2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)

Constructive Policy: Reinforcement Learning Approach for Connected Multi-Agent Systems

Sayyed Jaffar Ali Raza[1] and Mingjie Lin[2]
jaffar@knights.ucf.edu, mingjie@eecs.ucf.edu

***Abstract*— Policy based reinforcement learning methods are widely used for multi-agent systems to learn optimal actions given any state; with partial or even no model representation. However multi-agent systems with complex structures (curse of dimensionality) or with high constraints (like bio-inspired snake or serpentine robots) show limited performance in such environments due to sparse-reward nature of environment and no fully observable model representation. In this paper we present a constructive learning and planning scheme that reduces the complexity of high-diemensional agent model by decomposing it into identical, connected and scaled down multi-agent structure and then apply learning framework in layers of local and global ranking. Our layered hierarchy method also decomposes the final goal into multiple sub-tasks and a global task (final goal) that is bias-induced function of local sub-tasks. Local layer deals with learning 'reusable' local policy for a local agent to achieve a sub-task optimally; that local policy can also be reused by other identical local agents. Furthermore, global layer learns a policy to apply right combination of local policies that are parameterized over entire connected structure of local agents to achieve the global task by collaborative construction of local agents. After learning local policies and while learning global policy, the framework generates sub-tasks for each local agent, and accepts local agents' intrinsic rewards as positive bias towards maximum global reward based of optimal sub-tasks assignments. The advantage of proposed approach includes better exploration due to decomposition of dimensions, and reusability of learning paradigm over extended dimension spaces. We apply the *constructive policy method* to serpentine robot with hyper-redundant degrees of freedom (DOF), for achieving optimal control and we also outline connection to hierarchical apprenticeship learning methods which can be seen as layered learning framework for complex control tasks.**

I. Introduction

Reinforcement learning (RL) has remarkably enabled multi-agent systems to achieve policies that can deliver state-of-the art control in complex learning environments like playing Atari games or solving Go game with infinitely broad state-action horizon [1]. RL approaches are being widely used for training agents to learn control over complex gait maneuvers (like crawling, walking or running) solely based on their own learning experiences and corresponding rewards associated with that experience. Well established RL frameworks allow agents to learn through trial and error and stay motivated by improving performance over selecting better actions iteratively at next time step to yield high reward. Most of the RL methods assume that agents are at

[1]Department of Electrical and Computer Engineering, University of Central Florida, Orlando FL, USA
[2]Mingjie Lin is with the faculty of Electrical and Computer Engineering, University of Central Florida, Orlando FL, USA

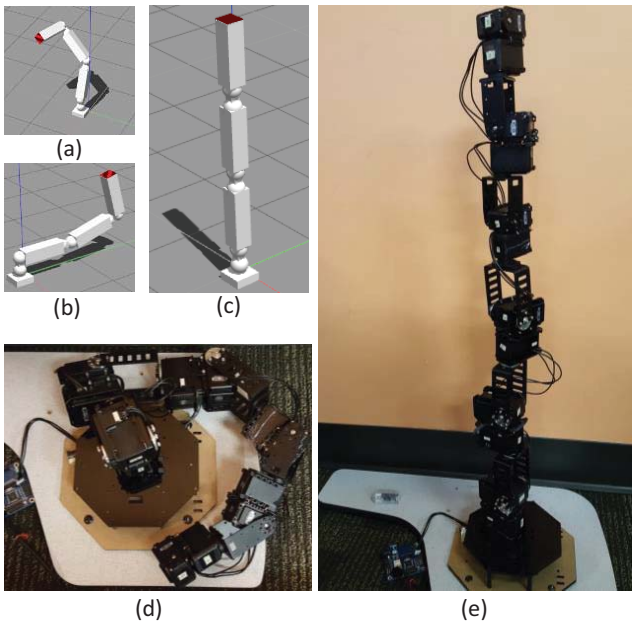

Fig. 1: 12-DOF serpentine robot that mimics octopus-like manipulation. Trained using Constructive Policy method as *6-DOFx2* agents. (a, b, c): ROS Gazebo simulation environment. (d,e): real-world hand-crafted serpentine robot

freedom to choose any actions that satisfies the condition of achieving maximum expected reward in return – hence reward being the major performance factor overall in action selection [2]. Reward engineering is considered to be one of the significant challenges for learning an action-selection policy for real-world problem scenario because, in real-world behavior; rewards are often delayed and are also decayed with time [3], [4]. In multi-agent setting, policy search algorithms need to be modeled with generic behavioral rewards such that a policy must not go adverse to any agent and should also suffice for overall shared goals of all agents – not just a single agent. One of the fundamental challenges for multi-agent robots is learn an equivalently beneficial policy for all agents spawned in complex dynamic environments with sparse or delayed feedback behavior [3]. Learning in such setting requires completely observable representation of model for efficient exploration, or a well defined heuristic for action-selection to narrow down action space; however modeling fully observable environment is mostly unattainable or requires exceedingly complicated computations. Policy search algorithms have been in use for

978-1-7281-0356-3/19/$31.00 ©2019 IEEE

a long time to address action-selection problem by learning a function $\pi(s)$ (or a rule) that can return a decision to choose an action given available information of the current state s given $s \in S$, even with partial or no model representations. The problem with the concept of a policy is that it refers to any method for determining an action given a state, and as a result it covers a wide range of algorithmic strategies, each suited to different problems with different computational requirements [5]. Current policy search algorithms also provide notable performance for learning an optimal policy for continuous domains. Such methods commonly perform non-linear function approximation coupled with off-the-shelf policy gradient algorithms to learn state-action transitions over high-dimensional continuous spaces [6]–[10].

Although these methods significantly improve convergence and give notable control capabilities in high-dimensional continuous spaces; most of these methods can only perform under full model observability and deterministic beliefs of optimality irrespective of notorious behavior of sparse rewards and also exploration is mostly engineered heuristically (ex: by injecting noise) [11]–[13] or is set by initializing a stochastic policy with high entropy [13]. Agents that possess intricate geometrical structure (like bio-inspired robots or soft robots) cannot be explicitly modeled due to their redundant kinematic properties and due to larger pool of possible sub-optimal solutions that make it difficult to converge towards most optimal solution [14] [15]. Secondly, agents with complex structures tend to have exceedingly convoluted action spaces due to redundant joints causing curse of dimensionality issues; any sub-optimal action selection at low level will embrace overall deteriorated action selection. Agents with complex reward behavior due to physical constraints can only receive sparse rewards because reward distribution for such agents (ex. bio-inspired manipulator) is based on expectation of reward as function of entire action-selection transitions – and action space is continuous and vectorized sequence of connected acts if an agent has connected structure (ex:bio-inspired robots) [16], [17]. Since most traditional algorithms with "vanilla" policy gradient methods are conceived entirely in context of Markov Decision Processes (MDP), these methods significantly enjoy high throughput over control but at a cost of exhaustive complexity in terms of extended exploration requirement and longer learning periods [18]. Also such methods easily overlook abstract simplicites available at temporal scales [19].

We present an approach that learns a policy in high-dimensional continuous horizon by scaling down the dimensions into identical sub-dimensions while keeping the continuous horizon intact, and by ranking the learning problem into local and global learning layers such that global learning layer is eventually engineered quickly with positive learning bias of its local layers' knowledge. The approach is reminiscent to human nature of doing sequential but lengthy tasks in daily routine. For the sake of simplicity; [illegible] opening a drawer– a person first needs to maneuver the arm forward (like a servo maneuver) towards handle, grasp it (consider grasping as already learned skill), and move the arm backwards against the drawer. The backward maneuver (servo maneuver) actually shares a similar problem model as forward movement; but the immediate reward for backward joint movement might get negative if drawn by forward maneuver policy. However the global or overall reward should still be optimal since the main goal was to open a drawer. Our constructive policy approach ranks such identical tasks as local tasks – generating a pool of local policies. Next step involves to learn deploying optimal combination of policies by sampling them with optimal policy parameters. While learning global policy we modify the single agent into scaled down hierarchy shadowing apprenticeship learning methodology that allows the system to accept advice or bias from multiple local apprentices operating at lower layers [10]. The contribution of our study is build upon decomposition of high-dimensional spaces into connected multi-agent space scenario that offer practical benefits for real-world robot models as well. We demonstrate our method to control remarkably complex robotic arm design with 12 DOFs – once trained, the arm follows continuous trajectories in continuous domain.

II. Related Work

Policy learning for agents operating in continuous horizons and sparse reward environments has been studied as temporal abstraction problem that inherently constitutes a Semi-MDP (SMDP) learning model such that an agent has a policy with parameters $\theta \exists \pi_\theta{:}S \times A \rightarrow [0,1]$, and follows a stochastic termination condition $\beta{:}S^+ \rightarrow [0,1]$. The agent takes an action $a_t \in A$ and continues to evaluate towards a_{t+1} with Markovian property until it terminates stochastically with probability distribution of β. When the evaluation is terminated the agent can stochastically choose to timeout and restart with newer policy parameter π_{θ^*} or continue determining actions with same parameters [18]. At higher-level, an agent can choose actions that draw varying amount of time to finish (randomly distributed time) and any sojourn time in given state is actually a continuous random variable with distribution depending on states [18], [20], [21].

With SMDPs, the expected recurrent time-average outcomes can be embedded under a stationary policy with recurrent states and can be generalized for primitive actions – furthermore those primitive actions are extended over temporal course of actions. This learning method is used for tasks that can be designed with shared meta-controls [22] and integrated temporal abstractions (*options framework, Sutton* [18]). For high dimensional or infinite spaces, approximation can be maximum entropy value approximation over established distribution models (ex: Boltzmann distribution or Thompson sampling) and choose actions with respect to randomly drawn belief that instantaneously is self-corrected over time steps following Bayesian control rule for calculating expected rewards. The rewards can be weighted for a cost utility function that quantifies overall desirability of corresponding belief [7], [23], [24]. Hierarchical or layered reinforcement learning is broadly studied topic in field of artificial intelligence. It is widely used for learning and opti-

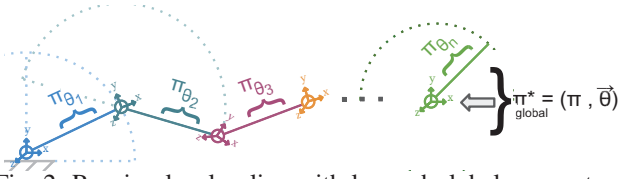

Fig. 2: Reusing local policy with learned global parameters. Colors show sub-agent intercepts operating under same local policy with different parameters

mizing policies for environments that cannot be modeled into discrete representations and are also infinitely huge in terms of model dimensions. The crucial part in modeling policy is adapting actions based on dynamic primitives. Policy based methods such as splines gradients or often called vanilla policy gradients algorithms are popular in discrete space behavioral policy modeling due to their trajectory centric representation of actions. These methods are widely seen to solve complex tasks like bi-pedal robot walking [25] or learning primitive motor angles to control joints [26]; also these vanilla policy gradient methods are relatively easy to implement and adapt in real-world as well. However such policies are limited to small-scale horizons that are often discretized with extended granularity resolution [9], [27].

Besides this, other notable approximation techniques [28]–[31] also involve Bayesian optimization of cost functions by formulating a conventional transform function type approach towards specific gradient trajectories with their covariance already estimated. Such approaches convert sparsity of rewards into probability map given that rewards are of finite and discounted nature. Next step is to take account of calculated covariance and apply traversal algorithm and update policy in that direction. The algorithm takes current and predicted probabilities and calculates the Bayesian optimality recursively. When it comes to integrating more "moving variables" for decision making and taking actions in continuous horizon, recent works intent to use deeper and larger systems that can simultaneously perform end-to-end control for wide range of task in parallel [29], [32], [33].

Such methods normally are build over networked systems like convolutional neural networks (CNN) with extended array of parameters that are trained using a guided policy search method [34] that represent the policy learning as supervised approach guided by the trajectory [32], [35]. Such methods are also seen to solve real-world problems like putting a cap on bottle [32] or stacking Lego blocks [36] Talking about hierarchies, researchers also consider RL problems as modular task and claim that every problem is actually composed of concurrent sub-problems with a matter of abstraction levels [37]. Modular RL or MRL treats agents with set of sub-goals that can possibly conflict with other agent's goal as well. In such setting every sub-agent reports unique numerical preference to a meta-controller and then meta-controller assigns an action as whole to agent systems. [illegible] on study for multi-stage learning was presented as *options* framework by Rich. S. Sutton [18] [illegible] be seen as a two stage system that learns abstract actions by selecting options over temporal sequences. An upgrade to *options* is presented as policy sketches [38] that annotate global goals as sequential sub-global task providing guidance at high level [39], [40]. Policy sketch method represent every sub-goal with a dedicated learned sub-policy that jointly maximizes a shared policy parameter among all policies; maximizing expected reward for global policy tied to shared parameter as well. Policy optimization for these sketches is done by actor-critic gradient computation [41].

III. Constructive Policy Architecture

We constitute the learning problem into layers by representing the a single complex agent into smaller and identical multi-agent models. We train a sub-agent independently within its own horizon, and then combine all sub-agents together by constructing a global policy based on composition of local policies. Each sub-agent is decomposed heuristically in such a way that it reminiscent a scaled-down physical structure of subsequent local agents connected in the global system—this representation is shown in fig 2. For our test case we implement our idea on a serpentine robot with 12-DOF joints to mimic octopus like maneuverability with control over redundant joints. Our goal is to control servo maneuvers in dynamic environment, which is a complex objective to be learned as a unified problem due to hyper-redundant servo joints. The redundancy is necessary to achieve bio-inspired gait but it comes with a problem of curse-of-dimensionality which exponentially increases the action space for an agent. We find serpentine design interesting because it can be seen as a chain of ideally connected small joint motors like a bio-inspired snake or octopus and the complexity can be understood by assuming taking a single joint action changes the decisions of future action selection entirely for every motor joint in the robot. Since each local agent represents a physical link in the robotic arm, they cannot move independently from each other. In other words, they are subjected to geometric constraints. This representation differentiates greatly from virtual agent-based multi-agent systems, such as two-player chess.

We divide the learning architecture into local and global levels. The local level considers an intercepted version of serpentine model (6-DOF joints in our case as shown with different colors in fig 2). At local level, the sub-agent learns a local policy π_{local} for independent maneuver. The policy π_{local} can be considered as a generic local control policy that can be used for any local agent in the system since each agent share similar physical structure. However local agents are constrained with movement of other sub-agents so the local policy cannot be reused with all sub-agents. Instead of using the local policy barely with agents we can parameterize it over independent policy parameters for each local agent. This parameterization is done on global level such that global policy π^* learns a parameter vector $\vec{\theta}$ sampled over distribution of local policy which was learned by a single sub-agent and now being used by all local agents but with different global parameters.

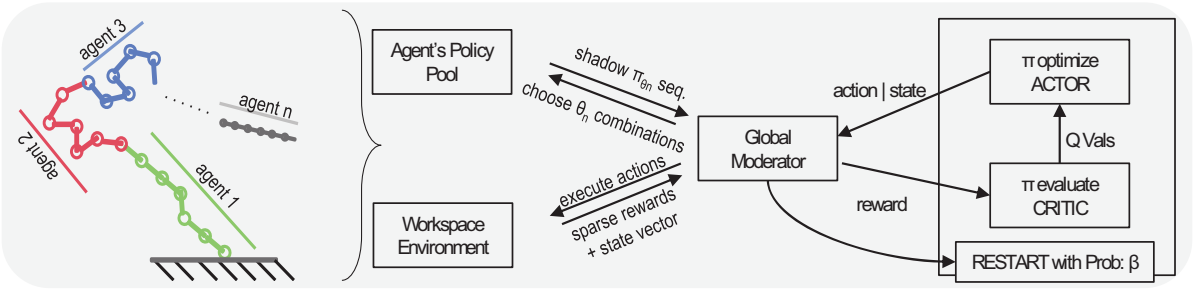

Fig. 3: Episodic information flow between local agents and a global moderator controller. Local agents are assigned subgoals and take actions within external environment based on policy parameters assigned to them. At the end of episode, the global moderator receives intrinsic rewards from local agents and scalar global reward from environment

The action space for global policy is the sub-goals combination for connected local agents. Global action space can be defined as a vector $G \Longrightarrow (\hat{g}_1 \dots \hat{g}_n) \in \mathcal{G}$.

$$\pi^*(\mathcal{G}, \mathcal{S}) \longrightarrow \pi_{local} \begin{bmatrix} (\hat{g_1}|s_1) \\ (\hat{g_2}|s_2) \\ \vdots \\ (\hat{g_n}|s_n) \end{bmatrix} \pi_p \left[(\mathcal{X}|\mathcal{S})\right]$$

The global policy chooses sub-goals from $\mathcal{G}$ and then parameterize those sub-goals with $\mathcal{X} \Longrightarrow (\theta_1 \dots \theta_n \in (0, 1)$. $\mathcal{X}$ is the parameter vector for global action. Action space is then given as

$$A = \bigcup_{\hat{g} \in \mathcal{G}} \{(\hat{g}, \theta) | \theta \in \mathcal{X}\}$$

A complete global action $(\hat{g} \in \mathcal{G}, \theta \in \mathcal{X})$ is build with sampling a parameter θ from $\pi_p(\mathcal{X}|\mathcal{S})$ and using parameters over sub-goal vector $\pi_p(\mathcal{X}|\mathcal{S})$. The global policy constructs the sub-goal trajectory that lead the final end-effector to terminal state or the global goal. Globally, parameter learning and action-selection (sub-goals) is learnt by alternate updating between parameter policy and action-value function iteration. We can formulate the global learning as an MDP $\langle \mathcal{S}, \mathcal{G}, \mathcal{R}, \gamma \rangle$ where reward $\mathcal{R}$ is a function of intrinsic signals from sub-agents that correspond to success at intercept level – however that does not assures success achieved globally in general. The intrinsic reward inherently motivates the global gradient to yield better sub-goals by signaling binary value that is expressed within global reward function. In order to maintain global success, the reward function contains a predictate coefficient that is multiplied with the overall intrinsic reward.

$$\mathcal{R} \simeq \left[\sum_i r_i\right] \cdot 1$$

$$1 = \begin{cases} 1, & \text{if } \mathcal{S}_t = \mathcal{S}_{goal} \\ 0, & \text{otherwise} \end{cases}$$

The predictate helps exploration to logically transition towards the sub-goal selection that is not just optimal intrinsically but returns optimality towards global goal as well. The [illegible] shadows the surrogate technique of proximal policy optimization or PPO [42]. But PPO is constrained with continuous model interaction during training which is expensive for dexterous models like snake robots. Our constructive method assumes that the local policy is well-trained over intercepted sub-agent and only requires a parametric approximation to be used over subsequent connected sub-agents. The global layer works as a moderator 3 to monitor the sug-goals and adapt the gradients for optimal parameters over their probability distributions.

$$\because P(a_i) = a_i \log(\theta_i) + (1 - a_i) \log(1 - \theta_i)$$

For simplicity, assume all agents choose parameter $\theta_i$then for single $agent_i$ update:

$$\frac{\partial}{\partial \theta_i{}^i} V^i_{\theta_i} = \underset{(S,\vec{A})}{R} \cdot \frac{\partial}{\partial \theta_i} \left[P(\hat{a_1}, \hat{a_2}, \dots, \hat{a_N} \right]$$

$$\frac{\partial}{\partial \theta_i{}^i} V^i_{\theta_i} = \underset{(S,\vec{A})}{R} \cdot \frac{\partial}{\partial \theta_i} \left[\frac{\vec{\hat{a}}}{\theta_i} - \frac{(1-\hat{a})}{(1-\theta_i)} \right]$$

Partial derivatives with respect to the parameter θ for value functions in and logarithmic transition probability gives simplified expression for value function that can be iterated recursively in place of classic bellman function reducing effort by utilizing localized probability distributions over shared θ parameters

IV. Experiments

We carry out and validate our experiments over a hyper-redundant serpentine robot arm with 12-DOF maneuvering capabilities. The arm resembles maneuvering properties of an octopus' tentacles. We performed experiments against Deterministic Proximal Policy Optimization (DPPO) method and Asynchronous Advantage Actor-Critic (A3C) method. Both methods have their own benefits in different domains; A3C provides an edge to process the algorithm over very low computing power and uses parallel threads over CPU instead of memory-hungry gradient update methods. However A3C does not provides leverage of experience replay buffers which relinquishes it's performance in sparse rewards behaviors which can be seen in fig: 5. Also looking at DPPO success rate, it is apparent that DPPO method has a promising gradient which verifies [43] (Schulman et. al.) observations over joint angle architectures. However DPPO achieves improved performance, but it requires more epoch cycles to reach to admissible outcomes. The proximal pol-

6+6 DOF	A3C	DPPO	CONSTRUCTIVE
Accuracy	<± 3.0 units	± 2.2 units	± 1.7 units
Success Rate	0.2243	0.398	**0.7451**
Epochs	>>6000	>6000	**<5000**

TABLE I: Comparison: accuracy, success rate and time taken

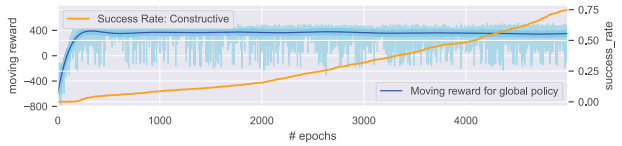

Fig. 4: Overall reward trend during training period of our global policy.

icy optimization updates alternate between sampling data through interaction with the environment, and optimizing a "surrogate" objective function using stochastic gradient ascent [42]. However for problems that involve dexterous manipulation like serpentine control the environment interaction is exhaustive and less likely to happen and this can be seen by the success rate of DPPO in Table I.

In contrast with DPPO and A3C, our proposed *Constructive policy* method performs relatively better and requires less time complexity to give better success rate. Quicker convergence trend stabilizes the effect of global policy learning for control of high-dimensional architecture in environments with rewards that are sparsely distributed. Our testing environment was setup within ROS ecosystem using Gazebo simulator for ROS to train and test robot models. ROS environment has proven to be standardized method to validate robot simulations [33]. We tested the performance over both 2D and 3D continuous spaces with Gazebo world. Furthermore we have also designed a physical robot arm for executing telemetries of simulation world over real world. Telemetry tests have been done, however our future work involves realtime testing of Constructive RL algorithm.

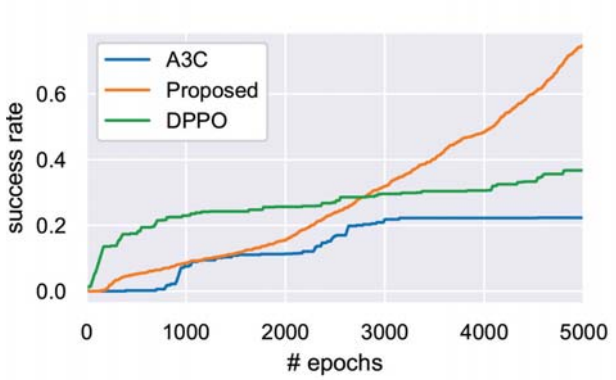

Fig. 5: Comparison plot for success rate (finish at terminal state) over 5000 epochs. A3C has many steep but slower [illegible] and DPPO shows quick but deferred progress. [illegible] proposed *constructive* policy method outperforms both methods and gives a success rate >74%

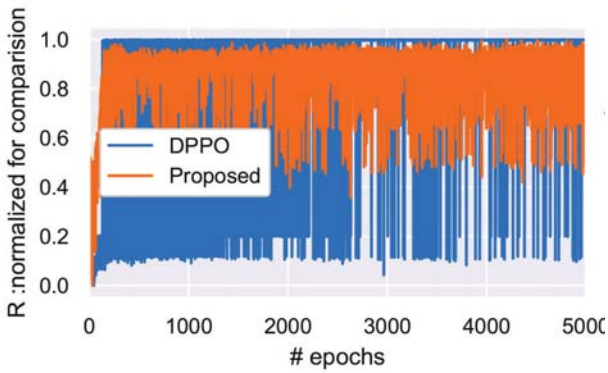

Fig. 6: Moving reward for proposed method vs. off-the-shelf DPPO. Though not worse but visibly DPPO has higher jitter in reward.

V. Conclusion

Our study contributes a promising approach in area of hierarchical deep reinforcement learning specially for complex robotic arms that are inspired by neurophysiology of living beings. Hyper-redundant robotic arms possess intricate designs and require exhaustive heuristics to formulate control functions. We believe that *"constructive policy learning"* contributes towards formal simplification of control task for bio-inspired robots by allowing structured composibility to divide high-dimensional problem spaces into significantly identical, smaller agents and treat unified control problem in distributed manner. Constructive policy method can be applied in environment where rewards are sparse or often accumulated with larger delaying factor – in contrast methods like ϵ-Greedy search or vanilla policy methods arguably deteriorate the expected reward outcome in sparse setting. Our method operates in distributed setting allowing local agent to share their states in entirety with global moderator such that the resulting global policy is fused with shared feedback of local agents' rewards – offering offline estimate or positive bias for Q-Value. We believe constructive Policy RL approach can be most useful in problems that can be decoupled into smaller sub-problems. Quick examples would be teaching a robot arm to open drawer, or teaching an octopus to slide-climb – dividing training process into "wrap around an object", then "push itself against object to climb". We have tested our idea in real-world simulation environment (ROS Gazebo) with active physics conditions and kinematics requirements. Our testing subject was a simulation robot arm that has 12 joints operating in continuous state and action space. Given any location within robot arm's workspace – it navigates to that Cartesian coordinate location. We observed success rate greater than 74% in simulation. We believe that this work along with recent advancements in connected robot learning area can contribute significantly towards greater developments. Also, our on-going work involves testing our method over a real physical robot (fig: 1) that has identical joints as simulation model.

REFERENCES

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] D. Dewey, "Reinforcement learning and the reward engineering principle," in *2014 AAAI Spring Symposium Series*, 2014.

[4] S. J. A. Raza and M. Lin, "Policy reuse in reinforcement learning for modular agents," in *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*, pp. 165–169, IEEE, 2019.

[5] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*, vol. 703. John Wiley & Sons, 2007.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[7] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in neural information processing systems*, pp. 3675–3683, 2016.

[8] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016.

[9] T. Haarnoja, V. Pong, A. Zhou, M. Dalal, P. Abbeel, and S. Levine, "Composable deep reinforcement learning for robotic manipulation," *arXiv preprint arXiv:1803.06773*, 2018.

[10] J. Z. Kolter, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning with application to quadruped locomotion," in *Advances in Neural Information Processing Systems*, pp. 769–776, 2008.

[11] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *ICML*, 2014.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. P. Wierstra, "Continuous control with deep reinforcement learning," Jan. 26 2017. US Patent App. 15/217,758.

[13] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," *arXiv preprint arXiv:1702.08165*, 2017.

[14] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, p. 503, 2015.

[15] H. Alshamsi, S. Jaffar, and M. Li, "Development of a loca prosthetic limb using artificial intelligence," 2016.

[16] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1704.03012*, 2017.

[17] E. Theodorou, J. Buchli, and S. Schaal, "Reinforcement learning of motor skills in high dimensions: A path integral approach," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 2397–2403, IEEE, 2010.

[18] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artificial intelligence*, vol. 112, no. 1-2, pp. 181–211, 1999.

[19] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning," in *Advances in neural information processing systems*, pp. 1281–1288, 2005.

[20] M. Baykal-Gürsoy and K. Gürsoy, "Semi-markov decision processes," *Wiley Encyclopedia of Operations Research and Management Sciences*, 2010.

[21] D. P. Bertsekas, "Neuro-dynamic programming," in *Encyclopedia of optimization*, pp. 2555–2560, Springer, 2008.

[22] K. Frans, J. Ho, X. Chen, P. Abbeel, and J. Schulman, "Meta learning shared hierarchies," *arXiv preprint arXiv:1710.09767*, 2017.

[25] P. Kormushev, B. Ugurlu, S. Calinon, N. G. Tsagarakis, and D. G. Caldwell, "Bipedal walking energy minimization by reinforcement learning with evolving policy parameterization," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pp. 318–324, IEEE, 2011.

[23] S. Agrawal and N. Goyal, "Analysis of thompson sampling for the multi-armed bandit problem," in *Conference on Learning Theory*, pp. 39–1, 2012.

[24] K. P. Körding and D. M. Wolpert, "Bayesian decision theory in sensorimotor control," *Trends in cognitive sciences*, vol. 10, no. 7, pp. 319–326, 2006.

[26] A. J. Ijspeert, J. Nakanishi, and S. Schaal, "Learning attractor landscapes for learning motor primitives," in *Advances in neural information processing systems*, pp. 1547–1554, 2003.

[27] C. Guestrin, M. Lagoudakis, and R. Parr, "Coordinated reinforcement learning," in *ICML*, vol. 2, pp. 227–234, Citeseer, 2002.

[28] M. L. Littman, A. R. Cassandra, and L. P. Kaelbling, "Learning policies for partially observable environments: Scaling up," in *Machine Learning Proceedings 1995*, pp. 362–370, Elsevier, 1995.

[29] E. Tzeng, C. Devin, J. Hoffman, C. Finn, X. Peng, S. Levine, K. Saenko, and T. Darrell, "Towards adapting deep visuomotor representations from simulated to real environments," *CoRR, abs/1511.07111*, 2015.

[30] E. Brochu, V. M. Cora, and N. De Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv preprint arXiv:1012.2599*, 2010.

[31] P. Henry, C. Vollmer, B. Ferris, and D. Fox, "Learning to navigate through crowded environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 981–986, IEEE, 2010.

[32] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[33] S. J. A. Raza, N. A. Gupta, N. Chitaliya, and G. R. Sukthankar, "Real-world modeling of a pathfinding robot using robot operating system (ros)," *arXiv preprint arXiv:1802.10138*, 2018.

[34] S. Levine and V. Koltun, "Guided policy search," in *International Conference on Machine Learning*, pp. 1–9, 2013.

[35] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine, "Combining model-based and model-free updates for trajectory-centric reinforcement learning," *arXiv preprint arXiv:1703.03078*, 2017.

[36] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, pp. 1057–1063, 2000.

[37] Ö. Şimşek, A. P. Wolfe, and A. G. Barto, "Identifying useful subgoals in reinforcement learning by local graph partitioning," in *Proceedings of the 22nd international conference on Machine learning*, pp. 816–823, ACM, 2005.

[38] J. Andreas, D. Klein, and S. Levine, "Modular multitask reinforcement learning with policy sketches," *arXiv preprint arXiv:1611.01796*, 2016.

[39] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 2169–2176, IEEE, 2017.

[40] N. Ono and K. Fukumoto, "Multi-agent reinforcement learning: A modular approach," in *Second International Conference on Multiagent Systems*, pp. 252–258, 1996.

[41] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[42] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[43] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.