Chandrashekar Lakshminarayanan

Mobile: +1(587)-778-2035
E-mail: chandrurec5@gmail.com

Short Summary My research work deals with developing Approximate Dynamic Programming (ADP) and Reinforcement Learning (RL) algorithms with provable performance and guaranteed convergence. I am also interested in applications of RL and stochastic optimization to problems arising in practice. Currently, I am a post-doctoral research fellow in the Department of Computing Science, University of Alberta, and I obtained my PhD from the Department of Computer Science and Automation, Indian Institute of Science.

Interests Markov Decision Processes, Approximate Dynamic Programming, Reinforcement Learning, Bandits, Machine Learning, Learning Theory, Stochastic Optimization, Stochastic Approximation and Stochastic Control.

Relevant Courses

- Stochastic Models and Applications, Stochastic Processes and Queuing Theory, Stochastic Approximation Algorithms
- Convex Optimization, Game Theory, Dynamics of Linear Systems
- Data mining, Pattern Recognition, Detection and Estimation Theory
- Real Analysis, Measure Theory, Topology (point-set), Linear Algebra

Thesis (Past Research)

Title:"Approximate Dynamic Programming and Reinforcement Learning - Algorithms, Analysis and an Application"
Advisor: Prof. Shalabh Bhatnagar (shalabh@csa.iisc.ernet.in), Department of Computer Science and Automation, Indian Institute of Science, Bangalore - 560012

Abstract: MDP is a useful mathematical framework to cast a variety of optimal sequential decision making problems under uncertainty in domains such as engineering, science and economics. However, computing optimal value function and optimal policy is difficult in practice because either the state space is too large or the model information is not available.

The primary investigations in the thesis were:

- *Approximate Dynamic Programming* refers to a gamut of methods that compute an approximate value function and a sub-optimal policy. The thesis investigated a widely used ADP method namely the Approximate Linear Programming (ALP) formulation. In particular, analytical tools were developed to bound the performance degradation that occurs when the constraint of the ALP are reduced or approximated. The analysis is based on ideas of monotone projections in tropical linear algebra.
- *Reinforcement Learning* algorithms are stochastic approximation (SA) schemes and solve the MDP by making use of sample trajectories. Actor-Critic algorithms are two timescale SA schemes since they make use of different step-size schedules. The thesis investigated the conditions under which two timescale SA schemes are stable and convergent.
- *Crowd Sourcing* is a new mode of organizing work in multiple groups of smaller chunks of tasks and outsourcing them to a distributed and large group of people in the form of an open call. An important task attribute that affects the completion time of a task is its price, and incorrect pricing leads to task starvation. In the thesis, the pricing problem is formulated in the MDP framework to compute a

pricing policy that achieves predictable completion times in simulations as well as real world experiments.

Publications (from the thesis)

Journal

- Chandrashekar, L. and Bhatnagar, S. "A Stability Criterion for Two Timescale Stochastic Approximation Schemes", Accepted for publication in Automatica, 2017.

International Conferences

- Chandrashekar, L.; Bhatnagar, S., "Approximate Dynamic Programming with $(\min, +)$ linear function approximation for Markov Decision Processes," 53rd IEEE Annual Conference on Decision and Control (CDC), December $15 - 17$, 2014, Los Angeles California, USA.
- Chandrashekar, L.; Bhatnagar, S., "A Generalized Reduced Linear Program for Markov Decision Processes," Twenty-Ninth AAAI Conference on Artificial Intelligence, January $25 - 30$, 2015, Austin Texas, USA.
- Chandrashekar, L.; Dubey, A.; Bhatnagar, S. and Chithralekha, B., "A Markov Decision Process framework for predictable job completion times on crowdsourcing platforms", Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, Pittsburgh, Nov. 2-4, 2014.
- Maity, R. K.; Chandrashekar, L.; Padakandla, S.; Bhatnagar, S., " Shaping Proto-Value Functions Using Rewards", European Conference on Artificial Intelligence 2016.

Work under consideration

- Chandrashekar, L.; Bhatnagar, S and Szepesvári C., "A Generalized Reduced Linear Program for Markov Decision Processes", under review in IEEE Transactions on Automatic Control.

Work in Progress

- "A stochastic gradient approach to parameter tuning in Hadoop".
- "Robust Temporal Difference Learning Algorithms".
- "Acceleration in Temporal Difference Learning Algorithms".

Patents

- Methods and Systems for Crowdsourcing of Tasks (US 20160071048 A1)
- System and method for improving dynamic performance of a circuit (US 7538701 B2)
- System and method for reducing power dissipation in an analog to digital converter (US 7821436 B2)

Industry Experience

Designation: Analog Design Engineer
Organization: Cosmic Circuits Pvt Ltd, Bangalore
Period: 3 years ($2005 - 2008$)
Role: I was part of the High Speed Analog to Digital Converters (ADC) Team. I was involved the following tasks:

- Design of blocks such as
  - Amplifiers with state-of-the-art feedback compensation schemes.
  - Comparators.
  - Digital Error Correction Logic.
- Full system simulation, testing and debugging of 10 to 12 bit ADCs.
- Worked in CMOS technologies varying from 0.360μm to 65nm.

Industrial Collaboration

Organization: Xerox Research Center India
Project Title: Methods and Systems for Crowdsourcing of Tasks
Period: $2012 - 2014$
Objective: To propose automatic pricing of tasks posted in a crowdsourcing platform
Co-investigator: Dr. Suijt P. Gujar and Dr. Chithralekha Balamurugan, Xerox Research Center, India, and Dr. Shalabh Bhatnagar and Srujana Sadula, Indian Institute of Science, Bangalore.
Achievements: Filed a patent titled "Methods and Systems for Crowdsourcing of Tasks (US 20160071048 A1)" and a published a paper titled "A Markov Decision Process framework for predictable job completion times on crowdsourcing platforms", Proceedings of the Second AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, Pittsburgh, Nov. 2-4, 2014.

Talks and Posters

- *Approximate Dynamic Programming with* $(\min, +)$ *linear function approximation for Markov Decision Processes*, 53rd IEEE Annual Conference on Decision and Control (CDC), December $15 - 17$, 2014, Los Angeles California, USA.
- *A Generalized Reduced Linear Program for Markov Decision Processes*, Twenty-Ninth AAAI Conference on Artificial Intelligence, January $25 - 30$, 2015, Austin Texas, USA.
- *A Generalized Reduced Linear Program for Markov Decision Processes*, 3^{rd} IKDD Conference on Data Science, March $13 - 16$, 2016, Pune, India.
- *Introduction to Reinforcement Learning*, I-CARE Big Data Analytics and Cognitive Computing winter school, Bangalore 2014.
- *Introduction to Probability*, Computer Science and Automation Undergraduate Summer School, Bangalore 2013.
- *Introduction to Linear Algebra*, Computer Science and Automation Undergraduate Summer School, Bangalore 2014.

Accomplishments

1. I was All India Rank 1 in the Graduate Aptitude Test in Engineering conducted in the year 2008 with a score of 1000/1000.
2. Presented with Medal of Honor for Best Pass out student in Systems Science and Automation, Indian Institute of Science.
3. Best research presentation award in the IIST-Research Scholars Day held at Indian Institute of Space Technology, Trivandrum, December 2013.
4. Best research presentation award in the Electrical Engineering and Computer Science Colloquium held at IISc, Bangalore, February 2015.

Academics

Ph.D.[2010-2015]
Department of Computer Science and Automation (CSA),
Indian Institute of Science (IISc), Bangalore, India.

M.E. [2008 - 2010]
Systems Science and Automation,
Department of Electrical Engineering (EE),
Indian Institute of Science (IISc), Bangalore, India.
C.G.P.A: 7.2/8

B.Tech. [2001 - 2005]
Instrumentation and Control Engineering,
National Institute of Technology, Tiruchirapalli.
C.G.P.A: 8.61/10

References

Csaba Szepesvari (Professor),
Department of Computing Sciences,
University of Alberta,
Edmonton, Canada, T6G2E8
e-mail: szepesva@ualberta.ca

Shalabh Bhatnagar (Professor),
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore-560012.
e-mail: shalabh@csa.iisc.ernet.in

Y. Narahari (Professor),
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore-560012.
e-mail: hari@csa.iisc.ernet.in

Shirish Shevade (Professor),
Department of Computer Science and Automation,
Indian Institute of Science,
Bangalore-560012.
e-mail: shirish@csa.iisc.ernet.in

Teaching Plan

Chandrashekar L

I owe it to my teachers and colleagues for all that I have learnt, and I sincerely hope to draw inspiration from them while teaching young minds. I believe teaching is not merely repeating, but an opportunity to present ideas in a original and refreshing way, a process which is not only an end in itself but a key stepping stone for research.

I. Short Term Goals

My course work, experience as an analog design engineer and research training at the Indian Institute of Science and the University of Alberta, enables me to offer a variety of courses related to computer science, electrical engineering and basic mathematics at the undergraduate level. Some of the courses (not limited to)

Computer Science

1) Design and Analysis of Algorithms[9]
2) Programming and Data Structure
3) Probability, Stochastic Process and Statistics[20]

Electrical Engineering

1) Digital Design [15]
2) Microprocessors [12]
3) Network/Circuit Theory [13]
4) Analog Electronics [21]
5) Linear Integrated Circuits [10]
6) Signals and Systems [17]
7) Classical and Modern Control Theory [16]

Basic Engineering Mathematics

1) Linear Algebra [11]
2) Engineering Mathematics [14]

Laboratory

1) Programming and Data Structures
2) Digital and Analog Circuits

II. Long Term Goals

I am also interested in designing advanced undergraduate and graduate level courses related to my field of expertise namely Reinforcement learning. Reinforcement Learning (RL) algorithms learn from samples obtained via direct interaction with the evinornment, and are different from supervised or unsupervised machine learning algorithms. Reinforcement learning lies at the intersection of machine learning (ML), stochastic control, optimization and operations research. My primary research focus would be RL and I would like to design courses that enable students to be up to speed with topics in RL. To this end, I would like to design the following advanced undergraduate and graduate courses in ML and RL.

A. Introduction to Reinforcement Learning

Introduction to reinforcement learning, introduction to stochastic dynamic programming, finite and infinite horizon models, the dynamic programming algorithm, infinite horizon discounted cost and average cost problems, numerical solution methodologies, full state representations, function approximation techniques, approximate dynamic programming, partially observable Markov decision processes, Q-learning, temporal difference learning, actor-critic algorithms.

B. Introduction to Machine Learning

Introduction to machine learning. Classification: nearest neighbor, decision trees, perceptron, support vector machines, VC-dimension. Regression: linear least squares regression, support vector regression. Additional learning problems: multiclass classification, ordinal regression, ranking. Ensemble methods: boosting. Probabilistic models: classification, regression, mixture models (unconditional and conditional), parameter estimation, EM algorithm. Beyond IID, directed graphical models: hidden Markov models, Bayesian networks. Beyond IID, undirected graphical models: Markov random fields, conditional random fields. Learning and inference in Bayesian networks and MRFs: parameter estimation, exact inference (variable elimination, belief propagation), approximate inference (loopy belief propagation, sampling). Additional topics: semi-supervised learning, active learning, structured prediction

III. Learning Outside of Classrooms

Having been a student at the National Institute of Technology, Trichy and at the Indian Institute of Science, I believe that a lot of learning happens when students discuss problems in groups. Hence, I would be interested in moderating and guiding 'Technology' clubs by students. For example, a robotics club might a nice place for students from the various disciplines (such as mechanical, electrical and computer science) to get a hands-on experience in building systems and to get a taste of open ended problems. Technical skills apart, interaction amongst peers can help the students in improving their soft skills such as team co-ordination, communication, and ability to set goals etc. By publishing the activities of the club as blogs/videos, the student community can interact with fellow researchers from across the world.

IV. Experience

I now list some of my relevant teaching and mentoring experience.

- **Teaching Assistantship:** At the Indian Institute of Science, I was a teaching assistant (TA) for courses such as Linear Algebra and Applications (Aug-Dec 2012, 2013, 2014) and Foundations of Data Sciences (Jan-May 2014). My duties as a TA involved conducting tutorials, setting quizzes and evaluating scripts.
- **Summer School:** I was a member of the organizing committee for the Undergraduate Summer School [1] conducted by the Department of CSA, IISc. The main aim of the summer school was to introduce undergraduate students to cutting-edge research in computer science via talks, demos, and hands-on sessions. I have given expository talks on linear algebra [2] and probability at the UG summer school. As a member of the organizing committee I was involved in the identification of interesting topics/speakers and selection of candidates.
- **Project Guidance:** I have guided masters students of the Stochastic Systems Lab (SSL) at Dept. of CSA, IISc. My work in crowdsourcing [7] was a joint effort with Masters students of the SSL.

References

[1] CSA Undergraduate Summer School, http://events.csa.iisc.ernet.in/summerschool2014/.

[2] Introduction to Linear Algebra, https://www.youtube.com/watch?v=qr-icg9wrz8.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume II. Athena Scientific, Belmont, MA, 4th edition, 2013.

[4] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. TRIM, 2008.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] L. Chandashekar and S. Bhatnagar. A generalized reduced linear program for markov decision processes. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA.*, pages 2722–2728, 2015.

[7] L. Chandrashekar, A.Dubey, S. Bhatnagar, and B. Chithralekha. A markov decision process framework for predictable job completion times on crowdsourcing platforms. In *Proceedings of*

the Seconf AAAI Conference on Human Computation and Crowdsourcing, HCOMP 2014, November 2-4, 2014, Pittsburgh, Pennsylvania, USA, 2014.

[8] L. Chandrashekar and S. Bhatnagar. Approximate dynamic programming with (min; +) linear function approximation for markov decision processes. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*, pages 1588–1593, 2014.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

[10] S. Franco. *Design with operational amplifiers and analog integrated circuits*. McGraw-Hill, 2001.

[11] Gilbert G. S. Strang. *Introduction to Linear Algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.

[12] Ramesh S Gaonkar. *Microprocessor architecture, programming, and applications with the 8085*. Prentice-Hall, Inc., 1995.

[13] W. H. Hayt, J. E. Kemmerly, and S. M. Durbin. *Engineering circuit analysis*. McGraw-Hill New York, 1986.

[14] E. Kreyszig. *Advanced engineering mathematics*. John Wiley & Sons, 1988.

[15] M. M. Mano. *Digital design*. EBSCO Publishing, Inc., 2002.

[16] Katsuhiko Ogata. *Modern control engineering*. Prentice Hall PTR, 2001.

[17] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals and systems*. Pearson, 2014.

[18] H. V. Poor. *An introduction to signal detection and estimation*. Springer Science & Business Media, 2013.

[19] S. M. Ross. *Stochastic processes*, volume 2. John Wiley & Sons New York, 1996.

[20] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[21] A. S. Sedra and K. C. Smith. *Microelectronic circuits*. Holt, Rinehart and Winston, 1982.

[22] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

Algorithms for Planning and Decision making Under Uncertainty

Research Proposal, Chandrashekar L

I. Introduction

Real world is often uncertain, starting from the time taken to service a customer in a queue, to the return-on-investment in a particular stock. Decision making in the face of such uncertainty is hence crucial while designing practical systems with good performance. As Fig. 1 illustrates, *decision making under uncertainty* is at the heart of several important research areas that have had major practical impact in the last few decades. In practice, the underlying uncertainty is captured using an appropriate stochastic model and an optimal decision is made based on the model. However, in complex real world systems, due to the *curse-of-dimensionality* deriving such decision rules often become computationally intractable. While dimensionality free methods [3] have had partial success in alleviating this problem, it is still an active area of research. More importantly, with the advent of the big data era (**http://dst.gov.in/big-data-initiative-1**) , it is crucial to develop decision making algorithms which are data driven (Fig. 2) as opposed to just being model based.

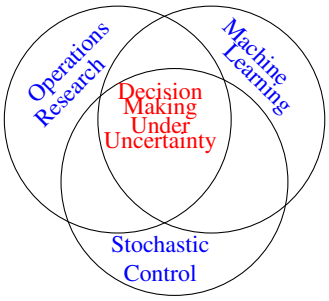

Fig. 1: Decision making under uncertainty is at the heart of many important areas such as Machine Learning, Operations Research and Stochastic Control.

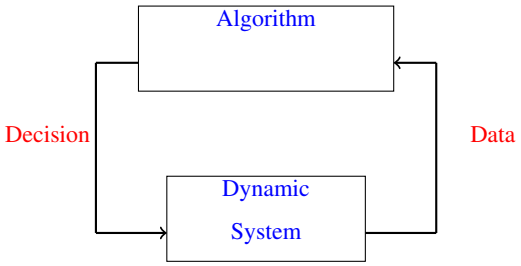

Fig. 2: The paradigm of data-driven approach to decision making is characterized by 'system-in-loop' or 'feedback' control.

II. Background

Fig. 3 shows the various aspects to be taken into account while adopting a data-driven approach towards decision making. In what follows, we discuss these aspects briefly before highlighting the challenges and the research gaps that this proposal wishes to address.

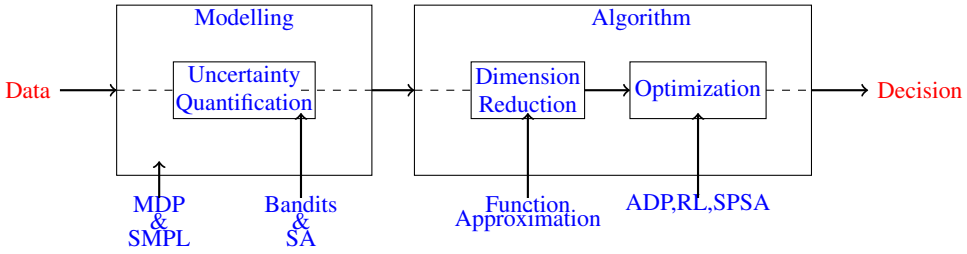

Fig. 3: The figure shows the various steps starting with *data*, followed by *modelling* and then by *algorithm* leading to the *decision rule* . The modelling step involves *uncertainty quantification* and the algorithmic step can further be split into *dimensionality reduction* and *optimization*. Markov Decision Processes (MDPs) and Stochastic Max-Plus (SMPL) system are useful mathematical models of uncertainty. Multi-arm bandit and Stochastic Approximation (SA) theories helps us in uncertainty quantification. Function approximation techniques are used to address the curse-of-dimensionality. *Approximate Dynamic Programming* (ADP), *Reinforcement Learning* (RL) and *Simultaneous Perturbation Stochastic Approximation* (SPSA) algorithms are optimization routines that eventually lead to the final decision rule.

A. Modeling

Markov Decision Processes (MDP): Many real world decision making problems such as control of sensor networks, control of queuing systems, traffic control, inventory control, terrain exploration by robots, sequential drug administration, allocation of tasks and payments in service systems etc are examples of sequential decision making problems that can be modeled as MDP. Further, the Markovian nature of the system dynamics enables MDPs to be solved using the principle of Dynamic Programming (DP) leading us to the optimal decision rule.

Stochastic Max-Plus (SMPL) Systems: are the class of discrete event systems where elementary components interact via 'synchronization' [13]. The inherent randomness in the different interacting components causes the stochastic behavior. Examples of such systems include railway networks, production chain, scheduling, queuing and digital systems.

B. Uncertainty Quantification

Multi-Armed Bandit: Theory [2] characterizes the classical *exploration-exploitation* trade-off. Bandit theory addresses the question of optimal sampling and at the same time with lesser compromise in performance, and is helpful in scenarios where samples need to be collected in an efficient way.

Stochastic Approximation (SA): Theory [7] helps us to build data-drive algorithms wherein sample data can be plugged in the place of the unknown ground truth.

C. Algorithm

Dimensionality Reduction: *Curse-of-dimensionality* denotes the fact that the number of states in the system grows exponentially in the number of state variables. Function approximation is a widely used dimensionality reduction wherein the each state is represented by a feature vector and computations are carried by taking linear combinations of the features. Dimensionality reduction is achieved by letting the dimension of the feature vector to be much less than the number of states.

Approximate Dynamic Programming: (ADP) algorithms [3] tackle the complexity of systems (i.e, of large number of states) by cleverly combining approximate representations and mathematical optimization.

Reinforcement Learning: In most cases, the underlying system model in not known explicitly. However, the samples can be obtained via simulation or direct interaction with the system. Reinforcement Learning (RL) algorithms [16] learn the optimal decision via feedback obtained by directly interacting with the system. RL algorithms such as Q-learning [18], temporal difference learning [17] have been successful in domains such as Backgammon, elevator control etc.

III. Objective: Open Problems and Potential Directions

We now list below several research gaps that we propose to address (Table I).

1) **ADP with Performance Guarantees:** Since the ADP methods are not exact, in most cases they only compute approximate or sub-optimal decision rules. Several well known ADP algorithms suffer from convergence issues [4] and the performance loss of the sub-optimal policy cannot be ascertained. However, some of our recent works [11, 14] have shown conditions under which ADP algorithms are convergent and yield provably *good* decision rule. The focus here is to explore newer ADP algorithms and approximation architectures that can be guaranteed to converge and yield a good decision rule.
2) **ADP for Constrained and Risk Sensitive MDPs:** The discounted cost and the average cost are the most widely studied formulations in the classical MDP setting. However, in practice the decisions need to balance multiple costs and sensitivities to the various costs can be different. Constrained MDP formulation allows us to take into account the various costs involved. However, ADP algorithms for constrained MDPs have not been sufficiently explored. The Approximate Linear Programming (ALP) [12] approach to MDP can accommodate multiple cost functions. A possible research direction is to extend [14] to constrained MDP to derive new ADP algorithms. *Risk* sensitive formulation of the

Problem	ADP	RL	Applications
Constrained MDP	*	[6], +	Urban Planning
Risk Sensitive MDP	*	[8], +	Health Care
SMPL	[13]	*	Traffic
Bandit	[1]	[2], +	Crowdsourcing

TABLE I: Summary of open problems and potential contributions. Here '$*$' denotes potential for fundamental contribution, and '$+$' denotes scope for significant additions.

MDP considers exponential cost and hence can be used in applications where risk avoidance is key. Exploring ADP algorithms for the risk sensitive setting is another line of research.

3) **ADP for Stochastic Max-Plus Systems:** Most of the current approaches are based on model predictive control [13]. Exploring machine learning and stochastic optimization [5] inspired ADP algorithms for SMPL systems is of interest.
4) **Reinforcement Learning:** It is of interest to explore sample efficient, stable and convergent RL algorithms for systems with large number of states especially for the constrained, risk sensitive formulations.
5) **Multi-Arm Bandits:** Several variants of the problem including linear/convex/contextual bandits, bandits with complex or partial feedback have been considered in literature. While upper bounds for regret is known in various cases finding matching lower bounds is a challenging research direction that can be explored.

IV. Significance: Applications and Impact

Addressing the open problems in Section III will help us in developing algorithms for the following domains.

Urban Planning: has separate costs related to the development of public transportation (roads/rail), facility development (housing, hospitals, industries) and resource distribution (power, water and sewage). Further, only a fixed budget is available at any given time. We propose *Constrained MDPs* that will help us to take care of these multiple costs. Also, ADP algorithms for SMPL systems can be applied to synchronize commuter traffic across the road and rail networks.

Health Care: An interesting recent advancement is to use data-driven decision making in the domain of health care. In particular, exploiting the patient specific data (history, congenital defects etc) to decide the remedy/drug-dosage to be administered from time to time is a challenging problem [15]. Another potential challenge is to use mobile devices for patient monitoring and for preventive interventions (alerting patients on the intake of medicine, therapy and rest). The algorithms for *Risk Sensitive* MDPs can be applied to factor the *risk* associated with wrong treatments.

Sensor Networks: Sensors serves as an important backbone of real-time data-driven decision making. In general, most sensors are mobile and battery operated, it is important to spend as less battery power as possible. As a result energy efficient transmission has attracted the attention in recent times [9]. This can be achieved by forming a network of sensors with one centralized controller that collates information from other nodes, and the center also has control over the *sleep-wake* schedules of all the other nodes. Multi-agent RL algorithms can be used to derive decentralized decision rules.

Human Computing: Crowdsourcing (crowd) is a new mode of organizing work in multiple groups of smaller chunks of tasks and outsourcing them to a distributed and large group of people in the form of an open call. We propose Bandit theory can be used to develop algorithms that use real time data (number of workers in the crowd, the number of pending tasks, the quality of the workers etc) to derive efficient decisions for pricing the tasks [10], allocating them to the crowd workers to their ensure timely completion.

In all the application domains, we propose to develop new algorithms that are robust, scalable and real time, and in cases where previous work exists, extend significantly the state of the art by gaining novel

A Stability Criterion for Two Timescale Stochastic Approximation Schemes

Chandrashekar Lakshminarayanan [a], Shalabh Bhatnagar [a],

[a] *Department of Computer Science and Automation,*
Indian Institute of Science,
Bangalore-560012,
India.

Abstract

We present *the first* sufficient conditions that guarantee stability of two-timescale stochastic approximation schemes. Our analysis is based on the ordinary differential equation (ODE) method and is an extension of the results in [8] for single-timescale schemes. As an application of our result, we show the stability of iterates in a two-timescale stochastic approximation scheme arising in reinforcement learning.

Key words: Simulation, two-timescale stochastic approximation, stability of iterates, limiting ODE, reinforcement learning.

1 Introduction

Many applications require stochastic computations in *nested* loops. For instance, consider parameter tuning in the case of a queuing system. Here, the performance of the system for a given parameter setting needs to be estimated first and then the parameters have to be tuned based on the estimation. Thus the estimation constitutes the *inner* loop and the parameter tuning, the *outer* loop. Another example of such nested computations arises in reinforcement learning (especially actor-critic algorithms [5]), wherein, the inner loop estimates the value of the agent's behavior while the outer loop tunes the agent's behavior. A large class of such nested stochastic computations fall under the category of two timescale stochastic approximation schemes [6, 2, 3, 4] given as under.

$$x_{n+1} = x_n + a(n)[h(x_n, y_n) + M^{(1)}_{n+1}], \tag{1a}$$

$$y_{n+1} = y_n + b(n)[g(x_n, y_n) + M^{(2)}_{n+1}], \tag{1b}$$

where, $x_n \in \mathbf{R}^{d_1}$, $y_n \in \mathbf{R}^{d_2}, n \geq 0$ are the iterates, $h\colon \mathbf{R}^{d_1+d_2} \to \mathbf{R}^{d_1}$, $g\colon \mathbf{R}^{d_1+d_2} \to \mathbf{R}^{d_2}$ are Lipschitz continuous functions, $M^{(i)}_n, i = 1, 2$ are (martingale difference) noise terms and $a(n)$ and $b(n)$ are both diminishing step-size schedules such that $\frac{b(n)}{a(n)} \to 0$ as $n \to \infty$. The term *stochastic approximation* (SA) signifies that the quantities $h(\cdot)$ and $g(\cdot)$ are corrupted by noise. The term *two timescale* signifies the fact that there are two different step-size schedules namely $a(n)$ and $b(n)$, with x_n and y_n being the corresponding *faster* and *slower* timescale iterates respectively (due to the relative magnitude of the step-sizes). The condition $\frac{b(n)}{a(n)} \to 0$ ensures that, at any given instant, the slower timescale iterates do not change as much compared to the faster timescale iterates, thereby producing an analogous effect as a *nested*-loop.

Under certain suitable conditions [10, 7] one can analyze the convergence of the iterates in (1). For instance, in the case of reinforcement learning (RL), y_n dictates the agent's behavior i.e., policy, and x_n estimates the value accumulated by the behavior. Thus for a given RL algorithm [5], it is of interest to show that $y_n \to y^*$, $x_n \to x^*$, where $y^* \in \mathbf{R}^{d_2}$ and $x^* \in \mathbf{R}^{d_1}$ are the best agent's behavior/policy and the value the agent accumulates under the same. An important condition required to show convergence [10, 7] is the boundedness/stability of the iterates i.e., $\sup_n ||x_n|| < \infty$ and $\sup_n ||y_n|| < \infty$. It is always possible to project the iterates after each update onto a compact set $\mathcal{C} \subset \mathbf{R}^{d_1+d_2}$ to ensure their boundedness. However, a shortcoming of such an approach is that the desired solution (i.e. (x^*, y^*)) might lie outside of the compact set $\mathcal{C}$. Thus it is of interest and importance to be able to analytically establish the stability of the iterates (without making use of projections).

In this paper, we present the conditions that imply stability of iterates in a two timescale SA scheme, and apply our anal-

Email addresses: chandrul@csa.iisc.ernet.in (Chandrashekar Lakshminarayanan), shalabh@csa.iisc.ernet.in (Shalabh Bhatnagar).

ysis to establish the stability of iterates in a two-timescale SA scheme arising in an application in reinforcement learning [5]. Our analysis is based on the ordinary differential equation (ODE) method [7, 6, 8] for stochastic approximation schemes. In what follows, in Section 2 we first present an overview of the convergence and stability analysis [8] of single timescale SA schemes (a degenerate case of two timescale SA schemes). In Section 3, we state the assumptions as well as discuss our contributions. Sections 4, 5.1 and 5.2 contain the main body of our analysis (Theorem 7 and Theorem 10). In Section 6, we use our analysis to show the stability of iterates for a two timescale RL algorithm presented in [5] and also present a numerical example. And finally we provide our concluding remarks in Section 7.

2 Overview of the ODE method

Consider the following (single timescale) SA scheme:

$$z_{n+1} = z_n + a(n)[f(z_n) + M_{n+1}], n \geq 0, \qquad (2)$$

where $z_n \in \mathbf{R}^d$ are the iterates, $f: \mathbf{R}^d \to \mathbf{R}^d$ is a Lipschitz continuous function, M_{n+1} are zero-mean noise terms and $a(n)$ are diminishing step-sizes. Notice that (2) is a degenerate version of (1), i.e., it has only one step-size schedule $a(n)$ as opposed to two step-sizes $a(n)$ and $b(n)$. The ODE method has been employed to analyze the convergence of single timescale SA schemes [8] as well as two timescale SA schemes [6]. While an ODE based stability analysis for single timescale SA schemes exists [8], there exists no ODE based stability analysis in the literature for two timescale SA schemes. Before we present our stability analysis for two timescale SA schemes, we present an overview of the ideas involved in the convergence and the stability analysis of single timescale SA schemes in this section. The presentation here is only aimed at motivating the ODE method and we refer the reader to [7] for a detailed presentation.

Convergence Analysis: The ODE method considers the iterates $\{z_n\}$ of (2) as a noisy-discretized version of the trajectory $\phi(t, z_0)$, $t \geq 0$ of the ODE $\dot{z}(t) = f(z(t))$ with initial condition $z_0 \in \mathbf{R}^d$. In order to study $\{z_n\}$, $n \geq 0$, which is a countable set, and the trajectory $\phi(t, z_0)$, that is a function of continuous time t (for a given initial condition z_0), we need the notions of *timescale* and *interpolated trajectory*. The time instants that define timescale are $t(n) \stackrel{def}{=} \sum_{i=0}^{n-1} a(i)$ and the interpolated trajectory is defined as $\bar{z}(t) = z_n + (z_{n+1} - z_n)\frac{t-t(n)}{t(n+1)-t(n)}, t \in [t(n), t(n+1)]$. We can then show that $\bar{z}(t)$ 'tracks' $\phi(t, z_0)$ (Lemma 1, Chapter 2, [7]), i.e. the difference between them can be made arbitrarily small. Since, z_ns are embedded in $\bar{z}(t)$, it can be said that $\{z_n\}$ tracks the ODE $\dot{z}(t) = f(z(t))$. The convergence analysis can be captured formally (i.e., the statement of Theorem 2, Chapter 2, [7]) as follows.

Theorem 1 *Almost surely, the sequence $\{z_n\}$ generated by (2) converges to a (possibly sample path dependent) compact connected internally chain transitive invariant set of the ODE $\dot{z}(t) = f(z(t))$.*

In many applications, we have $f(z) = \nabla J(z)$, for some performance measure $J: \mathbf{R}^d \to \mathbf{R}^d$, and in such cases Theorem 1 implies that the iterates will converge to the set $\{z : \nabla J(z) = 0\}$, i.e., the iterates converge to a local extremum or stay within a compact connected set of local extrema.

There are two sources of error that need to be contained while analyzing the iterates of an SA scheme via its corresponding ODE. These are, the discretization error, and the error due to the noise term. Convergence analysis based on the ODE method can be carried out under suitable assumptions (**(A1)-(A4)**, Chapter 2 of [7]) involving boundedness of the iterates, Lipschitz continuity of f, non-summability and square summability of the step-sizes and quadratic variation of the martingale noise terms to contain these errors. In particular, Lipschitz continuity helps in containing the discretization error and the conditions on the quadratic variation and step-sizes help to contain the error due to the noise.

Stability Analysis: The *rescaling* technique (Chapter 3, [7]) is an important method used to establish the stability of single timescale SA schemes. The aim of the stability analysis is to show that the iterates are bounded (the convergence analysis needs this as an assumption see **(A4)**, Chapter 2 of [7], Theorems1-8 of [10]). Since the boundedness of iterates is not known, in the case when the iterates go out of the unit ball around the origin, we can rescale the iterates into the unit ball using an appropriate scaling factor $c \geq 1$. This rescaling can be done every $T > 0$ instants in a periodic manner. This periodic rescaling gives rise to scaled iterates $\hat{z}_n$, scaled functions $f_c(z) \stackrel{def}{=} f(cz)/c, c \geq 1$, scaled interpolated trajectory $\hat{z}(t)$ and a scaled ODE $\dot{z}(t) = f_c(z(t))$. The convergence analysis holds for the scaled iterates (since their boundedness is ensured by construction). Consider the case when the iterates are not bounded, then the iterates have to keep leaving the unit ball, which in turn implies that the scaling factor $c \to \infty$. Then the iterates have to track the limiting ODE given by $\dot{z}(t) = f_\infty(z(t))$, where $f_\infty(z) \stackrel{def}{=} \lim_{c\to\infty} f_c(z)$. Now, if the limiting ODE has the origin as its unique globally asymptotic stable equilibrium, then the iterates have to fall at an exponential rate into the unit ball. Thus, in order to escape to infinity, the sizes of the jumps from within the unit ball to its outside should be unbounded, but an application of Gronwall's inequality (see Lemma 6, Chapter 3, [7]) shows that this cannot happen. This in turn implies that the iterates are stable and bounded. This assumption of existence of stable equilibrium on the limiting ODE is weaker than assuming stable equilibrium for the original ODE (consider the case when $f^1(z) = -z + cos(z)$ for which the corresponding $f^1_\infty(z) = -z$).

The stability analysis (Chapter 3 of [7]) requires assumptions (**(A5)**, Chapter 3 of [7]) on the existence of scaled

and limiting functions and the corresponding ODEs. Since the iterates track a scaled ODE, the analysis also needs results (Lemmas 1, 2 and Corollary 3, Chapter 3 of [7]) that show that the scaled ODE tracks the limiting ODE for large enough c.

3 Assumptions and Contributions

The stability analysis available for single timescale SA schemes is not directly applicable for two timescale SA schemes. In particular, the iterates evolve in both the timescales simultaneously and it not enough to rescale in the faster or the slower timescale alone. As it turns out, in this paper, we rescale the iterates in both the timescales and the analysis involves scaled as well as limiting ODEs in the faster as well as the slower timescales. To this end, we assume the following conditions (A1-A5) and show that they imply the stability of the iterates in the two timescale SA scheme in (1).

A1 $h\colon \mathbf{R}^{d_1+d_2} \to \mathbf{R}^{d_1}$ and $g\colon \mathbf{R}^{d_1+d_2} \to \mathbf{R}^{d_2}$ are Lipschitz continuous functions.

A2 $\{M_n^{(1)}\}$, $\{M_n^{(2)}\}$ are martingale difference sequences w.r.t. the increasing sequence of σ-fields $\{\mathcal{F}_n\}$, where $\mathcal{F}_n \stackrel{def}{=} \sigma(x_m, y_m, M_m^{(1)}, M_m^{(2)}, m \leq n), n \geq 0$, with $\mathbf{E}[\| M_{n+1}^{(i)} \|^2 | \mathcal{F}_n] \leq K(1+ \| x_n \|^2 + \| y_n \|^2), i = 1, 2, n \geq 0$, for some constant $K > 0$.

A3 $\{a(n)\}$, $\{b(n)\}$ are two step-size schedules satisfying $a(n) > 0$, $b(n) > 0$, $\sum_n a(n) = \sum_n b(n) = \infty, \sum_n (a(n)^2 + b(n)^2) < \infty$, $\frac{b(n)}{a(n)} \to 0$ as $n \to \infty$.

A4 The functions $h_c(x, y) \stackrel{def}{=} \frac{h(cx,cy)}{c}$, $c \geq 1$, satisfy $h_c \to h_\infty$ as $c \to \infty$, uniformly on compacts for some h_∞. Also, the *limiting* ODE $\dot{x}(t) = h_\infty(x(t), y)$ has a unique globally asymptotically stable equilibrium (a.s.e.) $\lambda_\infty(y)$, where $\lambda_\infty\colon \mathbf{R}^{d_2} \to \mathbf{R}^{d_1}$, is a Lipschitz map. Further $\lambda_\infty(0) = 0$, i.e., the ODE $\dot{x}(t) = h_\infty(x(t), 0)$ has the origin in $\mathbf{R}^{d_1}$ as its unique globally a.s.e.

A5 The functions $g_c(y) \stackrel{def}{=} \frac{g(c\lambda_\infty(y),cy)}{c}, c \geq 1$, satisfy $g_c \to g_\infty$ as $c \to \infty$, uniformly on compacts for some g_∞. Also, the *limiting* ODE $\dot{y}(t) = g_\infty(y(t))$ has the origin in $\mathbf{R}^{d_2}$ as its unique globally asymptotically stable equilibrium.

While convergence of two timescale iterates have been shown in prior work [6, 10, 7], they nevertheless assume boundedness of the iterates (in this paper, however, we provide conditions that imply boundedness). Further, conditions A4 and A5 are *weaker* (see Section 2 or Chapter 3 of [7]) than **(A4), (A5)**, Chapter 6, [7] or A3 and A4 of [10]. We present our arguments in two parts, one part each for the two timescales (Section 5.1 for the faster timescale and Section 5.2 for the slower timescale). Assumptions A1, A2 and A3 are the same as those required for the convergence analysis of two timescale SA schemes (Chapter 6, [7]). We require A1, A2 and A3 to reuse the results from Chapter 6, [7] and show that the scaled iterates are convergent (both in the faster as well as the slower timescale). A4 and A5 summarize the conditions on the scaled and limiting ODE in the faster and the slower timescales respectively. While the scaled ODE and the limiting ODE in the case of single timescale SA schemes are homogeneous ODEs (see **(A5)**, Chapter 3, [7]), the scaled ODE and the limiting ODE in A4 have an external input. For our arguments in Section 5.1, we would require the results on the scaled and the limiting ODE in A4. To this end, we extend the results known (Lemmas 1, 2 and Corollary 3, Chapter 3 of [7]) for homogeneous ODEs to ODEs with external input in Section 4. However, the scaled and the limiting ODEs for the slower timescale are homogeneous ODEs and hence we reuse the results in Chapter 3, [7].

4 Results on ODE with external input

The ODE method for analysis of single timescale SA schemes deals with homogeneous ODEs such as $\dot{z}(t) = f(z(t))$. However, in the case of two timescale iterates we need to study coupled ODEs i.e., those with external input such as in A4. In this section, we extend the results known for homogeneous ODEs in Chapter 3, [7], to ODEs with external inputs. To this end, we define the following:

$D1$ Let $\eta^{y(t)}(t, x)$ denote the solution to the ODE $\dot{x}(t) = h(x(t), y(t))$, $t \geq 0$, with initial condition $x \in \mathbf{R}^{d_1}$ and the external input $y(t) \in \mathbf{R}^{d_2}$. The superscript $y(t)$ in $\eta^{y(t)}(t, x)$ is to be understood as a symbol that indicates the fact that the external input is $y(t)$.

$D2$ Let $B(x_0, r) = \{x\colon \| x - x_0 \| \leq r\}$ denote the closed ball of radius r around x_0.

$D3$ Let $\eta_c^{y(t)}(t, x)$ and $\eta_\infty^{y(t)}(t, x)$ be solutions to the ODEs

$$\dot{x}(t) = h_c(x(t), y(t)), \tag{3}$$
$$\dot{x}(t) = h_\infty(x(t), y(t)), \tag{4}$$

respectively, with $x(0) = x$.

In the case when $y(t) = y, \forall t \geq 0$ for some $y \in \mathbf{R}^{d_2}$ we make use of the notations $\eta^y(t, x)$, $\eta_c^y(t, x)$ and $\eta_\infty^y(t, x)$ respectively. The lemma below shows that the trajectory of the ODE tends towards and stays inside a small neighborhood of its asymptotic stable equilibrium within a given time period.

Lemma 2 *Let $K \subset \mathbf{R}^{d_1}$ be a compact set and $y \in \mathbf{R}^{d_2}$ be a fixed external input. If the limiting ODE in A4 has a unique globally asymptotically stable equilibrium (a.s.e) $\lambda_\infty(y) \in \mathbf{R}^{d_1}$, then given any $\delta > 0$, there exists a $T_\delta > 0$ such that for all initial conditions $x \in K$, we have $\eta_\infty^y(t, x) \in B(\lambda_\infty(y), \delta), \forall t \geq T_\delta$.*

Proof: See Lemma 1, Chapter 3, [7].

The following lemma shows that for large values of c, the trajectories of (3) and (4) are close enough.

Lemma 3 *Let $x \in \mathbf{R}^{d_1}$, $y \in \mathbf{R}^{d_2}$, $[0,T]$ be a given time interval and $r > 0$ be a small positive constant. Let $y'(t) \in B(y,r),\ \forall t \in [0,T]$, then*

$$\| \eta_c^{y'(t)}(t,x) - \eta_\infty^y(t,x) \| \leq (\epsilon(c) + Lr)Te^{LT},\ \forall t \in [0,T],$$

where $\epsilon(c)$ is independent of x and y and $\epsilon(c) \to 0$, as $c \to \infty$.

Proof: See Lemma 2, Chapter 3 of [7].
The following lemma uses the results of Lemma 2 and Lemma 3 to show that for a fixed $y \in \mathbf{R}^{d_2}$ any trajectory of (3) starting within a δ-neighborhood of $\lambda_\infty(y)$ stays within an ϵ-neighborhood of $\lambda_\infty(y)$.

Lemma 4 *Let $y \in \mathbf{R}^{d_2}$, then given any $\epsilon > 0$ and $T > 0$, there exist $c_{\epsilon,T} > 0$, $\delta_{\epsilon,T} > 0$ and $r_{\epsilon,T} > 0$ such that $\forall t \in [0,T), \forall x \in B(\lambda_\infty(y), \delta_{\epsilon,T}),\ \forall c > c_{\epsilon,T}$ and external input $y'(s)$ with $y'(s) \in B(y, r_{\epsilon,T}), s \in [0,T]$, we have $\eta_c^{y'(t)}(t,x) \in B(\lambda_\infty(y), 2\epsilon), \forall t \in [0,T]$.*

Proof: Given any $\epsilon > 0$, it follows from Lyapunov stability that there exists $\delta_{\epsilon,T} > 0$ such that any trajectory of (4) starting in $B(\lambda_\infty(y), \delta_{\epsilon,T})$ stays within the ball $B(\lambda_\infty(y), \epsilon)$. Without loss of generality we can assume $\delta_{\epsilon,T} < \epsilon$. From Lemma 3 we know that there exist $c_{\epsilon,T}$ and $r_{\epsilon,T}$ such that $\| \eta_c^{y'(t)}(t,x) - \eta_\infty^y(t,x) \| \leq \epsilon$. Hence, $\forall t \in [0,T],\ \forall y'(t) \in B(y, r_{\epsilon,T}),\ \forall x \in B(\lambda_\infty(y), \delta_{\epsilon,T})$ and $\forall c > c_{\epsilon,T}$,

$$\begin{aligned}
&\| \eta_c^{y'(t)}(t,x) - \lambda_\infty(y) \| \\
&\leq \| \eta_c^{y'(t)}(t,x) - \eta_\infty^y(t,x) \| + \| \eta_\infty^y(t,x) - \lambda_\infty(y) \| \\
&\leq \delta/2 + \epsilon \\
&\leq 2\epsilon.
\end{aligned}$$

The following lemma shows that for a fixed $y \in \mathbf{R}^{d_2}$ and large enough c the trajectory of (3) falls within a neighborhood of $\lambda_\infty(y)$ within a given time period and continues to stay in that neighborhood.

Lemma 5 *Let $x \in B(0,1) \subset \mathbf{R}^{d_1}$, $y \in K' \subset \mathbf{R}^{d_2}$ and let $\lambda_\infty(y)$ be the unique globally asymptotically stable equilibrium of* (4)*. Then, given $\epsilon > 0$, there exists $c_\epsilon \geq 1$, $r_\epsilon > 0$ and $T_\epsilon > 0$ such that for any external input $y'(s)$ satisfying*

$$y'(s) \in B(y, r_\epsilon), \forall s \in [0,T], \tag{5}$$

we have $\| \eta_c^{y'(t)}(t,x) - \lambda_\infty(y) \| \leq 2\epsilon, \forall t \geq T_\epsilon, \forall c > c_\epsilon$.

Proof: Given any $\epsilon > 0$, it follows from Lyapunov stability that there exists $\delta > 0$ such that any trajectory of (4) starting in $B(\lambda_\infty(y), \delta)$ stays within the ball $B(\lambda_\infty(y), \epsilon)$. Without loss of generality we can assume $\delta < \epsilon$. Let $K = B(0,1) \cup B(\lambda_\infty(y), \delta)$, then from Lemma 2, there exists a $T_{\delta/2} > 0$ such that $\eta_\infty^y(t,x) \in B(\lambda_\infty(y), \delta/2)$, $\forall t \geq T_{\delta/2}$. Pick $T_\epsilon \triangleq T_{\delta/2}$ as given by Lemma 2 and divide the timeline into intervals T_ϵ apart, i.e., $t \in \bigcup_n [nT_\epsilon, (n+1)T_\epsilon), n \geq 0$. We know $\eta_\infty^y(t,x) \in B(\lambda_\infty(y), \delta/2), \forall t \geq T_\epsilon$. From Lemma 3, it follows that there exists $c^1_{\epsilon,T_\epsilon}$ and $r^1_{\epsilon,T_\epsilon}$ such that $\| \eta_c^{y'(t)}(T_\epsilon, x) - \eta_\infty^y(T_\epsilon, x) \| \leq \delta/2, \forall c > c^1_{\epsilon,T_\epsilon}$ and $y'(t)$ satisfying (5). This implies that $\eta_c^{y'(t)}(T_\epsilon, x) \in B(\lambda_\infty(y), \delta) \subset B(\lambda_\infty(y), 2\epsilon), \forall c > c^1_{\epsilon,T_\epsilon}$ and with $y'(t)$ satisfying (5). Thus starting from x, the trajectory $\eta_c^{y'(t)}(t,x)$ falls into the ball $B(\lambda(\infty), \delta)$ for all $c > c^1_{\epsilon,T_\epsilon}$ and $y'(t)$ satisfying (5).
It is easy to note that the trajectory $\eta_c^{y'(t)}(t,x)$ of the ODE (3) in the time interval $[T_\epsilon, 2T_\epsilon)$ starting from x is the same as the trajectory of the ODE (3) in the time interval $[0, T_\epsilon]$ but starting from the initial condition $\eta_c^{y'(T_\epsilon)}(T_\epsilon, x)$. Now we know from Lemma 3 that $\eta_c^{y'(t)}(t, \eta_c^{y'(t)}(T_\epsilon, x))$ and $\eta_\infty^y(t, \eta_c^{y'(t)}(T_\epsilon, x))$ track each other closely in the time $t \in [0, T_\epsilon)$. Formally, $\forall t \in [T_\epsilon, 2T_\epsilon)$,

$$\begin{aligned}
&\| \eta_c^{y'(t)}(t,x) - \eta_\infty^y(t - T_\epsilon, \eta_c^{y'(t)}(T_\epsilon, x)) \| \\
&= \| \eta_c^{y'(t)}(t - T_\epsilon, \eta_c^{y'(t)}(T_\epsilon, x)) - \eta_\infty^y(t - T_\epsilon, \eta_c^{y'(t)}(T_\epsilon, x)) \| \\
&\leq \delta/2.
\end{aligned} \tag{6}$$

Since $\eta_\infty^y(T_\epsilon, \eta_c^{y'(t)}(T_\epsilon, x)) \in B(\lambda_\infty(y), \delta/2)$ and from (6) we can conclude that $\eta_c^{y'(t)}(2T_\epsilon, x) \in B(\lambda_\infty(y), \delta)$. Also, since $\eta_c^{y'(t)}(T_\epsilon, x) \in B(\lambda_\infty(y), \delta)$, we know from Lemma 4 that there exists $c^2_{\epsilon,T_\epsilon}$ and $r^2_{\epsilon,T_\epsilon}$ such that $\eta_c^{y'(t)}(t,x) \in B(\lambda_\infty(y), 2\epsilon), \forall t \in [T_\epsilon, 2T_\epsilon)$.
Notice that by arguing in a similar manner one can show that $\eta_c^{y'(t)}(nT_\epsilon, x) \in B(\lambda_\infty(y),\ \delta), \forall n \geq 0$ and $\eta_c^{y'(t)}(t,x) \in B(\lambda_\infty(y), 2\epsilon), \forall t \in [nT_\epsilon, (n+1)T_\epsilon)$. The proof is complete by choosing $c_\epsilon = \max(c^1_{\epsilon,T_\epsilon}, c^2_{\epsilon,T_\epsilon})$ and $r_\epsilon = \min(r^1_{\epsilon,T_\epsilon}, r^2_{\epsilon,T_\epsilon})$.

5 Main Results

The idea of rescaling will be used twice, first for the faster timescale and then for the slower timescale recursions.

5.1 Faster Timescale Analysis

D4 Define the timescale according to $t(n) \stackrel{def}{=} \sum_{i=0}^{n-1} a(i)$. Let $z = (x,y)$ and $\bar{z}(t) = z_n + (z_{n+1} - z_n)\frac{t - t(n)}{t(n+1) - t(n)}, t \in [t(n), t(n+1)]$.

D5 Given a timescale $t(n), n \geq 0$ and a positive constant $T > 0$, define sampling instants $T_n, n \geq 0$ as $T_0 = 0$ and $T_n = \min\{t(m) : t(m) \geq T_{n-1} + T\}$. Note that $T_n = t(m(n))$ for some $m(n) \uparrow \infty$ as $n \uparrow \infty$.

D6 Define the scaling sequence $r(n) \stackrel{def}{=} \max(r(n-1), \| \bar{z}(T_n) \|, 1)$.

D7 The scaled iterates for $m(n) \leq k \leq m(n+1)-1$ are given by $\hat{x}_{m(n)} = \frac{x_{m(n)}}{r(n)}, \hat{y}_{m(n)} = \frac{y_{m(n)}}{r(n)}$,

$$\begin{aligned} \hat{x}_{k+1} &= \hat{x}_k + a(k)[h_c(\hat{x}_k, \hat{y}_k) + \hat{M}^{(1)}_{k+1}], \\ \hat{y}_{k+1} &= \hat{y}_k + a(k)[\epsilon_k + \hat{M}^{(2)}_{k+1}], \end{aligned} \tag{7}$$

where $c = r(n)$, $\epsilon_k = \frac{b(k)}{a(k)}\left(\frac{g(c\hat{x}_k, c\hat{y}_k)}{c}\right)$, $\hat{M}^{(1)}_{k+1} = \frac{M^{(1)}_{k+1}}{r(n)}$, $\hat{M}^{(2)}_{k+1} = \frac{M^{(2)}_{k+1}}{r(n)}$. Here h_c is as defined in A4.

D8 $\hat{z}(t) = \hat{z}_n + (\hat{z}_{n+1} - \hat{z}_n)\frac{t-t(n)}{t(n+1)-t(n)}, t \in [t(n), t(n+1)]$.

D9 Let $z_n(t) = (x_n(t), y_n(t)),\ t \in [T_n, T_{n+1}]$, denote the trajectory of the ODEs $\dot{x}(t) = h_{r(n)}(x(t), y(t)), \dot{y}(t) = 0$, with initial conditions $x_n(T_n) = \hat{x}(T_n)$ and $y_n(T_n) = \hat{y}(T_n)$ respectively.

Here T_ns are instants that are roughly $T > 0$ apart i.e., $T_{n+1} - T_n \approx T$. These time instants in the faster timescale are used to monitor the growth of the iterates. Also, note that the scaled iterates in D6 are bounded (by the very nature of their definition). The following lemma (Lemma 6) uses the convergence results for two timescale SA iterates [6, 7] to show the convergence of the scaled iterates in D6.

Lemma 6 *Under A1-A3, we have*

(i) The sequence $\hat{\zeta}^{(1)}_n = \sum_{m=0}^{n-1} a(m)\hat{M}^{(1)}_{m+1}, n \geq 1$ *is a.s. convergent.*

(ii) For $0 < k \leq m(n+1) - m(n)$, $\| \hat{z}(t(m(n)+k)) \| \leq K_2$, *a.s. for some* $K_2 > 0$.

(iii) $\lim_{n\to\infty} \| \hat{z}(t) - z_n(t) \| = 0, a.s., \forall t \in [T_n, T_{n+1})$.

Proof: See Chapter 3 and Chapter 6, [7].
Lemma 6 (i) shows that the martingale difference sequence $\hat{M}^{(1)}_{k+1}$ participating in the scaled recursion in (7) is convergent. Lemma 6 (ii) shows that in the faster timescale, between instants T_n and T_{n+1}, starting from within a unit ball around origin, the norm of the iterates can grow only by a factor $K_2 > 0$. Lemma 6 (iii) shows that scaled iterates track a corresponding scaled ODE.

Theorem 7 *Under A1-A4, we have the following:*

(i) For n *large and* $T \stackrel{def}{=} T_{1/4}$ *(where* T *is the sampling period used in D5 and* $T_{1/4}$ *is* T_ϵ *as in Lemma 5 above with* $\epsilon = 1/4$*), if* $\| \bar{x}(T_n) \| > C_1(1 + \| \bar{y}(T_n) \|)$, *then* $\| \bar{x}(T_{n+1}) \| < \frac{3}{4} \| \bar{x}(T_n) \|$.

(ii) $\| \bar{x}(T_n) \| \leq C^*(1 + \| \bar{y}(T_n) \|)$ *almost surely, for some* $C^* > 0$.

(iii) $\| x_n \| \leq K^*(1 + \| y_n \|)$ *almost surely for some* $K^* > 0$.

Proof: (i) $\| \bar{x}(T_n) \| > C_1(1 + \| \bar{y}(T_n) \|)$ implies that $r(n) \geq C_1$, $\| \hat{y}(T_n) \| < \frac{1}{C_1}$ and $\| \hat{x}(T_n) \| > \frac{1}{1+1/C_1}$. Let $\eta^{y(t)}_\infty(t, x)$ and $\eta^{y(t)}_c(t, x)$ be the solutions to the ODEs $\dot{x}(t) = h_\infty(x(t), y(t))$ and $\dot{x}(t) = h_c(x(t), y(t))$, respectively, with initial condition x in both. Let $y'(t - T_n) = y_n(t),\ \forall t \in [T_n, T_{n+1}]$. It is easy to see that $x_n(t) = \eta^{y'(t)}(t - T_n, \hat{x}(T_n)), \forall t \in [T_n, T_{n+1}]$, where $x_n(t)$ and $y_n(t)$ are as in D9.
We also know from Lemma 5 that there exists $r_{1/4}$, $c_{1/4}$, and $T_{1/4}$ such that $\| \eta^{y'(t)}_c(t, \hat{x}(T_n)) \| \leq \frac{1}{4},\ \forall t \geq T_{1/4}$, $\forall c \geq c_{1/4}$, whenever $y'(t) \in B(0, r_{1/4}), \forall t \in [0, T]$.
Now, let us pick $C_1 > \max(c_{1/4}, \frac{2}{r_{1/4}})$ and the positive constant $T > 0$ as $T \stackrel{def}{=} T_{1/4}$. Since $y'(t - T_n) = y_n(t) = \hat{y}(T_n), \forall t \in [T_n, T_{n+1}]$, we have for our choice of C_1, $y'(s) \in B(0, r_{1/4}), \forall s \in [0, T]$. We also know from Lemma 6 (iii) that $\| \hat{x}(T^-_{n+1}) - x_n(T_{n+1}) \| < \frac{1}{4}$ for sufficiently large n. Since $\| x_n(T_{n+1}) \| = \| \eta^{y'(t)}(T_{n+1} - T_n, \hat{x}(T_n)) \| \leq 1/4$, we have $\| \hat{x}(T^-_{n+1}) \| \leq \| \hat{x}(T^-_{n+1}) - x_n(T_{n+1}) \| + \| x_n(T_{n+1}) \| \leq \frac{1}{2}$. Since $\frac{\bar{x}(T_{n+1})}{\bar{x}(T_n)} = \frac{\hat{x}(T^-_{n+1})}{\hat{x}(T_n)}$, it follows that $\| \bar{x}(T_{n+1}) \| < \frac{1+1/C_1}{2} \| \bar{x}(T_n) \|$, and the result holds by assuming without loss of generality that $C_1 > \max(c_{1/4}, \frac{2}{r_{1/4}}) > 2$.
(ii) On a set of positive probability, let us assume on the contrary that there exists a monotonically increasing sequence $\{n_k\}$ for which $C_{n_k} \uparrow \infty$ as $k \to \infty$ and $\| \bar{x}(T_{n_k}) \| \geq C_{n_k}(1 + \| \bar{y}(T_{n_k}) \|)$. Now from Theorem 7 (i), we know that if $\| \bar{x}(T_n) \| > C_1(1 + \| \bar{y}(T_n) \|)$, then $\| \bar{x}(T_k) \|$ for $k \geq n$ falls at an exponential rate until it is within the ball of radius $C_1(1 + \| \bar{y}(T_k) \|)$. Thus corresponding to the sequence $\{n_k\}$, there must exist another sequence $\{n'_k\}$ such that $n_{k-1} \leq n'_k \leq n_k$ and $\| \bar{x}(T_{n'_k - 1}) \|$ is within the ball of radius $C_1(1 + \| \bar{y}(T_{n'_k - 1}) \|)$ but $\| \bar{x}(T_{n'_k}) \|$ is greater than $C_{n_k}(1 + \| \bar{y}(T_{n'_k}) \|)$. However, from Lemma 6 (ii) we know that the iterates can grow only by a factor of K_2 between $m(n'_k - 1)$ and $m(n'_k)$. This leads to a contradiction. So we conclude that $\| \bar{x}(T_n) \| \leq C^*(1 + \| \bar{y}(T_n) \|)$ for some $C^* > 0$.
(iii) We showed that $\| \bar{x}(T_n) \| \leq C^*(1 + \| \bar{y}(T_n) \|)$. From Lemma 6 (ii), we know that $\| \bar{z}(t) \| \leq K_2 \| \bar{z}(T_n) \|,\ \forall t \in [T_n, T_{n+1})$. The result follows by choosing $K^* = K_2 C^*$.

Remark: In the above proof of Theorem 7, (i) follows from the fact that (see Lemma 6 (iii)) the scaled iterates in (7) track the scaled ODE in D9, which in turn falls within an ϵ ball around the stable equilibrium (see Lemma 5) of the limiting ODE in A4. (ii) follows from (i) and Lemma 6 (ii). (iii) follows from (ii) and Lemma 6 (ii). We also observe that the stability result (Theorem 7, Chapter 3, [7]) for single timescale SA schemes can be recovered by letting $y_n = 0,\ \forall n \geq 0$ in Theorem 7.

D8 Given any $\epsilon > 0$ and $y \in \mathbf{R}^{d_2}$, define the set $A^\epsilon(y) \subset \mathbf{R}^{d_1}$ as $A^\epsilon(y) \stackrel{def}{=} \{x\colon \| x - \lambda_\infty(y) \| < \epsilon\}$.

Theorem 8 *For any given* $\epsilon > 0$, *there exists* $c_\epsilon > 0$ *such that, if* $r(n) > c_\epsilon$ *for some* n, *then it follows that* $(\hat{x}_k, \hat{y}_k) \in (A^\epsilon(\hat{y}_k), \hat{y}_k), \forall k \geq m(n)$.

Proof: The proof follows by a repeated application of Lemma 6 and Lemma 5 to the intervals $[T_w, T_{w+1}], \forall k \geq n$ and using the fact that $r(w) \geq r(n), \forall w \geq n$.

Theorem 8 shows that for large n and large scaling factor $c >> 1$, the faster timescale iterates are within an ϵ neighborhood of the stable equilibrium of the limiting ODE in A4.

5.2 Slower Timescale Analysis

The following arguments are with respect to the slower timescale. Thus, the notions of timescale, the corresponding sampling instants T_n and the scaled iterates are re-defined with respect to the slower timescale as below:

D10 Define now the timescale to be $t(n) \stackrel{def}{=} \sum_{i=0}^{n-1} b(i)$. Let $z = (x, y)$ and $\bar{z}(t) = z_n + (z_{n+1} - z_n)\frac{t-t(n)}{t(n+1)-t(n)}, t \in [t(n), t(n+1)]$.

D11 Given a timescale $t(n), n \geq 0$ and a positive constant $T > 0$, define sampling instants $T_n, n \geq 0$ as $T_0 = 0$ and $T_n = \min\{t(m) : t(m) \geq T_{n-1} + T\}$. Note that $T_n = t(m(n))$ for some $m(n) \uparrow \infty$ as $n \uparrow \infty$.

D12 Define scaling sequence $r(n) \stackrel{def}{=} \max(r(n-1), \| \bar{z}(T_n) \|, 1)$.

D13 The scaled iterates for $m(n) \leq k \leq m(n+1) - 1$ are given by $\hat{x}_{m(n)} = \frac{x_{m(n)}}{r(n)}, \hat{y}_{m(n)} = \frac{y_{m(n)}}{r(n)}$,

$$\begin{aligned} \hat{x}_{k+1} &= \hat{x}_k + a(k)[h_c(\hat{x}_k, \hat{y}_k) + \hat{M}^{(1)}_{k+1}], \\ \hat{y}_{k+1} &= \hat{y}_k + b(k)[g_c(\hat{x}_k, \hat{y}_k) + \hat{M}^{(2)}_{k+1}], \end{aligned} \tag{8}$$

where $c = r(n)$, $\hat{M}^{(1)}_{k+1} = \frac{M^{(1)}_{k+1}}{r(n)}$, $\hat{M}^{(2)}_{k+1} = \frac{M^{(2)}_{k+1}}{r(n)}$. Here h_c and g_c are as defined in A4 and A5 respectively.

D14 $\hat{z}(t) = \hat{z}_n + (\hat{z}_{n+1} - \hat{z}_n)\frac{t-t(n)}{t(n+1)-t(n)}, t \in [t(n), t(n+1)]$.

D15 We let $y_n(t), t \in [T_n, T_{n+1}]$, denote the trajectory of the ODEs $\dot{y}(t) = g_c(y(t))$ with initial conditions $y_n(T_n) = \hat{y}(T_n)$.

Lemma 9 *Under assumptions A1-A3, we have the following:*

(i) The sequence $\hat{\zeta}^{(2)}_n = \sum_{m=0}^{n-1} a(m)\hat{M}^{(2)}_{m+1}, n \geq 1$ *is a.s. convergent.*

(ii) For $0 < k \leq m(n+1) - m(n)$*, we have a.s.* $\| \hat{z}(t(m(n)) + k)) \| \leq K_3$*, for some* $K_3 > 0$*.*

(iii) For sufficiently large n *we have* $\sup_{t \in [T_n, T_{n+1})} \| \hat{y}(t) - y_n(t) \| \leq \epsilon(c) L T e^{L(L+1)T}$*, a.s, where* $\epsilon(c) \to 0$ *as* $c \to \infty$*.*

Proof: See Chapter 3 and Chapter 6, [7].

Lemma 9 is similar to Lemma 6 and uses the result of Theorem 7 i.e., in the slower timescale the scaled iterates in (8) obey $||\hat{x}_k|| \leq K^*(1 + ||\hat{y}_k||), \ \forall k \geq 0$. Let $g_c \colon \mathbf{R}^{d_2} \to \mathbf{R}^{d_2}$ and $g_\infty \colon \mathbf{R}^{d_2} \to \mathbf{R}^{d_2}$ be functions as defined in A5, and let $\chi_\infty(t, y)$ denote the solution to the ODE $\dot{y}(t) = g_\infty(y(t))$, with initial condition y, and let $\chi_c(t, y)$ denote the solution to the ODE $\dot{y}(t) = g_c(y(t))$, with initial condition y.

Theorem 10 *Under A1-A5, we have the following:*

(i) Let K^* *be as in Theorem 7 (iii), then it follows that* $\| \hat{y}(T_n) \| \geq \frac{1}{K^*+2}$ *for sufficiently large* $\| \bar{y}(T_n) \|$*.*

(ii) For n *large,* $T \stackrel{def}{=} T_{1/4}$ *(with* T *as in D11 and* $T_{1/4}$ *is* T_ϵ *as in Lemma 5 with* $\epsilon = 1/4$*), if* $\| \bar{y}(T_n) \| > C$*, then* $\| \bar{y}(T_{n+1}) \| < \frac{1}{2} \| \bar{y}(T_n) \|$*.*

(iii) $\| \bar{y}(T_n) \| \leq C'$ *a.s., for some* $C' > 0$*.*

(iv) $\sup_n \| y_n \| < \infty$*, a.s. and* $\sup_n \| x_n \| < \infty$*, a.s.*

Proof: (i) From Theorem 7 (iii), we know that $\| r(n) \| \leq \| \bar{y}(T_n) \| + K^*(1 + \| \bar{y}(T_n) \|)$. Also, $\| \hat{y}(T_n) \| = \frac{\|\bar{y}(T_n)\|}{r(n)} \geq \frac{\|\bar{y}(T_n)\|}{\|\bar{y}(T_n)\| + K^*(1+\|\bar{y}(T_n)\|)} = \frac{1}{1 + \frac{K^*}{\|\bar{y}(T_n)\|} + K^*}$. The claim follows for any $\| \bar{y}(T_n) \| > K^*$.

(ii) By A5 we have $\mathbf{0} \in \mathbf{R}^{d_2}$ is the unique globally asymptotically stable equilibrium of the ODE $\dot{y}(t) = g_\infty(y(t))$, and as a consequence of Lemma 5 there exist $c_{1/4}$ and $T_{1/4}$ such that $\| \chi_c(t, y) \| < \frac{1}{4(K^*+2)}, \forall t \geq T_{1/4}, c > c_{1/4}$.

Also, if $\| \bar{y}(T_n) \| > K^*$, it follows from (i) above that $\| \hat{y}(T_n) \| \geq \frac{1}{K^*+2}$. We know from Lemma 9 (iii) that for sufficiently large n, there exists $C_1 > 0$ such that $\| \hat{y}(T^-_{n+1}) - y_n(T_{n+1}) \| < \frac{1}{4(K^*+2)}$, for $r(n) > C_1$. Now, let us pick $C = \max(c_{1/4}, C_1, K^*)$ and $T = T_{1/4}$. Then for sufficiently large n it follows that $\| \hat{y}(T^-_{n+1}) \| \leq \| \hat{y}(T^-_{n+1}) - y_n(T_{n+1}) \| + \| y_n(T_{n+1}) \| \leq \frac{1}{2(K^*+2)}$. Since $\frac{\bar{y}(T_{n+1})}{\bar{y}(T_n)} = \frac{\hat{y}(T^-_{n+1})}{\bar{y}(T_n)}$, it follows that $\| \bar{y}(T_{n+1}) \| < \frac{1}{2} \| \bar{y}(T_n) \|$.

(iii) Let us assume on the contrary that on a set of positive probability, there exists a sequence $\{n_k\}$ such that $C_{n_k} \uparrow \infty$ as $k \to \infty$ and $\| \bar{y}(T_{n_k}) \| \geq C_{n_k}$. From Theorem 10 (ii) we know that if $\| \bar{y}(T_n) \| > C$, then $\| \bar{y}(T_k) \|$ for $k \geq n$ falls at an exponential rate to the ball of radius C. Thus corresponding to the sequence $\{n_k\}$, there exists another sequence $\{n'_k\}$ such that $n_{k-1} \leq n'_k \leq n_k$ and $\| \bar{y}(T_{n'_k - 1}) \|$ is within the ball of radius C but jumps outside this ball of radius C (i.e.,$\| \bar{y}(T_{n'_k}) \| > C$) to points which are at a distance greater than C_{n_k} from the origin. However, from Lemma 9 (ii) we know that the iterates can grow only by a factor of K_3 between $m(n'_k - 1)$ and $m(n'_k)$. This leads to a contradiction and the claim follows.

(iv) We know that $\| \bar{y}(T_n) \| \leq C'$. From Lemma 9 (ii), we know that $\| \bar{y}(t) \| \leq K_3 \| \bar{y}(T_n) \|, \ \forall t \in [T_n, T_{n+1})$. The result follows by noting that $\| y_n \| \leq K_3 C'$ almost surely. The fact that $\sup_n \| x_n \| < \infty$ follows from Theorem 7.

Remark: The proof above for Theorem 10 is along the lines of the proof of Theorem 7. Here, (ii) follows from the fact that the scaled iterates in (8) track the scaled ODE in D15, which in turn follows the limiting ODE in A5. Now, since the limiting ODE is stable to the origin, the iterates fall at an

exponential rate into the unit ball around the origin. Together with the fact that the iterates can grow only by a constant factor (Lemma 9 (ii)) within a given time period (iii) follows. Then, (iv) follows from (iii), Lemma 9 (ii) and Theorem 7.

To summarize, we first showed in Section 5.1 that the faster timescale iterates are bounded by the growth of slower timescale iterates, and in Section 5.2 we showed that the slower timescale iterates themselves are bounded, thus completing our stability analysis of two timescale stochastic approximation schemes.

6 An Application in Reinforcement Learning

Reinforcement Learning (RL) algorithms are sample trajectory based methods for solving Markov Decision Processes (MDP). In the example presented in this paper, we consider an MDP whose state space is denoted by $S = \{s_1, s_2, \ldots, s_n\}$, and the action space by $A = \{a_1, a_2, \ldots, a_m\}$. In state s and under action a, let $c_a(s)$ denote the single-stage cost incurred and $p_a(s, s')$ be the probability of transition to s'. By a policy, we mean a sequence $\pi = \{\pi_1, \pi_2, \ldots, \pi_k, \ldots\}$, where each $\pi_k, k = 1, 2, \ldots,$ specifies a way by which states are mapped to actions. In this paper, we consider the class of time stationary policies with the *Gibbs* parameterization given as under.

$$\pi^\theta(s,a) = \frac{e^{\phi_{sa}^\top \theta}}{\sum_{a'} e^{\phi_{sa'}^\top \theta}}, \tag{9}$$

where $\phi_{sa} \in \mathbf{R}^{d_2}$ is a d_2-dimensional feature vector for the state-action tuple (s, a), and $\theta \in \mathbf{R}^{d_2}$ is the parameter. We assume that under a policy π^θ the MDP is an irreducible and aperiodic Markov chain with probability transition matrix given by P_{π^θ}. The *average*-cost associated with a policy π^θ is given by $J(\theta) = \lim_{N\to\infty} \frac{1}{N}\mathbf{E}[\sum_{n=0}^{N-1} c_{a_n}(s_n)|\pi^\theta]$, where a_n is sampled from the distribution $\pi(s_n, \cdot), \forall n \geq 0$ and given s_n, s_{n+1} is distributed as $p_{a_n}(s_n, \cdot)$. Other important quantities of interest are the differential state-action cost given by $Q^\theta(s) = \mathbf{E}[\sum_{n=0}^{\infty}(c_{a_n}(s_n) - J(\theta))|\pi^\theta]$, the differential cost $V^\theta(s) = \sum_{a\in A} \pi^\theta(s,a)Q^\theta(s)$ and the advantage function $A^\theta(s,a) = Q^\theta(s,a) - V^\theta(s)$. The average and differential costs are connected by the Bellman equation

$$Q^\theta(s,a) = [c_a(s) - J(\pi) + \sum_a p_a(s,s')V^\theta(s')]. \tag{10}$$

We also assume that under any given policy π^θ the MDP is an aperiodic irreducible Markov chain with unique stationary distribution d^π. In what follows, since π^θ is fixed for a fixed θ, we use π^θ, θ and π interchangeably to denote π^θ.

Actor-Critic RL algorithms are a sub-class of policy gradient RL algorithms, where the aim is find the best policy amongst the class of parameterized policies (such as the Gibbs parameterization in (9)). AC algorithms are two timescale SA schemes and have two components namely, the actor which is responsible for control and updates the policy parameters (i.e. θ in (9)) in the slower timescale, and the critic which is responsible for prediction and evaluates the policy in the faster timescale. We now verify our stability conditions for the iterates in Algorithm 1 of [5], which is an actor-critic algorithm (repeated here as Algorithm 1). However, there are certain minor differences between Algorithm 1 of [5] and the Algorithm 1 presented here. Algorithm 1 of [5] corresponds to the average reward setting while the algorithm here is for the setting of average cost. Also, the update in equation (23) of Algorithm 1 of [5] makes use of a projection operator Γ in order to ensure boundedness of the iterates. However, in the actor update in line 5 of Algorithm 1 (here), we do not make use of projection and instead we introduce an additional term $\epsilon\theta_n$ (where $\epsilon > 0$ is a small positive constant). This is equivalent to adding a quadratic penalty term $\epsilon\theta^2$ to the objective $J(\theta)$. We now write the iterates in the standard

Algorithm 1 The Actor-Critic Algorithm

1: **for** $n = 0, 1, 2, \ldots$ **do**
2: Average Cost Update: $\hat{J}_{n+1} = \hat{J}_n + a(n)(c_{a_n}(s_n) - \hat{J}_n)$
3: TD Error: $\delta_n = c_{a_n}(s_n) - \hat{J}_n + v_n^\top \gamma_{s_{n+1}} - v_n^\top \gamma_{s_n}$
4: Critic Update: $v_{n+1} = v_n + a(n)\delta_n \gamma_{s_n}$
5: Actor Update: $\theta_{n+1} = \theta_n - b(n)(\delta_n \psi_{s_n a_n} + \epsilon\theta_n)$
6: **end for**

form as below:

$$\hat{J}_{n+1} = \hat{J}_n + a(n)[h^1(\hat{J}_n, v_n, \theta_n) + M_{n+1}^{(1)}], \tag{11}$$

$$v_{n+1} = v_n + a(n)[h^2(\hat{J}_n, v_n, \theta_n) + M_{n+1}^{(2)}], \tag{12}$$

$$\theta_{n+1} = \theta_n + b(n)[g(\hat{J}_n, v_n, \theta_n) + M_{n+1}^{(3)}], \tag{13}$$

where

- $h^1(\hat{J}_n, v_n, \theta_n) = \mathbf{E}[c_{a_n}(s_n)|\mathcal{F}_n] - \hat{J}_n$
  $= \sum_{s,a} d^\pi(s)\pi(s,a)\big(c_a(s) - \hat{J}_n\big)$,
- $h^2(\hat{J}_n, v_n, \theta_n) = \mathbf{E}[\delta_n \gamma_{s_n}|\mathcal{F}_n] = \sum_{s,a} d^\pi(s)\pi(s,a)$
  $\big(c_a(s) - \hat{J}_n + \sum_{s'} p_a(s,s')v_n^\top \gamma_{s'} - v_n^\top \gamma_s)\gamma_s$,
- $g(\hat{J}_n, v_n, \theta_n) = \mathbf{E}[\delta_n \psi_{s_n a_n}|\mathcal{F}_n]$
  $= -\sum_s d^\pi(s) \sum_a \nabla\pi(s,a)\hat{A}^\theta(s,a) - \epsilon\theta_n$.

Note that, in the above equations, π stands for π^{θ_n} and $\mathcal{F}_n \stackrel{def}{=} \sigma(\theta_m, v_m, \hat{J}_m, M_m^{(1)}, M_m^{(2)}, M_m^{(3)}, 0 \leq m \leq n)$. We assume for simplicity here that states s_n are sampled independently according to the stationary distribution $d^{\pi^\theta}(s)$ of the underlying Markov chain (see Chapter 6 of [1]). Further, for the feature matrix F with s^{th} row being f_s we assume that $Fe \neq e$, where $e = (1, \ldots, 1)$. A similar assumption has also been made in [5, 11]. Here, the differential cost $V^\theta(s)$ is approximated as $V^\theta(s) \approx v^{\pi\top}\gamma_s$ (where $\gamma_s \in \mathbf{R}^{d_1}$ is the feature vector of

state s, $v^\theta \in \mathbf{R}^k$ is a learnt weight vector) and $\hat{A}^\theta(s,a) = c_a(s) - \hat{J}_n + \sum_{s'} p_a(s,s')v_n^\top \gamma_{s'} - v_n^\top \gamma_s$ is the approximate *advantage*-function (see [5]). The martingale terms are given by $M_{n+1}^{(1)} = c_{a_n}(s_n) - \mathbf{E}[c_{a_n}(s_n)|\mathcal{F}_n], M_{n+1}^{(2)} = \delta_n \gamma_{s_n} - \mathbf{E}[\delta_n \gamma_{s_n}|\mathcal{F}_n], M_{n+1}^{(3)} = \delta_n \psi_{s_n a_n} - \mathbf{E}[\delta_n \psi_{s_n a_n}|\mathcal{F}_n]$. It is easy to see that there exist $C_i, i = 1,2,3$ such that $\mathbf{E}[||M_{n+1}^{(i)}||^2|\mathcal{F}_n] \leq C_i(1 + ||\theta_n||^2 + ||v_n||^2 + ||\hat{J}_n||^2)$. We show that A1-A5 hold for the iterates in Algorithm 1 in the proposition below.

Proposition 11

(1) The functions $h^1 \colon \mathbf{R}^{1+d_1+d_2} \to \mathbf{R}, h^2 \colon \mathbf{R}^{1+d_1+d_2} \to \mathbf{R}^{d_1}, g \colon \mathbf{R}^{1+d_1+d_2} \to \mathbf{R}^{d_2}$ are Lipschitz continuous.

(2) The functions $h_c^1(\hat{J}, v, \theta) \stackrel{def}{=} \frac{h^1(c\hat{J}, cv, c\theta)}{c}, c \geq 1$ are Lipschitz continuous, and satisfy $h_c^1 \to h_\infty^1$, as $c \to \infty$, uniformly on compacts.

(3) The functions $h_c^2(\hat{J}, v, \theta) \stackrel{def}{=} \frac{h^2(c\hat{J}, cv, c\theta)}{c}$ are Lipschitz continuous, and satisfy $h_c^2 \to h_\infty^2$ as $c \to \infty$, uniformly on compacts.

(4) The ODE $(\dot{\hat{J}}(t), \dot{v}(t)) = h_\infty(\hat{J}(t), v(t), \theta)$ has a unique globally asymptotically stable equilibrium $\lambda_\infty(\theta)$, where $\lambda_\infty \colon \mathbf{R}^{d_2} \to \mathbf{R}^{1+d_1}$ is a Lipschitz map that satisfies $\lambda_\infty(\mathbf{0}) = \mathbf{0}$.

(5) The functions $g_c(\theta) \stackrel{def}{=} \frac{g(c\theta, c\lambda_\infty(\theta))}{c}$ are Lipschitz continuous and satisfy $g_c \to g_\infty$, as $c \to \infty$, uniformly on compacts and the ODE $\dot{\theta} = g_\infty(\theta(t))$ has the origin in $\mathbf{R}^{d_2}$ as its unique globally asymptotically stable equilibrium.

Proof:

(1) Recall that $h^1(\hat{J}, v, \theta) = \sum_s d^\pi(s) \sum_a \pi(s,a) c_a(s) - \hat{J}$. Then h^1 is clearly Lipschitz (in fact linear) in $\hat{J}$. We now show that the derivatives of h^1 are bounded w.r.t. θ. $\nabla_\theta h^1(\hat{J}, v, \theta) = \sum_s \nabla_\theta d^\pi(s) \sum_a \pi(s,a) c_a(s) + \sum_s d^\pi(s) \sum_a \nabla_\theta \pi(s,a) c_a(s)$. We know from [9] that $d^\pi(s)$ is continuously differentiable in θ and has a bounded derivative. Now since $d^\pi(s) = \sum_{s'} d^\theta(s') \sum_a \pi(s',a) \sum_s p_a(s',s)$, and $\nabla \pi_\theta(s,a) = \pi(s,a)(\phi_{sa} - \sum_{a'} \phi(s,a')\pi(s,a'))$, it follows that $d^\pi(s)$ also has a bounded derivative w.r.t θ. Using similar arguments on the boundedness of the derivatives of $d^\pi(s)$ and $\pi(s,a)$ w.r.t θ we can show that h^2 and g are Lipschitz as well.

Given θ, define quantity $\pi_c(\theta) \stackrel{def}{=} \pi^{c\theta}, c \geq 1$. For any s, let $a^* = \arg\max_a \phi_{sa}^\top \theta$, and let $\pi_\infty^\theta(s,a) \stackrel{def}{=} \mathbf{1}_{\{a=a^*\}}$, where $\mathbf{1}$ is the indicator function. We show that $\pi_c^\theta(s,a) \to \pi_\infty^\theta(s,a)$, uniformly on compacts. For any $a \neq a^*$, $\pi_c^\theta(s,a) = \frac{e^{c\phi_{sa}^\top \theta}}{\sum_{a'} e^{c\phi_{sa'}^\top \theta}} \leq \frac{e^{c\phi_{sa}^\top \theta}}{e^{c\phi_{sa^*}^\top \theta}} = e^{c(\phi_{sa}^\top - \phi_{sa^*}^\top)\theta}$. Now since $a \neq a^*$ and $\pi_c^\theta(s,a^*) = 1 - \sum_{a \neq a^*} \pi_c^\theta(s,a)$, we have $\pi_c^\theta(s,a) \to 0$, as $c \to \infty$, $\forall s \in S, a \in A$, uniformly on compacts. We now drop the superscript θ and simply use π_∞ and π_c for notational simplicity.

(2) $h_c^1(\hat{J}, v, \theta) = \frac{\sum_s d^{\pi_c}(s) \sum_a \pi_c(s,a) c_a(s) - c\hat{J}}{c}$, thus we have $h_\infty^1(\hat{J}, v, \theta) = -\hat{J}$.

(3) In a similar fashion, $h_c^2(\hat{J}, v, \theta) \stackrel{def}{=} \Big(\sum_s d^{\pi_c}(s) \sum_a \pi_c(s,a)[c_a(s) - c\hat{J} + \sum_{s'} p_a(s,s') v_n^\top f_{s'} - v_n^\top f_s] \Big)/c$, and $h_\infty^2(\hat{J}, v, \theta) = F^\top D^{\pi_\infty}(-I + P_\pi)Fv - \hat{J}$.

(4) Consider the ODE $(\dot{\hat{J}}(t), \dot{v}(t)) = h_\infty(\hat{J}, v, \theta)$, where $h_\infty = (h_\infty^1, h_\infty^2)$. It is straightforward to see that $\dot{\hat{J}}(t) = -\hat{J}(t)$ is stable to the origin in $\mathbf{R}$. Now, we know from [5] (provided $A3$ of [5] holds) that $\dot{v}(t) = F^\top D^{\pi_\infty}(-I + P_\pi)Fv$ has the origin in $\mathbf{R}^{d_1}$ as its unique asymptotically stable equilibrium. Thus, $\lambda(\theta) = (0, \mathbf{0}) \in \mathbf{R}^{1+d_1}, \forall \theta \in \mathbf{R}^{d_2}$, where $0 \in \mathbf{R}$ and $\mathbf{0} \in \mathbf{R}^{d_1}$.

(5) Now, using the fact that $\lambda_\infty(\theta) = (0, \mathbf{0})$, we have $g_c(\theta) = \Big(\sum_s d^{\pi_c}(s) \sum_a \nabla \pi_c(s,a)[c_a(s) - c \times 0 + \sum_{s'} p_a(s,s') c\, \mathbf{0}^\top f_{s'} - c\, \mathbf{0}^\top f_s] \Big)/c - \frac{\epsilon c\theta}{c}$, and $g_\infty(\theta) = \lim_{c \to \infty} \frac{g(c\theta)}{c}$. Also, it is easy to see that all the quantities in the numerator of the first term on the right hand side of $g_c(\theta)$ are bounded and therefore, $g_\infty(\theta) = -\epsilon\theta$. Then the ODE $\dot{\theta}(t) = -\theta(t)$, has the origin in $\mathbf{R}^{d_2}$ as its unique globally asymptotically stable equilibrium.

The claim follows. We have thus shown that the iterates θ_n, v_n and $\hat{J}_n$ are stable. It is important to note that for the stability analysis we have not explicitly projected the iterates θ_n as in [5].

We now present a numerical implementation of Algorithm 1. We considered an MDP with 50 states and 10 actions. For each state $s \in S$ and action $a \in A$, the transition probabilities $p_a(s, \cdot)$ were chosen at random and then normalized so that $\sum_{s' \in S} p_a(s,s') = 1$. For each state-action pair (s,a), the cost $c_a(s)$ was chosen uniformly from integers 0 to 10. The feature $\gamma_s \in \mathbf{R}^{d_2}$ for state $s \in S$, was chosen such that each co-ordinate was 0 or 1 w.p. $\frac{1}{2}$. The features ϕ_{sa_i} were then made to be $\phi_{sa_i} = (\underbrace{0, \ldots, 0}_{k \times (i-1)}, \gamma_s, \underbrace{0, \ldots, 0}_{k \times (m-i)})^\top \in \mathbf{R}^{d_2}$. We chose $d_1 = 5$ and by the manner of constructing ϕ_{sa}, we have $d_2 = d_1 \times m = 50$. The plot below shows stability (and convergence) of all the faster timescale iterates and some of the slower timescale iterates (since θ has 50 co-ordinates we only show plots for some of the co-ordinates).

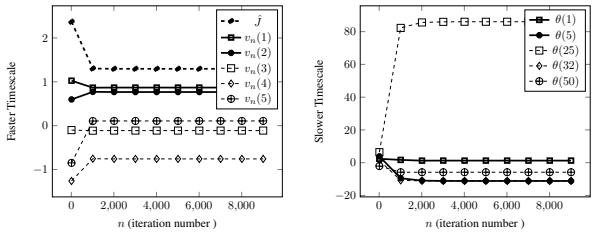

Fig. 1. Stability and convergence of faster timescale iterates (on the left) and slower timescale iterates (on the right). Algorithm 1 was run for $10,000$ iterations and the plots show the iterates sampled every $1,000$ iterations.

7 Conclusions

In this paper, we derived verifiable sufficient conditions that guarantee the stability of two-timescale stochastic approximation algorithms. This problem had not been addressed previously in the literature. The sufficient conditions turn out to be weaker than those needed to establish convergence (Chapter 6 of [7]). Our stability analysis of two-timescale stochastic approximation is quite general and may easily be extended to the case of multi-timescale stochastic approximation, where the number of timescales is more than two. We also presented an application of our results to an actor-critic algorithm in reinforcement learning and also presented a numerical example using this algorithm.

8 Acknowledgement

"This work was partially supported by the Robert Bosch Centre, Indian Institute of Science, Bangalore."

References

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996.

[2] S. Bhatnagar. Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 15(1):74–107, 2005.

[3] S. Bhatnagar and V. S. Borkar. A two timescale stochastic approximation scheme for simulation-based parametric optimization. *Probability in the Engineering and Informational Sciences*, 12:519–531, 1998.

[4] S. Bhatnagar, M. C. Fu, S. I. Marcus, and I. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Trans. Model. Comput. Simul.*, 13:180–209, 2003.

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

[6] V. S. Borkar. Stochastic approximation with two time scales. *Systems & Control Letters*, 29(5):291 – 294, 1997.

[7] V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge Univ. Press, 2008.

[8] V. S. Borkar and S. P. Meyn. The ODE method for convergence of stochastic approximation and reinforcement learning. *SIAM Journal on Control and Optimization*, 38(2):447–469, 2000.

[9] Paul J. Schweitzer. Perturbation theory and finite Markov chains. *Journal of Applied Probability*, 5(2):pp. 401–413, 1968.

[10] Vladislav B Tadić. Almost sure convergence of two time-scale stochastic approximation algorithms. In *American Control Conference, 2004. Proceedings of the 2004*, volume 4, pages 3802–3807. IEEE, 2004.

[11] J. N. Tsitsiklis and B. Van Roy. Average cost temporal-difference learning. *Automatica*, 35(11):1799–1808, 1999.

A Generalized Reduced Linear Program for Markov Decision Processes

Chandrashekar Lakshminarayanan and Shalabh Bhatnagar
Department Computer Science and Automation
Indian Institute of Science, Bangalore-560012, India
{chandrul,shalabh}@csa.iisc.ernet.in

Abstract

Markov decision processes (MDPs) with large number of states are of high practical interest. However, conventional algorithms to solve MDP are computationally infeasible in this scenario. Approximate dynamic programming (ADP) methods tackle this issue by computing approximate solutions. A widely applied ADP method is approximate linear program (ALP) which makes use of linear function approximation and offers theoretical performance guarantees. Nevertheless, the ALP is difficult to solve due to the presence of a large number of constraints and in practice, a reduced linear program (RLP) is solved instead. The RLP has a tractable number of constraints sampled from the original constraints of the ALP. Though the RLP is known to perform well in experiments, theoretical guarantees are available only for a specific RLP obtained under idealized assumptions.

In this paper, we generalize the RLP to define a generalized reduced linear program (GRLP) which has a tractable number of constraints that are obtained as positive linear combinations of the original constraints of the ALP. The main contribution of this paper is the novel theoretical framework developed to obtain error bounds for any given GRLP. Central to our framework are two $\max$-norm contraction operators. Our result theoretically justifies linear approximation of constraints. We discuss the implication of our results in the contexts of ADP and reinforcement learning. We also demonstrate via an example in the domain of controlled queues that the experiments conform to the theory.

Introduction

Markov decision process (MDP) is an important mathematical framework to study optimal sequential decision making problems that arise in science and engineering. Solving an MDP involves computing the optimal *value-function* (J^*), a vector whose dimension is the number of states. MDPs with small number of states can be solved easily by conventional solution methods such as value/ policy iteration or linear programming (LP) (Bertsekas 2013). Dynamic programming is at the heart of all the conventional solution methods for MDPs.

The term *curse-of-dimensionality* (or in short *curse*) denotes the fact that the number of states grows exponentially in the number of state variables. Most practical MDPs suffer from the curse, i.e., have large number of states and J^* is difficult to compute. A practical way to tackle the curse is to compute an approximate value function $\tilde{J}$ instead of J^*. The methods that compute $\tilde{J}$ instead of J^* are known as approximate dynamic programming (ADP) methods whose success depends on the quality of approximation, i.e., on the quantity $||J^* - \tilde{J}||$. Most ADP methods employ linear function approximation (LFA), i.e., let $\tilde{J} = \Phi r^*$, where Φ is a feature matrix and r^* is a learnt weight vector. Dimensionality reduction is achieved by choosing Φ to have far fewer columns in comparison to the number of states and this makes computing $\tilde{J}$ easier.

Approximate linear program (ALP) (de Farias and Roy 2003) employs LFA in the linear programming formulation (Bertsekas 2013) of MDP. The ALP computes an approximate value function and offers sound theoretical guarantees. A serious shortcoming of the ALP is the large number of constraints (of the order of the number of states). A technique studied in literature that tackles the issue of large number of constraints is constraint sampling (de Farias and Roy 2004; Farias and Roy 2006) wherein one solves a reduced linear program (RLP) with a small number of constraints sampled from the constraints of the ALP. (de Farias and Roy 2004) presents performance guarantees for the RLP when the constraints are sampled with respect to the stationary distribution of the optimal policy. Such an idealized assumption on the availability of the optimal policy (which in turn requires knowledge of J^*) is a shortcoming. Nevertheless, the RLP has been shown to perform empirically well (de Farias and Roy 2004; de Farias and Roy 2003; Desai, Farias, and Moallemi 2009) even when the constraints are not sampled using the stationary distribution of the optimal policy.

Motivated by the gap between the limited theoretical guarantees of the RLP as currently available in the literature and its successful practical efficacy, in this paper, we provide a novel theoretical framework to characterize the error due to constraint reduction/approximation. The novelty and salient points of our contribution are listed below:

• We define a generalized reduced linear program (GRLP) which has a tractable number of constraints that are obtained as positive linear combinations of the original constraints of

the ALP.

• We develop a novel analytical framework in order to relate $\hat{J}$, the solution to the GRLP, and the optimal value function J^*. In particular, we come up with two novel max-norm contraction operators, viz., the least upper bound (LUB) projection operator and the approximate least upper bound projection operator (ALUB).

• We show that $||J^* - \hat{J}|| \leq (c_1 + c_2)$, where $c_1 > 0$, $c_2 > 0$ are constants. While the term c_1 corresponds to the error inherent to the ALP itself, the term c_2 constitutes the additional error introduced due to constraint approximation.

• The results from the GRLP framework solve the problem of theoretically justifying linear approximation of constraints. Unlike the bounds in (de Farias and Roy 2004) that hold only for specific RLP, our bounds hold for any GRLP and consequently for any RLP.

• We also discuss qualitatively the relative importance of our results in the context of ADP and their implication in the reinforcement learning setting.

• We demonstrate via an example in controlled queues that the experiments conform to the theory developed.

The rest of the paper is organized as follows. First, we present the basics of MDP. We then discuss the ALP technique, the basic error bounds as well as the issues and proposed solutions in literature, followed by the long-standing open questions, that we address in this paper. Finally, we present the main results of the paper namely the GRLP and its error analysis. We then present a qualitative discussion of our result followed by the numerical example.

Markov Decision Process (MDP)

In this section, we briefly discuss the basics of Markov Decision Process (MDP) (the reader is referred to (Bertsekas 2013; Puterman 1994) for a detailed treatment).

The MDP Model: An MDP is a 4-tuple $< S, A, P, g >$, where S is the state space, A is the action space, P is the probability transition kernel and g is the reward function. We consider MDPs with large but finite number of states, i.e., $S = \{1, 2, \ldots, n\}$ for some large n, and the action set is given by $A = \{1, 2, \ldots, d\}$. For simplicity, we assume that all actions are feasible in all states. The probability transition kernel P specifies the probability $p_a(s, s')$ of transitioning from state s to state s' under the action a. We denote the reward obtained for performing action $a \in A$ in state $s \in S$ by $g_a(s)$.

Policy: A policy μ specifies the action selection mechanism, and is described by the sequence $\mu = \{u_1, u_2, \ldots, u_n, \ldots\}$, where $u_n \colon S \to A$, $\forall n \geq 0$. A stationary deterministic policy (SDP) is one where $u_n \equiv u$, $\forall n \geq 0$ for some $u \colon S \to A$. By abuse of notation we denote the SDP by u itself instead of μ. In the setting that we consider, one can find an SDP that is optimal (Bertsekas 2013; Puterman 1994). In this paper, we restrict our focus to the class U of SDPs. Under an SDP u, the MDP is a Markov chain with probability transition kernel P_u.

Value Function: Given an SDP u, the infinite horizon discounted reward corresponding to state s under u is denoted by $J_u(s)$ and is defined by $J_u(s) \triangleq \mathbf{E}[\sum_{n=0}^{\infty} \alpha^n g_{a_n}(s_n) | s_0 = s, a_n = u(s_n)\ \forall n \geq 0]$, where $\alpha \in (0, 1)$ is a given discount factor. Here $J_u(s)$ is known as the value of the state s under the SDP u, and the vector quantity $J_u \triangleq (J_u(s), \forall s \in S) \in R^n$ is called the value-function corresponding to the SDP u.

The optimal SDP u^* is obtained as $u^*(s) \triangleq \arg\max_{u \in U} J_u(s)$[1].

The optimal value-function J^* is the one obtained under the optimal policy, i.e., $J^* = J_{u^*}$.

The Bellman Equation and Operator: Given an MDP, our aim is to find the optimal value function J^* and the optimal policy u^*. The optimal policy and value function obey the Bellman equation (BE) as under: $\forall s \in S$,

$$J^*(s) = \max_{a \in A} \big(g_a(s) + \alpha \sum_{s'} p_a(s, s') J^*(s')\big), \tag{1a}$$

$$u^*(s) = \arg\max_{a \in A} \big(g_a(s) + \alpha \sum_{s'} p_a(s, s') J^*(s')\big). \tag{1b}$$

Typically J^* is computed first and u^* is obtained by substituting J^* in (1b).

The Bellman operator $T \colon \mathbf{R}^n \to \mathbf{R}^n$ is defined using the model parameters of the MDP as follows:

$$(TJ)(s) = \max_{a \in A} \big(g_a(s) + \alpha \sum_{s'} p_a(s, s') J(s')\big), J \in \mathbf{R}^n.$$

Basis Solution Methods: When the number of states of the MDP is small, J^* and u^* can be computed exactly using conventional methods such as value/policy iteration and linear programming (LP) (Bertsekas 2013).

Curse-of-Dimensionality is a term used to denote the fact that the number of states grows exponentially in the number of state variables. Most MDPs occurring in practice suffer from the curse, i.e., have large number of states and it is difficult to compute $J^* \in \mathbf{R}^n$ exactly in such scenarios.

Approximate Dynamic Programming (Bertsekas 2013) (ADP) methods compute an approximate value function $\tilde{J}$ instead of J^*. In order to make the computations easier, ADP methods employ function approximation (FA) where in $\tilde{J}$ is chosen from a parameterized family of functions. The problem then boils down to finding the optimal parameter which is usually of lower dimension and is easily computable.

Linear Function Approximation (LFA) (de Farias and Roy 2003; Nedić and Bertsekas 2003; Konidaris, Osentoski, and Thomas 2011; Mahadevan and Liu 2010; Mahadevan and Maggioni 2007) is a widely used FA scheme where the approximate value function $\tilde{J} = \Phi r^*$, with $\Phi = [\phi_1 | \ldots | \phi_k]$ being an $n \times k$ feature matrix and r^*, is the parameter to be learnt.

Approximate Linear Programming

We now present the linear programming formulation of the MDP which forms the basis for ALP. The LP formulation is

[1] Such u^* exists and is well defined in the case of infinite horizon discounted reward MDP, for more details see (Puterman 1994).

obtained by unfurling the $\max$ operator in the BE in (1) into a set of linear inequalities as follows:

$$\min_{J \in \mathbf{R}^n} c^\top J$$
$$\text{s.t } J(s) \geq g_a(s) + \alpha \sum_{s'} p_a(s,s') J(s'),\ \forall s \in S, a \in A, \quad (2)$$

where $c \in \mathbf{R}^n_+$ is a probability distribution and denotes the relative importance of the various states. One can show that J^* is the solution to (2) (Bertsekas 2013). The LP formulation in (2) can be represented in short[2] as,

$$\min_{J \in \mathbf{R}^n} c^\top J$$
$$\text{s.t } J \geq TJ. \quad (3)$$

The approximate linear program (ALP) is obtained by making use of LFA in the LP, i.e., by letting $J = \Phi r$ in (3) and is given as

$$\min_{r \in \mathbf{R}^k} c^\top \Phi r$$
$$\text{s.t } \Phi r \geq T\Phi r. \quad (4)$$

Unless specified otherwise we use $\tilde{r}_c$ to denote the solution to the ALP and $\tilde{J}_c = \Phi \tilde{r}_c$ to denote the corresponding approximate value function. The following is a preliminary error bound for the ALP from (de Farias and Roy 2003):

Theorem 1 *Let* $\mathbf{1}$*, i.e., the vector with all-components equal to* 1*, be in the span of the columns of* Φ *and* c *be a probability distribution. Then, if* $\tilde{J}_c = \Phi \tilde{r}_c$ *is an optimal solution to the ALP in* (4)*, then* $||J^* - \tilde{J}_c||_{1,c} \leq \frac{2}{1-\alpha} \min_r ||J^* - \Phi r||_\infty$*, where* $||x||_{1,c} = \sum_{i=1}^n c(i)|x(i)|$.

For a more detailed treatment of the ALP and sophisticated bounds, the reader is referred to (de Farias and Roy 2003). Note that the ALP is a linear program in k ($<< n$) variables as opposed to the LP in (3) which has n variables. Nevertheless, the ALP has nd constraints (same as the LP) which is an issue when n is large and calls for constraint approximation/reduction techniques.

Related Work

Constraint sampling and The RLP: The most important work in the direction of constraint reduction is constraint sampling (de Farias and Roy 2004) wherein a reduced linear program (RLP) is solved instead of the ALP. While the objective of the RLP is same as that of the ALP, the RLP has only $m << nd$ constraints. These m constraints are *sampled* from the original nd constraints of the ALP according to a special sampling distribution $\psi_{u^*,V}$, where u^* is the optimal policy and V is a Lyapunov function (see (de Farias and Roy 2004) for a detailed presentation). If $\tilde{r}$ and $\tilde{r}_{RLP}$ are the solutions to the ALP and the RLP respectively, from (de Farias and Roy 2004) we know that $||J^* - \Phi\tilde{r}_{RLP}||_{1,c} \leq ||J^* - \Phi\tilde{r}||_{1,c} + \epsilon||J^*||_{1,c}$. A major gap in the theoretical analysis is that the error bounds are known for only a specific RLP formulated using idealized assumptions, i.e., under knowledge of u^*.

Other works: Most works in literature make use of the underlying structure of the problem to cleverly reduce the number of constraints of the ALP. A good example is (Guestrin et al. 2003), wherein the structure in factored linear functions is exploited. The use of basis function also helps constraint reduction in (Morrison and Kumar 1997). In (Borkar, Pinto, and Prabhu 2009), the constraints are approximated indirectly by approximating the square of the Lagrange multipliers. In (Petrik and Zilberstein 2009) the transitional error is reduced ignoring the representational and sampling errors. Empirical successes include repeated application of constraint sampling to solve Tetris (Farias and Roy 2006).

Long-Standing Open Questions: The fact that RLP works well empirically goads us to build a more elaborate theory for constraint reduction. In particular, one would like to answer the following questions related to constraint reduction in ALP that have so far remained open.

• As a natural generalization of the RLP, what happens if we define a generalized reduced linear program (GRLP) whose constraints are positive linear combinations of the original constraints of the ALP?

• Unlike (de Farias and Roy 2004) which provides error bounds for a specific RLP formulated using an idealized sampling distribution, is it possible to provide error bounds for any GRLP (and hence any RLP)? In this paper, we address both of the questions above.

Generalized Reduced Linear Program

We define the generalized reduced linear program (GRLP) as below:

$$\min_{r \in \chi} c^\top \Phi r,$$
$$\text{s.t } W^\top \Phi r \geq W^\top T \Phi r, \quad (5)$$

where $W \in \mathbf{R}^{nd \times m}_+$ is an $nd \times m$ matrix with all positive entries and $\chi \subset \mathbf{R}^k$ is any bounded set such that $\hat{J}_c \in \chi$. Thus the i^{th} ($1 \leq i \leq m$) constraint of the GRLP is a positive linear combination of the original constraints of the ALP, see Assumption 1. Constraint reduction is achieved by choosing $m << nd$. Unless specified otherwise we use $\hat{r}_c$ to denote the solution to the GRLP in (5) and $\hat{J}_c = \Phi \hat{r}_c$ to denote the corresponding approximate value function. We assume the following throughout the rest of the paper:

Assumption 1 $W \in \mathbf{R}^{nd \times m}_+$ *is a full rank* $nd \times m$ *matrix with all non-negative entries. The first column of the feature matrix* Φ *(i.e.,*ϕ_1*) is* $\mathbf{1}$[3] $\in \mathbf{R}^n$ *and that* $c = (c(i), i = 1, \ldots, n) \in \mathbf{R}^n$ *is a probability distribution, i.e.,* $c(i) \geq 0$ *and* $\sum_{i=1}^n c(i) = 1$*. It is straightforward to see that a RLP is trivially a GRLP.*

As a result of constraint reduction the feasible region of the GRLP is a superset of the feasible region of the ALP (see Figure 1). In order to bound $||J^* - \tilde{J}_c||$, (de Farias and Roy 2003) makes use of the property that $\Phi\tilde{r}_c \geq T\Phi\tilde{r}_c$.

[2] $J \geq TJ$ is a shorthand for the nd constraints in (2). It is also understood that constraints $(i-1)n+1, \ldots, in$ correspond to the i^{th} action.

[3] $\mathbf{1}$ is a vector with all components equal to 1.

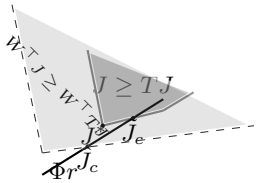

Figure 1: The outer lightly shaded region corresponds to GRLP constraints and the inner dark shaded region corresponds to the original constraints. The main contribution of the paper is to provide a bound for $||J^* - \hat{J}_c||$.

However, in the case of the GRLP, this property does not hold anymore and hence it is a challenge to bound the error $||J^* - \hat{J}_c||$. We tackle this challenge by introducing two novel $\max$-norm contraction operators called the least upper bound projection (LUBP) and approximate least upper bound projection (ALUBP) operators denoted by Γ and $\tilde{\Gamma}$ respectively. We first present some definitions before the main result and a sketch of its proof. The least upper bound (LUB) projection operator $\Gamma \colon \mathbf{R}^n \to \mathbf{R}^n$ is defined as below:

Definition 2 *Given $J \in \mathbf{R}^n$, its least upper bound projection is denoted by ΓJ and is defined as*

$$(\Gamma J)(i) \triangleq \min_{j=1,\ldots,k} (\Phi r_{e_j})(i),\ \forall i = 1,\ldots,n, \qquad (6)$$

where $V(i)$ denotes the i^{th} component of the vector $V \in \mathbf{R}^n$. Also in (6)*, e_j is the vector with 1 in the j^{th} place and zeros elsewhere, and r_{e_j} is the solution to the linear program in* (7) *for $c = e_j$.*

$$r_c \triangleq \min_{r \in \chi} c^\top \Phi r,$$
$$s.t\ \Phi r \geq TJ. \qquad (7)$$

Remark 1

1. *Given Φ and $J \in \mathbf{R}^n$, define $\mathcal{F} \triangleq \{\Phi r | \Phi r \geq TJ\}$. Thus $\mathcal{F}$ is the set of all vectors in the span of Φ that upper bound TJ. By fixing c in the linear program in* (7) *we select a unique vector $\Phi r_c \in \mathcal{F}$. The LUB projection operator Γ picks n vectors $\Phi r_{e_i}, i = 1,\ldots,n$ from the set $\mathcal{F}$ and ΓJ is obtained by computing their component-wise minimum.*
2. *Even though ΓJ does not belong to the span of Φ, ΓJ in some sense collates the various best upper bounds that can be obtained via the linear program in* (7)*.*
3. *The LUB operator Γ in* (6) *bears close similarity to the ALP in* (4)*.*

We define an approximate least upper bound (ALUB) projection operator which has a structure similar to the GRLP and is an approximation to the LUB operator.

Definition 3 *Given $J \in \mathbf{R}^n$, its approximate least upper bound (ALUB) projection is denoted by $\tilde{\Gamma} J$ and is defined as*

$$(\tilde{\Gamma} J)(i) \triangleq \min_{j=1,\ldots,k} (\Phi r_{e_j})(i),\ \forall i = 1,\ldots,n, \qquad (8)$$

where r_{e_j} is the solution to the linear program in (9) *for $c = e_j$, and e_j is same as in Definition 2.*

$$r_c \triangleq \min_{r \in \chi} c^\top \Phi r,$$
$$s.t\ W^\top \Phi r \geq W^\top TJ, W \in \mathbf{R}_+^{nd \times m}. \qquad (9)$$

Definition 4 *The LUB projection of J^* is denoted by $\bar{J} = \Gamma J^*$, and let $r^* \triangleq \arg\min_{r \in \mathbf{R}^k} ||J^* - \Phi r^*||$.*

Main Result

Theorem 5

$$||J^* - \hat{J}_c||_{1,c} \leq \frac{6||J^* - \Phi r^*||_\infty + 2||\Gamma \bar{J} - \tilde{\Gamma} \bar{J}||_\infty}{1 - \alpha}. \qquad (10)$$

Proof: Here we provide a sketch of the proof (see (Lakshminarayanan and Bhatnagar 2014) for the detailed proof). Figure 2 gives an idea of the steps that lead to the result. First, one shows that the operators Γ and $\tilde{\Gamma}$ have the $\max$-norm contraction property with factor α. As a result, operators Γ and $\tilde{\Gamma}$ have fixed points $\tilde{V} \in \mathbf{R}^n$ and $\hat{V} \in \mathbf{R}^n$ respectively. This leads to the inequalities $\tilde{J}_c \geq \tilde{V} \geq J^*$ and $\hat{J}_c \geq \hat{V}$ (see Figure 2), followed by which one can bound the term $||J^* - \hat{V}||_\infty$ and then go on to show that any solution $\tilde{r}_c$ to the GRLP is also a solution to the program in (11).

$$\min_{r \in \chi} ||\Phi r - \hat{V}||_{1,c}$$
$$\text{s.t}\ W^\top \Phi r \geq W^\top T \Phi r. \qquad (11)$$

One then obtains the bound $||J^* - \hat{J}_c||_{1,c}$ as in (10) using the fact that $||J^* - \bar{J}||_\infty \leq 2||J^* - \Phi r^*||_\infty$ where r^* is as in Definition 4.

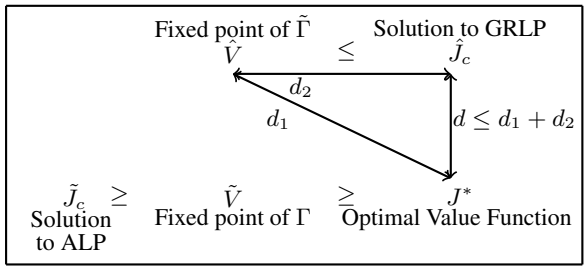

Figure 2: A schematic of the error analysis. Here $d = ||J^* - \hat{J}_c||_{1,c}$.

It is important to note that $\Gamma/\tilde{\Gamma}$ are only analytical constructs that lead us to the error bounds, and need not be calculated in practice for systems with large n.

Discussion

We now make various important qualitative observations about the result in Theorem 5.

Error Terms: The error term is split into two factors, the

first of which is related to the best possible projection while the second factor is related to constraint approximation. The second factor $||\Gamma\bar{J}-\tilde{\Gamma}\bar{J}||_\infty$ is completely defined in terms of Φ, W and T, and does not require knowledge of stationary distribution of the optimal policy. It makes intuitive sense since given that Φ approximates J^*, it is enough for W to depend on Φ and T without any additional requirements. Unlike the result in (de Farias and Roy 2004) which holds only for a specific RLP formulated under ideal assumptions, our bounds hold for any GRLP and as a result for any given RLP. Another interesting feature of our result is that it holds with probability 1. Also by making use of appropriate Lyapunov functions as in (de Farias and Roy 2003), the error bound in (10) can also be stated using a weighted L_∞-norm, thereby indicating the relative importance of states.

Additional insights on constraint sampling: It is easy to notice from Definitions 2, 3 and 4 that for any given state $s \in S, \Gamma\bar{J}(s) \geq J^*(s)$, and that $\Gamma\bar{J}(s) \geq \tilde{\Gamma}\bar{J}(s)$. If the state s is selected in the RLP, then it is also true that $\Gamma\bar{J}(s) \geq \tilde{\Gamma}\bar{J}(s) \geq J^*(s)$. Thus the additional error $|\Gamma\bar{J}(s) - \tilde{\Gamma}\bar{J}(s)|$ due to constraint sampling is less than the original projection error $|\Gamma\bar{J}(s) - J^*(s)|$ due to function approximation. This means that the RLP is expected to perform well whenever *important* states are retained after constraint sampling. Thus the sampling distribution need not be the stationary distribution of the optimal policy as long as it samples the important states, an observation that might theoretically explain the empirical successes of the RLP (de Farias and Roy 2003; Farias and Roy 2006; Desai, Farias, and Moallemi 2009).

Relation to other ADP methods:

ADP Method	Empirical	Theoretical
Projected Bellman Equation	✓	✗-Policy Chattering
ALP	✗-Large number of Constraints	✓
RLP	✓	✗- Only under ideal assumptions

A host of the ADP methods such as (Lagoudakis and Parr 2003; Nedić and Bertsekas 2003; Boyan 1999; Tsitsiklis and Roy 1997) are based on solving the projected Bellman equation (PBE). The PBE based methods have been empirically successful and also have theoretical guarantees for the approximate value function. However, a significant shortcoming is that they suffer from the issue of *policy-chattering* (see section 6.4.3 of (Bertsekas 2013)), i.e., the sequence of policies might oscillate within a set of bad policies. A salient feature of the ALP based methods is that they find only one approximate value function $\tilde{J}_c$ and one sub-optimal policy derived as a greedy policy with respect to $\tilde{J}_c$. As a result there is no such issue of policy-chattering for the ALP based methods. By providing the error bounds for the GRLP, our paper provides the much required theoretical support for the RLP. Our GRLP framework closes the long-standing gap in the literature of providing a theoretical framework to bound the error due to constraint reduction in ALP based schemes.

GRLP is linear function approximation of the constraints: In order to appreciate this fact consider the Lagrangian of the ALP and GRLP in (12) and (13), respectively, i.e.,

$$\tilde{L}(r,\lambda) = c^\top \Phi r + \lambda^\top (T\Phi r - \Phi r), \quad (12)$$

$$\hat{L}(r,q) = c^\top \Phi r + q^\top W^\top (T\Phi r - \Phi r). \quad (13)$$

The insight that the GRLP is linear function approximation of constraints (i.e., the Lagrangian multipliers) can be obtained by noting that $Wq \approx \lambda$ in (13). Note that while the ALP employs LFA in its objective, the GRLP employs linear approximation both in the objective as well as the constraints. This has significance in the context of the reinforcement learning setting (Sutton and Barto 1998) wherein the model information is available in the form of noisy sample trajectories. RL algorithms make use of stochastic approximation (SA) (Borkar 2008) and build on ADP methods to come up with incremental update schemes to learn from noisy samples presented to them and linear approximation architectures are found to be useful in this setting. An SA scheme to solve the GRLP in the RL setting can be derived in a manner similar to (Borkar, Pinto, and Prabhu 2009).

Application to Controlled Queues

We take up an example in the domain of controlled queues to show that experiments are in agreement with the theory developed. More specifically, we look at the error bounds for different constraints reduction schemes to demonstrate the fact that whenever value of $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ is less, the GRLP solution is close to the optimal value function.

The queuing system consists of $n = 10^4$ states and $d = 4$ actions. We chose $n = 10^4$ because it was possible to solve both the GRLP and the exact LP (albeit with significant effort) so as to enumerate the approximation errors. We hasten to mention that while we could run the GRLP for queuing systems with $n > 10^4$ without much computational overhead, solving the exact LP was not possible for $n > 10^4$, as a result of which the approximation error could not be computed.

Queuing Model: The queuing model used here is similar to the one in Section 5.2 of (de Farias and Roy 2003). We consider a single queue with arrivals and departures. The state of the system is the queue length with the state space given by $S = \{0, \ldots, n-1\}$, where $n-1$ is the buffer size of the queue. The action set $A = \{1, \ldots, d\}$ is related to the service rates. We let s_t denote the state at time t. The state at time $t+1$ when action $a_t \in A$ is chosen is given by $s_{t+1} = s_t + 1$ with probability p, $s_{t+1} = s_t - 1$ with probability $q(a_t)$ and $s_{t+1} = s_t$, with probability $(1 - p - q(a_t))$. For states $s_t = 0$ and $s_t = n-1$, the system dynamics is given by $s_{t+1} = s_t + 1$ with probability p when $s_t = 0$ and $s_{t+1} = s_t - 1$ with probability $q(a_t)$ when $s_t = n-1$. The service rates satisfy $0 < q(1) \leq \ldots \leq q(d) < 1$ with $q(d) > p$ so as to ensure 'stabilizability' of the queue. The reward associated with the action $a \in A$ in state $s \in S$ is given by $g_a(s) = -(s + 60q(a)^3)$.

Choice of Φ : We make use of polynomial features in Φ (i.e., $1, s, \ldots, s^{k-1}$) since they are known to work well for this domain (de Farias and Roy 2003). This takes care of the term $||J^* - \Phi r^*||_\infty$ in (10).

Selection of W: For our experiments, we choose two contenders for the W-matrix:

(**i**) W_c- matrix that corresponds to sampling according to c. This is justified by the insights obtained from the error term $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ and the idea of selecting the important states. (**ii**) W_a state-aggregation matrix, a heuristic derived by interpreting W to be the feature matrix that approximates the Lagrange multipliers as $\lambda \approx Wq$, where $\lambda \in \mathbf{R}^{nd}, r \in \mathbf{R}^m$. One can show (Dolgov and Durfee 2006) that the optimal Lagrange multipliers are the discounted number of visits to the 'state-action' pairs under the optimal policy u^*, i.e., $\lambda^*(s,u^*(s)) = \big(c^\top(I - \alpha P_{u^*})^{-1}\big)(s) = \big(c^\top(I + \alpha P_{u^*} + \alpha^2 P_{u^*}^2 + \ldots)\big)(s), \lambda^*(s,u^*(s)) = 0, \forall a \neq u^*(s)$, where P_{u^*} is the probability transition matrix with respect to the optimal policy. Even though we might not have the optimal policy u^* in practice, the fact that λ^* is a linear combination of $\{P_{u^*}, P_{u^*}^2, \ldots\}$ hints at the kind of features that might be useful for the W matrix. Our choice of W_a matrix to correspond to aggregation of nearby states is motivated by the observation that P^n captures n^{th} hop connectivity/neighborhood information. The aggregation matrix W_a is defined as below: $\forall i = 1, \ldots, m$,

$$W_a(i,j) = 1, \ \forall j \text{ s.t } j = (i-1)\frac{n}{m} + k + (l-1)n,$$
$$k = 1, \ldots, \frac{n}{m}, l = 1, \ldots, d,$$
$$= 0, \text{ otherwise.} \quad (14)$$

In order to provide a contrast between good and bad choices of W matrices we also make use of two more matrices, an ideal matrix W_i generated by sampling according to the stationary distribution of the optimal policy as in (de Farias and Roy 2004) and W_c generated by sampling using c, as well as W_r, a random matrix in $\mathbf{R}_+^{nd \times m}$. For the sake of comparison, we compute $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ for the different W matrices. Though computing $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ might be hard in the case of large n, since $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ is completely dependent on the structure of Φ, T and W, we can compute it for small n instead and use it as a surrogate. Accordingly, we first chose a smaller system, Q_S, with $n = 10$, $d = 2$, $k = 2$, $m = 5$, $q(1) = 0.2$, $q(2) = 0.4$, $p = 0.2$ and $\alpha = 0.98$. In the case of Q_S, W_a ((14) with $m = 5$) turns out to be a 20×5 matrix where the i^{th} constraint of the GRLP is the average of all constraints corresponding to states $(2i-1)$ and $2i$ (there are four constraints corresponding to these two states). The various error terms are listed in Table 1 and plots are shown in Figure 3. It is clear from Table 1 that W_a, W_i and W_c have much better $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ than randomly generated positive matrices. Since each constraint is a hyperplane, taking linear combinations of non-adjacent hyperplanes might drastically affect the final solution. This could be a reason why W_r (random matrix) performs poorly in comparison with other W matrices.

Error Term	W_i	W_c	W_a	W_r
$\|\|\Gamma\bar{J} - \tilde{\Gamma}\bar{J}\|\|_\infty$	39	84	54.15	251.83

Table 1: Shows various error terms for Q_S.

Next we consider a moderately large queuing system Q_L with $n = 10^4$ and $d = 4$ with $q(1) = 0.2$, $q(2) = 0.4$, $q(3) = 0.6$, $q(4) = 0.8$, $p = 0.4$ and $\alpha = 0.98$. In the case of Q_L, we chose $k = 4$ (i.e., we used $1, s, s^2$ and s^3 as basis vectors) and we chose W_a (14), W_c, W_i and W_r with $m = 50$. We set $c(s) = (1-\zeta)\zeta^s$, $\forall s = 1, \ldots, 9999$, with $\zeta = 0.9$ and $\zeta = 0.999$ respectively. The results in Table 2 show that performance exhibited by W_a and W_c is better by several orders of magnitude over 'random' in the case of the large system Q_L and is close to the ideal sampler W_i. Also note that a better performance of W_a and W_c in the larger system Q_L is in agreement with a lower value of $||\Gamma\bar{J} - \tilde{\Gamma}\bar{J}||_\infty$ in the smaller system Q_S.

Error Terms	W_i	W_c	W_a	W_r
$\|\|J^* - \hat{J}_c\|\|_{1,c}$ for $\zeta = 0.9$	32	32	220	5.04×10^4
$\|\|J^* - \hat{J}_c\|\|_{1,c}$ for $\zeta = 0.999$	110	180.5608	82	1.25×10^7

Table 2: Shows performance metrics for Q_L.

Conclusion

Solving MDPs with large number of states is of practical interest. However, when the number of states is large, it is difficult to calculate the exact value function. ALP is a widely studied ADP scheme that computes an approximate value function and offers theoretical guarantees. Nevertheless, the ALP is difficult to solve due to its large number of constraints and in practice a reduced linear program (RLP) is solved. Though RLP has been shown to perform well empirically, theoretical guarantees are available only for a specific RLP formulated under idealized assumptions. This paper provided a more elaborate treatment of constraint reduction/approximation. Specifically, we generalized the RLP to formulate a generalized reduced linear program (GRLP) and provided error bounds. Our results addressed a major long-standing open problem of analytically justifying linear function approximation of the constraints. We discussed the implications of our results in the contexts of ADP and reinforcement learning. We found that our experiments conform to the theory developed in this paper on an example in the domain of controlled queues. Future directions include providing more sophisticated error bounds based on Lyapunov functions, a two-time scale actor-critic scheme to solve the GRLP, and basis function adaptation schemes to tune the W matrix.

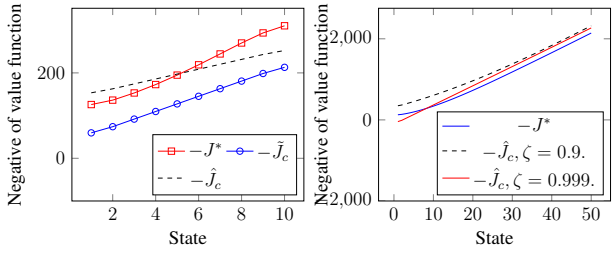

Figure 3: Plot corresponding to Q_S on the left and Q_L on the right. The GRLP here used W_a in (14) with $m = 5$ for Q_S and $m = 50$ for Q_L.

References

[Bertsekas 2013] Bertsekas, D. P. 2013. *Dynamic Programming and Optimal Control*, volume II. Belmont,MA: Athena Scientific, 4^{th} edition.

[Borkar, Pinto, and Prabhu 2009] Borkar, V. S.; Pinto, J.; and Prabhu, T. 2009. A new learning algorithm for optimal stopping. *Discrete Event Dynamic Systems* 19(1):91–113.

[Borkar 2008] Borkar, V. S. 2008. *Stochastic Approximation: A Dynamical Systems Viewpoint*. TRIM.

[Boyan 1999] Boyan, J. A. 1999. Least-squares temporal difference learning. In *ICML*, 49–56. Citeseer.

[de Farias and Roy 2003] de Farias, D. P., and Roy, B. V. 2003. The linear programming approach to approximate dynamic programming. *Operations Research* 51(6):850–865.

[de Farias and Roy 2004] de Farias, D. P., and Roy, B. V. 2004. On constraint sampling in the linear programming approach to approximate dynamic programming. *Math. Oper. Res.* 29(3):462–478.

[Desai, Farias, and Moallemi 2009] Desai, V. V.; Farias, V. F.; and Moallemi, C. C. 2009. A smoothed approximate linear program. In *NIPS*, 459–467.

[Dolgov and Durfee 2006] Dolgov, D. A., and Durfee, E. H. 2006. Symmetric approximate linear programming for factored mdps with application to constrained problems. *Annals of Mathematics and Artificial Intelligence* 47(3-4):273–293.

[Farias and Roy 2006] Farias, V. F., and Roy, B. V. 2006. Tetris: A study of randomized constraint sampling. In *Probabilistic and Randomized Methods for Design Under Uncertainty*. Springer. 189–201.

[Guestrin et al. 2003] Guestrin, C.; Koller, D.; Parr, R.; and Venkataraman, S. 2003. Efficient solution algorithms for factored MDPs. *J. Artif. Intell. Res.(JAIR)* 19:399–468.

[Konidaris, Osentoski, and Thomas 2011] Konidaris, G.; Osentoski, S.; and Thomas, P. S. 2011. Value function approximation in reinforcement learning using the Fourier basis. In *AAAI*.

[Lagoudakis and Parr 2003] Lagoudakis, M. G., and Parr, R. 2003. Least-squares policy iteration. *Journal of Machine Learning Research* 4:1107–1149.

[Lakshminarayanan and Bhatnagar 2014] Lakshminarayanan, C., and Bhatnagar, S. 2014. A generalized reduced linear program for markov decision processes. *CoRR* abs/1409.3536v2.

[Mahadevan and Liu 2010] Mahadevan, S., and Liu, B. 2010. Basis construction from power series expansions of value functions. In *Advances in Neural Information Processing Systems*, 1540–1548.

[Mahadevan and Maggioni 2007] Mahadevan, S. S., and Maggioni, M. 2007. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision Processes. *Journal of Machine Learning Research* 8(16):2169–2231.

[Morrison and Kumar 1997] Morrison, J. R., and Kumar, P. R. 1997. New linear program performance bounds for queueing networks. Technical Report 3, Journal of Optimization Theory and Applications.

[Nedić and Bertsekas 2003] Nedić, A., and Bertsekas, D. P. 2003. Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems* 13(1-2):79–110.

[Petrik and Zilberstein 2009] Petrik, M., and Zilberstein, S. 2009. Constraint relaxation in approximate linear programs. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 809–816. ACM.

[Puterman 1994] Puterman, M. L. 1994. *Markov Decision Processes: Discrete Stochastic Programming*. New York: John Wiley.

[Sutton and Barto 1998] Sutton, R. S., and Barto, A. G. 1998. *Introduction to Reinforcement Learning*. Cambridge, MA, USA: MIT Press, 1st edition.

[Tsitsiklis and Roy 1997] Tsitsiklis, J. N., and Roy, B. V. 1997. An analysis of temporal-difference learning with function approximation. Technical report, IEEE Transactions on Automatic Control.

Approximate dynamic programming with $(\min, +)$ linear function approximation for Markov decision processes

Chandrashekar L[1] and Shalabh Bhatnagar[2]

***Abstract*— Markov Decision Process (MDP) is a useful framework to study problems of optimal sequential decision making under uncertainty. Given any MDP the aim here is to find the optimal action selection mechanism i.e., the optimal policy. Typically, the optimal policy (u^*) is obtained by substituting the optimal value-function (J^*) in the Bellman equation. Alternatively, u^* is also obtained by learning the optimal state-action value function Q^* known as the Q value-function. However, it is difficult to compute the exact values of J^* or Q^* for MDPs with large number of states. Approximate Dynamic Programming (ADP) methods address this difficulty by computing lower dimensional approximations of J^*/Q^*. Most ADP methods employ linear function approximation (LFA), i.e., the approximate solution lies in a subspace spanned by a family of pre-selected basis functions. The approximation is obtained via a linear least squares projection of higher dimensional quantities and the L_2 norm plays an important role in convergence and error analysis. In this paper, we discuss ADP methods for MDPs based on LFAs in the $(\min, +)$ algebra. Here the approximate solution is a $(\min, +)$ linear combination of a set of basis functions whose span constitutes a subsemimodule. Approximation is obtained via a projection operator onto the subsemimodule which is different from linear least squares projection used in ADP methods based on conventional LFAs. MDPs are not $(\min, +)$ linear systems, nevertheless, we show that the monotonicity property of the projection operator helps us establish the convergence of our ADP schemes. We also discuss future directions in ADP methods for MDPs based on the $(\min, +)$ LFAs.**

I. Introduction

Markov Decision Process (MDP) is a framework to pose optimal sequential decision making problems. In particular, an MDP is a four tuple $< S, A, P, g >$, where S is the state space, A is the action space, P is the probability transition kernel and g is the reward function. In this paper, we consider MDPs with $|S| = n$ states and $|A| = d$ actions. A policy is a map $u\colon S \to A$ and under u action $u(s)$ is chosen in states $s \in S$. The value of a state s under policy u is denoted by $J_u(s)$, and $J_u = (J_u(s), s \in S) \in \mathrm{R}^n$ is the value function corresponding to policy u. Solving an MDP involves solving two sub-problems namely *prediction* and *control*. The *prediction* problem constitutes evaluating the value function $J_u \in \mathrm{R}^n$ of a given policy u. The *control* problem involves coming up with desirable policies. The optimal value function is the fixed point of the Bellman equation given as $J^* = TJ^*$, where T is the Bellman operator. Once J^* is known u^* can be obtained by substituting J^* in the Bellman equation. The problem of prediction and control can be addressed simultaneously by computing $Q^* \in \mathrm{R}^{n\times d}$, known as the Q value function. Q^* encodes both J^* and u^*, and obeys the equation $Q^* = HQ^*$, where H is the Q-Bellman operator. When n is small, conventional methods such as value iteration (VI), policy iteration (PI) and linear programming (LP) can be used to compute J^*/Q^* and u^*. VI computes J^*/u^* and is based on the contraction property of T/H. PI computes J_{u_n} (*prediction*) of policy u_n based on which obtains an improved policy u_{n+1} (*control*) such that the sequence $u_n \to u^*$ as $n \to \infty$. The LP method computes J^* by representing the BE as a set of linear inequalities.

When n is large computing exact values of J^*/u^* is not possible and we need to resort to Approximate Dynamic Programming (ADP) methods that compute an approximate value-function $\tilde{J}$ and a sub-optimal policy $\tilde{u}$. Most ADP methods employ linear function approximators (LFAs), i.e., approximate the value function is given by $\tilde{J} = \Phi r^*$, where Φ is a $n\times k$ feature matrix and $r^* \in \mathrm{R}^k$ ($k << n$) is a weight vector to be computed. Approximate policy iteration (API) is the ADP analogue of PI in that it computes $\tilde{J}_u \approx J_u$ that obeys the Projected Bellman Equation (8) and then uses $\tilde{J}_u$ to perform policy improvement. An important shortcoming is that only $||J_u - \tilde{J}_u||_2$ can be bounded under certain restricted assumptions. However in order to guarantee policy improvement one needs to bound $||J_u - \tilde{J}_u||_\infty$. Consequently policy improvement is not guaranteed and API might not produce a sequence of convergent policies always and even in the cases when API converges to a sub-optimal policy $\tilde{u}$ its performance cannot be ascertained. Also, there is no convergent ADP based on conventional LFA to compute approximate J^*/Q^*. Approximate Linear Programming (ALP) [1] offers guarantee on the approximate policy but is limited by the large number of constraints. As a result, ADP methods based on conventional LFAs either do not address both the prediction and control problems (in the case of API) or do not alleviate the curse altogether (in the case of ALP).

The $(\min, +)$ $\big((\max, +)\big)$ algebra differs from conventional algebra, in that '$+$' and '$\times$' operators are replaced by '$\min$' ('$\max$') and '$+$' respectively. It is known that finite horizon deterministic optimal control problems based on maximizing a reward criterion are $(\max, +)$ linear transformations. A lot of work has been reported in the literature [2], [3], [4], [5] that make use of $(\max, +)$ basis to compute approximate value-functions for deterministic optimal con-

Work was supported by projects from Department of Science and Technology, Government of India as well as Xerox Corporation, USA.

[1]Chandrashekar L is a researcher with the Department of Computer Science and Automation Indian Institute of Science, Bangalore-560012, India. chandrul@csa.iisc.ernet.in

[2]Shalabh Bhatnagar is a faculty member with the Department of Computer Science and Automation Indian Institute of Science, Bangalore-560012, India. shalabh@csa.iisc.ernet.in

trol problems. The primary focus of the paper is to explore ADP schemes for MDPs based on $(\min, +)$ LFAs as opposed to conventional LFAs. The organization of the paper and our specific contributions in this paper are listed below.

1) In section II we briefly introduce the MDP setting and then discuss the basic solution methods.
2) In section III we discuss the ADP methods based on conventional LFAs and their shortcomings.
3) In section IV we introduce the $(\min, +)$ linear basis and introduce the PBE in the $(\min, +)$ basis.
4) In sections V and VI we present the main contribution of this paper namely the Approximate Q Iteration (AQI) and Variational Approximate Q Iteration schemes respectively. Both AQI and VAQI make use of the projected Bellman operator (denoted by Π_M) and the variational form of the projected Bellman operator (denoted by Π_M^W) respectively to compute $\tilde{Q} \approx Q^*$. We show that Π_M and Π_M^W are contraction maps in the L_∞-norm. Since AQI and VAQI schemes compute an approximate Q value they address the prediction and control problems, and using the L_∞ norm contraction property of Π_M and Π_M^W one can show convergence of these schemes and provide error bounds for the sub-optimal policies obtained by them.

II. Markov Decision Processes

An MDP is characterized by its state space S, action space A, the reward function $g\colon S \times A \to \mathrm{R}$, and the probability of transition from state s to s' under action a denoted by $p_a(s, s')$. The reward for selecting an action a in state s is denoted by $g_a(s)$. We consider MDP with state space $S = \{1, 2, \ldots, n\}$ and action set $A = \{1, 2, \ldots, d\}$. For simplicity, we assume that all actions $a \in A$ are feasible in all states $s \in S$. A policy is a map $u\colon S \to A$ that describes the action selection mechanism[1]. Under a policy u the MDP is a Markov chain and we denote its probability transition kernel by $P_u = (p_{u(i)}(i, j), i = 1 \text{ to } n, j = 1 \text{ to } n)$. The expected discounted reward starting from state s when following policy u is denoted by $J_u(s)$ and is defined as

$$J_u(s) = \mathbf{E}[\sum_{t=0}^{\infty} \alpha^t g_{a_t}(s_t) | s_0 = s, u].$$

Here $\{s_t\}$ is the trajectory or sequence of states of the Markov chain under u when $s_0 = s$, and $a_t = u(s_t), \forall t \geq 0$. Also $\alpha \in (0, 1)$ is a given discount factor. We call $J_u = (J_u(s), \forall s \in S) \in \mathrm{R}^n$ the value-function corresponding to policy u. The optimal policy denoted as u^* is given by $u^* = (u^*(s), s \in S)$ with

$$u^*(s) = \arg\max_{u \in U} J_u(s), \forall s \in S,$$

where U is the set of all SDPs. The optimal value function is then $J^*(s) = J_{u^*}(s), \forall s \in S$. The optimal value function and optimal policy are related by the Bellman Equation as below:

$$J^*(s) = \max_{a \in A}(g_a(s) + \alpha \sum_{s'=1}^{n} p_a(s, s') J^*(s')), \tag{1a}$$

$$u^*(s) = \arg\max_{a \in A}(g_a(s) + \alpha \sum_{s'=1}^{n} p_a(s, s') J^*(s')). \tag{1b}$$

Usually, J^* is computed first and u^* is obtained by substituting J^* in (1b). One can also define the state-action value-function of a policy u, i.e., the Q value-function as follows:

$$Q_u(s, a) = \mathbf{E}[\sum_{t=0}^{\infty} \alpha^t g_{a_t}(s_t) | s_0 = s, a_0 = a],$$

with $a_t = u(s_t), \forall t > 0$. The optimal Q function obeys the Q Bellman equation given below:

$$Q^*(s, a) = g_a(s) + \alpha \sum_{s'} p(s, s') \max_{a'} Q^*(s', a'),$$

$\forall s \in S, a \in A$. It is easy to see that $J^*(s) = \max_a Q^*(s, a)$. The optimal policy can be computed as $u^*(s) = \arg\max_a Q^*(s, a)$. Thus, computing Q^* addresses both the problems of *prediction* and *control* since it encodes both J^* as well as u^*.

A. Basic Solution Methods

It is important to note that J^* and Q^* are the fixed points of maps T and H respectively defined as follows: For $J \in \mathrm{R}^n$ and $Q \in \mathrm{R}^{n \times d}$

$$(TJ)(s) = \max_{a \in A}(g_a(s) + \alpha \sum_{j=1}^{n} p_a(s, s') J(s')), \tag{2a}$$

$$(HQ)(s, a) = (g_a(s) + \alpha \sum_{j=1}^{n} p_a(s, s') \max_{a \in A} Q(s', a)), \tag{2b}$$

T and H are called the Bellman and Q-Bellman operators respectively. Given $J \in \mathrm{R}^n, Q \in \mathrm{R}^{n \times d}$, TJ and HQ are the 'one-step' greedy value-functions. We summarize certain useful properties of H in the following lemmas (see [7] for proofs).

Lemma 1: H is a max-norm contraction operator, i.e., given $Q_1, Q_2 \in \mathbf{R}^{n \times d}$

$$||HQ_1 - HQ_2||_\infty \leq \alpha ||Q_1 - Q_2||_\infty. \tag{3}$$

Corollary 1: Q^* is a unique fixed point of H.

Lemma 2: H is a monotone map, i.e., given $Q_1, Q_2 \in \mathrm{R}^{n \times d}$ such that $Q_1 \geq Q_2$, it follows that $HQ_1 \geq HQ_2$.

Lemma 3: Given $Q \in \mathrm{R}^{n \times d}$, $k \in \mathrm{R}$ and $\mathbf{1} \in \mathrm{R}^{n \times d}$ - a vector with all entries 1, we have

$$H(Q + k\mathbf{1}) = HQ + \alpha k \mathbf{1}. \tag{4}$$

It is easy to check that Lemmas1, 2 and 3 hold for T as well ([7]).

Value iteration (VI) is the most basic method to compute J^*/Q^* and works using the fixed point iterations

$$J_{n+1} = TJ_n, \tag{5a}$$

$$Q_{n+1} = HQ_n. \tag{5b}$$

[1] A policy thus defined is known as stationary deterministic policy (SDP). Policies can also be non-stationary and randomized. However since there exists an optimal policy that is SDP ([6]) we restrict our treatment to SDPs alone.

Iterations in (5) constitute an exact method, and the contraction property of the Bellman operator ensures that $J_n \to J^*$ in (5a) and $Q_n \to Q^*$ in (5b), respectively as $n \to \infty$. They are also referred to as *look-up-table* methods or full state representation methods, as opposed to methods employing function approximation. u^* can be computed by substituting J^* in (1b). Another basic solution method is Policy Iteration (PI) presented in Algorithm 1. Alternatively, J^* can also be

Algorithm 1 Policy Iteration

1: Start with any policy u_0
2: **for** $i = 0, 1, \ldots, n$ **do**
3: Evaluate policy u_i by computing J_{u_i}.
4: Improve policy $u_{i+1}(s) = \arg\max_a(g_a(s) + \alpha \sum_{s'} p_a(s, s')J_{u_i}(s'))$.
5: **end for**
6: **return** u_n

computed using the linear programming (LP) formulation given by

$$\min \; c^\top J \tag{6}$$
$$s.t \quad J(s) \geq g(s, a) + \alpha \sum_{s'} p_a(s, s')J(s'), \forall s \in S, a \in A;$$

VI and PI form the basis of ADP methods explained in the next section.

III. Approximate Dynamic Programming in conventional LFAs

The phenomenon called *curse-of-dimensionality* denotes the fact that the number of states grows exponentially in the number of state variables. Due to the *curse*, as the number of variables increase, it is hard to compute exact values of J^*/Q^* and u^*. Approximate Dynamic Programming (ADP) methods make use of (1) and dimensionality reduction techniques to compute an approximate value-function $\tilde{J}$. Then $\tilde{J}$ can be used to obtain an approximate policy $\tilde{u}$ which is greedy with respect to $\tilde{J}$ as follows:

$$\tilde{u}(s) = \arg\max_a(g_a(s) + \alpha \sum_{s'} p_a(s, s')\tilde{J}(s')). \tag{7}$$

A bound on the error between the value-function $J_{\tilde{u}}$ corresponding to the greedy policy in (7) and the optimal value function is given by the following result.

Lemma 4: Let $\tilde{J} = \Phi r^*$ be the approximate value function and $\tilde{u}$ be as in (7), then

$$||J_{\tilde{u}} - J^*||_\infty \leq \frac{2}{1-\alpha}||J^* - \tilde{J}||_\infty.$$

Proof: See [7]. ■

Thus a good ADP method is one that addresses both the *prediction* (i.e., computing $\tilde{J}$) and the *control* (i.e., computing $\tilde{u}$) problems with desirable approximation guarantees.

LFA has been widely employed for its simplicity and ease of computation. Here, one typically lets $\tilde{J} \in V \subset \mathrm{R}^n$, where V is the subspace spanned by a set of preselected basis functions $\{\phi_i \in \mathrm{R}^n, i = 1, \ldots, k\}$. Let Φ be the $n \times k$ matrix with columns $\{\phi_1, \ldots, \phi_k\}$, and $V = \{\Phi r | r \in \mathrm{R}^k\}$, then the approximate value function $\tilde{J}$ is of the form $\tilde{J} = \Phi r^*$ for some $r^* \in \mathrm{R}^k$. Here, r^* is a weight vector to be learnt, and due to dimensionality reduction ($k << n$), computing $r^* \in \mathrm{R}^k$ is easier than computing $J^* \in \mathrm{R}^n$.

We now discuss popular ADP methods namely approximate policy evaluation (APE) and approximate policy iteration (API), which are analogous to VI and PI. APE and API are based on solving the projected Bellman equation which is analogous to Bellman equation in (1).

A. Approximate Policy Evaluation

J^* or Q^* are usually not known and hence finding their projections onto V is not possible. Nevertheless, one may use the Bellman operator and the linear least squares projection operator to write down a projected Bellman equation (PBE) as below:

$$\Phi r^* = \Pi T_u \Phi r^*, \tag{8}$$

where $\Pi = \Phi(\Phi^\top D \Phi)^{-1}\Phi^\top$ is the projection operator, D is a diagonal matrix with all diagonal entries strictly greater than 0 and T_u is the Bellman operator restricted to a policy u and is given by

$$(T_u J)(s) = g_{u(s)}(s) + \alpha \sum_{s'} p_{u(s)}(s, s')J(s'), J \in \mathrm{R}^n.$$

The approximate policy evaluation (APE) in this setting is the following iteration:

$$\Phi r_{n+1} = \Pi T_u \Phi r_n. \tag{9}$$

Here, $\Pi T_u \colon \mathrm{R}^n \to \mathrm{R}^n$ is the projected Bellman operator and the convergence of (9) can be established by showing that ΠT_u is a contraction map.

Lemma 5: If $\Pi = \Phi(\Phi^\top D \Phi)^{-1}\Phi^\top$, and D is a diagonal matrix with the i^{th} diagonal entry being the stationary probability of visiting state i under policy u, then ΠT_u is a contraction map with factor α.

Proof: See [7]. ■

Corollary 2: The iteration in (9) converges to Φr^* such that $\Phi r^* = \Pi T_u \Phi r^*$.

The error bound for Φr^* is given by

$$||J_u - \Phi r^*||_D \leq \frac{1}{\sqrt{1-\alpha^2}}||J_u - \Pi J_u||_D. \tag{10}$$

One is inclined to think that iteration (9) being an ADP analogue of (5a) would yield an approximation to J^*. However, it is important to note that (9) only computes $\tilde{J}_u$ because (9) contains T_u and not T. Since the operator ΠT might not be a contraction map in the L_2 norm, T_u cannot be replaced by T in iteration (9), and the PBE in (8).

B. Approximate Policy Iteration

Approximate Policy Iteration (Algorithm 2) tackles both prediction and control problems, by performing APE and policy improvement at each step.

Algorithm 2 Approximate Policy Iteration (API)

1: Start with any policy u_0
2: **for** $i = 0, 1, \ldots, n$ **do**
3: Approximate policy evaluation $\tilde{J}_i = \Phi r_i^*$, where $\Phi r_i^* = \Pi T_{u_i} \Phi r_i^*$.
4: Improve policy $u_{i+1}(s) = \arg\max_a (g_a(s) + \alpha \sum_{s'} p_a(s, s') \tilde{J}_i(s'))$.
5: **end for**
6: **return** Suboptimal policy $\tilde{u}$=u_n

The performance guarantee of API can be stated as follows:

Lemma 6: If at each step i one can guarantee that $||\tilde{J}_i - J_{u_i}||_\infty \leq \delta$, then one can show that $\lim_{n\to\infty} ||J_{u_i} - J^*|| \leq \frac{2\delta\alpha}{(1-\alpha)^2}$.

Note that the error bound required by Lemma 6 is in the L_∞ norm, whereas (10) is only in the L_2 norm. So API cannot guarantee an approximate policy improvement at each step which is a shortcoming. Also, even though each evaluation step (line 3 of Algorithm 2) converges, the sequence $u_n, n \geq 0$ is not guaranteed to converge. This is known as policy *chattering* and is another shortcoming of conventional LFAs. Thus the problem of *control* is only partially addressed by API, i.e, suffers in both convergence and performance guarantees.

C. LFAs for Q value-function

To alleviate the shortcomings in API, it is then natural to look for an approximation method that computes a policy in a more direct fashion. Since we know that by computing Q^* we obtain the optimal policy directly, it is a good idea to approximate Q^*. The PBE version of (5b) is a plausible candidate and the iterations are given by

$$\Phi r_{n+1} = \Pi H \Phi r_n. \tag{11}$$

The above scheme runs into the following problems:

1) The H operator (2b) contains a $\max$ term, and it is not straightforward to show that ΠH is a contraction map in L_2 norm, and consequently one cannot establish the convergence of iterates in (11).
2) The issue pertaining to $\max$ operator can be alleviated by restricting H to a policy u, i.e., by considering iterations of the form

$$\Phi r_{n+1} = \Pi H_u \Phi r_{n+1}, \tag{12}$$

where H_u is the Q-Bellman analog of the operator T_u. However, iterates in (12) attempt to approximate Q_u and not Q^*, which means the problem of *control* is unaddressed.

D. Approximate Linear Programming

An ADP method making use of LFA based on the LP formulation (6) of the MDP is the ALP formulation given by

$$\min \; c^\top \Phi r \tag{13}$$
$$s.t \quad \Phi r(s) \geq g_a(s) + \alpha \sum_{s'} p_a(s, s')(\Phi r)(s'), \forall s \in S, a \in A.$$

The ALP computes an approximate value function $\tilde{J}$ and a sub-optimal policy $\tilde{u}$ can be obtained by substituting $\tilde{J}$ in the BE. The ALP method does offer performance guarantees [1] for $\tilde{u}$, however it is limited by the large number of constraints. We conclude the section by making the following observations about ADPs based on conventional LFAs.

- Methods such as APE, API which make use of PBE in (8) neither have guaranteed convergence nor performance bounds for the sub-optimal policy $\tilde{u}$.
- The ALP, while it offers performance guarantees for $\tilde{u}$, is limited by the large number of constraints.

We now present the main contribution of the paper namely the Approximate Q Iteration (AQI) and Variational Approximate Q Iteration (VAQI) schemes. In particular, VAQI does not suffer from the aforementioned shortcomings of the ADP methods based on conventional LFAs.

IV. $(\min, +)$ LINEAR FUNCTIONS

The $\mathbf{R}_{\min}$ semiring is obtained by replacing multiplication $(\times)$ by $+$, and addition $(+)$ by $\min$.

Definition 1:

$$\text{Addition:} \qquad x \oplus y = \min(x, y).$$
$$\text{Multiplication:} \qquad x \otimes y = x + y.$$

Henceforth we use, $(+, \times)$ and $(\oplus, \otimes)$ to respectively denote the conventional and $\mathbf{R}_{\min}$ addition and multiplication respectively. A Semimodule over a semiring can be defined in a similar manner to vector spaces over fields. In particular we are interested in the semimodule $\mathcal{M} = \mathbf{R}_{\min}^n$ (since $J^* \in \mathbf{R}_{\min}^n$). Given $u, v \in \mathbf{R}_{\min}^n$, and $\lambda \in \mathbf{R}_{\min}$, we define addition and scalar multiplication as follows:

Definition 2:

$$(u \oplus v)(i) = \min\{u(i), v(i)\} = u(i) \oplus v(i), \forall i = 1, 2, \ldots, n.$$
$$(u \otimes \lambda)(i) = u(i) \otimes \lambda = u(i) + \lambda, \forall i = 1, 2, \ldots, n.$$

Similarly one can define the $\mathrm{R}_{\max}$ semiring which has the operators $\max$ as addition and $+$ as multiplication.

It is a well known fact that deterministic optimal control problems with cost/reward criterion are $(\min, +)$/$(\max, +)$ linear ([2], [3], [4], [5]). However, it is straightforward to show that the Bellman operator T (as well as H) in (2) corresponding to infinite horizon discounted reward MDPs is neither $(\min, +)$ linear nor $(\max, +)$ linear. Nevertheless, the motivation behind developing ADP methods based on $(\min, +)$ LFAs is to explore them as an alternative to the conventional basis representation.

Given a set of basis functions $\{\phi_i, i = 1, \ldots, k\}$, we define the $(\min, +)$ linear span to be $\mathcal{V} = \{v | v = \Phi \otimes r \stackrel{def}{=} \min(\phi_1 + r(1), \ldots, \phi_k + r(k)), r \in \mathbf{R}_{\min}^k\}$. Thus $\mathcal{V}$ is a subsemimodule. In the context of value function approximation, we would want to project quantities in $\mathbf{R}_{\min}^n$ onto $\mathcal{V}$. The $(\min, +)$ projection operator Π_M works as follows ([2], [8], [3]):

$$\Pi_M u = \min\{v | v \in \mathcal{V}, v \geq u\}, \forall u \in \mathcal{M}. \tag{14}$$

We can write the PBE in the $(\min,+)$ basis:

$$v = \Pi_M T v, v \in \mathcal{V}$$
$$\Phi \otimes r^* = \min\{\Phi \otimes r^* \in \mathcal{V} | \Phi \otimes r^* \geq T\Phi \otimes r^*\}. \quad (15)$$

Thus by making use of $(\min,+)$ LFAs and the projection operator Π_M we aim to find the minimum upper bound to J^*/Q^*.

V. Approximate Q Iteration

We now present an ADP scheme based on solving the PBE in $(\min,+)$ basis to compute $\tilde{Q}(\approx Q^*)$. Our ADP scheme successfully addresses the two shortcomings of the ADP scheme in conventional basis. First, we establish contraction of the projected Bellman operator in the L_∞ norm. This enables us to show that our recursion to compute $\tilde{Q}$ converges. We compute a greedy policy $\tilde{u}(s) = \max_a \tilde{Q}(s,a)$, and an approximate value function $\tilde{J}(s) = \max_a \tilde{Q}(s,a)$. Secondly, we also bound $||\tilde{Q} - Q^*||$ in the max norm which along with Lemma 4 gives a guarantee for $J_{\tilde{u}}$.

We are interested in solving the following PBE:

$$\Phi \otimes r^* = \Pi_M H \Phi \otimes r^*. \quad (16)$$

Since we want to approximate Q^*, Φ is an $nd \times k$ feature matrix, and $Q^*(s,a) \approx \tilde{Q}(s,a) = \phi^{(s-1)\times d + a} \otimes r^*$, where ϕ^i is the i^{th} row of Φ. The Approximate Q Iteration (AQI) algorithm is given by

$$\Phi \otimes r_{n+1} = \Pi_M H \Phi \otimes r_n. \quad (17)$$

The following results help us to establish the fact that the projected Bellman operator $\Pi_M H \colon \mathbf{R}_{\min}^{n\times d} \to \mathbf{R}_{\min}^{n\times d}$ is a contraction map in the L_∞ norm. In the discussion that follows, $\mathbf{1} \in \mathrm{R}^n \times d$ is a vector with all components equal to 1.

Lemma 7: For $Q_1, Q_2 \in \mathbf{R}_{\min}^{n\times d}$, such that $Q_1 \geq Q_2$, we have $\Pi_M Q_1 \geq \Pi_M Q_2$.

Proof: Follows from the definition of Π_M in (14). ■

Lemma 8: Let $Q \in \mathbf{R}_{\min}^{n\times d}$, $V_1 = \Pi_M Q$ be its projection onto $\mathcal{V}$. For $k \in \mathrm{R}$ the projection of $Q + k\mathbf{1}$ is $V_2 = \Pi_M Q + k\mathbf{1}$.

Proof: We know that since $V_1 \geq Q$, $V_1 + k\mathbf{1} \geq Q + k\mathbf{1}$, and from the definition of the Π_M in (14) $V_2 \leq V_1 + k\mathbf{1}$. Similarly $V_2 - k\mathbf{1} \geq Q$, so $V_1 \leq V_2 - k\mathbf{1}$. ■

Theorem 9: The projected Bellman operator $\Pi_M H$ is a contraction map in L_∞ norm with modulus α.

Proof: Let $Q_1, Q_2 \in \mathbf{R}_{\min}^{n\times d}$, and $\epsilon \stackrel{def}{=} ||Q_1 - Q_2||_\infty$, then by Lemma 7 we have

$$\begin{aligned}\Pi_M H Q_1 - \Pi_M H Q_2 &\leq \Pi_M H(Q_2 + \epsilon\mathbf{1}) - \Pi_M H Q_2 \\ &= \Pi_M(H Q_2 + \alpha\epsilon\mathbf{1}) - \Pi_M H Q_2 \\ &= \alpha\epsilon\mathbf{1}. \end{aligned} \quad (18)$$

Here the equality in second line is due to Lemma 3, and the final line follows from Lemma 8. Similarly $\Pi_M H Q_2 - \Pi_M H Q_1 \leq \alpha\epsilon\mathbf{1}$, so it follows that $||\Pi_M H Q_1 - \Pi_M H Q_2||_\infty \leq \alpha ||Q_1 - Q_2||_\infty$. ■

Corollary 3: AQI in (17) converges to a fixed point r^* such that $\Phi \otimes r^* = \Pi_M H \Phi \otimes r^*$.

VI. Variational Approximate Q Iteration

The projection operator Π_M used in (17) is exact. Let $v = \Pi_M u$, then for test vectors $w_i \in \mathbf{R}_{\min}^n, i = 1, \ldots, m$ it follows from the definition of Π_M that

$$w_i^\top v \geq w_i^\top u$$

where the dot product $x^\top y = \min_{i=1}^n (x(i) + y(i))$. Let W denote the $nd \times m$ test matrix whose columns are w_i. Now we shall define the approximate projection operator to be

$$\Pi_M^W u = \min\{v \in \mathcal{V} | W^\top v \geq W^\top u\}. \quad (19)$$

The superscript in Π_M^W denotes the test matrix W. The Variational Approximate Q Iteration (VAQI) is given by

$$\Phi \otimes r_{n+1} = \Pi_M^W H \Phi \otimes r_n. \quad (20)$$

Lemmas 7, 8, and Theorem 9 continue to hold if Π_M is replaced with Π_M^W. Thus by Corollary 3, we know that (20) converges to a unique fixed point r_W^* such that $\Phi \otimes r_W^* = \Pi_M^W H \Phi \otimes r_W^*$.

VII. Error Analysis

Theorem 10: Let $\tilde{r}$ be such that $\tilde{r} = \arg\min_r ||Q^* - \Phi \otimes r||_\infty$. Let r_W^* be the fixed point of the iterates in (20), then

$$\begin{aligned}||Q^* - \Phi \otimes r_W^*||_\infty \leq \frac{2}{1+\alpha}(&||Q^* - \Phi \otimes \tilde{r}||_\infty \\ &+ ||\Phi \otimes \tilde{r} - \Pi_M^W \Phi \otimes \tilde{r}||_\infty). \end{aligned} \quad (21)$$

Proof: Let $\epsilon = ||Q^* - \Phi \otimes \tilde{r}||_\infty$. By contraction property of H (Lemma 1) we know that

$$||HQ^* - H\Phi \otimes \tilde{r}||_\infty \leq \alpha\epsilon.$$

So we have $||\Phi \otimes \tilde{r} - H\Phi \otimes \tilde{r}||_\infty \leq (1+\alpha)\epsilon$. Then

$$\begin{aligned}||\Phi \otimes \tilde{r} - \Pi_M^W H \Phi \otimes \tilde{r}||_\infty &= ||\Phi \otimes \tilde{r} - \Pi_M^W \Phi \otimes \tilde{r}||_\infty \\ &\quad + ||\Pi_M^W \Phi \otimes \tilde{r} - \Pi_M^W H \Phi \otimes \tilde{r}||_\infty.\end{aligned}$$

$$\begin{aligned}\text{Now, } \Pi_M^W \Phi \otimes \tilde{r} - \Pi_M^W H \Phi \otimes \tilde{r} &\leq \Pi_M^W \Phi \otimes \tilde{r} \\ &\quad - \Pi_M^W(\Phi \otimes \tilde{r} - (1+\alpha)\epsilon) \\ &= (1+\alpha)\epsilon.\end{aligned}$$

Similarly $\Pi_M^W H \Phi \otimes \tilde{r} - \Pi_M^W \Phi \otimes \tilde{r} \leq (1+\alpha)\epsilon$, and hence

$$||\Phi \otimes \tilde{r} - \Pi_M^W H \Phi \otimes \tilde{r}||_\infty \leq (1+\alpha)\epsilon + \beta, \quad (22)$$

where $\beta = ||\Phi \otimes \tilde{r} - \Pi_M^W \Phi \otimes \tilde{r}||_\infty$. Now consider the iterative scheme in (20) with $r_0 = \tilde{r}$, and

$$\begin{aligned}||Q^* - \Phi \otimes r_W^*||_\infty &= ||Q^* - \Phi \otimes r_0 + \Phi \otimes r_0 \\ &\quad - \Phi \otimes r_1 + \ldots - \Phi \otimes r_W^*||_\infty \\ &\leq ||Q^* - \Phi \otimes r_0||_\infty + \\ &\quad ||\Phi \otimes r_0 - \Phi \otimes r_1||_\infty \\ &\quad + ||\Phi \otimes r_1 - \Phi \otimes r_2||_\infty + \ldots \\ &\leq \epsilon + (1+\alpha)\epsilon + \beta + \alpha((1+\alpha)\epsilon + \beta) \\ &\quad + \ldots \\ &= \epsilon(\frac{1+\alpha}{1-\alpha} + 1) + \frac{\beta}{1-\alpha} \\ &= \frac{2\epsilon + \beta}{1-\alpha}.\end{aligned}$$

■

The error bound for AQI is obtained by letting $\beta = 0$ in Theorem 10.

Corollary 4: Let r^* and r^*_W be the fixed points of iterations (17) and (20) respectively, and let $\tilde{Q} = \Phi \otimes r^*$, $\tilde{Q}_W = \Phi \otimes r^*_W$. The greedy policies $\tilde{u}(s) = \arg\max_a \tilde{Q}(s,a)$ and $\tilde{u}_W(s) = \arg\max_a \tilde{Q}_W(s,a)$ obey

$$||J_{\tilde{u}} - J^*|| \leq \frac{2(2\epsilon)}{(1-\alpha)^2}, ||J_{\tilde{u}_W} - J^*|| \leq \frac{2(2\epsilon+\beta)}{(1-\alpha)^2}, \quad (23)$$

where $\epsilon = \min_r ||Q^* - \Phi \otimes r||_\infty$.

Proof: Follows from Theorem 10, Lemma 4, and the observation that the term β is the error due to the usage of Π^W_M. Thus for solution to (16), $\beta = 0$. ■

VIII. Experiments

We test our ADP schemes on a randomly generated MDP with 100 states, i.e., $S = \{1, 2, \ldots, 100\}$, and action set $A = \{1, \ldots, 5\}$. The reward $g_a(s)$ is a random integer between 1 and 10, and we let the discount factor $\alpha = 0.9$. We now describe feature selection, where $\{\phi_j, j = 1, \ldots, k\}, \phi_j \in \mathbf{R}^{n\times d}_{\min}$ and $\{\phi^i, i = 1, \ldots, n \times d\}, \phi^i \in \mathbf{R}^k_{\min}$ denote the columns and rows respectively of the feature matrix Φ. The feature corresponding to a state-action pair (s,a) is given by $\phi^{(s-1)\times d+a}$. Let ϕ^x, ϕ^y be features corresponding to state-action pairs (s_x, a_x) and (s_y, a_y) respectively, then

$$< \phi^x, \phi^y >= \phi^x(1) \otimes \phi^y(1) \oplus \ldots \oplus \phi^x(k) \otimes \phi^y(k). \quad (24)$$

We desire the following in the feature matrix Φ.

1) Features ϕ^i should have unit norm, i.e., $||\phi^i|| =< \phi^i, \phi^i >= \mathbf{0}$, since $\mathbf{0}$ is the multiplicative identity in the $(\min, +)$ algebra.
2) For dissimilar state-action pairs, we prefer $< \phi^x, \phi^y >= +\infty$, since $+\infty$ is the additive identity in $(\min, +)$ algebra.

Keeping these in mind, we design the feature matrix Φ for the random MDP as in (25). For state-action pair (s,a) let $x = (s-1) \times d + a$, then the feature

$$\phi^x(i) = \begin{cases} 0 & : g_a(s) \in [g_{\min} + \frac{(i-1)L}{k}, g_{\min} + \frac{(i)L}{k}] \\ 1000 & : g_a(s) \notin [g_{\min} + \frac{(i-1)L}{k}, g_{\min} + \frac{(i)L}{k}], \end{cases} \quad \forall i = 1, \ldots, k. \quad (25)$$

We use 1000 in place of $+\infty$. Note that in (25) state-action pairs with similar rewards have similar features. The results are plotted in Fig. 1 and Fig. 2. Here J^* is the optimal value-function, $\tilde{J}_{EP}(s) = \max_a \tilde{Q}_{EP}(s,a)$, where $\tilde{Q}_{EP}$ is the value obtained via the iterative scheme in (17) and subscript EP denotes the fact that the projection employed in *exact* (Π_M). $\tilde{J}_W(s) = \max_a \tilde{Q}_W(s,a)$, where $\tilde{Q}_W$ is the value obtained via the iterative scheme in (20) and subscript W denotes the fact that the projection employed is Π^W_M. $u_{EP}(s) = \arg\max_a \tilde{Q}_{EP}(s,a)$ and $u_W(s) = \arg\max_a \tilde{Q}_W(s,a)$ are greedy policies and $J_{u_{EP}}$, J_{u_W} are their respective value functions. We also evaluated the performance of an *arbitrary* policy u_{arbt}, wherein we chose a fixed but arbitrary action for each state. We observed that $||J^* - J_{u_{arbt}}||_\infty = 40.5$.

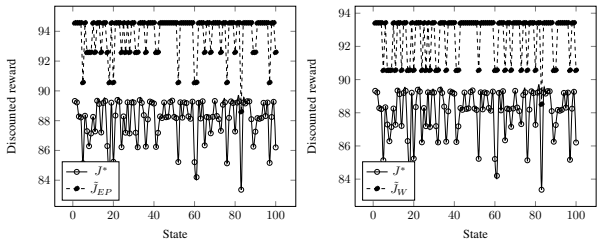

Fig. 1. $||J^* - \tilde{J}_{EP}||_\infty = 6.47, ||J^* - \tilde{J}_W||_\infty = 6.35$

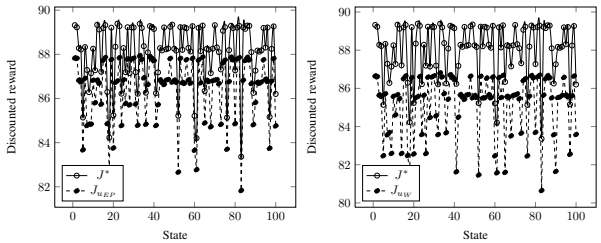

Fig. 2. $||J^* - J_{u_{EP}}||_\infty = 2.61, ||J^* - J_{u_W}||_\infty = 5.61$

IX. Conclusion

We developed two ADP schemes AQI and VAQI based on the $(\min, +)$ basis to compute an approximation to Q^*. The convergence of AQI and VAQI followed from the L_∞ norm contraction property of the projected Bellman operators $\Pi_M H$ and $\Pi^W_M H$. We also showed that the approximation error is bounded in the L_∞ norm (Theorem 10), and characterized the performance of the greedy policy (Corollary 4). The preliminary experiments on randomly generated MDPs show that AQI and VAQI converge in practice. We also observe that the policies computed by AQI and VAQI perform much better than arbitrary policies.

References

[1] D. P. de Farias and B. V. Roy, "The linear programming approach to approximate dynamic programming," *Operations Research*, vol. 51, no. 6, pp. 850–865, 2003.

[2] M. Akian, S. Gaubert, and A. Lakhoua, "The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis," *SIAM Journal on Control and Optimization*, vol. 47, no. 2, pp. 817–848, 2008.

[3] W. M. McEneaney, A. Deshpande, and S. Gaubert, "Curse-of-complexity attenuation in the curse-of-dimensionality-free method for hjb pdes," in *American Control Conference, 2008*. IEEE, 2008, pp. 4684–4690.

[4] W. M. McEneaney and L. J. Kluberg, "Convergence rate for a curse-of-dimensionality-free method for a class of hjb pdes," *SIAM Journal on Control and Optimization*, vol. 48, no. 5, pp. 3052–3079, 2009.

[5] S. Gaubert, W. McEneaney, and Z. Qu, "Curse of dimensionality reduction in max-plus based approximation methods: Theoretical estimates and improved pruning algorithms," in *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*. IEEE, 2011, pp. 1054–1061.

[6] M. Puterman, *Markov Decision Processes: Discrete Stochastic Programming*. New York: John Wiley, 1994.

[7] D. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Belmont,MA: Athena Scientific, 2007, vol. II.

[8] G. Cohen, S. Gaubert, and J.-P. Quadrat, "Kernels, images and projections in dioids," in *Proceedings of WODES96*, 1996, pp. 151–158.