\|\boldsymbol{v}\|^2_{\boldsymbol{H}^{(j)}} = \boldsymbol{v}^{\intercal}\boldsymbol{H}^{(j)}\boldsymbol{v}$ and $\boldsymbol{H}^{(j)}$ denotes the Hessian defined as:

$$\boldsymbol{H}^{(j)} = -\mathbb{E}_{\boldsymbol{\tau}^{(j)} \sim p_{\tilde{\boldsymbol{\theta}}^{\star}_j}\left(\boldsymbol{\tau}^{(j)}\right)} \left[\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \sum_{m=1}^{M^{(j)}} \nabla^2_{\tilde{\boldsymbol{\theta}}_j, \tilde{\boldsymbol{\theta}}_j} \log \boldsymbol{\pi}_{\tilde{\boldsymbol{\theta}}_j}\left(\boldsymbol{u}^{(j)}_m \mid \boldsymbol{s}^{(j)}_m\right) \right].$$

The problems of PG-ELLA. Although successful, as shown in [9, 23], PG-ELLA is unscalable to a large number of tasks, which is typical for the lifelong learning setting. Here, we differentiate the two inefficiency sources of PG-ELLA. First, we note that although the dependency on all tasks has been remedied using a second-order expansion, computing the operating point $\tilde{\boldsymbol{\theta}}^{\star}_j$ can be expensive. In essence, determining $\tilde{\boldsymbol{\theta}}^{\star}_j$ equates to solving a local MDP (i.e., the j^{th}) defined over $\mathbb{T}^{(j)}$. This led the authors to propose a one-step gradient update given a *small set* of observed trajectories for a task j. Apart from the reduction in

the number of available trajectories, gradient methods are known to exhibit slow convergence rates and are prone to local minima. Consequently, a PG-ELLA update can lead to a low-quality latent repository $\boldsymbol{L}$ due to the error in approximating $\tilde{\boldsymbol{\theta}}_j^\star$.

The second inefficiency reducing PG-ELLA's scalability arises when updating the shared repository $\boldsymbol{L}$. On one hand, the update involves an inversion of a $dp \times dp$ matrix leading to a complexity of $\mathcal{O}\left(d^3p^2\right)$ in the best case. Consequently, such a method is not scalable for neither high-dimensional policy parametrization nor latent dimensions p. As, as mentioned in [23], gradient-based methods share the problems mentioned above. On the other hand, as the number of tasks grows large, these centralized methods (updating $\boldsymbol{L}$ or computing $\tilde{\boldsymbol{\theta}}_j^\star$) become intractable due to computational and memory constraints.

5. Scalable Lifelong Policy Search

Our strategy to remedy the two problems above consists of both scaling PG-ELLA to multiple trajectories and multiple tasks through more processing units, as well as generalizing gradient methods to algorithms with faster convergence rates. We assume the existence of multiple processors defined through undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with $\mathcal{V}$ denoting the set of nodes and $\mathcal{E}$ the set of edges. We assume a natural ordering among the nodes from $1, \ldots, |\mathcal{V}|$ and $e_{ij} \in \mathcal{E}$ denotes the edge between nodes i and j with $i < j$. We define a block Matrix $\boldsymbol{A}$ to have $|\mathcal{E}|$ rows and $|\mathcal{V}|$ columns. Suppose edge e_{ij} is the kth edge in $\mathcal{E}$ associating to the kth row in matrix $\boldsymbol{A}$, then, the entry (k, i) of matrix $\boldsymbol{A}$ equals identity matrix $\boldsymbol{I} \in \mathbb{R}^{d \times d}$, i.e., $\boldsymbol{A}(k, i) = \boldsymbol{I}$, and the block $\boldsymbol{A}(k, j)$ of matrix A equals -$\boldsymbol{I}$, i.e., $\boldsymbol{A}(k, j) = -\boldsymbol{I}$ and other entries are set to matrix $\boldsymbol{0}$. $\boldsymbol{A}(\boldsymbol{I})$ denotes the general edge-node incident matrix with identity matrix $\boldsymbol{I}$.

The intuition is that we distribute all the tasks among the nodes in the graph and compute the local variables by local variables from other nodes, which connected by edges in the set $\mathcal{E}$. One potential drawback is the communication cost between nodes.

5.1. Scaling the Computation of $\tilde{\boldsymbol{\theta}}_j^\star$

As mentioned earlier, the first problem of current lifelong policy search algorithms relates to the scalability in determining the local operating point $\tilde{\boldsymbol{\theta}}_j^\star$ for the second-order Taylor expansion in Equation (2). Given a set of trajectories $\mathbb{T}^{(j)}$ for a task j, $\tilde{\boldsymbol{\theta}}_j^\star$ is determined as the solution to:

$$\begin{aligned}\tilde{\boldsymbol{\theta}}_j^\star &= \arg\min_{\tilde{\boldsymbol{\theta}}} -\mathcal{J}_{\mathcal{L},\tilde{\boldsymbol{\theta}}}^{(j)}\left(\tilde{\boldsymbol{\theta}}\right) = \arg\min_{\tilde{\boldsymbol{\theta}}} \mathrm{KL}\left(p_{\boldsymbol{\theta}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\middle\|p_{\tilde{\boldsymbol{\theta}}_j}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\right)\\ &= \arg\min_{\tilde{\boldsymbol{\theta}}} \int_{\boldsymbol{\tau}^{(j)}\in\mathbb{T}^{(j)}} p_{\boldsymbol{\theta}^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\log\left[\frac{p_{\tilde{\boldsymbol{\theta}}^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}\right]d\boldsymbol{\tau}^{(j)}.\end{aligned}$$

Given the above processing unit, we assume a random split of the set $\mathbb{T}^{(j)}$ among the nodes $1,\ldots,|\mathcal{V}|$ such that $\mathbb{T}^{(j)} = \cup_{i=1}^{|\mathcal{V}|}\mathbb{T}_i^{(j)}$ and thus rewrite the optimization problem of $\tilde{\boldsymbol{\theta}}_j^\star$ as:

$$\begin{aligned}\min_{\tilde{\boldsymbol{\theta}}_1^{(j)}:\tilde{\boldsymbol{\theta}}_{|\mathcal{V}|}^{(j)}} &\sum_{i=1}^{|\mathcal{V}|}\left[\int_{\boldsymbol{\tau}^{(j)}\in\mathbb{T}_i^{(j)}} p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\log\left[\frac{p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}\right]d\boldsymbol{\tau}^{(j)}\right] \\ &\text{s.t. } \tilde{\boldsymbol{\theta}}_i^{(j)} = \tilde{\boldsymbol{\theta}}_j^{(j)}, \text{ for all } e_{ij}\in\mathcal{E} \text{ and } i<j,\end{aligned} \tag{3}$$

where the constraints $\tilde{\boldsymbol{\theta}}_i^{(j)} = \tilde{\boldsymbol{\theta}}_j^{(j)}$, for all $e_{ij}\in\mathcal{E}$ and $i<j$ have been added so as to ensure agreement across the different processing units in $\mathcal{G}$. To rewrite the constraint above using a compact representation we defined the following: a vector $\boldsymbol{\theta}_:^{(j)} = \left[\tilde{\boldsymbol{\theta}}_1^{\mathsf{T},(j)},\ldots,\tilde{\boldsymbol{\theta}}_{|\mathcal{V}|}^{\mathsf{T},(j)}\right]^\mathsf{T}\in\mathbb{R}^{d|\mathcal{V}|}$ and matrix $\tilde{\boldsymbol{A}} = \boldsymbol{A}(\boldsymbol{I}_{d\times d})\in\mathbb{R}^{d|\mathcal{E}|\times d|\mathcal{V}|}$, where $\boldsymbol{I}_{d\times d}$ is a d-dimensional identity matrix. Using the above notation, we reformulate the problem in Equation (3) as:

$$\begin{aligned}\min_{\tilde{\boldsymbol{\theta}}_1^{(j)}:\tilde{\boldsymbol{\theta}}_{|\mathcal{V}|}^{(j)}} &\sum_{i=1}^{|\mathcal{V}|}\left[\int_{\boldsymbol{\tau}^{(j)}\in\mathbb{T}_i^{(j)}} p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\log\left[\frac{p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}\right]d\boldsymbol{\tau}^{(j)}\right] \\ &\text{s.t. } \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\theta}}_:^{(j)} = \mathbf{0}.\end{aligned}$$

The above is a constraint optimization problem that we aim to solve efficiently. Generally, this problem can be intractable due to the non-convexity of each of the i objectives. Under the broad-class of exponential family policies, however,

each of the local objectives is convex in $\tilde{\boldsymbol{\theta}}_i^{(j)}$ and can thus be solved optimally and efficiently. Our technique for determining the solution relies on a blend between an augmented Lagrangian and a decomposition-coordination procedure similar to the accelerated direction method of multiples ADMM [24] algorithm. In contrast to ADMM, our method is required to determine the feasible point based exclusively on local interactions among the computational nodes. This setting is well known in the distributed optimization literature [25]. Unfortunately, current methods for distributed ADMM mostly focus on the univariate case (as detailed in [25]). The univariate case is limited by its one-dimensional setting — we generalize a distributed version of ADMM to our multi-dimensional setting to propose a scalable and efficient solver for lifelong policy search. Let $\boldsymbol{\lambda} \in \mathbb{R}^{d|\mathcal{E}|}$ be a vector of Lagrange multipliers associating with the constraint $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{\theta}}_{:}^{(j)} = \mathbf{0}$ and a constant $\rho > 0$ of penalty term $\left|\left|\tilde{\boldsymbol{A}}\tilde{\boldsymbol{\theta}}_{:}^{(j)}\right|\right|_2^2$. We augmented Lagrangian methods (similar to existing work [24, 25]) so that the Lagrangian function becomes:

$$\begin{aligned}
&\mathcal{L}_{\text{Aug}}\left(\tilde{\boldsymbol{\theta}}_{:}^{(j)}, \boldsymbol{\lambda}\right) \\
&\quad = \sum_{i=1}^{|\mathcal{V}|}\left[\int_{\boldsymbol{\tau}^{(j)} \in \mathbb{T}_i^{(j)}} p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \log\left[\frac{p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}\right] d\boldsymbol{\tau}^{(j)}\right] \\
&\qquad\qquad - \boldsymbol{\lambda}^{\mathsf{T}} \tilde{\boldsymbol{A}}\tilde{\boldsymbol{\theta}}_{:}^{(j)} + \frac{\rho}{2}\left|\left|\tilde{\boldsymbol{A}}\tilde{\boldsymbol{\theta}}_{:}^{(j)}\right|\right|_2^2.
\end{aligned}$$

At this stage, we can use standard ADMM for acquiring primal-dual updates and thus determining the solution $\tilde{\boldsymbol{\theta}}_{:}^{\star,(j)}$. Standard ADMM, however, is not readily applicable to our setting due to the need of a distributed solution in which each node performs parameter updates based on only local information exchange between its neighbors. To achieve such a distributed solution, we generalize the approach in [25] to the multi-dimensional setting and associate a dual variable with the constraint on each edge. Each processor i keeps a local decision estimate $\tilde{\boldsymbol{\theta}}_i^{(j)}$ and a vector of dual variables $\boldsymbol{\lambda}_{ki}$ with $k < i$. We define two sets of the neighbors of a node i, denote by $\mathbb{P}_i$ and $\mathbb{S}_i$. Here, $\mathbb{P}_i$ collects all nodes having an index lower than i, i.e., $\mathbb{P}_i = \{j | e_{ij} \in \mathcal{E}, j < i\}$ and $\mathbb{S}_i$ all

Algorithm 1 Scalable Operating Point Computation

1: Initialize $\tilde{\boldsymbol{\theta}}_i^{(j)}$, $\forall i \in \{1, \ldots, |\mathcal{V}|\}$, and set $\rho > 0$.

2: **for** $k = 0, \ldots, K$ **do**

3: Each agent i updates, $\tilde{\boldsymbol{\theta}}_i^{(j)}$ in a sequential order from $i = 1, \ldots, |\mathcal{V}|$ using:

$$\tilde{\boldsymbol{\theta}}_i^{(k+1),(j)} = \arg\min_{\tilde{\boldsymbol{\theta}}_i^{(j)}} \left[\int_{\boldsymbol{\tau}^{(j)} \in \mathbb{T}_i^{(j)}} p_{\boldsymbol{\theta}_i^{(j)}}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right) \log \left[\frac{p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right)} \right] d\boldsymbol{\tau}^{(j)} \right.$$
$$\left. + \frac{\rho}{2} \sum_{l \in \mathbb{P}_i} \left|\left| \tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \tilde{\boldsymbol{\theta}}_i^{(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{li}^{(k)} \right|\right|_2^2 + \frac{\rho}{2} \sum_{l \in \mathbb{S}_i} \left|\left| \tilde{\boldsymbol{\theta}}_i^{(j)} - \tilde{\boldsymbol{\theta}}_l^{(k),(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{il}^{(k)} \right|\right|_2^2 \right].$$

4: Each agent updates $\boldsymbol{\lambda}_{li}$ for $l \in \mathbb{P}_i$ as:

$$\boldsymbol{\lambda}_{li}^{(k+1)} = \boldsymbol{\lambda}_{li}^{(k)} - \rho \left(\tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \tilde{\boldsymbol{\theta}}_i^{(k+1),(j)} \right).$$

5: **end for**

nodes with index higher than i, i.e., $\mathbb{S}_i = \{j | e_{ij} \in \mathcal{E}, i < j\}$. Our algorithm to determine $\tilde{\boldsymbol{\theta}}_j^{\star}$ is summarized by the set of instructions in Algorithm 1. Algorithm 1 consists of two steps. First, primal updates are performed by each node in $\mathcal{G}$ (line 3). Second, given the updated primal, the dual variables are then computed (line 4).

Note that in the case of Gaussian policies, which also can be generalized to any policy from the exponential family, the primal updates on line 3 of Algorithm 1 can be computed in closed form (see Appendix B for derivation):

$$\tilde{\boldsymbol{\theta}}_i^{(k+1),(j)} = \left[-\frac{\rho \deg_i}{2} \boldsymbol{I}_{d \times d} + \mathbb{E}_{\boldsymbol{\tau}^{(j)} \sim p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}(\cdot)} \left[\mathfrak{R}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right) \sum_{m=0}^{M^{(j)}-1} \boldsymbol{\Phi}^{\mathsf{T},(j)}(\boldsymbol{x}_m) \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}^{\mathsf{T},(j)}(\boldsymbol{x}_m) \right] \right]^{-1}$$
$$\times \left[\mathbb{E}_{\boldsymbol{\tau}^{(j)} \sim p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}(\cdot)} \left[\mathfrak{R}^{(j)} \left(\boldsymbol{\tau}^{(j)}\right) \sum_{m=0}^{M^{(j)}-1} \boldsymbol{\Phi}^{\mathsf{T},(j)}(\boldsymbol{x}_m) \boldsymbol{\Sigma}^{-1} \boldsymbol{u}_m^{(j)} \right] \right.$$
$$\left. - \frac{\rho}{2} \left(\sum_{l \in \mathbb{S}_i} \left[\tilde{\boldsymbol{\theta}}_l^{(k),(j)} + \frac{1}{\rho} \boldsymbol{\lambda}_{il}^{(k)} \right] + \sum_{l \in \mathbb{P}_i} \left[\tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{li}^{(k)} \right] \right) \right].$$

where $\deg_i$ is the degree of processing unit i, i.e., set of neighbors, and $\Phi(\cdot)$ is a

feature map of the system's state space. Having the problem solved in computing the linearization operating point, the second step needed for scaling lifelong policy search is handling the inefficiency in updating the shared repository. We detail the solution to this step in the next section.

Theoretical Guarantees: We now show the theorem derives an improvement to a *linear* convergence rate for our proposed algorithm.

Theorem 1. *The sequence* $\left\{\boldsymbol{\theta}_{:}^{(k),(j)}, \boldsymbol{\lambda}^{(k)}\right\}_{k\geq 0}$ *consists of*

$$\boldsymbol{\theta}_{:}^{(k),(j)} = \left[\tilde{\boldsymbol{\theta}}_1^{\mathsf{T},(j)}, \ldots, \tilde{\boldsymbol{\theta}}_{|\mathcal{V}|}^{\mathsf{T},(j)}\right]^{\mathsf{T}} \text{ and } \boldsymbol{\lambda}^{(k)} = \left[\boldsymbol{\lambda}_1^{(k),\mathsf{T}}, \ldots, \boldsymbol{\lambda}_{|\mathcal{E}|}^{(k),\mathsf{T}}\right]^{\mathsf{T}},$$

where $\boldsymbol{\lambda}_i^{(k)} \in \mathbb{R}^d$ *is a dual vector. Following Algorithm 1, the sequence converges linearly with a rate given by* $\mathcal{O}\left(1/k\right)$.

(The interested reader can find the proof in Appendix A.)

5.2. Scaling the Model Update

The lifelong policy search problem is framed as the solution to the optimization problem in Equation (2). It is easy to see that such a problem is not convex in both the shared repository, $\boldsymbol{L}$, and the co-efficient vectors, $\boldsymbol{s}_j$, simultaneously. Holding one variable fixed, however, leads to a convex problem in the other. Consequently, following such an alternating procedure, one can solve a sequence of convex optimization problems iteratively. The computations over the $\boldsymbol{s}_j$ vectors are relatively simple to scale. The scalability problem arises when trying to update the latent repository $\boldsymbol{L}$ due to its dependency on all tasks observed thus far. To scale the computation of the shared repository $\boldsymbol{L}$, we follow a similar strategy to that of the previous section. Holding $\boldsymbol{s}_j$ fixed, computing $\boldsymbol{L}$, then we need to solve:

$$\min_{\boldsymbol{L}} \frac{1}{t}\sum_{j=1}^{t} \left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \boldsymbol{L}\boldsymbol{s}_j\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2||\boldsymbol{L}||_{\mathsf{F}}^2.$$

First, we introduce the $vec(\boldsymbol{Z})$ notation, which vectorizes a matrix $\boldsymbol{Z} \in \mathbb{R}^{d\times p}$ to a vector of size $\mathbb{R}^{dp}$, i.e.,

$$vec(\boldsymbol{Z}) = [a_{1,1}, \ldots, a_{d,1}, a_{1,2}, \ldots, a_{d,2}, \ldots, a_{d,1}, \ldots, a_{d,p}]^{\mathsf{T}},$$

where $a_{i,j}$ denotes the (i,j)-th element of matrix $\boldsymbol{Z}$. Consequently, we can write $vec(\boldsymbol{L}\boldsymbol{s}_j) = \left(\boldsymbol{s}_j^{\mathsf{T}} \otimes \boldsymbol{I}_{d\times d}\right) vec(\boldsymbol{L})$, with $\otimes$ being the Kronecker product and $\boldsymbol{I}_{d\times d}$ a d-dimensional identity matrix. Hence, we can rewrite the optimization problem in the following equivalent form:

$$\min_{vec(\boldsymbol{L})} \frac{1}{t}\sum_{j=1}^{t} \left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec(\boldsymbol{L})\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 ||vec(\boldsymbol{L})||_2^2,$$

where $||\boldsymbol{L}||_{\mathsf{F}}^2 = ||vec(\boldsymbol{L})||_2^2$. Splitting this equation across $\mathcal{G}$ leads us to the following:

$$\min_{vec(\boldsymbol{L}_1):vec\left(\boldsymbol{L}_{|\mathcal{V}|}\right)} \sum_{i=1}^{|\mathcal{V}|}\left(\frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right)\right)$$
$$\text{s.t.} \quad vec\left(\boldsymbol{L}_l\right) = vec\left(\boldsymbol{L}_i\right) \text{ for all } e_{li} \in \mathcal{E} \text{ and } l < i,$$

where $vec\left(\boldsymbol{L}_i\right)$ is the chunk of the shared repository to be learned using processing unit i and t_i is the total number of tasks assigned to that processing unit. Similarly, we can rewrite the constraints compactly in the form of $\tilde{\boldsymbol{A}}\bar{vec}(\boldsymbol{L}) = \boldsymbol{0}$, where $\bar{vec}(\boldsymbol{L}) = \left[vec\left(\boldsymbol{L}_1\right)^{\mathsf{T}}, \ldots, vec\left(\boldsymbol{L}_{|\mathcal{V}|}\right)^{\mathsf{T}}\right]^{\mathsf{T}}$. Thus, we can write the augmented Lagrangian as:

$$\mathcal{L}_{\text{Aug}_{\boldsymbol{L}}}\left(\bar{vec}(\boldsymbol{L}), \boldsymbol{\lambda}_{\boldsymbol{L}}\right) = \sum_{i=1}^{|\mathcal{V}|}\left(\frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right)\right)$$
$$- \boldsymbol{\lambda}_{\boldsymbol{L}}^{\mathsf{T}}\tilde{\boldsymbol{A}}\bar{vec}(\boldsymbol{L}) + \frac{\rho}{2}\left|\left|\tilde{\boldsymbol{A}}\bar{vec}(\boldsymbol{L})\right|\right|_2^2,$$

where $\rho > 0$ is the penalty parameter and $\boldsymbol{\lambda}_{\boldsymbol{L}}$ is a vector of Lagrange multipliers. At this stage, we can follow a similar strategy to that of the previous section to determine the optimal solution $\bar{vec}(\boldsymbol{L})^{\star}$. To update the primal variables, each processing unit in $\mathcal{G}$ solves the following problem:

$$vec\left(\boldsymbol{L}_i^{(k+1)}\right) = \arg\min_{vec(\boldsymbol{L}_i)}\left[\frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right)\right.$$
$$\left. + \frac{\rho}{2}\sum_{l\in\mathbb{P}_i}\left|\left|vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_i\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right|\right|_2^2 + \frac{\rho}{2}\sum_{l\in\mathbb{S}_i}\left|\left|vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_l^{(k)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right|\right|_2^2\right], \tag{4}$$

Algorithm 2 Scalable Operating Point Computation for $\boldsymbol{L}$

1: Initialize $vec\left(\boldsymbol{L}_i^{(0)}\right)$, $\forall i \in \{1,\ldots,|\mathcal{V}|\}$, and set $\rho > 0$.

2: **for** $k = 0,\ldots,K$ **do**

3: Each agent i updates, $vec\left(\boldsymbol{L}_i^{(k+1)}\right)$ in a sequential order from $i = 1,\ldots,|\mathcal{V}|$ using:

$$
\begin{aligned}
vec\left(\boldsymbol{L}_i^{(k+1)}\right) = \arg\min_{vec(\boldsymbol{L}_i)} \Bigg[& \frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right) \\
& + \frac{\rho}{2}\sum_{l\in\mathbb{P}_i}\left|\left|vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_i\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right|\right|_2^2 \\
& + \frac{\rho}{2}\sum_{l\in\mathbb{S}_i}\left|\left|vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_l^{(k)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},il}^{(k)}\right|\right|_2^2\Bigg],
\end{aligned}
$$

4: Each agent updates $\boldsymbol{\lambda}_{li}$ for

$$
\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k+1)} = \boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)} - \rho\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right).
$$

5: **end for**

where k is the iteration number. Given an updated primal, the dual variables, $\boldsymbol{L}$, are then computed according to:

$$
\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k+1)} = \boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)} - \rho\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right). \tag{5}
$$

Equation (5) is similar to step 4 in Algorithm 1. We provide the pseudocode in Algorithm 2. Note that the update rule can be acquired in a closed form the Gaussian policy assumption, leading to the following: (See Appendix B for details):

$$
\begin{aligned}
vec\left(\boldsymbol{L}_i^{(k+1)}\right) = & \left[\frac{1}{t_i}\sum_{j=1}^{t_i}\boldsymbol{\Gamma}_j^{\intercal}\boldsymbol{H}_j\boldsymbol{\Gamma}_j + \frac{\rho \mathrm{deg}_i}{2}\boldsymbol{I}_{dk\times dk}\right]^{-1}\left(\frac{1}{t_i}\sum_{j=1}^{t_i}\boldsymbol{\Gamma}_j\boldsymbol{H}_j\tilde{\boldsymbol{\theta}}_j^{\star}\right. \\
& \left. + \frac{\rho}{2}\left(\sum_{l\in\mathbb{S}_i}\left[vec\left(\boldsymbol{L}_l^{(k+1)}\right) + \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right] + \sum_{l\in\mathbb{P}_i}\left[vec\left(\boldsymbol{L}_l^{(k+1)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},il}^{(k)}\right]\right)\right),
\end{aligned}
$$

with $\boldsymbol{\Gamma}_j = \boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}$. Given the solution to the shared repository, the task-specific coefficients are performed locally using a LASSO [26] step for each of

the tasks t_i.

Theoretical Guarantees: Our next theorem derives a *linear* convergence rate for our proposed algorithm.

Theorem 2. *The sequence* $\left\{ v\bar{e}c\left(\boldsymbol{L}^{(k)}\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} \right\}_{k\geq 0}$ *consists of*

$$v\bar{e}c\left(\boldsymbol{L}^{(k)}\right) = \left[vec\left(\boldsymbol{L}_1\right)^{(k),\mathsf{T}}, \ldots, vec\left(\boldsymbol{L}_{|\mathcal{V}|}\right)^{(k),\mathsf{T}} \right]^{\mathsf{T}} \text{ and } \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} = \left[\boldsymbol{\lambda}_{\boldsymbol{L},1}^{(k),\mathsf{T}}, \ldots, \boldsymbol{\lambda}_{\boldsymbol{L},|\mathcal{E}|}^{(k),\mathsf{T}} \right]^{\mathsf{T}},$$

where $\boldsymbol{\lambda}_{\boldsymbol{L},i}^{(k)} \in \mathbb{R}^{dp}$ *is a dual vector. Following Equations* (4) *and 5, the sequence converges linearly with a rate given by* $\mathcal{O}\left(1/k\right)$.

(The interested reader can find the proof in Appendix A.)

Theorems 1 and 2 show that our technique improves over PG-ELLA in both the policy gradient update step and as well as in the computations of the shared repository. This improvement is significant as it shows linear converges in both steps.

6. Experimental Results

Next, we present an empirical validation on five benchmark dynamical systems introduced in [4, 23] .

Cart Pole: The cart pole (CP) system is controlled by the cart's mass m_c in kg, the pole's mass m_p in kg and the pole's length l in meters. The state is given by the cart's position and velocity v, as well as the pole's angle θ and angular velocity $\dot{\theta}$. The goal is to control the pole in an upright position.

Double Inverted Pendulum: The double inverted pendulum (DIP) is an extension of the cart pole system. It has one cart with mass m_0 in kg and two poles in which the corresponding lengths are l_1 and l_2 in meters. We assume the poles have no mass and there are two masses m_1 and m_2 in kg on the top of each pole. The state consists of the cart's position x_1 and velocity v_1, the lower pole's angle θ_1 and angular velocity $\dot{\theta}_1$, as well as the upper pole's θ_2 and angular velocity $\dot{\theta}_2$. The goal is also to learn a policy to control the two poles in a specific state.

Helicopter: This linearized model of a CH-47 (HC) tandem-rotor helicopter assumes horizontal motion at 40 knots. It characterized by two matrices $A \in \mathbb{R}^{4\times4}$ and $B \in \mathbb{R}^{4\times2}$. The main goal is to stabilize the helicopter by controlling the collective and differential rotor thrust.

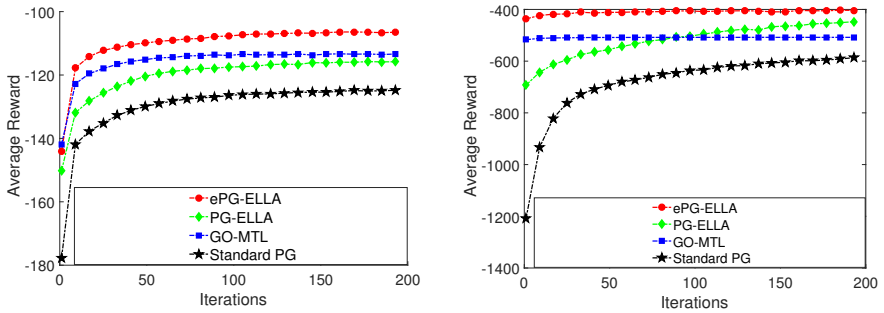
Simple Mass: The simple mass (SM) system is characterized by the spring constant k in N/m, the damping constant d in Ns/m and the mass m in kg. The system's state is given by the position x and the velocity, v, of the mass. The goal is to train a policy for guiding the mass to a specific state.

Double Mass: The double mass (DM) is an extension of the simple mass system. It has two masses m_1, m_2 in kg and two springs in which the corresponding springs constants are k_1 and k_2 in N/m, as well as the damping constant d_1 and d_2 in Ns/m. The state consists of the big mass' position x_1 and velocity v_1, as well as the small mass' position x_2 and velocity v_2. The goal is also to learn a policy to control the two mass in a specific state.

Table 1: Parameter ranges used in the experiments

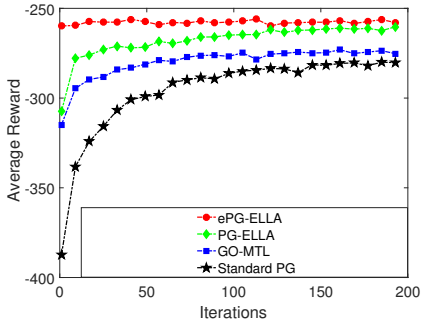
CP	DIP	HC
$m_c, m_p \in [0, 1]$ $l \in [0.2, 0.8]$	$m_0 \in [1.5, 3.5]$ $m_1, m_2 \in [0.055, 0.1]$ $l_1 \in [0.4, 0.8], l_2 \in [0.5, 0.9]$	A with rand entry $A(i,j) \in [-32, 3]$ B with rand entry $A(i,j) \in [-9, 1]$
SM	DM	
$m \in [3, 5]$ $k \in [1, 7]$	$m_1 \in [1, 7], m_2 \in [1, 10]$ $k_1 \in [1, 5], k_2 \in [1, 7]$ $d_1, d_2 \in [0.01, 0.1]$	

We generated 50 tasks for each domain by varying the dynamical parameters of each of the domains (for 250 in total). For reproducibility, we summarize these ranges in Table 1. The reward was given by $-\sqrt{\|\boldsymbol{x}_m - \dot{\boldsymbol{x}}\|_2^2} - \sqrt{\boldsymbol{u}_m^\intercal \boldsymbol{u}_m}$, where $\dot{\boldsymbol{x}}$ was the goal state, $\boldsymbol{x}_m$ is the current state and $\boldsymbol{u}_m$ is the action that the agent has chosen. We run each task for a total of 200 iterations. At each iteration, the learner observed a task through 50 trajectories of 150 steps and updated

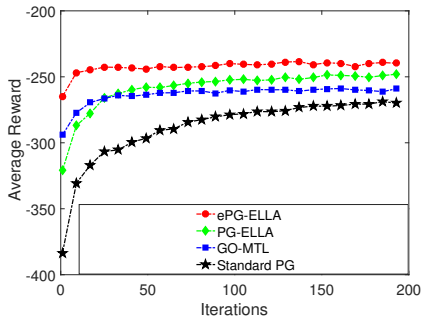


(a) SM

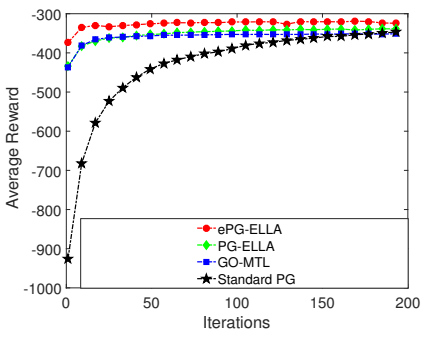
(b) DM



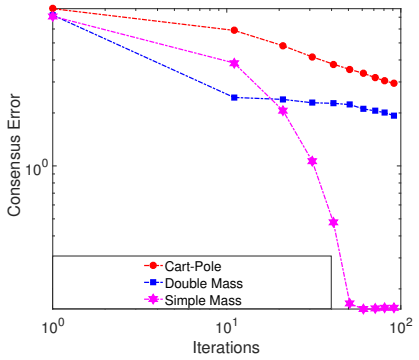
(c) CP



(d) DCP



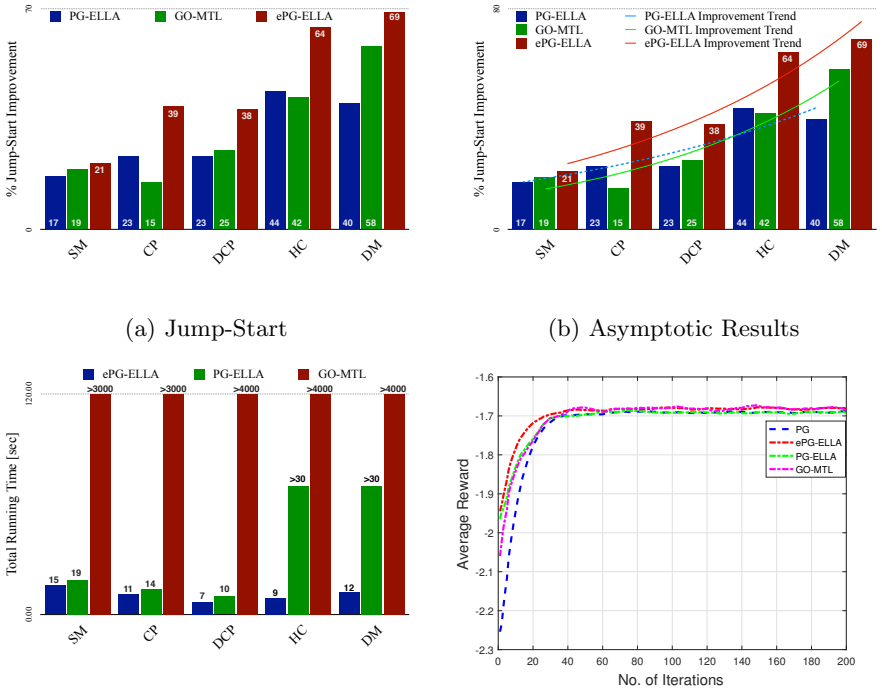
(e) HC



(f) Agreement

Figure 1: Figures (a)-(e) report average reward versus iterations and demonstrate that ePG-ELLA is capable of outperforming other methods. Figure (f) shows the agreement error across processors on 3 sample systems.

performed algorithmic updates. We used eNAC [27], a standard PG algorithm, as the base learner. We compare our method (ePG-ELLA) to standard PG, PG-



(a) Jump-Start

(b) Asymptotic Results

(c) Time to Convergence

(d) Novel Domain

Figure 2: Figures (a) and (b) show that ePG-ELLA is capable of outperforming others in both the jump-start and asymptotic performance. Figure (c) shows ePG-ELLA takes less time to optimized the objective function. Finally, Figure (d) demonstrates the generalization capability of our method in terms of average reward on unobserved tasks in HC domain.

ELLA [9], and GO-MTL [22]. To distribute our computations, we made use of MATLAB's parallel pool running on 10 nodes. For distributive computation in ePG-ELLA, we equally assigned 50 tasks to 10 agents, that is, each agent learns 5 tasks in parallel. The edges between agents were generated randomly without overlapping, and the agents are able to exchange information via the graph network. To make sure every node in the graph has at least one predecessor and at least one successor, we connect node 1 to node 2, node 2 to node 3, ..., node 9 and 10. We then randomly assigned 20 edges to all nodes to ensure a connected graph.

Figures 1a—1e report the average reward performance on the different bench-

mark in terms of the number of iterations. In all these systems, our method is capable of outperforming the others in terms of total reward. Figure 1f shows that our method is capable of acquiring feasible solutions by reporting the consensus error on 3 different systems. Figures 2a and 2b demonstrate that our algorithm is capable of outperforming PG-ELLA and GO-MTL in terms of jump-starts and asymptotic performance. Figure 2c demonstrates that our method takes less time to optimize the objective function in terms of time consumption.

Transfer provides significant advantages when the lifelong learning agent faces a novel task domain. To evaluate this, we chose the most complex of the task domains (HC) and trained the lifelong learner by unseen 49 tasks to yield an effectively shared knowledge base for each of the algorithms. Then, we evaluated the algorithms' ability to learn a new (unseen) task from the helicopter domain, comparing the benefits of ePG-ELLA transfer from $\boldsymbol{L}_{\text{ePGELLA}}$ with PG-ELLA (from $\boldsymbol{L}_{\text{PG-ELLA}}$) , GO-MTL (from $\boldsymbol{L}_{\text{GO-MTL}}$) and PG. Figure 2d depicts the result of learning on a novel domain, averaged over HC tasks, showing that ePG-ELLA converges fastest in this scenario in terms of average reward.

Negative transfer may occur during training with a small probability. However, we run the experiments over 200 iterations and take the average results to smooth the results. Any cases where transfer was negative were statistically overwhelemed by the positive results in our setting.

7. Conclusion and Future Work

We proposed a scalable and efficient lifelong learner for policy gradient methods. Our approach achieves linear convergence rates in both acquiring linearizing operating points and updating shared knowledge bases. We have shown that our new technique can outperform state-of-the-art methods on a variety of benchmark control systems. Future work will include targeting the more-general lifelong learning setting in different domains and running on real world applications such as robotics.

8. Acknowledgements

We thank the anonymous reviewers for their useful suggestions and comments. This research has taken place in part at the Intelligent Robot Learning (IRL) Lab, Washington State University. IRL's support includes NASA NNX16CD07C, NSF IIS-1149917, NSF IIS-1643614, and USDA 2014-67021-22174.

Appendix A. Convergence and Convergence Rate

In this section, we will show the proof of Theorem 2. Theorem 1 can be proved in a similar way. To show that the sequence $\left\{ v\bar{e}c\left(\boldsymbol{L}^{(k)}\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} \right\}_{k\geq 0}$ converges with rate $\mathcal{O}(\frac{1}{k})$, let the function

$$\begin{aligned}\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}\right)\right) &= \sum_{i=1}^{|\mathcal{V}|} f_i\left(vec\left(\boldsymbol{L}_i\right)\right) \\ &= \sum_{i=1}^{|\mathcal{V}|}\left(\frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right)\right)\end{aligned}$$

and the Lagrangian function

$$\begin{aligned}\mathcal{L}_{\text{Aug}_{\boldsymbol{L}}}\left(v\bar{e}c(\boldsymbol{L}), \boldsymbol{\lambda}_{\boldsymbol{L}}\right) = \sum_{i=1}^{|\mathcal{V}|}\left(\frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right)\right) \\ - \boldsymbol{\lambda}_{\boldsymbol{L}}^{\mathsf{T}}\tilde{\boldsymbol{A}}v\bar{e}c(\boldsymbol{L}) + \frac{\rho}{2}\left|\left|\tilde{\boldsymbol{A}}v\bar{e}c(\boldsymbol{L})\right|\right|_2^2,\end{aligned} \tag{A.1}$$

where

$$v\bar{e}c\left(\boldsymbol{L}^{(k)}\right) = \left[vec\left(\boldsymbol{L}_1^{(k)}\right)^{\mathsf{T}}, \ldots, vec\left(\boldsymbol{L}_{|\mathcal{V}|}^{(k)}\right)^{\mathsf{T}}\right]^{\mathsf{T}} \text{ and } \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} = \left[\boldsymbol{\lambda}_{\boldsymbol{L},1}^{(k),\mathsf{T}}, \ldots, \boldsymbol{\lambda}_{\boldsymbol{L},|\mathcal{E}|}^{(k),\mathsf{T}}\right]^{\mathsf{T}},$$

with $\boldsymbol{\lambda}_{\boldsymbol{L},i}^{(k)} \in \mathbb{R}^{dk}$ being a dual vector. It is easy to verify that each cost function

$$f_i\left(vec\left(\boldsymbol{L}_i\right)\right) = \frac{1}{t_i}\sum_{j=1}^{t_i}\left(\left|\left|\tilde{\boldsymbol{\theta}}_j^{\star} - \left(\boldsymbol{s}_j \otimes \boldsymbol{I}_{d\times d}\right) vec\left(\boldsymbol{L}_i\right)\right|\right|_{\boldsymbol{H}_j}^2 + \mu_2 \left|\left|vec\left(\boldsymbol{L}_i\right)\right|\right|_2^2\right)$$

is convex. Also, we needs following assumptions

Assumption 1 (Existence of a Saddle Point)**.** *The Lagrangian function* $\mathcal{L}_{Aug_{\boldsymbol{L}}}$ *has a saddle point, that is, there is a solution* $(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}^*)$ *such that*

$$\mathcal{L}_{Aug_{\boldsymbol{L}}}(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}) \leq \mathcal{L}_{Aug_{\boldsymbol{L}}}(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}^*) \leq \mathcal{L}_{Aug_{\boldsymbol{L}}}(v\bar{e}c(\boldsymbol{L}), \boldsymbol{\lambda}_{\boldsymbol{L}}^*) \quad \text{(A.2)}$$

for all $v\bar{e}c(\boldsymbol{L}) \in \mathbb{R}^{dp|\mathcal{V}|}$ *and* $\boldsymbol{\lambda}_{\boldsymbol{L}} \in \mathbb{R}^{dp}$*.*

Assumption 2. *The penalty term* $\left\|\tilde{\boldsymbol{A}}v\bar{e}c(\boldsymbol{L})\right\|_2^2$ *is bounded by a constant* $\gamma > 0$*.*

We extended Wei and Ozdaglar's results to multiple dimension [25]. Before we provide the proofs, we need following lemmas:

Lemma 1. *The sequence* $\left\{v\bar{e}c\left(\boldsymbol{L}^{(k)}\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)}\right\}_{k\geq 0}$ *consists of*

$$v\bar{e}c\left(\boldsymbol{L}^{(k)}\right) = \left[vec\left(\boldsymbol{L}_1^{(k)}\right)^{\mathsf{T}}, \ldots, vec\left(\boldsymbol{L}_{|\mathcal{V}|}^{(k)}\right)^{\mathsf{T}}\right]^{\mathsf{T}} \text{ and } \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} = \left[\boldsymbol{\lambda}_{\boldsymbol{L},1}^{(k),\mathsf{T}}, \ldots, \boldsymbol{\lambda}_{\boldsymbol{L},|\mathcal{E}|}^{(k),\mathsf{T}}\right]^{\mathsf{T}},$$

where $\boldsymbol{\lambda}_{\boldsymbol{L},i}^{(k)} \in \mathbb{R}^{dp}$ *is a dual vector. Then, for all* $k \geq 0$*, we have following relation,*

$$\begin{aligned}
&\mathcal{F}(v\bar{e}c(\boldsymbol{L})) - \mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right) + \left(v\bar{e}c\left(\boldsymbol{L}^{(}k\right) - v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right)^{\mathsf{T}} \\
&\cdot \left(-\tilde{\boldsymbol{A}}^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)} - \rho\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(k)}\right)\right) + \rho\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) v\bar{e}c\left(\boldsymbol{L}^{(k)}\right)\right)\right) \geq 0
\end{aligned}$$

for all $v\bar{e}c(\boldsymbol{L}) \in \mathbb{R}^{dp|\mathcal{V}|}$*, where matrix* $\tilde{\boldsymbol{A}}$ *is the general edge-node incident matrix and* $\tilde{\boldsymbol{B}} = \min\left\{\boldsymbol{0}, \tilde{\boldsymbol{A}}\right\}$*, and the* min *is taken element-wise.*

Proof. Let

$$\begin{aligned}
g_i^{(k)}(vec(\boldsymbol{L}_i)) = \frac{\rho}{2}\sum_{l\in\mathbb{P}_i}\left\|vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec(\boldsymbol{L}_i) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right\|_2^2 \\
+ \frac{\rho}{2}\sum_{l\in\mathbb{S}_i}\left\|vec(\boldsymbol{L}_i) - vec\left(\boldsymbol{L}_l^{(k)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},il}^{(k)}\right\|_2^2
\end{aligned}$$

From equation 4, the optimal value of function $g_i^k + f_i$ is $vec\left(\boldsymbol{L}_i^{(k+1)}\right)$, which implies that there exits a subgradient $sg\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) \in \partial f_i(vec(\boldsymbol{L}_i))$ such that $sg\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \nabla g_i^{(k)}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) = 0$, and we obtain

$$\left[vec(\boldsymbol{L}_i) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}}\left(sg\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \nabla g_i^{(k)}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right)\right) = 0, \quad \text{(A.3)}$$

for all $vec(\boldsymbol{L}_i) \in \mathbb{R}^{dk}$. Due to the definition of subgradient, we have

$$f_i\left(vec\left(\boldsymbol{L}_i\right)\right) \geq f_i\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}} \cdot sg\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) \tag{A.4}$$

Combining Equation (A.3) and A.4 yield:

$$f_i\left(vec\left(\boldsymbol{L}_i\right)\right) - f_i\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}} \cdot \nabla g_i^{(k)}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) \geq 0. \tag{A.5}$$

The gradient of $g_i^{(k)}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right)$ is

$$\begin{aligned}\nabla g_i^{(k)}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) = &-\rho \sum_{l \in \mathbb{P}_i}\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right) \\ &+ \rho \sum_{l \in \mathbb{S}_i}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right) - vec\left(\boldsymbol{L}_l^{(k)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},il}^{(k)}\right)\end{aligned}$$

Then, we substitute the gradient in Equation (A.5) with above equation, we have

$$\begin{aligned}&f_i\left(vec\left(\boldsymbol{L}_i\right)\right) - f_i\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}} \\ &\quad \cdot \left(-\rho \sum_{l \in \mathbb{P}_i}\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}\right)\right. \\ &\quad \left. + \rho \sum_{l \in \mathbb{S}_i}\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right) - vec\left(\boldsymbol{L}_l^{(k)}\right) - \frac{1}{\rho}\boldsymbol{\lambda}_{\boldsymbol{L},il}^{(k)}\right)\right) \geq 0.\end{aligned}$$

Since the dual variables $\boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k)}$ have the relation in Equation (5), then,

$$\begin{aligned}&f_i\left(vec\left(\boldsymbol{L}_i\right)\right) - f_i\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}} \\ &\quad \cdot \left(\sum_{l \in \mathbb{P}_i} \boldsymbol{\lambda}_{\boldsymbol{L},li}^{(k+1)} + \sum_{l \in \mathbb{S}_i} -\boldsymbol{\lambda}_{\boldsymbol{L},il}^{(k+1)} + \sum_{l \in \mathbb{S}_i} \rho\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_l^{(k)}\right)\right)\right) \geq 0.\end{aligned}$$

Then, by the definition of matrix $\tilde{\boldsymbol{A}}$, we can rewrite above equation as

$$\begin{aligned}&f_i\left(vec\left(\boldsymbol{L}_i\right)\right) - f_i\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}} \\ &\quad \cdot \left(-\tilde{\boldsymbol{A}}(\cdot, i)^{\mathsf{T}} \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)} + \sum_{l \in \mathbb{S}_i} \rho\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_l^{(k)}\right)\right)\right) \geq 0.\end{aligned}$$

where $\tilde{\boldsymbol{A}}(\cdot, i)$ denotes the ith column of matrix $\tilde{\boldsymbol{A}}$. We sum the above equation

from $k=1$ to $|\mathcal{V}|$,

$$\sum_{i=1}^{|\mathcal{V}|} f_i\left(vec\left(\boldsymbol{L}_i\right)\right) - \sum_{i=1}^{|\mathcal{V}|} f_i\left(vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right) + \sum_{i=1}^{|\mathcal{V}|}\left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}} \cdot\left(-\tilde{\boldsymbol{A}}(\cdot,i)^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)} + \sum_{l\in\mathbb{S}_i}\rho\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_l^{(k)}\right)\right)\right) \geq 0. \tag{A.6}$$

By the definition of matrices $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$, we obtain

$$-\sum_{i=1}^{|\mathcal{V}|}\left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}}\tilde{\boldsymbol{A}}(\cdot,i)^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)} = -\left[v\bar{e}c\left(\boldsymbol{L}\right) - v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right]^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)}$$

and

$$\sum_{i=1}^{|\mathcal{V}|}\left[vec\left(\boldsymbol{L}_i\right) - vec\left(\boldsymbol{L}_i^{(k+1)}\right)\right]^{\mathsf{T}}\sum_{l\in\mathbb{S}_i}\rho\left(vec\left(\boldsymbol{L}_l^{(k+1)}\right) - vec\left(\boldsymbol{L}_l^{(k)}\right)\right) = \rho\left[v\bar{e}c\left(\boldsymbol{L}\right)v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right]^{\mathsf{T}}\left(-\tilde{\boldsymbol{A}}+\tilde{\boldsymbol{B}}\right)^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right).$$

The result follows by plugging preceding two equations into Equation (A.6). □

Lemma 2. *The sequence* $\left\{v\bar{e}c\left(\boldsymbol{L}^{(k)}\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)}\right\}_{k\geq 0}$ *consists of*

$$v\bar{e}c\left(\boldsymbol{L}^{(k)}\right) = \left[vec\left(\boldsymbol{L}_1^{(k)}\right)^{\mathsf{T}}, \ldots, vec\left(\boldsymbol{L}_{|\mathcal{V}|}^{(k)}\right)^{\mathsf{T}}\right]^{\mathsf{T}} \text{ and } \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} = \left[\boldsymbol{\lambda}_{\boldsymbol{L},1}^{(k),\mathsf{T}}, \ldots, \boldsymbol{\lambda}_{\boldsymbol{L},|\mathcal{E}|}^{(k),\mathsf{T}}\right]^{\mathsf{T}},$$

where $\boldsymbol{\lambda}_{\boldsymbol{L},i}^{(k)} \in \mathbb{R}^{dp}$ *is a dual vector. Assume that* $\left(v\bar{e}c\left(\boldsymbol{L}^*\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right)$ *is the solution of problem A.1. Let matrix* $\tilde{\boldsymbol{A}}$ *is the general edge-node incident matrix and* $\tilde{\boldsymbol{B}} = \min\left\{\boldsymbol{0}, \tilde{\boldsymbol{A}}\right\}$*, and the* min *is taken element-wise. Then, we have following relation:*

$$\begin{aligned} v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}(\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)} - \boldsymbol{\lambda}_{\boldsymbol{L}}^*) &+ \rho(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(k)}\right)\right) \\ &+ \rho\left(v\bar{e}c\left(\boldsymbol{L}^*\right) - v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right)^{\mathsf{T}}\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(k)}\right)\right) \\ &= \frac{1}{2\rho}\left(\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} - \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right\|^2 - \left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{(k+1)} - \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right\|^2\right) \\ &+ \frac{\rho}{2}\left(\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k)}\right) - v\bar{e}c\left(\boldsymbol{L}^*\right)\right)\right\|^2 - \left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^*\right)\right)\right\|\right) \\ &- \frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(k)}\right)\right) - \tilde{\boldsymbol{A}}v\bar{e}c\left(\boldsymbol{L}^{(k+1)}\right)\right\|^2 \end{aligned}$$

Proof. See the proofs of Lemma 4.2 in [25] □

Lemma 3. *Let* $(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}^*)$ *be a saddle point of Equation* (A.1)*. Then we have*

$$\tilde{\boldsymbol{A}} v\bar{e}c(\boldsymbol{L}^*) = 0.$$

Proof. The result follows by the Assumption 1. □

With aforementioned Lemmas, we are able to show the main theoretical result in the paper.

Proof of Theorem 1. To show the convergence with rate $\mathcal{O}(\frac{1}{k})$, we need an additional variable $y^k = \frac{1}{k}\sum_{s=0}^{k-1} v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)$ which is the average of $v\bar{e}c\left(\boldsymbol{L}^k\right)$ until time k. Then, We need to bound

$$\mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(y^k, \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right) - \mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right)$$

First, $\mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(y^k, \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right) - \mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right) \geq 0$ since $(v\bar{e}c(\boldsymbol{L}^*), \boldsymbol{\lambda}_{\boldsymbol{L}}^*)$ is the saddle point of Equation (A.1). Then, we derive the upper bound of above equation. By setting $v\bar{e}c(\boldsymbol{L}) = v\bar{e}c(\boldsymbol{L}^*)$ in Lemma 1, we have following relation,

$$\begin{aligned}
&\mathcal{F}\left(v\bar{e}c(\boldsymbol{L}^*)\right) - \mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right) + \left(v\bar{e}c(\boldsymbol{L}^*) - v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right)^{\mathsf{T}} \\
&\cdot \left(-\tilde{\boldsymbol{A}}^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^{(s+1)} - \rho\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)\right) + \rho\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s_1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)\right)\right) \geq 0
\end{aligned}$$

We have $\tilde{\boldsymbol{A}} v\bar{e}c(\boldsymbol{L}^*) = 0$ from Lemma 3, so $v\bar{e}c(\boldsymbol{L}^*)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}} = 0$. Thence, the above equation can be reduced to

$$\begin{aligned}
&\mathcal{F}\left(v\bar{e}c(\boldsymbol{L}^*)\right) - \mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right) + v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^{(s+1)} \\
&+ \left(\rho v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{B}} + v\bar{e}c\left(\boldsymbol{L}^{(s)}\right) + \rho\left(v\bar{e}c(\boldsymbol{L}^*) - v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right)\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\right) \\
&\cdot \left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)\right) \geq 0
\end{aligned}$$

By adding $(\boldsymbol{\lambda}_{\boldsymbol{L}}^*)^{\mathsf{T}}\tilde{\boldsymbol{A}} v\bar{e}c\left(\boldsymbol{L}^{(s)}\right) - (\boldsymbol{\lambda}_{\boldsymbol{L}}^*)^{\mathsf{T}}\tilde{\boldsymbol{A}} v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)$, we obtain

$$\begin{aligned}
&\mathcal{F}\left(v\bar{e}c(\boldsymbol{L}^*)\right) - \mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right) + v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\boldsymbol{\lambda}_{\boldsymbol{L}}^* \\
&+ v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}(\boldsymbol{\lambda}_{\boldsymbol{L}}^{(s+1)} - \boldsymbol{\lambda}_{\boldsymbol{L}}^*) + \rho v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)\right) \\
&+ \rho\left(v\bar{e}c(\boldsymbol{L}^*) - v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right)^{\mathsf{T}}\tilde{\boldsymbol{B}}^{\mathsf{T}}\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right) - v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)\right) \geq 0
\end{aligned}$$

Using Lemma 2 and Lemma 3, it yields,

$$\begin{aligned}
&\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)-\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right)+\left(\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\\
&+\frac{1}{2\rho}\left(\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{(s)}-\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right\|^{2}-\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{(s+1)}-\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right\|^{2}\right)\\
&+\frac{\rho}{2}\left(\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)-v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)\right\|^{2}-\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)-v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)\right\|\right)\\
&-\frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}(vec(\boldsymbol{L}^{(s+1)})-vec(\boldsymbol{L}^{(s)}))-\tilde{\boldsymbol{A}}vec(\boldsymbol{L}^{(s+1)})\right\|^{2}\geq 0
\end{aligned}$$

Then, we sum aforementioned inequality from $s=0,\dots k-1$ and after telescoping cancellation, we have

$$\begin{aligned}
&k\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)-\sum_{s=0}^{k-1}\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right)+\left(\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}\sum_{s=0}^{k-1}v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\\
&+\frac{1}{2\rho}\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{0}-\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right\|^{2}+\frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{0}\right)-v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)\right\|^{2}\\
&\geq\frac{1}{2\rho}\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{k}-\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right\|^{2}+\frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{l}\right)-v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)\right\|^{2}\\
&+\sum_{s=0}^{k-1}\frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)-v\bar{e}c\left(\boldsymbol{L}^{(s)}\right)\right)-\tilde{\boldsymbol{A}}v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right\|^{2}\geq 0
\end{aligned}$$

Since $\mathcal{F}$ is a convex function due to the convexity of f_i, we have $\sum_{s=0}^{k-1}\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right)\geq k\mathcal{F}(y^{k})$ and with the definition of $y^{k}=\frac{1}{k}\sum_{s=0}^{k-1}v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)$,

$$\begin{aligned}
&k\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)-k\mathcal{F}(y^{k})+k\left(\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}y^{k}\\
&+\frac{1}{2\rho}\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{0}-\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right\|^{2}+\frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{0}\right)-v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)\right\|^{2}\geq 0
\end{aligned}$$

Taking $-\frac{1}{k}$ on both sides of the above equation and $\left(\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}v\bar{e}c\left(\boldsymbol{L}^{*}\right)=0$ by Lemma 3, we obtain

$$\begin{aligned}
&\mathcal{F}(y^{k})-\left(\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}y^{k}-\mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)+\left(\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right)^{\mathsf{T}}\tilde{\boldsymbol{A}}^{\mathsf{T}}v\bar{e}c\left(\boldsymbol{L}^{*}\right)\\
&\leq\frac{1}{k}\left(\frac{1}{2\rho}\left\|\boldsymbol{\lambda}_{\boldsymbol{L}}^{0}-\boldsymbol{\lambda}_{\boldsymbol{L}}^{*}\right\|^{2}+\frac{\rho}{2}\left\|\tilde{\boldsymbol{B}}\left(v\bar{e}c\left(\boldsymbol{L}^{0}\right)-v\bar{e}c\left(\boldsymbol{L}^{*}\right)\right)\right\|^{2}\right)
\end{aligned}$$

By Assumption 2 and definition of y^{k},

$$\frac{\rho}{2}\left\|\tilde{\boldsymbol{A}}y^{k}\right\|_{2}^{2}\leq\frac{\rho}{2k}\sum_{s=0}^{k-1}\left\|\tilde{\boldsymbol{A}}v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)\right\|_{2}^{2}\leq\frac{\rho}{2k}k\gamma=\frac{\gamma\rho}{2}\tag{A.7}$$

Then,

$$\mathcal{F}(y^k) - (\boldsymbol{\lambda}_{\boldsymbol{L}}^*)^\mathsf{T} \tilde{\boldsymbol{A}}^\mathsf{T} y^k + \frac{\rho}{2} \left\| \tilde{\boldsymbol{A}} y^k \right\|_2^2 y^k - \mathcal{F}\left(v\bar{e}c\left(\boldsymbol{L}^*\right)\right) + (\boldsymbol{\lambda}_{\boldsymbol{L}}^*)^\mathsf{T} \tilde{\boldsymbol{A}}^\mathsf{T} v\bar{e}c\left(\boldsymbol{L}^*\right) - \frac{\rho}{2} \left\| \tilde{\boldsymbol{A}} v\bar{e}c\left(\boldsymbol{L}^*\right) \right\|_2^2$$
$$\leq \frac{1}{k} \left(\frac{1}{2\rho} \left\| \boldsymbol{\lambda}_{\boldsymbol{L}}^0 - \boldsymbol{\lambda}_{\boldsymbol{L}}^* \right\|^2 + \frac{\rho}{2} \left\| \tilde{\boldsymbol{B}} \left(v\bar{e}c\left(\boldsymbol{L}^0\right) - v\bar{e}c\left(\boldsymbol{L}^*\right) \right) \right\|^2 + \frac{\gamma\rho}{2} \right) \tag{A.8}$$

Combining Equation (A.8) and (A.7), we have

$$\mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(y^k, \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right) - \mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(v\bar{e}c\left(\boldsymbol{L}^*\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right)$$
$$\leq \frac{1}{k} \left(\frac{1}{2\rho} \left\| \boldsymbol{\lambda}_{\boldsymbol{L}}^0 - \boldsymbol{\lambda}_{\boldsymbol{L}}^* \right\|^2 + \frac{\rho}{2} \left\| \tilde{\boldsymbol{B}} \left(v\bar{e}c\left(\boldsymbol{L}^0\right) - v\bar{e}c\left(\boldsymbol{L}^*\right) \right) \right\|^2 + \frac{\gamma\rho}{2} \right)$$

Due to the convexity of function $\mathcal{F}$ and linearity of $(\boldsymbol{\lambda}_{\boldsymbol{L}}^*)^\mathsf{T} \tilde{\boldsymbol{A}}^\mathsf{T} v\bar{e}c\left(\boldsymbol{L}^*\right)$ and $\frac{\rho}{2} \left\| \tilde{\boldsymbol{A}} v\bar{e}c\left(\boldsymbol{L}^*\right) \right\|_2^2$, the function $\mathcal{L}_{\mathrm{Aug}_{\boldsymbol{L}}}\left(v\bar{e}c\left(\boldsymbol{L}\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^*\right)$ is strict convex and has a unique minimizer, i.e., $v\bar{e}c\left(\boldsymbol{L}^*\right)$. Hence if $k \to \inf$, then the sequence $\left\{ v\bar{e}c\left(\boldsymbol{L}^{(k)}\right), \boldsymbol{\lambda}_{\boldsymbol{L}}^{(k)} \right\}_{k\geq 0}$ converges to the optimal minimizer $v\bar{e}c\left(\boldsymbol{L}^*\right)$. In practice, we could use the additional variables $y^k = \frac{1}{k} \sum_{s=0}^{k-1} v\bar{e}c\left(\boldsymbol{L}^{(s+1)}\right)$ to achieve $\mathcal{O}\left(\frac{1}{k}\right)$ convergence rate. □

Appendix B. Update Rules

In this section, we provide the derivation of update rules in Section 5.1. The derivation of Section 5.2 can be obtained by similar methods. Using Leibniz integration rule, the gradient can be computed as:

$$\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}} \left[\int_{\boldsymbol{\tau}^{(j)} \in \mathbb{T}_i^{(j)}} p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \log \left[\frac{p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)} \right] d\boldsymbol{\tau}^{(j)} \right.$$
$$\left. \frac{\rho}{2} \sum_{l \in \mathbb{P}_i} \left\| \tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \tilde{\boldsymbol{\theta}}_i^{(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{li}^{(k)} \right\|_2^2 + \frac{\rho}{2} \sum_{l \in \mathbb{S}_i} \left\| \tilde{\boldsymbol{\theta}}_i^{(j)} - \tilde{\boldsymbol{\theta}}_l^{(k),(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{li}^{(k)} \right\|_2^2 \right]$$
$$= \int_{\boldsymbol{\tau}^{(j)} \in \mathbb{T}_i^{(j)}} \nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}} \underbrace{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \log \left[\frac{p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}{p_{\boldsymbol{\theta}_i^{(j)}}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right) \mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)} \right]}_{f_{\mathrm{RL}_i^{(j)}}} d\boldsymbol{\tau}^{(j)}$$
$$+ \rho \mathrm{deg}_i \tilde{\boldsymbol{\theta}}_i^{(j)} - \rho \left(\sum_{l \in \mathbb{S}_i} \left[\tilde{\boldsymbol{\theta}}_l^{(k),(j)} + \frac{1}{\rho} \boldsymbol{\lambda}_{il}^{(k)} \right] + \sum_{l \in \mathbb{P}_i} \left[\tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{li}^{(k)} \right] \right),$$

with $\deg_i$ is the degree of processing unit i, i.e., set of neighbors. Assuming a Gaussian policy for the i^{th} split of task j, we can $f_{\mathrm{RL}_i^{(j)}}$ as:

$$
\begin{aligned}
f_{\mathrm{RL}_i^{(j)}} &= p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\Re^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\log\left[p^{(j)}_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\right] - \text{cnst.} \\
&= p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\Re^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\sum_{m=0}^{M^{(j)-1}} -\frac{1}{2}\left[\boldsymbol{u}_m^{(j)} - \boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)\tilde{\boldsymbol{\theta}}_i^{(j)}\right]^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\left[\boldsymbol{u}_m^{(j)} - \boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)\tilde{\boldsymbol{\theta}}_i^{(j)}\right],
\end{aligned}
$$

with $\boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m) \in \mathbb{R}^{m\times d}$ being a feature representation of the state, and $\boldsymbol{\Sigma} \in \mathbb{R}^{m\times m}$ being the covariance matrix of the Gaussian policy. Hence:

$$
\begin{aligned}
&\int_{\boldsymbol{\tau}^{(j)}\in\mathbb{T}_i^{(j)}} \nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}} p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\Re^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\log\left[\frac{p^{(j)}_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)}{p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\Re^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}\right] d\boldsymbol{\tau}^{(j)} \\
&= -\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}} p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\Re^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\Bigg[\sum_{m=0}^{M^{(j)}-1}\boldsymbol{u}_m^{(j),\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{u}_m^{(j)} \\
&\qquad\qquad - 2\sum_{m=0}^{M^{(j)}-1}\boldsymbol{u}_m^{(j),\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}\left(\boldsymbol{x}_m\right)\tilde{\boldsymbol{\theta}}_i^{(j)} \\
&\qquad\qquad + \sum_{m=0}^{M^{(j)}-1}\tilde{\boldsymbol{\theta}}_i^{(j),\mathsf{T}}\boldsymbol{\Phi}^{(j),\mathsf{T}}\left(\boldsymbol{x}_m\right)\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}\left(\boldsymbol{x}_m\right)\tilde{\boldsymbol{\theta}}_i^{(j)}\Bigg] \\
&= -p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\Re^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\Bigg[\sum_{m=0}^{M^{(j)}-1}\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\boldsymbol{u}_m^{(j),\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{u}_m^{(j)} \\
&\qquad\qquad - 2\sum_{m=0}^{M^{(j)}-1}\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\boldsymbol{u}_m^{(j),\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}\left(\boldsymbol{x}_m\right)\tilde{\boldsymbol{\theta}}_i^{(j)} \\
&\qquad\qquad + \sum_{m=0}^{M^{(j)}-1}\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\tilde{\boldsymbol{\theta}}_i^{(j),\mathsf{T}}\boldsymbol{\Phi}^{(j),\mathsf{T}}\left(\boldsymbol{x}_m\right)\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}\left(\boldsymbol{x}_m\right)\tilde{\boldsymbol{\theta}}_i^{(j)}\Bigg].
\end{aligned}
$$

At this stage, we handle each of the above three gradients separately. The first is zero as the expression does not depend on $\tilde{\boldsymbol{\theta}}_i^{(j)}$. For the second we have:

$$
\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\boldsymbol{u}_m^{(j),\mathsf{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}\left(\boldsymbol{x}_m\right)\tilde{\boldsymbol{\theta}}_i^{(j)} = \boldsymbol{\Phi}^{(j),\mathsf{T}}\left(\boldsymbol{x}_m\right)\boldsymbol{\Sigma}^{-1}\boldsymbol{u}_m^{(j)}.
$$

Furthermore, we have:

$$
\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\tilde{\boldsymbol{\theta}}_i^{(j),\mathsf{T}}\boldsymbol{\Phi}^{(j),\mathsf{T}}\left(\boldsymbol{x}_m\right)\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}\left(\boldsymbol{x}_m\right)\tilde{\boldsymbol{\theta}}_i^{(j)} = (\boldsymbol{A}+\boldsymbol{A}^{\mathsf{T}})\tilde{\boldsymbol{\theta}}_i^{(j)},
$$

with $\boldsymbol{A} = \boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)$. Since $\boldsymbol{A} = \boldsymbol{A}^{\mathsf{T}}$, then we have:

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\tilde{\boldsymbol{\theta}}_i^{(j),\mathsf{T}}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(i)}(\boldsymbol{x}_m)\,\tilde{\boldsymbol{\theta}}_i^{(j)} &= 2\boldsymbol{A}\tilde{\boldsymbol{\theta}}_i^{(j)} \\ &= 2\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)\,\tilde{\boldsymbol{\theta}}_i^{(j)}.\end{aligned}$$

Plugging these results back in the original gradient equation, yields:

$$\begin{aligned}&\int_{\boldsymbol{\tau}^{(j)}\in\mathbb{T}_i^{(j)}} \nabla_{\tilde{\boldsymbol{\theta}}_i^{(j)}} p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\log\left[\frac{p^{(j)}_{\tilde{\boldsymbol{\theta}}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)}{p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)}\right]d\boldsymbol{\tau}^{(j)} \\ &= -p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}\left(\boldsymbol{\tau}^{(j)}\right)\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\Bigg[-2\sum_{m=0}^{M^{(j)}-1}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{u}_m^{(j)} \\ &\qquad\qquad + 2\sum_{m=0}^{M^{(j)}-1}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)\Bigg] \\ &= 2\mathbb{E}_{\boldsymbol{\tau}^{(j)}\sim p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}(\cdot)}\left[\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\sum_{m=0}^{M^{(j)}-1}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{u}_m^{(j)}\right] \\ &\qquad\qquad - 2\mathbb{E}_{\boldsymbol{\tau}^{(j)}\sim p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}(\cdot)}\left[\mathfrak{R}^{(j)}(\boldsymbol{\tau}^{(j)})\sum_{j=0}^{M^{(j)}-1}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)\,\tilde{\boldsymbol{\theta}}_i^{(j)}\right].\end{aligned}$$

Considering the penalty terms of ADMM, and setting the above to zero, we can update $\tilde{\boldsymbol{\theta}_i^{(j)}}$ in closed-form according to:

$$\begin{aligned}&- 2\mathbb{E}_{\boldsymbol{\tau}^{(j)}\sim p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}(\cdot)}\left[\mathfrak{R}^{(j)}(\boldsymbol{\tau}^{(j)})\sum_{j=0}^{M^{(j)}-1}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}^{(j)}(\boldsymbol{x}_m)\,\tilde{\boldsymbol{\theta}}_i^{(j)}\right] \\ &\qquad + 2\mathbb{E}_{\boldsymbol{\tau}^{(j)}\sim p^{(j)}_{\boldsymbol{\theta}_i^{(j)}}(\cdot)}\left[\mathfrak{R}^{(j)}\left(\boldsymbol{\tau}^{(j)}\right)\sum_{m=0}^{M^{(j)}-1}\boldsymbol{\Phi}^{(j),\mathsf{T}}(\boldsymbol{x}_m)\,\boldsymbol{\Sigma}^{-1}\boldsymbol{u}_m^{(j)}\right] + \rho\mathrm{deg}_i\tilde{\boldsymbol{\theta}}_i^{(j)} \\ &\qquad - \rho\left(\sum_{l\in\mathbb{S}_i}\left[\tilde{\boldsymbol{\theta}}_l^{(k),(j)} + \frac{1}{\rho}\boldsymbol{\lambda}_{il}^{(k)}\right] + \sum_{l\in\mathbb{P}_i}\left[\tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \frac{1}{\rho}\boldsymbol{\lambda}_{li}^{(k)}\right]\right) = 0.\end{aligned}$$

Solving the above yields:

$$\tilde{\boldsymbol{\theta}}_i^{(k+1),(j)} = \left[-\frac{\rho \text{deg}_i}{2} \boldsymbol{I}_{d\times d} + \mathbb{E}_{\boldsymbol{\tau}^{(j)} \sim p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}(\cdot)} \left[\mathfrak{R}^{(j)} \left(\boldsymbol{\tau}^{(j)} \right) \sum_{m=0}^{M^{(j)}-1} \boldsymbol{\Phi}^{\mathsf{T},(j)}(\boldsymbol{x}_m) \boldsymbol{\Sigma}^{-1} \boldsymbol{\Phi}^{\mathsf{T},(j)}(\boldsymbol{x}_m) \right] \right]^{-1}$$
$$\times \left[\mathbb{E}_{\boldsymbol{\tau}^{(j)} \sim p_{\tilde{\boldsymbol{\theta}}_i^{(j)}}(\cdot)} \left[\mathfrak{R}^{(j)} \left(\boldsymbol{\tau}^{(j)} \right) \sum_{m=0}^{M^{(j)}-1} \boldsymbol{\Phi}^{\mathsf{T},(j)}(\boldsymbol{x}_m) \boldsymbol{\Sigma}^{-1} \boldsymbol{u}_m^{(j)} \right] \right.$$
$$\left. - \frac{\rho}{2} \left(\sum_{l \in \mathbb{S}_i} \left[\tilde{\boldsymbol{\theta}}_l^{(k),(j)} + \frac{1}{\rho} \boldsymbol{\lambda}_{il}^{(k)} \right] + \sum_{l \in \mathbb{P}_i} \left[\tilde{\boldsymbol{\theta}}_l^{(k+1),(j)} - \frac{1}{\rho} \boldsymbol{\lambda}_{li}^{(k)} \right] \right) \right].$$

Similar steps can be performed to determine the update equation for the shared repository by taking the gradient with respect to $vec(\boldsymbol{L})$.

[1] J. Kober, J. R. Peters, Policy search for motor primitives in robotics, in: Advances in neural information processing systems, 2009, pp. 849–856.

[2] S. A. Murphy, D. W. Oslin, A. J. Rush, J. Zhu, Methodological challenges in constructing effective treatment sequences for chronic psychiatric disorders, Neuropsychopharmacology 32 (2007) 257–262.

[3] J. Pineau, M. G. Bellemare, A. J. Rush, A. Ghizaru, S. A. Murphy, Constructing evidence-based treatment strategies using methods from computer science, Drug and alcohol dependence 88 (2007) S52–S60.

[4] R. S. Sutton, A. G. Barto, Introduction to Reinforcement Learning, 1st ed., MIT Press, Cambridge, MA, USA, 1998.

[5] A. Wilson, A. Fern, S. Ray, P. Tadepalli, Multi-task reinforcement learning: a hierarchical bayesian approach, in: Proceedings of the 24th international conference on Machine learning, ACM, 2007, pp. 1015–1022.

[6] M. E. Taylor, P. Stone, Transfer Learning for Reinforcement Learning Domains: A Survey, Journal of Machine Learning Research 10 (2009) 1633–1685.

[7] A. Lazaric, M. Ghavamzadeh, Bayesian multi-task reinforcement learning, in: Proceedings of the 27th International Conference on Machine Learning, 2010.

[8] H. Li, X. Liao, L. Carin, Multi-task reinforcement learning in partially observable stochastic environments, The Journal of Machine Learning Research 10 (2009) 1131–1186.

[9] H. Bou-Ammar, E. Eaton, P. Ruvolo, M. Taylor, Online Multi-task Learning for Policy Gradient Methods, in: T. Jebara, E. P. Xing (Eds.), Proceedings of the 31st International Conference on Machine Learning, JMLR Workshop and Conference Proceedings, 2014.

[10] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine learning 8 (1992) 229–256.

[11] S. Bhatnagar, R. S. Sutton, M. Ghavamzadeh, M. Lee, Natural actor–critic algorithms, Automatica 45 (2009) 2471–2482.

[12] J. Peters, S. Schaal, Natural actor-critic, Neurocomputing 71 (2008) 1180–1190.

[13] P. Ruvolo, E. Eaton, Ella: An efficient lifelong learning algorithm, in: Proceedings of the 30th International Conference on Machine Learning, 2013.

[14] S. Thrun, J. O'Sullivan, Discovering Structure in Multiple Learning Tasks: The TC Algorithm, in: International Conference on Machine Learning, Morgan Kaufmann, 1996.

[15] W. Caarls, E. Schuitema, Parallel online temporal difference learning for motor control, IEEE transactions on neural networks and learning systems 27 (2016) 1457–1468.

[16] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates, arXiv preprint arXiv:1610.00633 (2016).

[17] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, S. Levine, Collective robot reinforcement learning with distributed asynchronous guided policy search, arXiv preprint arXiv:1610.00673 (2016).

[18] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, Journal of Machine Learning Research 17 (2016) 1–40.

[19] M. P. Deisenroth, P. Englert, J. Peters, D. Fox, Multi-task policy search for robotics, in: Robotics and Automation (ICRA), 2014 IEEE International Conference on, 2014, pp. 3876–3881.

[20] A. Wilson, A. Fern, S. Ray, P. Tadepalli, Multi-Task Reinforcement Learning: A Hierarchical Bayesian Approach, in: Machine Learning, Proceedings of the 24th International Conference Corvallis, Oregon, USA, June 20-24, year = 2007,, ????

[21] M. Snel, S. Whiteson, Learning potential functions and their representations for multi-task reinforcement learning, Autonomous agents and multi-agent systems 28 (2014) 637–681.

[22] A. Kumar, H. Daume, Learning task grouping and overlap in multi-task learning, in: Proceedings of the 29th International Conference on Machine Learning, 2012, pp. 1383–1390.

[23] H. Bou Ammar, E. Eaton, J. M. Luna, P. Ruvolo, Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2015.

[24] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning 3 (2011) 1–122.

[25] E. Wei, A. Ozdaglar, Distributed alternating direction method of multipliers, in: Decision and Control, 2012 IEEE 51st Annual Conference on, IEEE, 2012, pp. 5445–5450.

[26] R. Tibshiranit, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society. Series B (Methodological) 58 (1996) pp. 267–288.

[27] J. Peters, S. Schaal, Natural Actor-Critic, Neurocomputing 71 (2008).