Bridging Control, Optimization and Reinforcement Learning

Zongqiang Pang,Liping Bai *Member, IEEE,*

***Abstract*—As a master student from college of automation who focused on reinforcement learning, i tried to apply for appropriate labs in automation department. What I find out is that there are lot of researchers trained in control theory would like to incorporate the latest development in reinforcement learning into their research, yet when they made the journey to study that subject, they find tidious/nonstandarized notations and lack of rigirous proof. The defining textbook for reinforcement learning is that of Richard Sutton, yet he is giered towards an audience with Computer Scinece background. The notations is filled with subscript and superscript such that it would make it easy to be implemented by computer. The proof in Richard Sutton's book is also not satisfying from a control theorist point of view, since the main objective of him is to get the concepts acrossed rather than prove the logic behind them. In this papyer, I hope to provide control researchers tools to make that jump into reinforcement learning by introducing simplified notations and provide proof to the key ideas in RL. It also serves as a survey on the researchs at the intercept between control theory and reinforcement learning.**

***Index Terms*—Reinforcement Learning, Control Theory, Optimal Control, Optimization**

I. Introduction

CONTROL, Optimization and Reinforcement Learning are three distinct disciplines studying the same thing: controlling in a dynamic environment. Unfortunately, the silos of institutions means these three communities have their own channel of exchange. For instance, in Chinese institutions, Control and Optimization is usually housed in the department of Automation where the bread and butter of their work is modelling based on differential equations and solutions of control laws based on matrix theory and stochastic process. Their publication is usually funnelled through IEEE Transactions. Reinforcement Learning on the other hand, is usually housed in the Computer Science Department, where they use Statistical Learning theory to control a dynamic system with or without a physical model.Their research is usually funnelled through ACM affiliated publications.

The success of reinforcement learning has lured people from traditional control background to incorporate those techniques into their research. Recently, there are interdiscipline conferences held specifially for the purpose of briding the gaps between these three groups of people, for instance the L4DC (Learning for Dynamic Control) conference and Intersections between Control, Learning and Optimization Workshop.

One of the obsticle for bringing everyone together is the notations. Much like a language, the style of notation is how researchers of various disciplines disttinguish one another. The starting point of reinforcement learning theory is Markov Decision Process(MDP) which is not so for the vast majority of control theory. For control theoriest, a system is usually expressed as a step-wise update: $x_{t+1} = \{(x_t, u_t, e_t)$ where x is the state of the system, u is the control measure, e is the error where stochastidy is encoded. MDP encode stochastidy via transition matrix, $\mathcal{P}^a_{ss'} := Pr\{S_{t+1} = s' | S_t = s, A_t = a\}$ where s' is the new state, a is the action taken. A control theoriest would write it as $\mathcal{P}^u_{x'x} := Pr\{X_{t+1} = x' | X_t = x, U_t = u\}$ If the transition matrix in MDP is equivolent to the error in step wise update, then clearly, this two ways of describing a system is equivolent.In this paper, the style of notation would be the step wise update with error since the target audience is control theoriests rather than computer scientists.

Another obsticle facing control theoriest is how tidious the notation in reinforcement learning is. Often times, there are multiple subscripts and superscripts to denote time, action, etc. Proof with this set of notation is extremely confusing. Quite often, control theoriests find themselves spending more time sorting out the transcripts than they would like to. In this paper, we introduce a set of simplifed notations based on the book of D. P. Bertsekas. Proof of key results in reinforcement learning is listed with this set of simplifed notations.

The aim of this paper is to provide resources to control theoriest who want to embark on the journey of reinforcement learning. All the milestones in reinforcement learning are casted in the light of control theory. In the first section, the notation is introduced. In the second section, detailed proof is provided. In the third section, a list of intersections between control and reinforcement learning provided.

II. Simplied Notation

The theories of phased array were fully formulated during the WWII era where an array of radars was deployed to detect an accurate angle of arrival [**?**]. Today, phased array hardware is widely available as a commercial product as shown in Figure 1. Together with various forms of Space Time Signal Processing(STSP), phased array and the beamforming technology has become the enabling components for future communication networks. There are different configurations of arrays, in this paper, we only consider one dimension uniform linear array.

Nanjing Unversity of Posts and Telecommunications, College of Automation & College of Artificial Intelligence, Nanjing, Jiangsu,210000 China email:zqpang@njupt.edu.cn

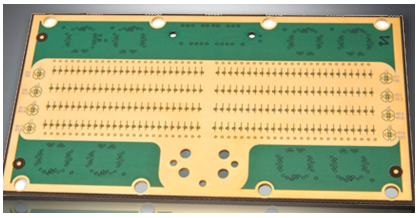

Fig. 1: Pivotal 39GHz Beamformer

A. Channel Model

Suppose there are p propogation path from transmitter to receiver. the gain for each path is denoted by α_i. The channel is modeled as a sum of each path. When Line of Sight(LoS) not available and the number of path is large, Rayleigh fading and a pleathera of channel modelling techniques can be applied to capture Non Line of Sight channel gain. In this paper, we only consider the environment with direct Line of Sight transmission and no reflection path.

B. beamforming codebook $\mathcal{F}$

Let Angle of Departure(AoD) be denoted as φ. Suppose the the range of adjustment for the beamformer is ζ degrees and is deicretized into N portions, each with the angle adjustment of $\frac{\zeta}{N}$ degree. For the i^{th} code in the codebook with AoD of φ_i, the beamforming vector is computed as the following:

$$\mathbf{f} := \mathbf{a}(\varphi_i) = [1, e^{jdcos(\varphi_i)}, ..., e^{jd(M-1)cos(\varphi_i)}]^T \quad (1)$$

C. System Model

The schematics of wirelessly powered communication network is shown in Figure 2. L energy transmitting node, each equipped with M radiating elements arranged as a uniform linear array, transmit power to K energy receivers scattered in an open field.

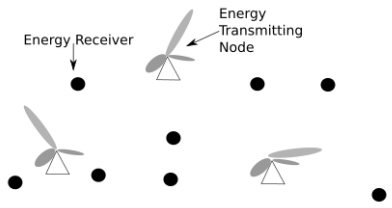

Fig. 2: Wirelessly Powered Network

One challenge for energy beamforming is lack of channel information. Time Division Duplex is a common strategy to implement joint communication and energy transfer, where the wireless power transfer happens from time 0 to P and wireless information transfer(WIT) happens from time P to T. Therefore, no information can be sent before enough energy is stored during the power transfer phase. Ideally, pilots signals should be sent and decoded systematically for channel estimation. [**?**], yet this function is not available for energy beamforming because WIT and WPT functions are realized with separate circuits [**?**], where the latter does not provide decoding capacity. Previous works have proposed methods of channel estimation with only one-bit feedback [**?**], we would adopt this minimum feedback scheme in this paper.

Because the energy beamforming signal **x** does not carry any information, it is assumed to be independent sequences with zero mean and unit variance. [**?**] Furthermore, because we consider the beamformer to be analog, x_1 to x_M are the same signal $x \in \mathbb{C}$.The power in noise is significantly weaker than the energy signal, therefore it can be ignored for practical purposes.

Let $y_{j,p}$ denote the signal received on the j^{th} receiver from the p^{th} transmitter. x_p be the signal transmitted from node p. $\mathbf{f}_p$ be the beamforming code for node p. $h^j_{i,p}$ denote the line of signt channel connecting j^{th} receiver to i^{th} radiating element from p^{th} transmitting node.

$$y_{j,p} = \begin{bmatrix} h^j_{1,p} & h^j_{2,p} & ... & h^j_{M,p} \end{bmatrix} \begin{bmatrix} f_{1,p} \\ \cdot \\ \cdot \\ f_{M,p} \end{bmatrix} x_p$$

In this paper, we assume that the radiating elements from the same node share the same path gain α. Let $\alpha_{j,p}$ denote path gain for line of sight channel connecting j^{th} receiver to p^{th} transmitter.$\varphi_{j,p}$ denote the Angle of Departure connecting j^{th} receiver to p^{th} transmitter. $\mathbf{a}(\varphi_{j,p})^* = [1, e^{jdcos(\varphi_{j,p}}, ..., e^{jd(M-1)cos(\varphi_{j,p})})]$. Therefore:

$$y_{j,p} = \alpha_{j,p}\mathbf{a}(\varphi_{j,p})^*\mathbf{f}_p x_p \quad (2)$$

The received signal on the j^{th} receiver is a summation of signals delievered from all the transmitters to this receiver.

$$y_j = \sum_{p=1}^{L} \alpha_{j,p}\mathbf{a}(\varphi_{j,p})^*\mathbf{f}_p x_p \quad (3)$$

Let the wireless power transfer happening for duration P. Received energe on the j^{th} receiver for duration P is:

$$e_j = \int_0^P |y_j(t)|^2 dt = \int_0^P |\sum_{p=1}^{L} \alpha_{j,p}\mathbf{a}(\varphi_{j,p})^*\mathbf{f}_p x_p(t)|^2 dt \quad (4)$$

D. Wireless Power Transfer

The objective of energy beamforming control is to choose a beamforming code for each energy transmitting node such that the total received power is maximized while satisfying the minimum energy requirement of each energy receiver.

$$\begin{aligned} \underset{\mathbf{f}_p, \forall p}{\text{maximize}} \quad & \sum_{j \in \{1,2,..,L\}} e_j \\ \text{subject to} \quad & \mathbf{f}_p \in \mathcal{F}, \forall p; \\ & e_j \geq e_{min} \end{aligned}$$

III. Reinforcement Learning

A. problem setup

The impetus of reinforcement learning is that an agent can learn by interacting with the environment. In the intersection between control, optimization, and learning, the problem have different mathematical formulations. Here, we follow

the problem setup proposed by Richard Sutton in his book Introduction to Reinforcement Learning. [**?**]

Agent can observe the state at each step, denoted as S_t, where t is the t^{th} step taken. For our discussion, we focus only on the subset of problems where state s is fully observable by the agent. There are action choices for each state denoted as A_t. A reward is given for each action taken at step t denoted as R_t. The terminal step is denoted as t=T. For an episodic problem, T is a finite number, for a non-episodic problem, T=∞

An episode of data is registered as an alternating sequence of state, action, and reward:

$$S_0, A_0, R_0, S_1, A_1, R_1 .....S_{T-1}, A_{T-1}, R_{T-1}, S_T, A_T, R_T$$

Gain at step t is defined as the accumulative reward the agent can get from step t onward. A discounting factor γ between 0 to 1 is introduced to incorporate the sense of time, much like how interest rate encodes time in financial systems:

$$G_t := R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + ... + \gamma^{T-t} R_T \tag{5}$$

This can be written in its recursive form, known as Bellman Equation, which is the basis for an iteratively implemented backward induction algorithm:

$$G_t = R_t + \gamma G_{t+1} \tag{6}$$

Transition matrix is intruduced to encode the stochastidy in the environmental dynamics. Transaction Matrix $\mathcal{P}$ is defined as:

$$\mathcal{P}^a_{ss'} := Pr\{S_{t+1} = s' | S_t = s, A_t = a\} \tag{7}$$

State/Action Function q(s,a) is definied as expected gain starting from state s by taking action a:

$$\begin{aligned} q(s,a) :&= \mathbb{E}\{G_t | S_t = s, A_t = a\} \\ &= \mathbb{E}\{\sum_{k=0}^{T-t} \gamma^k R_{t+k+1} | S_t = s, A_t = a\} \end{aligned} \tag{8}$$

Policy is defined as:

$$\pi(s,a) := Pr(A = a | S = s) \tag{9}$$

Optimal Policy is defined as:

$$\pi^*(s) := \arg\max_a q(s,a) \tag{10}$$

Value Function v(s) is defined as the expected gain starting from state s:

$$\begin{aligned} v(s) :&= \mathbb{E}\{G_t | S_t = s\} \\ &= \mathbb{E}\{\sum_{k=0}^{T-t} \gamma^k R_{t+k+1} | S_t = s\} \\ &= \sum_{a \in \mathcal{A}} \pi(s,a) q(s,a) \end{aligned} \tag{11}$$

B. Without Approximation

One obvious approach to learning is to statistically construct a model of the environment, which is called Model-Based Learning. The most primitive form of model-based learning is Bellman Equation based backward induction. Statistical tactics, such as maximum likelihood, Bayesian methods, etc., can be deployed to approximate the model with the least amount of sampling. However, since the environment is implicitly embedded in v(s) and q(s,a), the model building process can be circumvented entirely, hence Model-Free Learning. Depending on whether the iteration rules is policy dependent, model-free learning can be subdivided into on-policy learning and off-policy learning.

One hindrance to the implementation of the brute force backward induction is its memory requirement. A more effective approach is to update q value and v value after one episode, one step, or n steps. They are called Monte Carlo Method, Temporal Difference Method, and $\lambda(n)$ Method respectively.

For online learning, ϵ-greedy Policy $\pi_\epsilon(s)$ is frequently deployed to balance exploration and exploitation, such that the environment can be encoded most efficiently. ϵ is initiated set to 1 and then asymptotically goes to 0 as the episode counts increases.

$$\pi_\epsilon(s,a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|} & \arg\max_a q_\epsilon(s,a) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases}$$

C. With Approximation

When the problem gets complex, state S becomes a rather large vector and function approximation with neuro networks can be utilized to facilitate learning. Reinforcement learning as a self-sustaining mathematical framework has been refined by Rich Sutton et al. since the 1980s. Only recently, the progress made with Deep Learning has been applied to the realm of Reinforcement Learning [**?**], rendering the computation tenable with existing hardare.

Let the value function and state/action function be parameterized with $\mathbf{w} : \hat{v}(s, \mathbf{w}) \approx v(s)$ and $\hat{q}(s, a, \mathbf{w}) \approx q(s, a)$

Let the i^{th} iteration of parameter be denoted as w_i. The Loss Function $\mathcal{L}(w_i)$ is defined as the following:

$$\mathcal{L}(w_i) := \mathbb{E}\{[v(s) - \hat{v}(s, w_i)]^2\} \tag{12}$$

$$\mathcal{L}(w_i) := \mathbb{E}\{[q(s,a) - \hat{q}(s, a, w_i)]^2\} \tag{13}$$

While the real value of v(s) and q(s,a) are not knowable, it can be approximated:

$$v(s) \approx \sum_{a \in A} R(s,a) + \gamma v(s', \mathbf{w}) \tag{14}$$

$$q(s,a) \approx r + \gamma \arg\max_a q(s', a', \mathbf{w}) \tag{15}$$

The Gradient of weighing paramter **w** can be derived from 12 and 13 with the real values substituted by 14 and 15 respectively. By convention, constant is omitted. Parameter is updated following Gradient Descent:

$$\mathbf{w}_i = \mathbf{w}_{i-1} - \nabla_{w_{i-1}} \mathcal{L}(w_{i-1}) \tag{16}$$

D. Policy Gradient Methods

Policy $\pi(s)$ can be written as a function parameterized by θ with s as input and a smooth distribution overall all actions as output.By adjusting parameter θ we can adjust the distribution over action choices for different states. This style of learning is called policy gradient-based learning.

Let us register a path sequence taken by the agent as τ such that the sequence is denoted as $\{S_{\tau 0}, A_{\tau 0}, R_{\tau 0} ... S_{\tau T}, A_{\tau T}, R_{\tau T}\}$. the gain of sequence τ is defined as the gain of this entire sequence of state, action, reward:

$$G(\tau) := \sum_{t=0}^{T} \gamma^t R_t \tag{17}$$

Denote $P(\tau, \theta)$ as the probability that path τ is travesed when the policy is parameterized by θ. The Objective Function can be defined in various ways. Here we adopt the definition as the following:

$$U(\theta) = \sum_{\tau} P(\tau, \theta) G(\tau) \tag{18}$$

The objective of the policy gradient method is to find the parameter θ to maximize the objective function.

The gradient of aforementioned utility function is:

$$\nabla_\theta U(\theta) = \nabla_\theta \sum_{\tau} P(\tau, \theta) G(\tau) \tag{19}$$

A mathematical sleight of hand called Importance Sampling is deployed to convert this theoretical expression of gradient into something that is algorithmically feasible.

$$\nabla_\theta U(\theta) \approx \frac{1}{N} \sum_{\tau=1}^{N} \sum_{t=0}^{T-1} \nabla_\theta ln \pi_\theta(s,a)|_{\theta_{old}} [q^{\pi_{\theta_{old}}}(s,a) - b] \tag{20}$$

We can use stochastic gradient descent(SGD) method to update θ:

$$\theta = \theta_{old} - \alpha \nabla_\theta ln \pi_\theta(s,a)|_{\theta_{old}} [q^{\pi_{\theta_{old}}}(s,a) - b] \tag{21}$$

Actor-Critic Method takes advantage of both policy gradient and function approximation to build a bootstrap structure that lead up to fast convergence. state/action function for policy $\pi_\theta(s)$ is approximated by $q^{\pi_\theta}(s, a, \mathbf{w})$.Baseline b is introduced into the bootstrap structure to foster convergence. Different algorithms define baseline differently. In advantage Actor-Critic algorithm, baseline is defined as a value function based on π_θ.Because the SGD updating process does not rely on the ordering of things, it is obvious that some of the aforementioned computations can be done asynchronously. Asynchronous Advantage Actor-Critic (A3C) is proven one of the most effective agents for renforcement learning, and is the one we will use in this paper.

E. Multiagent Reinforcement Learning

$A_t = \{A_t^1, A_t^2, ..., A_t^M\}$ M is the number of agents. The action space is cartician product of action choices available to each agent. $A_t(s) = A_t^1(s) \times A_t^2(s) \times ... \times A_t^M(s)$, which grows exponentially as the number of agents grows.

The Rollout method proposed by Dimitri Bertsekas breakdown this collective decision into its sequential components, reducing the complexity of action space while increasing the complexity of state space. It is proven that the intermediate state rollout method yields the same result as does the regular method. [**?**]

Without intermidiate state rollout, the sequences of data collected is: $...S_t, A_t, R_t, S_{t+1}...$ as shown in Figure 3

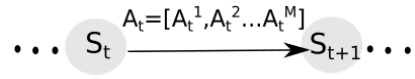

Fig. 3: Without Rollout

The intermediate states rollout technique converting action space complexity into state-space complexity by introducing intermediate states, denoted as S_t^k where k goes from 1 to M-1. The sequence of data is now: $...S_t, A_t^1, R_t^1, S_t^1, A_t^2, R_t^2, S_t^2, ..., S_t^{M-1}, A_t^M, R_t^M, S_{t+1}...$ as shown in Figure 4

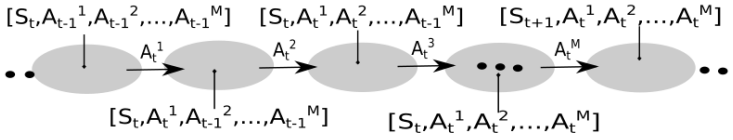

Fig. 4: With Rollout

suppose each agent has N choices. This formulation reduces the size action space from N^M to $N \times M$.

IV. Beamforming as a Multiagent Reinforcement Learning problem

A. Environment

The wirelessly powered communication network has L energy transmitting stations positioned at the corner of a 30m x 30m field. K energy receivers randomly scattered between 1m to 29m as illustrated by Figure 5. 0.5s of energy transfer is followed with 0.5s of information transfer. Assume no energy leftover at each cycle, such that at the beginning of the next energy transfer interval, the remaining power at each energy receiver is 0.

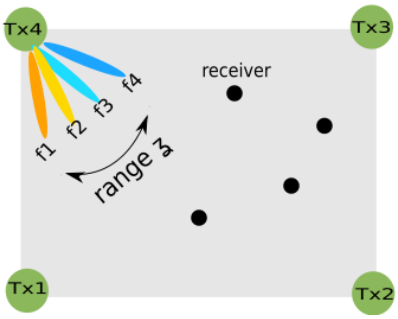

Fig. 5: Environment

Number of Energy Transmitting Nodes	L=4
Positions of Trasmitting Node	TX_1(0,0) TX_2(30,0) TX_3(30,30) TX_4(0,30)
Number of Radiation Elements per Trasmitting Node	M=64
Energy Carrier Frequency	8M Hz
Field of Energy Receivers	30m x 30m
Number of Energy Receivers	K
Energy Transfer Time	0.5s
Information Transfer Time	0.5s
Maximum Number of Steps	100

Observation Space:$\{e_1, e_2, ...e_K, c_1, c_2, c_3, c_4\}$ where e_j is the energy received at the j^{th} receiver, c_i is the codebook choice for the i^{th} energy emitting node.

Reward: If e_j < e_min, reward is deducted by 50 points each. If $e^n_{total}ew$ > $e^n_{total}ew$, reward is increased by 100 points. If $e^n_{total}ew$ < $e^n_{total}ew$, reward is deduced by 300 points.

B. A3C agent

Layers of Actor Network	3
Layers of Critic Network	3
Learning Rate for Actor	α_a=0.1
Learning Rate for Critic	α_c=0.1
Discount Rate	γ=0.9
Action Function	Softmax

C. Simulation Result

V. Conclusion

In this paper, we demonstrated the possibility to formulate WPT as a multiagent reinforcement learning problem, this lays the groundwork for further study towards fully locally computed control algorithms for wirelessly powered communication networks. Instead of group actions of all agents together, a multiagent rollout approach sees things sequentially, action taken by one agent becomes part of the state of another. This framework deduces the dimension of action space from exponential growth to multiplicative growth, and it can be applied to other problems. The most recent incarceration of beamforming technology is a passive reflective surface, or Intelligent Reflective Surface(IRS), where the reflective components are in the thousands. The multiagent approach proposed in this paper could be applied to IRS control as well, which should be a fruitful topic of future studies.

References

[1] A. A. Ahmadi and B. E. Khadir, "Learning dynamical systems with side information," ser. Proceedings of Machine Learning Research, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., vol. 120. The Cloud: PMLR, 10–11 Jun 2020, pp. 718–727. [Online]. Available: http://proceedings.mlr.press/v120/ahmadi20a.html

[2] A. Allibhoy and J. Cortes, "Data-driven distributed predictive control via network optimization," ser. Proceedings of Machine Learning Research, A. M. Bayen, A. Jadbabaie, G. Pappas, P. A. Parrilo, B. Recht, C. Tomlin, and M. Zeilinger, Eds., vol. 120. The Cloud: PMLR, 10–11 Jun 2020, pp. 838–839. [Online]. Available: http://proceedings.mlr.press/v120/allibhoy20a.html