Approximate Linear Programming for Logistic Markov Decision Processes*

Martin Mladenov†
TU Dortmund
martin.mladenov@cs.tu-dortmund.de

Craig Boutilier
Google Research
cboutilier@google.com

Dale Schuurmans‡
University of Alberta
daes@ualberta.ca

Ofer Meshi
Google Research
meshi@google.com

Gal Elidan
Google Research
elidan@google.com

Tyler Lu
Google Research
tylerlu@google.com

Abstract

Online and mobile interactions with users, in areas such as advertising and product or content recommendation, have been transformed by machine learning techniques. However, such methods have largely focused on *myopic prediction*, i.e., predicting immediate user response to system actions (e.g., ads or recommendations), without explicitly accounting for the long-term impact on user behavior, nor the potential need for planning action *sequences*. In this work, we propose the use of Markov decision processes (MDPs) to formulate this *long-term decision problem* and address two key questions that emerge in their application to user interaction.

The first focuses on model formulation, specifically, how best to construct MDP models of user interaction in a way that exploits the great successes of myopic prediction models. To this end, we propose a new model called *logistic MDPs*, an MDP formulation that allows the concise specification of transition dynamics. It does so by augmenting the natural factored form of *dynamic Bayesian networks* (DBNs) with *user response variables* that are captured by a logistic regression model (the latter being precisely the model used for myopic user interaction).

The second question we address is how best to solve large logistic MDPs of this type. A variety of methods have been proposed for solving MDPs that exploit the conditional independence reflected in the DBN representations, including *approximate linear programming (ALP)*. Despite their compact form, logistic MDPs do not admit the same conditional independence as DBNs, nor do they satisfy the linearity requirements for standard ALP. We propose a constraint generation approach to ALP for logistic MDPs that circumvents these problems by: (a) recovering compactness by conditioning on the logistic response variable; and (b) devising two procedures, one exact and one approximate, that linearize the search for violated constraints in the master LP. For the approximation procedure, we also derive error bounds on the quality of the induced policy. We demonstrate the effectiveness of our approach on advertising problems with up to several thousand sparse binarized features (up to 2^{54} states and 2^{39} actions).

*A shorter version of this paper appeared in *The Proceedings of the The 26th International Joint Conference on Artificial Intelligence (IJCAI-17)*, Melbourne, Aug. 2017.

†This work was performed while the author was a visiting intern at Google.

‡This work was performed while the author was a visiting scholar at Google.

1 Introduction

Online and mobile interactions with users have been transformed by the use of machine learning (ML) to predict user intent (e.g., search queries) preferences (e.g., product/content recommendations) and responses (e.g., advertising clicks and conversions). For example, in online advertising, models such as logistic regression, deep neural networks and random forests have been used very effectively to predict user responses to search, display and video ads in a variety of formats [34, 14, 22, 45, 30, 13, 29]. However, such methods have largely focused on *myopic* prediction, that is, predicting a user's "immediate" response to the system's action. As such, neither explicit modeling of the long-term impact of that action on the user's responses to future actions, nor the potential need for planning action *sequences*, is considered. For example, ad systems typically predict a user's click probability (or *click-through rate (CTR)*), "score" each eligible ad by combining its predicted CTR (pCTR) with its bid, and show the highest scoring ad.[1]

Implicitly, the process above defines a user interaction *policy*, which determines the entire sequence of interactions with a user. Empirical evidence has begun to show that failing to account for the long-term value of the induced policy can have detrimental effects on user satisfaction and system performance [26] (we discuss this evidence in detail below). Consequently, sequential models of user behavior and interaction optimization—such as *Markov decision processes (MDPs)* and corresponding online solution methods like *reinforcement learning (RL)*—have begun to receive increased attention [15, 28, 5, 4, 2, 40, 44]. However, such methods have yet to find widespread adoption for a variety of reasons (e.g., modeling difficulty, computational scalability, difficulty in defining objectives, exploration tradeoffs, etc.).

In this work, we consider model-based approaches to optimizing long-term user interaction, specifically, formulating and solving MDPs that characterize these interactions. While RL methods that learn policies or value functions directly from data are often valuable, we highlight several key benefits of the model-based philosophy. MDPs can be especially useful in settings where value-based RL might be unstable, and when fast approximate solutions are desired (e.g., for reward exploration, or for warm-starting model-free RL algorithms). We then address two key problems: learning MDP models from data, and solving very large MDPs.

To address the problem of model construction, we introduce a new model, *logistic MDPs*. We show how one can often take advantage of existing myopic models which predict some stochastic user response variable (e.g., CTR) to readily form a compact model of transition dynamics. The two key insights that drive this approach are: (a) the features defining the myopic model generally serve as a sufficient statistic of user history for long-term interaction (i.e., the features are self-predictive, thus form a Markov state space); and (b) the evolution of distinct features exhibits considerable conditional independence.

A consequence of this latter insight is that transition models can be expressed as a *dynamic Bayesian network (DBN)* [20, 9, 8] that directly incorporates the response model into the transition function. Such models are easily learned from the same data used to train myopic models. We focus on the case where the response variable in question is myopically modeled using logistic regression,

[1] This is a simplified view—a variety of other considerations often factor into such decisions, including other predicted quantities (e.g., conversion rate). But the point about myopic predictions and decisions remains.

and dub the resulting model a *logistic MDP*. We focus on linear logistic regression, but briefly discuss how our techniques can be extended to certain non-linear models (e.g., ReLU networks).

We then turn our attention to solution methods for logistic MDPs. Typically, MDPs represented using DBNs can be solved effectively—either exactly or approximately—by exploiting the structure in the transition model in both dynamic and linear programming methods [9, 25, 8]. One especially scalable approach is *approximate linear programming (ALP)* [23, 18], in which the value function is approximated using a linear combination of compact basis functions, and can be solved using techniques such as *constraint generation* [38]. Unfortunately, these algorithms cannot be applied directly to logistic MDPs because of the coupling of state and action features induced by the logistic function. We introduce two techniques to address this problem. The first is an exact method that treats the search for a violated constraint as a *sequence* of Boolean optimization problems (rather than a single optimization as in standard constraint generation). The second approach uses a (dynamic) piecewise-constant approximation of the logistic function to decompose the search for a violated constraint into a *set* of (independent, hence parallelizable) Boolean optimizations. We derive several distinct error bounds on the quality of this approximation.

Finally, we describe empirical results on a set of models derived from in-app display advertising data. We show that our methods can scale to problems of realistic size, which represent some of the largest discrete MDPs addressed in the literature.

A summary of the main contributions of this work is as follows:

- We provide detailed motivation for the use of non-myopic optimization and MDP/RL models of user interactions in Sections 3.1 and 3.2. We describe how one can leverage models (and the data used to derive them) commonly used to predict myopic user responses when developing MDP models of user behavioral dynamics.

- We introduce a new class of MDP models, *logistic MDPs*, that incorporate logistic regression response models into user dynamics. We also describe how factored DBN models can be used to specify such models concisely and naturally. See Section 3.3.

- We develop an approximate linear programming solution for solving factored logistic MDPs that uses a general constraint generation procedure to find maximally violated constraints in a relaxed version of the LP. The CG procedure is itself a non-linear Boolean optimization problem. See Section 4.1.

- In Section 4.2, we derive an exact search procedure for solving the CG optimization using linear Boolean optimization. The procedure allows for some problem decomposition and parallelization, and can also be approximated to arbitrary tolerance.

- We propose a simple decomposition of the CG optimization problem in Section 4.3 using a collection of linear Boolean optimization problems using a piecewise constant (PWC) approximation of the logistic function. The solution is "embarrassingly" parallelizable at the cost of some approximation in solution quality.

- In Section 4.4, we analyze the error associated with the approximation used in both forms of CG optimization. In the exact search procedure, error is induced only if the an approximation

tolerance is used. In the PWC approach, error is generally unavoidable. We derive bounds on the loss in expected value of the policies induced by our methods relative to the use of exact ALP. For the PWC approach, we also show how to choose good "decompositions" (i.e., PWC approximations) that help minimize these bounds.

- We examine the performance of our PWC approximation in Section 5 on a collection of problems of varying size. Even without parallelization, high quality solutions for very large logistic MDPs can be found quickly.
- In Section 6, we describe a number of important extensions of our methods, including dealing with non-linearities using certain types of deep neural networks and using cross-product features, using Lagrangian relaxation for CG optimization, and incorporating multiple user response predictions.

2 Background and Related Work

We begin by outlining necessary background and briefly discussing related work.

2.1 Logistic Regression for Online Advertising

Predicting user behavior is a critical component in any system that controls online and mobile interactions with users. The prediction of user responses to content or product recommendations, advertisements and other such interventions is especially common. Response predictions are routinely made for: click-through rates on ads, videos, product recommendations, search queries; conversion rates on ads; view-through rates on video content and video ads; install and usage rates on apps; and a host of other forms of behavior.

In many cases, the prediction is of a binary user response such as an ad click, app download, video watch, etc., with response probability predictions used to rank or score potential ads or recommendations (often combined with other factors, e.g., an advertiser's bid). A variety of models are used for this type of prediction ranging from linear logistic regression [34, 12, 22, 30, 13] to random forests and boosted trees [31, 45] to deep neural networks (DNNs) [29, 17, 16]

While many modeling approaches are used in online advertising (whether search, display, video, in-app, etc.) for CTR prediction, the "workhorse" remains logistic regression. Systems that scale to handle many trillions of training examples, and large numbers of predictive variables—which translate into billions of sparse, binarized features—are now routinely used in practice [12, 1]. We focus on logistic regression in this work (but briefly discuss the application of our methods to DNN models in Section 6.4).

For the logistic regression models that follow, we assume a finite set of *variables* $\mathcal{X} = \{X_1, \ldots, X_n\}$ that describe properties of the interaction. These may include both static and dynamic user characteristics (e.g., geolocation, gender, age, past purchase activity, device), action characteristics (e.g., advertiser, product type, creative characteristics), context (e.g., query, web site, app), and characteristics summarizing past interaction with the user (e.g., count of past ad impressions, clicks, etc.). Often variables with large discrete, ordered or continuous domains are either embedded into some more

compact representation or explicitly quantized. We assume each variable X_i has a discrete domain $Dom(X_i)$. For any $X \subseteq \mathcal{X}$, we use $Dom(X)$ to denote the cross-product $\prod\{Dom(X_i) : X_i \in X\}$. We later distinguish action variables from others, but for now we treat these uniformly. We generally use the notation x_i to refer to elements of $Dom(X_i)$ and $\mathbf{x}$ to elements of $Dom(X)$ (i.e., instantiations of the variables in X).

Models are often built using a "one-hot" encoding which transforms any element of $Dom(\mathcal{X})$ into a sparse binarized *feature vector* of length $\sum_i |Dom(X_i)|$. We will usually reserve the term *feature* to denote elements of this binarized representation to distinguish it from the underlying variables. However, we use the same notation $\mathbf{x}$ to denote both variable and feature vectors (context will usually be clear, and each uniquely determines the other).

Let φ be a binary *user response variable*, for instance, denoting the fact that a user clicks on an ad impression. A *(linear) logistic regression* predictor models the probability of a positive (resp., negative) user response given feature vector $\mathbf{x}$ as:

$$P(\varphi = \top|\mathbf{x}) = \sigma(\mathbf{u}^{\mathrm{T}}\mathbf{x}) = \frac{1}{1 + e^{-\mathbf{u}^{\mathrm{T}}\mathbf{x}}}, \tag{1}$$

where $\mathbf{u}$ is a learned weight vector. In ad settings, this is the typical form of a *pCTR model*.

2.2 Sequential User Modeling

While the use of sequential decision models to make recommendations or serve ads is relatively uncommon in practice, a significant amount of work has been applied to the sequential modeling of user online behavior, and some research into the use of MDPs and RL for optimizing interactions has been developed. We briefly describe a small selection of such work.

Within the advertising realm, Charikar et al. [15] assume the *web surfing behavior* of different user types can be modeled as a Markov process and construct policies to optimally allocate ad spend at different states (e.g., web sites) to maximize return (e.g., conversions), possibly as a function of user history. Archak et al. [4] tackle a related problem, offering a somewhat different model, but still assuming Markovian user behavior in a sponsored search setting. They propose a constrained MDP model that can be used to determine the optimal ad spend for a given user, assuming a given budget *for that user*. Interestingly, in [5] the same authors analyze user behavior empirically, demonstrating that users in a search context exhibit what might be called a "general-to-specific" search behavior that is approximately Markovian, and suggest that myopic click-through optimization will fail to optimize spend (hence motivating their MDP approach [4]). Boutilier and Lu [10] extend the MDP model in [4] to allow for the global optimization of an advertiser policy over a population of users (in different states of interaction) subject to an overall (rather than per-user) budget constraint. Li et al. [28] also consider Markov models in behavioral and contextual advertising.

More recent work has considered the application of RL methods to determine appropriate advertising and offer strategies to large populations of customers, learning optimal policies from data generated by online interactions with users [40, 44]. These methods are similar in their underlying user modeling assumptions, but attempt to learn *policies* directly from data rather than constructing a model of user dynamics.

One of the first MDP models for recommender systems was developed by Shani, Heckerman and Brafman [39], which uses the item sequence embodied in user purchases as the basis for an n-

gram transition model, and optimizes recommendations for long-term value. Little subsequent work has applied MDPs to recommendation tasks, though RL has been proposed (e.g., [42]). There has, however, been a increasing body of work that incorporates sequential modeling of user behavior into various forms of recommenders, ranging from (first- or higher-order) Markov models [33, 24] to HMMs [36] to recurrent networks and LSTMs [43, 48].

2.3 Factored MDPs and Approximate Linear Programming

In this section, we outline basic background on MDPs, briefly describe factored MDP representations that use dynamic Bayesian networks, and review approximate linear programming approaches, specifically how these can exploit DBNs.

2.3.1 Markov Decision Processes

A (finite) *Markov decision process (MDP)* [32] is given by: a finite state space S and action space A; a stochastic transition model P, with $P(s, a, s') = p^a_{ss'}$ denoting the probability of a transition from state s to s' when action a is taken; and a bounded, real-valued reward function $R(s, a) = r^a_s$ denoting the immediate (expected) reward of taking action a at state s. A *(stationary, deterministic) policy* π specifies an action $\pi(s) \in A$ to be taken in any state s.

We focus on discounted infinite-horizon problems with discount factor $0 \leq \gamma < 1$, where the aim is to find an optimal policy that maximizes the expected sum of discounted rewards. The (unique) *optimal value function* $V^* : S \to \mathbb{R}$ satisfies the Bellman equation Eq. 2, and induces the *optimal policy* Eq. 3, for all $s \in S$:

$$V^*(s) = \max_{a \in A} r^a_s + \gamma \sum_{s' \in S} p^a_{ss'} V^*(s'), \tag{2}$$

$$\pi^*(s) = \operatorname*{argmax}_{a \in A} r^a_s + \gamma \sum_{s' \in S} p^a_{ss'} V^*(s'). \tag{3}$$

Given an arbitrary (i.e., not necessarily optimal) value function (VF) $V : S \to \mathbb{R}$, we define the *action-value backup* for any state s and action a to be:

$$Q^V(s, a) = r^a_s + \gamma \sum_{s' \in S} p^a_{ss'} V(s'), \tag{4}$$

which denotes the expected value of taking action a at state s followed by acting according to some policy π with value V. When $V = V^\pi$, we sometimes write Q^π to denote this function.

We sometimes also use the vector representation $\mathbf{v}$ of (value or other) functions V defined over a state space S, where the ith element of $\mathbf{v}$ is $V(s_i)$ under a fixed indexing of S.

The optimal VF and policy can be computed using dynamic programming algorithms like value or policy iteration, or using natural linear programming (LP) formulations [32, 8]. In this work we focus on LP solutions. The primal LP formulation for a finite MDP is as follows:

$$\min_{v_s : s \in S} \quad \sum_{s \in S} \alpha_s v_s \tag{5}$$

$$\textit{s.t.} \quad v_s \geq Q^v(s, a), \forall s \in S, a \in A. \tag{6}$$

The variables v_s reflect the optimal VF at each state s, while $Q^v(\cdot, a)$ denotes the action-value backup, treating the vector v of variables as a VF. A state weighting function α is also required. Typically, α_s is interpreted as the probability that the process begins in state s; however, since optimal policies are uniformly optimal, any strictly positive vector α will give rise to the optimal VF. If α is a (strictly positive) initial state distribution, then the LP objective reflects the expected value of the optimal policy.[2]

Interestingly, the structure of this LP ensures that the simplex algorithm—which has worst-case exponential running time for general LPs—is strongly polynomial time for MDPs with fixed discount rates [49].

2.3.2 Factored MDPs

In domains involving user interaction, the underlying MDP is most naturally specified using finite sets of *state and action variables*, $\mathcal{X} = \{X_1, \ldots, X_n\}$ and $\mathcal{A} = \{A_1, \ldots, A_m\}$, respectively. Each state variable X_i has finite domain $Dom(X_i)$ (similarly for action variables). This implies that our state space $S = Dom(\mathcal{X})$ and action space $A = Dom(\mathcal{A})$ have exponential size. As such, some compact representation of the dynamics and reward function is needed even to specify the MDP in a practical manner. *Factored MDPs* [8] do just this.

Reward functions are often naturally structured, typically depending on only a small number of variables and/or factored into a set of additive functions reflecting the independent contributions of different variables to state quality or action cost. To this end, we assume reward is decomposable as the sum of r *reward factors* ρ_i:

$$R(\mathbf{x}, \mathbf{a}) = \sum_{i \leq r} \rho_i(\mathbf{x}[R_i], \mathbf{a}[R_i]), \tag{7}$$

where each $R_i \subset \mathcal{X} \cup \mathcal{A}$ is small set of state and/or action variables, and where the notation $\mathbf{x}[U]$ (resp., $\mathbf{a}[U]$) denotes the restriction of any variable instantiation to those variables in subset $U \subseteq \mathcal{X} \cup \mathcal{A}$.

Dynamics can be represented concisely in many domains using a *dynamic Bayesian network (DBN)* [20, 9, 8]. Two assumptions allow for the compact representation of the family of successor state distributions $\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ using a DBN. First, we assume that these distributions are products of marginals over the individual state variables:

$$\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}) = \prod_i \Pr_i(x_i^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}). \tag{8}$$

In other words, given any source state $\mathbf{x}^{(t)}$ at time t, an action $\mathbf{a}^{(t)}$ taken at time t has independent effects on each successor state variable.[3] Second, we assume that each variable X_i's local distribution is dependent on a small *parent set*, or small subset $Par_i \subset \mathcal{X} \cup \mathcal{A}$ of variables at the prior time. This means the local distributions themselves can be specified compactly since the number of parameters is bounded by some constant if the number of parents is itself bounded. As a result, the transition

[2] The role of the state weighting is less straightforward in approximate linear programming (see Section 2.3.3) [18].

[3] The strictness of this assumption can be relaxed easily: DBN representations admit correlations in post-action variables, and remain compact as long as the correlations exhibit suitable conditional independence (i.e., if the distribution can be represented using a Bayes net). For ease of exposition, we assume this "strict" independence; but our methods can be extended easily to handle such correlations.

function can be specified as:

$$\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}) = \prod_i \Pr_i(x_i^{(t+1)}|\mathbf{par}_i^{(t)}). \tag{9}$$

Here $\mathbf{par}_i^{(t)}$ is shorthand for $(\mathbf{x}^{(t)}[Par_i], \mathbf{a}^{(t)}[Par_i])$, the joint instantiation of $(\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ restricted to the state and action variables in Par_i.

DBN models of dynamics are often very compact as a consequence of this product decomposition: they are specified using, for each variable, a *conditional probability distribution (CPD)*, where the CPD for X_i specifies a distribution over $Dom(X_i)$ for each element of $Dom(Par_i)$. Furthermore, the structure in the transition distribution that gives rise to the compactness of DBNs can be exploited by various solution algorithms. For example, algorithms such as SPI, SPUDD and APRICODD [9, 25, 41, 8] use the DBN structure to (exactly or approximately) compute compact representations of optimal policies and value functions in the form of decision graphs or trees. These methods effectively impose the graph or tree structure on the value function and policy representations used by dynamic programming algorithms like value iteration and (modified) policy iteration. This same structure can be exploited in LP solutions to MDPs, as we discuss in the next section. Additional logical structure captured by relational or first-order representations can further compress the description of MDPs and lead to additional computational advantages [27, 47, 37].

2.3.3 Approximate Linear Programming

As noted above, MDPs can be solved using a direct LP formulation requiring $O(|S|)$ LP variables and $O(|S||A|)$ constraints. In addition, each constraint itself involves all $O(|S|)$ LP variables. When S and A are determined by instantiations of state and action variable sets $\mathcal{X}$ and $\mathcal{A}$, respectively, the exponential size of S and A make this direct formulation infeasible. To counter this, *approximate linear programming (ALP)* methods have been proposed to compress the representation and make the use of LP methods viable for computing approximate MDP solutions [23, 38, 6, 18].

In ALP, we assume that the value function is a linear combination of a set of *basis functions* or "features" $\mathcal{B} = \{\beta_1, \ldots, \beta_k\}$ defined over the state space. We further assume that each β_i is a function of some small number of state variables $B_i \subset \mathcal{X}$, whose joint domain $Dom(B_i)$ is relatively small. Thus, our VF is parameterized by a weight vector $\mathbf{w}$ and can be concisely specified as:

$$V(\mathbf{x}; \mathbf{w}) = \sum_{i \leq k} w_i \beta_i(\mathbf{x}[B_i]). \tag{10}$$

In matrix-vector form, the VF is expressed as:

$$\mathbf{v}_\mathbf{w} = \mathbf{B}\mathbf{w}, \tag{11}$$

where $\mathbf{B} = [\beta_1, \ldots, \beta_k]$ is the *basis matrix*, whose ith column is the vector representation of basis function β_i. Hence, $V(\mathbf{x}; \mathbf{w}) = \mathbf{B}_\mathbf{x}\mathbf{w}$, where $\mathbf{B}_\mathbf{x}$ denotes the row of $\mathbf{B}$ corresponding to state $\mathbf{x}$.

If we assume the state weighting function is factored in a similar way, the objective function in

the primal LP can be written compactly as:[4]

$$\min_{\mathbf{w}} \sum_{i \leq k} \sum_{\mathbf{b}_i \in Dom(B_i)} \alpha[\mathbf{b}_i] w_i \beta_i(\mathbf{b}_i). \tag{12}$$

We will assume the presence of a constant *bias factor* β_{bias} such that $\beta_{bias}(\mathbf{x}) = b$ for all $\mathbf{x}$. Notice that $B_{bias} = \emptyset$. We often take $b = 1$ w.l.o.g. so that (the vector representation of) $\beta_{bias} = \mathbf{1}$). This ensures the approximate LP (defined below) has a feasible solution.[5]

The DBN representation, combined with the structure of the linear VF, allows the LP constraints to be expressed concisely as well [23, 38]. Specifically, for any state $\mathbf{x}$, the l.h.s. of constraint 6 is simply $V(\mathbf{x}; \mathbf{w})$, which has the compact form given by Eq. 12. Furthermore, the action-value backup for a (factored) action $\mathbf{a}$ (i.e., the r.h.s. $Q^v(\mathbf{x}, \mathbf{a})$ of the constraint) can be written in a similarly compact form. Setting aside the reward function term, from Eq. 4 we see that $Q^v(\mathbf{x}, \mathbf{a})$ is the expectation of a linear combination of basis functions, hence is itself the weighted combination of *backprojected basis functions*, where the backprojection of β_i is a function that depends only on those variables in $Par_{B_i} = \cup_{X_j \in B_i} Par_j$ [9, 25, 23, 38]; in this way, there is dependence only on certain action variables as well. For each basis function β_i, we denote its backprojection by:

$$g_i(\mathbf{x}, \mathbf{a}) = g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) = \sum_{\mathbf{y} \in Dom(B_i)} \Pr(\mathbf{y} | \mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) \beta_i(\mathbf{y}). \tag{13}$$

When combined with the factored reward function, each constraint—requiring that the value of a state be at least as great as its action-value backup for each action, see Eq. 6—can be expressed concisely as:

$$0 \geq h(\mathbf{x}, \mathbf{a}; \mathbf{w}), \tag{14}$$

where $h(\mathbf{x}, \mathbf{a}; \mathbf{w}) = Q^V(\mathbf{x}, \mathbf{a}; \mathbf{w}) - V(\mathbf{x}; \mathbf{w})$, and has the form:

$$h(\mathbf{x}, \mathbf{a}; \mathbf{w}) = \sum_{i \leq k} w_i \Big(\gamma \cdot g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) - \beta_i(\mathbf{x}[B_i]) \Big) + \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j]). \tag{15}$$

Finally, we must deal with exponential number of LP constraints. This can be tackled in one of several ways. de Farias and Van Roy [18, 19] propose and analyze constraint sampling methods. Guestrin et al. [23] introduce a clever means by which the DBN structure is exploited to compile these constraints into a more concise representation that, under certain assumptions, ensures the entire set of constraints is expressible with a polynomial number of (somewhat more complicated) constraints. Schuurmans and Patrascu [38] propose the use of *constraint generation* to simplify the expression of the LP—this is the approach upon which we build in Sec 4.

ALP with constraint generation (ALPCG) repeatedly solves a relaxed LP (the *master*) using only those constraints corresponding to a subset $C \subseteq (\mathcal{X}, \mathcal{A})$ of state-action pairs. It attempts to exploit the fact that at the optimal solution only a small number of constraints will be active (bounded by the number of LP variables). At each iteration it checks whether any missing constraints are violated

[4] We note that any compact factorization of the weighting function will generally work with our methods—we assume this specific form for ease of exposition. To illustrate, suppose we have a uniform state weighting whose weights sum to κ. Then $\alpha[\mathbf{b}_i] = \kappa / |Dom(B_i)|$.

[5] For feasibility of the approximate LP, it is sufficient that $\mathbf{1}$ lies in the column space of $\mathbf{B}$—a bias factor is a simple way to ensure this.

at the solution of the current relaxed LP. If so, the *maximally violated constraint (MVC)* is added to C, thus tightening the relaxation, and the new LP is solved. If no violated constraints exist, then the current solution is optimal and ALPCG terminates.

Given the solution $\mathbf{w}$ to the master LP (see Eq. 12), the *subproblem* of detecting the MVC (if any) is a relatively straightforward *Boolean optimization problem (BOP)*:

$$\max_{\mathbf{x},\mathbf{a}} \quad \sum_{i \leq k} w_i \Big(g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}]) - \beta_i(\mathbf{x}[B_i]) \Big) + \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j]). \tag{16}$$

If the solution $(\mathbf{x}^*, \mathbf{a}^*)$ to Eq. 16 has positive objective value—i.e., if $h(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w}) > 0$—then the constraint (corresponding to) $(\mathbf{x}^*, \mathbf{a}^*)$ is the MVC and is added to the master.

The subproblem Eq. 16 contains Boolean (indicator) variables expressing the assignment of both state and action variables, as well as the selection of terms in each of the sub-functions g_i, β_i and ρ_j, with constraints enforcing logical consistency of the variable and function assignments. The subproblem can be solved using standard Boolean or mixed integer programming (MIP) solvers. Schuurmans and Patrascu [38] show that this approach can be quite effective for MDPs with compact DBN representations. The method also exhibits good anytime performance—e.g., by terminating ALPCG early, one can attain an approximate solution to the MDP. Finally, variations of this scheme can be applied (e.g., adding *some* violated constraint, not necessarily maximal; or adding multiple violated constraints at each iteration).

We make several remarks on the ALP method in general. First, recall that the full LP formulation Eq. 5 produces the optimal VF regardless of the state weighting function α (as long as it is strictly positive). By contrast, in ALP different weightings α can produce rather different VFs. Hence the choice of α may be important (as we discuss below).

Second, ALP (with or without constraint generation) will generally produce an approximation $V_{\mathbf{B}} = \mathbf{B}\mathbf{w}^*$ to the optimal VF within the span of the basis matrix $\mathbf{B}$. Note, however, that while $V_{\mathbf{B}}$ approximates the optimal value function, it is generally not a VF corresponding to any policy, hence the distance $V_{\mathbf{B}}$ between V^* (the optimal VF) does not accurately reflect the approximation error. de Farias and Van Roy [18] perform the following analysis of this error. Letting $\widetilde{\pi}$ denote the induced (greedy) policy according to $V_{\mathbf{B}}$, consider $\widetilde{V} = V_{\widetilde{\pi}}$, the VF corresponding to the greedy policy $\widetilde{\pi}$—it is $\widetilde{V}$ that expresses the true value of using the ALP solution. The error induced by ALP is then

$$\|V^* - \widetilde{V}\|_{1,\alpha} = \sum_{\mathbf{x}} \alpha(\mathbf{x}) \big(V^*(\mathbf{x}) - \widetilde{V}(\mathbf{x}) \big),$$

in other words, the expected value loss w.r.t. the initial state distribution. de Farias and Van Roy show that this error is bounded:

$$\|V^* - \widetilde{V}\|_{1,\alpha} \leq \frac{1}{1-\gamma} \|V^* - V_{\mathbf{B}}\|_{1,\mu},$$

where μ is the (normalized) total discounted occupancy frequency (over states) induced by $\widetilde{\pi}$ under α.

While informative, the practical value of this bound is limited since it depends on knowledge of μ, which itself depends on our solution $\widetilde{V}$ (via $\widetilde{\pi}$). However, it does suggest that, if one can roughly estimate states or regions of state space in which the (approximate) policy is more likely to spend

time, a state weighting that reflects this could be useful in ALP.[6] To overcome this limitation, de Farias and Van Roy [18] also propose an analysis that requires the design of a (domain-specific) Lyapunov function; however, such functions are generally difficult to construct.

3 Logistic MDPs

In this section, we turn our attention to the use of MDPs for optimizing long-term engagement with users. We first provide additional motivation for the use of non-myopic models for user engagement in Section 3.1. In Section 3.2, we describe in some detail how one can leverage existing myopic user models to build sequential models of user response behavior. Finally, we formally described factored *logistic MDPs* in Section 3.3. Users not interested in the precise details of how such models are constructed may skip directly to Section 3.3.

3.1 Non-myopic Models of User Interactions

As discussed in Section 2.1, practical systems for ad serving and content/product recommendations are almost exclusively myopic in the sense that they generally predict a user's response to some immediate system action (e.g., the ad served or item recommended) without considering what impact this action has on the ability to make good future decisions.

There are in fact two senses in which a system may be myopic: (a) not accounting for its own future actions; (b) predicting only immediate or near-term rather than long-term user response. Our focus is on the former, but it is worth noting that the majority of practical systems are myopic in the latter sense as well. We believe that doing (b) well also requires dealing with (a) in most instances. For example, if app recommendations are made that incorporate some prediction of the in-app purchases a user will make over some extended period, subsequent recommendations and other system actions will likely influence this behavior; and the mere act of making a recommendation may influence a user's response to these subsequent actions.

Intuitively, one would expect that system decisions made at any point in time have the potential to influence user responses to *future* actions as well. For example, exposure to a display ad may increase a user's propensity to click on the same (or related) ads in the future by increasing awareness through exposure, or may decrease this propensity by creating a "blindness" effect [26]. In product recommendation, exposure (or purchase) of one product may increase the odds of a user purchasing a subsequent complementary product, while simultaneously decreasing the odds of purchasing a (partial) substitute or clicking on a partially substitutable ad [7]. Advertising budgets similarly require making tradeoffs over time in one's interaction with a user [21, 4, 10].

Apart from such intuitions, compelling evidence has recently been published showing that overall ad load and ad quality can have an impact on a user's general propensity to click on online ads: Hohnhold et al. [26] demonstrate, using long-running cookie experiments, that users exposed to greater ad load or ads of lower quality eventually learn to click on ads at a lower rate. Finally, as we discuss

[6] de Farias and Van Roy [18] also suggest an iterative approach in which an ALP solution is determined, its discounted occupancy frequency is computed (itself a challenging problem in large factored models), and the ALP solution is updated using this new weighting.

further below, the nature of the models used to make myopic predictions of user response is such that any system action will usually change the user feature vector used to make the *next* prediction. As such, the current action often influences the predicted effectiveness of the next (and all subsequent) actions.

The aim in most ad and recommendation systems is not simply to maximize the odds of a single "successful" interaction with a user, but to do this over some "lifetime" of engagement with the user. Thus, it is most natural to model this interaction using an MDP, and construct an optimal policy by solving the resulting MDP directly, or using reinforcement learning (RL) to discover an optimal policy from direct interaction with users. Both model-based MDP methods [5, 4, 2, 10] and model-free RL [40, 44] have been proposed in the literature.[7]

In this work, we focus on model-based methods, though the way in which we exploit myopic models is relevant for model-free RL as well (see Section 3.2). One reason for our focus on model-based RL is the ability to quickly solve an MDP using the same dynamics, but different reward functions. This can be critical in allowing decision makers to explore tradeoffs between different objectives. For instance, in *app recommendations*, one may want to make recommendations that have high install rates, high levels of user app engagement (post-install), significant potential for revenue generation for the developer, and otherwise perform well w.r.t. many other factors. Quantifying these tradeoffs precisely in a reward function can be difficult *a priori*. However, the impact of different immediate reward functions (each making a different immediate tradeoff) on cumulative versions of these long-term objectives can be tested by solving the corresponding MDP and estimating the expected value of each metric given the optimal policy. This can be used to determine the most suitable reward function. Even with an approximate model, this process can be used to provide confidence in the reward function adopted in a model-free, online RL method. Model-based methods, as long as scalable solution algorithms exist, can also allow for greater data- and computational efficiency.

3.2 Exploiting Myopic Prediction Models

In order to solve an MDP reflecting long-term engagement with users, we require a transition model that captures the dynamics of user behavior, reflecting expected user actions under various conditions, the evolution of user state or history, and how users respond to system actions (e.g., specific recommendations or ads). In principle, with the proper encoding of state space, learning a transition model is straightforward with sufficient data (and sufficiently "exploratory" data).

A key challenge lies in specifying the *state space*, i.e., defining an appropriate set of features that are observable from logged interactions with users, that summarize user history adequately, and are both predictive of future user behavior—in particular, responses to ads or recommendations—and render the implied dynamics Markovian. Even with tremendous advances in deep learning and sequence modeling, we are nowhere near the point where features can be automatically discovered an extracted from massive, unfiltered, raw logged user interaction data to build such transition models.

Fortunately, we don't generally need to start from scratch. We can instead leverage the tremendous insight derived from years of experimentation, engineering effort, and experience in building

[7] Another reason to consider sequential modeling of user interactions is to explicitly model the learning of user characteristics and propensities, and to guide the exploration-exploitation tradeoff using measures like *value of information*. We do not address this here.

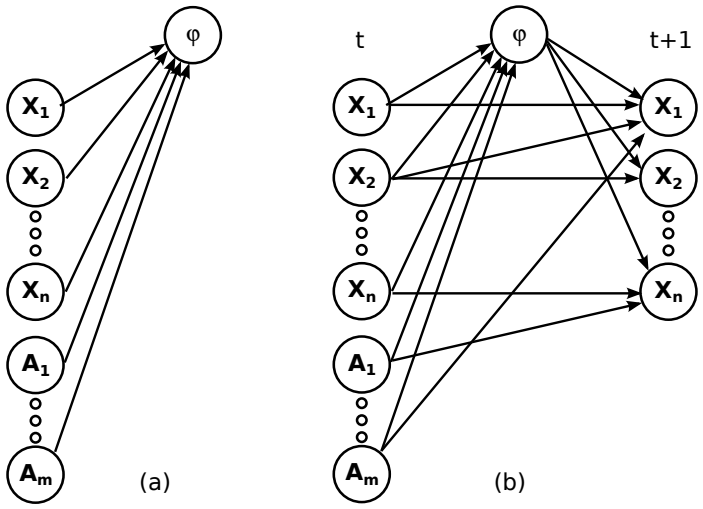


Fig. 1: An illustration of (a) a (myopic) logistic regression response model, and (b) the dependence structure in the DBN representation of a logistic MDP that extends the myopic response model.

myopic prediction models for user responses (see Section 2.1). Focusing on logistic regression, the "workhorse" of online advertising, a typical model is illustrated in Fig. 1(a), where the response variable φ (e.g., representing a positive ad click) depends on the variables X_i representing the user state, user history and any relevant context (e.g., web site visited, query issued, app being used, etc.), and variables A_i reflecting various properties of the system action (e.g., ad being shown, app or video being recommended).

As argued above, we often want to predict (and optimize) what happens to the response variable over time, for instance, to optimize cumulative CTR over a user session, week, or lifetime. Doing so requires not just predicting how actions influence the immediate response, but how they *change the variables we expect to see at the next point in time*, since it is these variables that are predictive of the user's *next* response, and so on. What we desire is a *model of the dynamics of these predictive variables* so that we can optimize the long-term (e.g., cumulative) expected value of the response variable (and perhaps other variables as well) by the suitable choice of actions. If the model is *Markovian*, then we can immediately formulate the MDP required to compute such an optimal policy.

Fig. 1(b) illustrates how, ideally, we would like to extend the myopic response model in the case of logistic regression. We explain the structure of this model formally in the next section, where we show that this is a variant of *dynamic Bayesian network (DBN)*. We view the variables X_i as *state variables* and the A_i as *action variables*. The model shows how the state variables at time $t + 1$ depend (probabilistically) on the state at time t and the action taken at time t.

Of course, one must ask whether an extension of this type is sensible. If fact, this provides a natural and practical way to construct the required MDP, which exploits several key characteristics of typical myopic response models:

- First, it is usually safe to assume that the existing variables or features form a suitable summary of user history and action properties for predicting the response variable φ. In other words, these form a *sufficient statistic* of all user (and action) data w.r.t. φ. This is due to the considerable

energy and experimentation that goes into defining a domain and understanding the relevant features required for accurate myopic response prediction.

- Predicting φ itself is not enough—the features themselves must be (roughly) "self-predictive;" that is, for each state variable X_i, it must be the case that the information contained in state and action variables at time t, along with the observed response ϕ at time $t+1$, is sufficient to predict the distribution over X_i at time $t+1$. In other words, the variables in the response model (together with ϕ) form a *sufficient statistic* for each X_i. It turns our that in a great many practical models, this is in fact the case. Many of the state variables capture static user properties, others capture various summary statistics of past user behavior and responses, while others still reflect the natural evolution of user actions. In each case, the variables in question depend in a natural way on the preceding state and action variables (and in some cases on ϕ itself). Some examples:
  - Static user variables are those which remain fixed over the duration of a session being modeled (e.g., demographic properties, such as age, gender, geo-region, device type). Local transition models are trivial (identities).[8]
  - Counting variables are often used to summarize relevant aspects of user history (e.g., number of impressions of ads from a specific campaign). These evolve deterministically in a way that depends on its prior value and some other variables (e.g., the action variables identifying the campaign).[9]
  - Other context features (e.g., query, app, etc.) require development of more refined models, but tend not to be influenced by actions, so are straightforward to learn from data (see next bullet).
- Given the above, one can readily develop models reflecting the evolution of these variables from historical interaction data (assuming sufficient "explicit" exploration or other induced randomness)—indeed, the same logged data used to construct myopic response models can be used directly for this purpose. Finally, we note that the prediction of (the distribution of) state variables X_i at time $t+1$ generally depends on only a small number of preceding state and action features and (sometimes) the response variable. This is an important property of the resulting DBN models that we exploit below.
- If some variables are non-Markovian w.r.t. the current set, they usually have a natural semantics that allows one to add additional variables that help predict the problematic variables, again rendering the dynamics Markovian.

We have focused on how to exploit myopic models to form the MDP required to explicitly compute an optimal policy. However, the observations above have significance for model-free RL as well. Specifically, if the properties above hold—i.e., if the variables from the myopic response model form

[8] Typical MDPs in recommendation and advertising domains have many static variables. Technically, we could treat each set of static features as inducing a distinct MDP (much like "context" in contextual bandits), since the subsets of states corresponding to distinct instantiations are non-communicating under any policy. However, we treat the process as a single MDP since there is considerable transition and value function generalization that occurs across these distinct sub-MDPs.

[9] Some natural "counting" variables are only approximately Markovian due to horizon effects (e.g., the number of ads from a specific campaign viewed in the last 24 hours).

a sufficient statistic for response prediction and are themselves "self-predictive"—then these variables form a suitable state space for RL.

3.3 Factored Logistic MDPs: Formal Model

We formalize the considerations above by defining *logistic MDPs*, in which a standard DBN representation is augmented with a binary response variable.[10] We focus on models with a single response variable, but discuss extensions to multiple such variables in Section 6.3.

A logistic MDP incorporates the same elements and relies on the same assumptions as in the factored MDP representation described in Section 2.3.2. Specifically, we assume state and action variables $\mathcal{X}$ and $\mathcal{A}$, respectively, with finite domains and augment these with a binary response variable φ.

To specify system dynamics, we assume the state-action pair $(\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ at time t determines $\Pr(\varphi^{(t+1)})$, and the triple $(\mathbf{x}^{(t)}, \mathbf{a}^{(t)}, \varphi^{(t+1)})$ dictates the next state distribution $\Pr(\mathbf{x}^{(t+1)})$. Hence the dynamics factor as:

$$\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}) = \sum_{\varphi \in \{\top, \bot\}} \Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}, \varphi^{(t+1)}) \Pr(\varphi^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}). \tag{17}$$

As in Section 2.3.2, we assume that $\Pr(\mathbf{x}^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)}, \varphi^{(t+1)})$ is a product distribution over the time $t+1$ state variables, with $\Pr(X_j^{(t+1)}|Par_j^{(t)}, \varphi^{(t+1)})$ depending on a small set of state and action parents. Any specific X_j may be independent of φ, but if not, we sometimes refer to φ as a parent of X_j. Furthermore, we assume that $\Pr(\varphi^{(t+1)}|\mathbf{x}^{(t)}, \mathbf{a}^{(t)})$ is given by a linear logistic regression model (see Eq. 1). This model will generally be that used in the myopic response prediction task, as outlined above. This ensures the specification of the dynamics remains compact. See Fig. 1(b) for an illustration of this dependence structure.

The reward function is also assumed to be factored as described in Section 2.3.2. The only modification is that reward factors are permitted to include the response variable φ (not just state or action variables):

$$R(\mathbf{x}, \mathbf{a}) = \sum_{i \leq r} \rho_i(\mathbf{x}[R_i], \mathbf{a}[R_i], \varphi), \tag{18}$$

where (as above) each $R_i \subset \mathcal{X} \cup \mathcal{A}$ is small set of state and/or action variables.

We make a few important observations about this model:

- As above, we assume that the dynamics induce no correlations among time $t+1$ state variables (i.e., these are independent given the prior state and action and the time $t+1$ response variable). Again, this is largely for ease of exposition: if within-slice dependencies are confined to parent sets that induce small tree width, our methods below apply with only small changes to notation.
- Notice from Fig. 1(b) that φ itself *correlates all of the $t+1$ variables to which it is connected, destroying the structure that is normally exploited in DBN planning algorithms*. However, we

[10] Technically, the resulting model remains a DBN as we discuss below; but since the response variable destroys much of the structure typically assumed in a DBN (as we outline below), we treat it very differently from other variables. We describe this as an "augmentation" of a DBN to emphasize this fact.

show in the next section that the particular structure of the logistic model can be exploited to develop an effective algorithmic approach that incurs only moderate overhead relative to standard DBN planning.

- The transition model is very compact: each local distribution X_i requires specification of $O(|Dom(Par_i)||Dom(X_i)|$ parameters (where we count φ as a parent if X_i depends on it), while the distribution over φ requires $O(\sum_i Dom(X_i))$ parameters.
- We limit our attention to models with one response variable. Algorithmic extensions to handle multiple response variables are discussed in Section 6.3.
- The limitation to discrete domains and linear logistic regression is not that restrictive. In practice, variables with continuous, ordered or large cardinality domains are often quantized or otherwise compressed (see Section 2.1).
- If non-linear interactions are present in the myopic prediction model, often several variables will be "crossed" to capture such interactions. For example, we can use a cross-product feature over $Dom(\{X_1, X_2\})$ in the logistic model, capturing the non-linear interactions between two variables X_1, X_2, while retaining the generalized linear form. For ease of exposition, we present our methods assuming no such crosses exist. Our techniques easily handle such crosses, as we outline in Section 6.1.

To conclude, we see that, by leveraging the existence of a myopic prediction model for the response variable, the specification of logistic MDP is not significantly more difficult than specifying a more standard factored MDP using a DBN.

4 Approximate Linear Programming for Logistic MDPs

We now develop an extension of ALPCG designed to handle logistic MDPs. We first outline the basic structure of our algorithm, which is identical to ALPCG, but uses a logistic MDP representation (i.e., a DBN augmented with a response variable). To implement the algorithm, we need to overcome two key complications introduced by the logistic extension when solving the constraint generation subproblem: the loss of strong conditional independence among state variables due to the coupling induced by the logistic function; and the non-linearity of the response model (in state-action space). We introduce a systematic approach for addressing these issues, and instantiate it in two distinct ways.

First, we develop a search procedure for the maximally violated constraint (MVC) that solves, instead of a single BOP, a *sequence* of BOPs. This procedure is guaranteed to find the MVC, and thus generalizes ALP by finding an exact solution to the ALP problem for a richer class of models.

Second, we develop an approximate version of ALP in which a discretization of probability space allows the search for the MVC to be formulated as a *set* of BOPs. This approximates the ALP solution rather than solving it exactly, but has an advantage over the exact procedure in that the required BOPs can be solved in parallel rather than in sequence. We derive error bounds that provide guidance on how to select an appropriate discretization.

4.1 ALP with Constraint Generation

With a logistic MDP, using the augmented DBN representation, we can generalize the ALP formulation in a straightforward way. We first observe that the action-value backup is complicated by the presence of the logistic response. Specifically, we notice the backprojection of any basis function (see Eq. 13) now depends not only on its immediate parents, but on *all* preceding state and action variables due to their influence on φ.

We can break this dependence by conditioning on φ. In particular, assuming a linear value function V given by Eq. 10, the action-value backup has the form:

$$Q^V(\mathbf{x}, \mathbf{a}; \mathbf{w}) = \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) \Big[\sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j], \varphi) + \gamma \sum_{i \leq k} w_i g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}], \varphi) \Big]. \quad (19)$$

Here the backprojection g_i for basis function β_i is defined as in Eq. 13, but where we *fix the value* of φ and then take an expectation w.r.t. realizations ($\top$ or $\bot$) of φ. The same conditioning provides us with the expected immediate reward component of the action-value backup ρ_j.

We re-express $h(\mathbf{x}, \mathbf{a}; \mathbf{w}) = Q^V(\mathbf{x}, \mathbf{a}; \mathbf{w}) - V(\mathbf{x}; \mathbf{w})$, as specified in Eq. 15, to reflect the dependence on φ:

$$h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}) = \sum_{j \leq r} \rho_j(\mathbf{x}[R_j], \mathbf{a}[R_j], \varphi) + \sum_{i \leq k} w_i \Big(\gamma g_i(\mathbf{x}[Par_{B_i}], \mathbf{a}[Par_{B_i}], \varphi) - \beta_i(\mathbf{x}[B_i]) \Big). \quad (20)$$

Finally, we obtain the ALP formulation for logistic MDPs:

$$\min_{\mathbf{w}} \quad \sum_{i \leq k} \sum_{\mathbf{b}_i \in Dom(B_i)} \alpha[\mathbf{b}_i] w_i \beta_i(\mathbf{b}_i) \quad (21)$$

$$\textit{s.t.} \quad 0 \geq \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}) \qquad \forall \mathbf{x} \in S, \mathbf{a} \in A. \quad (22)$$

We define

$$C(\mathbf{x}, \mathbf{a}; \mathbf{w}) = \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) \cdot h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}) \quad (23)$$

and on occasion write constraint 22 as $0 \geq C(\mathbf{x}, \mathbf{a}; \mathbf{w})$. This LP has the same desirable features as the ALP formulation for standard DBNs: a compact objective and compact constraints. As with standard ALPCG, we handle the exponential number of constraints using constraint generation. We turn our attention to subproblem optimization next.

4.2 Search for Maximally Violated Constraints

We now turn our attention to the constraint generation problem. Assuming that we have solved the ALP formulation outlined in Eqs. 21–22, using only a *subset* of the constraints 22, to obtain a solution $\mathbf{w}$ to the relaxed problem. We now wish to find a maximally violated constraint at the current solution. The corresponding subproblem optimization is:

$$\max_{\mathbf{x}, \mathbf{a}} \quad \sum_{\varphi \in \{\top, \bot\}} \Pr(\varphi | \mathbf{x}, \mathbf{a}) \cdot h(\mathbf{x}, \mathbf{a}, \varphi; \mathbf{w}). \quad (24)$$

Before describing our procedure, we introduce some notation. Let $f(\mathbf{x}, \mathbf{a}; \mathbf{u})$ denote the linear function of state-action features that is passed through the sigmoid in the logistic regression response model Eq. 1; that is, $\Pr(\varphi = \top|\mathbf{x}, \mathbf{a}) = \sigma(f(\mathbf{x}, \mathbf{a}; \mathbf{u}))$ where $\mathbf{u}$ are the fixed parameters for the predictor. We refer to these inputs as *logits* (i.e., log-odds of the induced probability), and the space of possible inputs to the f as *logit space*. Since $\mathbf{u}$ is fixed throughout the optimization procedure, we suppress it in our notation and write $f(\mathbf{x}, \mathbf{a})$. For any interval $[f_\ell, f_u]$ of logit space (i.e., input space to the sigmoid), we (ab)use the same notation to denote the set of states:

$$[f_\ell, f_u] = \{(\mathbf{x}, \mathbf{a}) : f_\ell \leq f(\mathbf{x}, \mathbf{a}) \leq f_u\}.$$

Notice that this interval also places lower and upper bounds, $\sigma_\ell = \sigma(f_\ell)$ and $\sigma_u = \sigma(f_u)$, respectively, on $\Pr(\varphi = \top|\mathbf{x}, \mathbf{a})$.

We now define a search procedure to find an MVC, i.e., to solve the subproblem optimization in Eq. 24. We denote the ALP procedure using this search by *ALP-SEARCH*. Our search for an MVC will proceed by searching in intervals of logit space. Specifically, given interval $[f_\ell, f_u]$, we describe a recursive method to find an MVC among those (associated with) states $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$ that searches in subintervals until a constraint is found that is provably an MVC. (We explicitly consider approximation of this MVC as well.) We can initiate the procedure over the entire space (leaving $(\mathbf{x}, \mathbf{a})$ unconstrained except for trivially computable maximum and minimum bounds on the logits w.r.t. the logistic weight vector $\mathbf{u}$), or by starting with a small number of subintervals and computing an MVC in each. In the latter case, the overall MVC is that with the maximum degree of violation from among the subinterval MVCs.

The search for an MVC in interval $[f_\ell, f_u]$ is complicated by the non-linear nature of the constraint function $C(\mathbf{x}, \mathbf{a}; \mathbf{w})$ in Eq. 23. Notice, however, that if we replace $\Pr(\varphi|\mathbf{x}, \mathbf{a})$ (hence, its converse) by some constant σ_* for all $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$, then the optimization to find an MVC in the interval becomes linear:

$$\max_{\mathbf{x},\mathbf{a}} \quad \sigma_* h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \sigma_*) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \tag{25}$$

$$\textit{s.t.} \quad f_\ell \leq f(\mathbf{x}, \mathbf{a}) \leq f_u. \tag{26}$$

Since $\sigma(z)$ is monotonically increasing in z, $\sigma_u = \sigma(f_u)$ provides an upper bound on $\Pr(\varphi|\mathbf{x}, \mathbf{a})$ for any $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$ (see Fig. 2 for an illustration).

The search for the MVC within $[f_\ell, f_u]$, as embodied in Eqs. 25–26, can be decomposed into a search over two disjoint classes of state-action pairs:

$$H^+_{\mathbf{w}} = \{(\mathbf{x}, \mathbf{a}) : h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) \geq h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w})\}, \tag{27}$$

$$H^-_{\mathbf{w}} = \{(\mathbf{x}, \mathbf{a}) : h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) < h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w})\}. \tag{28}$$

For any $(\mathbf{x}, \mathbf{a}) \in H^+_{\mathbf{w}}$, the expression in the objective Eq. 25 is monotonically non-decreasing in σ_*. (Similarly, for any $(\mathbf{x}, \mathbf{a}) \in H^-_{\mathbf{w}}$, it is monotonically non-increasing.) Hence, by restricting the search to $H^+_{\mathbf{w}}$, and setting the constant probability term to σ_u, we obtain the BOP:

$$\max_{\mathbf{x},\mathbf{a}} \quad \sigma_u \cdot h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \sigma_u) \cdot h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \tag{29}$$

$$\textit{s.t.} \quad (\mathbf{x}, \mathbf{a}) \in H^+_{\mathbf{w}} \cap [f_\ell, f_u], \tag{30}$$

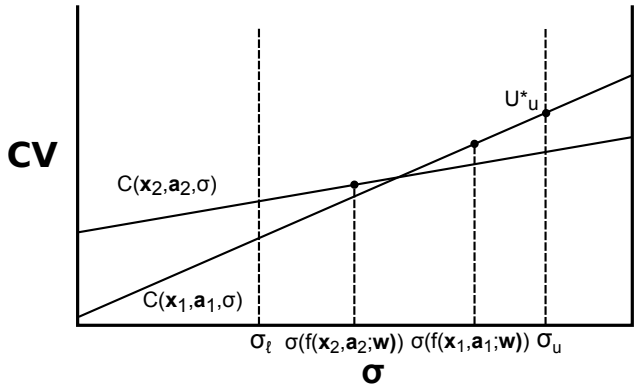


Fig. 2: An illustration of search for an MVC. The graph shows the degree of constraint violation C for two state-action pairs as a function of the constant σ approximating their corresponding response probabilities. Both $(\mathbf{x}_1, \mathbf{a}_1), (\mathbf{x}_2, \mathbf{a}_2) \in H^+_{\mathbf{w}}$ given their increasing nature. Both pairs lie in the interval $[f_\ell, f_u]$ of logit space, which is shown in the figure in probability space by $[\sigma_\ell, \sigma_u]$. When σ is set to σ_u, $(\mathbf{x}_1, \mathbf{a}_1)$ induces the upper bound U^*_u on constraint violation among *all* pairs in $H^+_{\mathbf{w}} \cap [f_\ell, f_u]$. Notice that $(\mathbf{x}_1, \mathbf{a}_1)$'s true response probability $\sigma(f(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w}))$ is less than σ_u, so the bound is not tight; but the gap with its true violation $C(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w})$ is small, so the search within this interval will terminate. Notice also that $(\mathbf{x}_1, \mathbf{a}_1)$ is indeed the MVC; but had $\sigma(f(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w}))$ been to the left of the intersection point, then $(\mathbf{x}_2, \mathbf{a}_2)$ would have been the MVC despite the upper bound suggesting otherwise. In that case, the gap $U^*_u - C(\mathbf{x}_1, \mathbf{a}_1; \mathbf{w})$ would be larger, inducing a split of the interval.

where constraint 30 represents the union of the corresponding linear constraints above. A solution of the BOP 29 provides a maximizer $(\mathbf{x}^*_u, \mathbf{a}^*_u)$ with objective value U^*_u that has the following properties (see Fig. 2):

- U^*_u is an upper bound on the maximal constraint violation in $H^+_{\mathbf{w}} \cap [f_\ell, f_u]$. This is so due to the monotonicity of the objective w.r.t. σ for any $(\mathbf{x}, \mathbf{a}) \in H^+_{\mathbf{w}}$ and the fact that σ_u is an upper bound on σ for any $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$.
- The true constraint violation $C(\mathbf{x}^*_u, \mathbf{a}^*_u; \mathbf{w})$ of the solution $(\mathbf{x}^*_u, \mathbf{a}^*_u)$ is a lower bound on the maximum constraint violation in $H^+_{\mathbf{w}} \cap [f_\ell, f_u]$.

We note that the corresponding optimization using $H^-_{\mathbf{w}}$ and the lower bound probability $\sigma_\ell = \sigma(f_\ell)$ provides an analogous solution $(\mathbf{x}^*_\ell, \mathbf{a}^*_\ell)$ with objective value U^*_ℓ, where:

- U^*_ℓ is an upper bound on the maximal constraint violation in $H^-_{\mathbf{w}} \cap [f_\ell, f_u]$.
- The true violation $C(\mathbf{x}^*_\ell, \mathbf{a}^*_\ell; \mathbf{w})$ is a lower bound on the maximal constraint violation in $H^-_{\mathbf{w}}[f_\ell, f_u]$.

As a result, $U^* = \max(U^*_\ell, U^*_u)$ is an upper bound on the maximal constraint violation in $[f_\ell, f_u]$ (we denote by $MCV(I)$ the maximum *degree* of constraint violation in interval I, which is distinct from the MVC, i.e., the constraint itself). The corresponding solution—denote this $(\mathbf{x}^*, \mathbf{a}^*)$—has a gap $U^* - C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w})$ that provides an indication of how close it is to provably being the MVC within $[f_\ell, f_u]$. This leads to the following conditions under which we can terminate the search for the MVC within this interval (see Fig. 2):

- If $U^* \leq \varepsilon$, for some small tolerance ε, we report that there is no (non-negligible) violation in $[f_\ell, f_u]$ and terminate the search in this interval.

- If $U^* \leq MCV(I)$, for some other interval I, we terminate the search in $[f_\ell, f_u]$ and report that the MVC (for the full subproblem) does not lie in this interval.[11]
- If $U^* \leq C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w}) + \varepsilon$, we terminate the search in $[f_\ell, f_u]$ and report that $(\mathbf{x}^*, \mathbf{a}^*)$ is (within ε of) the MVC in this interval.

Notice that the first condition induces some degree of approximation into the constraint generation process. We analyze this approximation in Section 4.4.1.

We can also exploit the discrete nature of state-action space to ensure an *exact* solution, which we also outline below.

Should none of these conditions hold, we split the interval and recursively search in the two subintervals $[f_\ell, \frac{f_\ell+f_u}{2}]$ and $[\frac{f_\ell+f_u}{2}, f_u]$ (we note that non-even splitting is also possible). We emphasize that when the subproblem search procedure terminates with an MVC $(\mathbf{x}^*, \mathbf{a}^*)$, we add the *true* constraint to the master, i.e., $0 \geq C(\mathbf{x}^*, \mathbf{a}^*; \mathbf{w})$, not the approximate constraint using the approximated probability.

It is straightforward to see that ALP-SEARCH will terminate with the identification of a violated constraint if there exists any constraint violated in the master to degree at least ε. Furthermore, it will never generate a constraint that is not violated. The latter point ensures that the master LP will never be overconstrained. The former means that when we terminate the constraint generation procedure, the final master LP may not include all violated constraints; but every unexpressed constraint will be within ε of being satisfied. We analyze the error associated with this relaxation relative to exact ALP in Section 4.4.1.

Exact Termination: While in practice, we expect ALP-SEARCH to use a small constraint violation error tolerance ε, the procedure is easily modified to produce an exact solution to the subproblem. Several different schemes can be used to test for exact termination. We outline an especially simple approach here, which we call the *identical witness test*.

When searching for the MVC in interval $[f_\ell, f_u]$, the following simple test ensures that our identified constraint is the true MVC in the interval: if the same state-action pair maximizes the degree of constraint violation using both constant probabilities σ_u and σ_ℓ, we can terminate with that constraint. This test can be implemented by solving two BOPs, over all $(\mathbf{x}, \mathbf{a}) \in [f_\ell, f_u]$, with constant probabilities σ_u and σ_ℓ, respectively. They produce, respectively, optimal values U and L and solutions $(\mathbf{x}_U, \mathbf{a}_U)$ and $(\mathbf{x}_L, \mathbf{a}_L)$. Note that U (resp., L) is an upper (resp., lower) bound on the maximum degree of constraint violation in $[f_\ell, f_u]$.

To account for the potential arbitrary selection of solutions due to ties, we must then evaluate: (a) the degree of constraint violation for $(\mathbf{x}_U, \mathbf{a}_U)$ using σ_ℓ—if it matches L, then $(\mathbf{x}_U, \mathbf{a}_U)$ is a true MVC; (b) the degree of constraint violation for $(\mathbf{x}_L, \mathbf{a}_L)$ using σ_u—if it matches U, then $(\mathbf{x}_L, \mathbf{a}_L)$ is a true MVC. If neither of these conditions holds, we split the interval. Since $\mathcal{X} \times \mathcal{A}$ is finite, this procedure is guaranteed to terminate. It is not hard to see that:

Observation 1. *ALP-SEARCH with the identical witness test constructs an exact solution to the ALP optimization problem.*

[11] We generalize this condition below to exploit parallelism in the search process.

Thus, an exact solution to the search problem is viable. In practice, however, we expect that a even a small approximation tolerance will encourage much faster convergence.

Parallelization and Other Speed-ups: ALP-SEARCH effectively builds a search tree of subintervals within logit space, recursively searching for MVCs in each subinterval. Once a subinterval is split, the search for MVCs in each child subinterval can be undertaken in a parallel or distributed manner, which can greatly accelerate wall-clock computation time. Moreover, with small amounts of communication of lower bounds across these parallel searches, early termination of the search within a subinterval becomes possible. Specifically, we can generalize the second termination condition above, which tests whether the upper bound U^* on the MCV in some interval lies below the true MCV in any other interval. Instead, we can compare this upper bound to the *lower bound* currently known for any other subinterval (and hence the global best bound for the entire problem). Lower bounds in any subinterval can be propagated to their parent intervals trivially and updated as search proceeds deeper in the tree, effectively giving us a form of bounding found in branch-and-bound. The process can, of course, be parallelized or distributed in a fashion similar to distributed versions of branch-and-bound.

A number of other techniques can be used to accelerate computation. The upper bound on the MCV described above supposes that the subproblem BOP is solved exactly within the interval in question, with approximation arising solely due to the use of an inexact probability. The LP relaxation of the BOP can be used to obtain an even looser upper bound by allowing fractional assignments to state and action variables. Since the relaxation can be solved much more quickly than the BOP itself, this may lead to earlier pruning of certain "nodes" in the search tree. Additional speed up may be attained by using the MVC solution at any interval to warm start the solution at the corresponding child subinterval.

4.3 A Piecewise Constant Approximation

The CG procedure above provides an "exact" solution (modulo the termination tolerance ε) to the subproblem by solving a *sequence* of BOPs—the formulation of one BOP depends on the results of a previous BOP. While this does lend itself to some forms of parallelization or distribution, it requires communication among the subinterval optimization problems. The idea of linearizing the subproblem using a constant approximation to the (sigmoid) response probability over some interval $I = [f_\ell, f_u]$ in logit space can be applied in a non-adaptive way as well, leading to an "embarrassingly" parallelizable approximation procedure. To do this, we fix a set of intervals in logit space, and approximate the sigmoid in each interval with a constant σ_I. Such a piecewise-constant (PWC) approximation allows us to compute the MVC within each interval using a single BOP; and since the interval subproblems are independent, they can be solved in parallel with no communication. We call the ALP method using this means of constraint generation *ALP-APPROX*.

More precisely, we partition logit space into m intervals $[\delta_{i-1}, \delta_i]$, $1 \leq i \leq m$, and associate a constant response probability σ_i with intervals $[\delta_{i-1}, \delta_i]$, where $\sigma_i = \sigma(f_i)$ for some $f_i \in [\delta_{i-1}, \delta_i]$. We require only that $\delta_0 \leq \min_{\mathbf{x},\mathbf{a}} f(\mathbf{x}, \mathbf{a})$ and $\delta_m \geq \max_{\mathbf{x},\mathbf{a}} f(\mathbf{x}, \mathbf{a})$. See Fig. 3 for an illustration. As above, the search for the MVC within the ith interval $[\delta_{i-1}, \delta_i]$ is now linear:

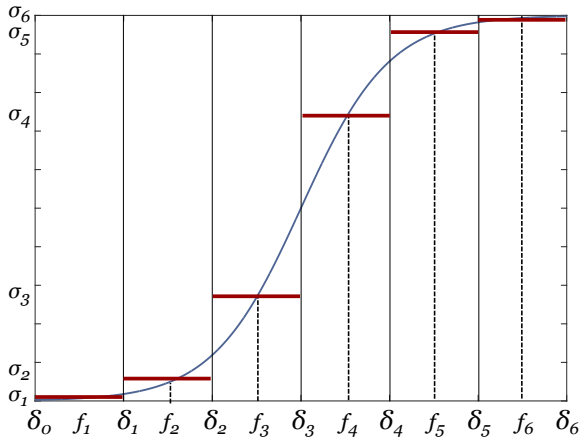


Fig. 3: A piecewise-constant approximation of the sigmoid.

$$\max_{\mathbf{x},\mathbf{a}} \quad \sigma_i h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \sigma_i) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \tag{31}$$

$$\textit{s.t.} \quad \delta_{i-1} \leq f(\mathbf{x}, \mathbf{a}) \leq \delta_i. \tag{32}$$

Unlike the search procedure ALP-SEARCH, we do not refine these intervals, but use a suitably chosen partitioning and fixed σ_i for each such interval. We solve the subproblem optimization for each interval I independently to obtain an approximately MVC $(\mathbf{x}^I, \mathbf{a}^I)$. We then compute the true degree of constraint violation $C(\mathbf{x}^I, \mathbf{a}^I; \mathbf{w})$ for each candidate and return to the master the (true) constraint for the candidate whose true violation is maximum. If the maximum true constraint violation is non-positive, we report no violated constraint and terminate the master with an approximately optimal solution.

Notice that this constraint generation procedure will generally incur some error, since it may fail to detect a violated constraint. This can arise for two reasons. First, the PWC approximation (within any interval I) may determine that no constraints are violated within I even though there may be a violation (whose degree has been underestimated). Second, while it may detect a state-action pair that has a violation under the approximation, if the true violation value for this pair is non-positive the procedure will claim no violation exists, yet it is possible that a different state-action pair still exhibits a true violation. If this occurs in all intervals, no violated constraint will be added and the master will terminate.[12] We analyze and bound the error induced by this occurrence in Section 4.4.

In particular, we show how the selection of intervals and their corresponding response probabilities influences approximation error.

A key advantage of ALP-APPROX is that the subproblems corresponding to the distinct intervals I at each iteration are independent, and hence can be solved in parallel. This stands in contrast with the exact approach ALP-SEARCH in which the refined subproblems within any branch of the search

[12]Note that should a constraint be missed in some interval I during one round of CG, as long as the master does not terminate (i.e., as long as a truly violated constraint was found in some other interval), a violated constraint may be found in I in some later round, since the subproblem changes with each new master iteration.

tree must be solved in sequence. By contrast, the sequencing of the exact approach—by refining intervals iteratively during a single round of CG—does not incur (significant) approximation error. Obvious hybrid approaches, of course, can be derived.

4.4 Approximation Error

We analyze the error in solution quality introduced by the two forms of approximation used above: allowing ε-tolerance when assessing the degree of maximal constraint violation in ALP-SEARCH; and the PWC approximation used in ALP-APPROX.

4.4.1 Analysis of ALP-SEARCH

We begin by analyzing the impact of inexact subproblem optimization on the quality of the ALPCG procedure. We do so by comparing the quality of the solution induced by exact ALP and *approximate ALP with ε constraint error (AALP(ε))*. The latter refers to any variant of ALP in which the constraint 14, for each $(\mathbf{x}, \mathbf{a})$, is replaced by the relaxed constraint

$$\varepsilon_{\mathbf{x},\mathbf{a}} \geq h(\mathbf{x}, \mathbf{a}; \mathbf{w}), \tag{33}$$

for some $0 \leq \varepsilon_{\mathbf{x},\mathbf{a}} \leq \varepsilon$ (i.e., the constraint for each state-action pair can be relaxed to a different extent). For example, a constraint generation method AALPCG(ε) that terminates with the guarantee that every ungenerated constraint $(\mathbf{x}, \mathbf{a})$ satisfies Eq. 33 is such a procedure.

Theorem 2. *Consider a fixed basis set $\mathcal{B}$ and a state weighting function α. Let $\widetilde{V}_{\mathbf{B}} = \mathbf{B}\widetilde{\mathbf{w}}$ be the value function approximation induced by the optimal solution to an AALP(ε) and $V_{\mathbf{B}} = \mathbf{B}\mathbf{w}$ be the VF approximation according to the optimal solution for (exact) ALP with the same MDP, $\mathcal{B}$ and α. Then*

$$0 \leq \sum_{\mathbf{x}} \alpha(\mathbf{x}) \left(V_{\mathbf{B}}(\mathbf{x}) - \widetilde{V}_{\mathbf{B}}(\mathbf{x}) \right) \leq \frac{\varepsilon}{1-\gamma}.$$

Proof. To simplify the analysis, we make several assumptions: (a) The basis functions are normalized so that $\mathbf{B1} = \mathbf{1}$. This is w.l.o.g. since each β_i can be binarized using a one-hot encoding, with each state corresponding to an indicator vector with a single non-zero entry per β_i, with normalization determined by $|\mathcal{B}|$. (b) The state weighting function is similarly normalized (as would be standard for a weighting function reflecting an initial or steady state distribution). (c) As mentioned in Section 2.3.3, the basis includes a bias term $\beta_{bias} = \mathbf{1}$.

Since AALP(ε) solves a relaxed version of the ALP problem, its objective provides a lower bound on the objective of ALP, $\sum_{\mathbf{x}} \alpha(\mathbf{x}) \widetilde{V}_{\mathbf{B}}(\mathbf{x}) \leq \sum_{\mathbf{x}} \alpha(\mathbf{x}) V_{\mathbf{B}}(\mathbf{x})$. Now consider a modification $\mathbf{w}'$ of the relaxed solution that increases the bias weight by $\frac{\varepsilon}{1-\gamma}$, i.e.: $w'_i = \widetilde{w}_i + \frac{\varepsilon}{1-\gamma}$ if i indicates the bias and $w'_i = \widetilde{w}_i$ otherwise. This new solution gives rise to a new value function approximation $V'_{\mathbf{B}}(\mathbf{x}) = \widetilde{V}_{\mathbf{B}}(\mathbf{x}) + \frac{\varepsilon}{1-\gamma}\beta_{\text{bias}}(\mathbf{x}) = \widetilde{V}_{\mathbf{B}}(\mathbf{x}) + \frac{\varepsilon}{1-\gamma}$, as the bias function is equal to one by assumption. The objective value for this new solution (in either ALP or AALP) provides a bounded increase relative to the approximate objective:

$$\sum_{\mathbf{x}} \alpha(\mathbf{x}) \left(V'_{\mathbf{B}}(\mathbf{x}) - \widetilde{V}_{\mathbf{B}}(\mathbf{x}) \right) = \sum_{\mathbf{x}} \alpha(\mathbf{x}) \frac{\varepsilon}{1-\gamma} \beta_{\text{bias}}(\mathbf{x}) = \frac{\varepsilon}{1-\gamma}, \tag{34}$$

due to α being normalized.

If $\mathbf{w}'$ is feasible for the (exact) ALP problem, then $\sum_{\mathbf{x}} \alpha(\mathbf{x}) V'_{\mathbf{B}}(\mathbf{x})$ is an upper bound on the optimal objective value, i.e., $\sum_{\mathbf{x}} \alpha(\mathbf{x}) \widetilde{V}_{\mathbf{B}}(\mathbf{x}) \leq \sum_{\mathbf{x}} \alpha(\mathbf{x}) V_{\mathbf{B}}(\mathbf{x}) \leq \sum_{\mathbf{x}} \alpha(\mathbf{x}) V'_{\mathbf{B}}(\mathbf{x})$, which establishes our result. We now show this is the case.

The feasibility of $\widetilde{\mathbf{w}}$ for the AALP(ε) ensures that $Q^{\widetilde{V}_{\mathbf{B}}}(\mathbf{x}, \mathbf{a}) \leq \widetilde{V}_{\mathbf{B}}(\mathbf{x}) + \varepsilon$ for any $\mathbf{x}, \mathbf{a}$. Now consider the action-value backup of $V'_{\mathbf{B}}$:

$$
\begin{aligned}
Q^{V'_{\mathbf{B}}}(\mathbf{x}, \mathbf{a}) &= Q^{\widetilde{V}_{\mathbf{B}} + \frac{\varepsilon}{1-\gamma}}(\mathbf{x}, \mathbf{a}) && (35)\\
&= R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{y}} Pr(\mathbf{y}|\mathbf{x}, \mathbf{a}) \left(\widetilde{V}_{\mathbf{B}}(\mathbf{y}) + \frac{\varepsilon}{1-\gamma} \right) && (36)\\
&= R(\mathbf{x}, \mathbf{a}) + \gamma \sum_{\mathbf{y}} Pr(\mathbf{y}|\mathbf{x}, \mathbf{a}) \widetilde{V}_{\mathbf{B}}(\mathbf{y}) + \gamma \sum_{\mathbf{y}} \frac{\varepsilon}{1-\gamma} Pr(\mathbf{y}|\mathbf{x}, \mathbf{a}) && (37)\\
&= Q^{\widetilde{V}_{\mathbf{B}}}(\mathbf{x}, \mathbf{a}) + \frac{\gamma\varepsilon}{1-\gamma} && (38)\\
&\leq \widetilde{V}_{\mathbf{B}}(\mathbf{x}) + \varepsilon + \frac{\gamma\varepsilon}{1-\gamma} && (39)\\
&= \widetilde{V}_{\mathbf{B}}(\mathbf{x}) + \frac{\varepsilon}{1-\gamma} && (40)\\
&= V'_{\mathbf{B}}(\mathbf{x}). && (41)
\end{aligned}
$$

Thus, since $\mathbf{w}'$ satisfies the backup constraint $Q^{V'_{\mathbf{B}}}(\mathbf{x}, \mathbf{a}) \leq V'_{\mathbf{B}}(\mathbf{x})$ for any $\mathbf{x}, \mathbf{a}$, it is feasible for ALP. □

ALP-SEARCH, using the search procedure in Section 4.2 with termination tolerance ε, is clearly an AALP(ε) procedure. Hence we have the following corollary:

Corollary 3. *Let $\widetilde{\mathbf{w}}$ be the optimal solution to ALP-SEARCH with termination tolerance ε, and $\widetilde{V}_{\mathbf{B}}$ its corresponding VF approximation. Let $\mathbf{w}$ and $V_{\mathbf{B}}$ denote the same for (exact) ALP. Then*

$$
0 \leq \sum_{\mathbf{x}} \alpha(\mathbf{x}) \left(V_{\mathbf{B}}(\mathbf{x}) - \widetilde{V}_{\mathbf{B}}(\mathbf{x}) \right) \leq \frac{\varepsilon}{1-\gamma}.
$$

4.4.2 Analysis of ALP-APPROX

The ALP-APPROX algorithm requires that we analyze the maximum degree of constraint relaxation induced by the PWC sigmoid approximation. There are several analytical approaches one could adopt; we focus on one specific approach that relates the maximum error in the value function to the transition function error introduced by the approximation, and then show how this can be used to influence the design of the PWC approximation itself.

We adapt a result of Boyen and Koller [11] on the error introduced in belief state estimation of a (factored) partially observable Markov chain (or factored HMM) when per-step belief state approximation is used. Specifically, suppose we have an underlying Markov chain $T(\mathbf{x}^{(t)}|\mathbf{x}^{(t-1)})$ over finite state space $\mathbf{X}$. Let $D^{(t)}$ represent a *belief state* distribution at time t , i.e., a distribution over $\mathbf{X}$, representing one's posterior over state space given the history of observations. Maintaining this belief

state is often intractable in factored models, so we assume an approximation process A is adopted of the following form, starting with an initial belief state $D^{(0)}$:

- Letting $\widetilde{D}^{(t-1)}$ denote the approximate belief state at time $t-1$, define $\widehat{D}^{(t)} = T\widetilde{D}^{(t-1)}$ to be the projection of the approximate belief state through the true transition function.
- Define $\widetilde{D}^{(t)} = A\left[\widehat{D}^{(t)}\right]$ to be the result of applying approximation A to the resulting projected belief state.
- Finally, define $D^{(t)} = TD^{(t-1)}$ to be the true belief state at time t of the process.

Note we have described a process with no observations, just the projection of an initial belief state through the dynamics. An approximation step A is typically used to maintain tractability of the representation of the distribution (e.g., Boyen and Koller analyze the effect projecting the joint posterior distribution at time t into a collection of independent factors). Boyen and Koller [11] show that the KL-divergence between the approximate and true belief state at any stage t is bounded under certain conditions. Specifically, if T *mixes* at rate at least α,[13] and the approximation A satisfies (for any t)

$$KL(D^{(t)} \| \widetilde{D}^{(t)}) - KL(D^{(t)} \| \widehat{D}^{(t)}) \leq \varepsilon,$$

then

$$KL(D^{(t)} \| \widetilde{D}^{(t)}) \leq \varepsilon/\alpha$$

for all t. In other words, if the approximation scheme introduces a bounded additional error at each stage of the approximation, then the maximum cumulative error is bounded as well, due to the mixing/contraction properties of the Markov process.

Given a fixed policy π, our PWC sigmoid approximation induces an a transition function $\widehat{T}_\pi$ that approximates the true dynamics T_π. We can implicitly interpret this as a Boyen-Koller approximation scheme by assuming that $\widehat{T}_\pi$ consists of first projecting a belief state $\widetilde{D}^{(t-1)}$ through the true dynamics T_π to obtain $\widehat{D}^{(t)}$, and then applying some (unspecified) approximation method A to obtain $\widetilde{D}^{(t)}$. Thus, if

$$KL(D^{(t)} \| T_\pi \widetilde{D}^{(t-1)}) - KL(D^{(t)} \| \widehat{T}_\pi \widetilde{D}^{(t-1)}) \leq \varepsilon \tag{42}$$

then we can bound the error between the true and approximate distribution induced by any policy π at any stage of the process, and more specifically, the error in the steady state distribution (which we exploit below).

Assume a PWC sigmoid approximation δ with each interval $[\delta_{i-1}, \delta_i]$ approximated by constant σ_i. For any $(\mathbf{x}, \mathbf{a})$, we write $\delta(\mathbf{x}, \mathbf{a})$ to denote the approximation of its true response probability $\sigma(\mathbf{x}, \mathbf{a})$. We show that the l.h.s. of Eq. 42 is in fact bounded:

Proposition 4.

$$KL(D^{(t)} \| T_\pi \widetilde{D}^{(t-1)}) - KL(D^{(t)} \| \widehat{T}_\pi \widetilde{D}^{(t-1)}) \leq \ln \max_{\mathbf{x} \in \mathbf{X}} \max \left[\frac{\delta(\mathbf{x}, \pi(\mathbf{x}))}{\sigma(\mathbf{x}, \pi(\mathbf{x}))}, \frac{1 - \delta(\mathbf{x}, \pi(\mathbf{x}))}{1 - \sigma(\mathbf{x}, \pi(\mathbf{x}))} \right].$$

[13] The mixing rate definition they use is as follows: $\min_{\mathbf{x}_1, \mathbf{x}_2} \sum \mathbf{x} \min\{\Pr(\mathbf{x}_1|\mathbf{x}), \Pr(\mathbf{x}_2|\mathbf{x})\}$, but can be easily generalized in several ways to be less restrictive.

Proof.

$$
\begin{aligned}
&KL(D^{(t)} \| T^\pi \widetilde{D}^{(t-1)}) - KL(D^{(t)} \| \widehat{T}^\pi \widetilde{D}^{(t-1)}) \\
&= \sum_{\mathbf{x}'} D^{(t)}(\mathbf{x}') \ln \frac{[\widehat{T}^\pi \widetilde{D}^{(t-1)}](\mathbf{x}')}{[T^\pi \widetilde{D}^{(t-1)}](\mathbf{x}')} \\
&\leq \max_{\mathbf{x}'} \ln \frac{[\widehat{T}^\pi \widetilde{D}^{(t-1)}](\mathbf{x}')}{[T^\pi \widetilde{D}^{(t-1)}](\mathbf{x}')} \\
&= \max_{\mathbf{x}'} \ln \frac{\sum_{\mathbf{x}} \left(\widehat{p}(\top|\mathbf{x}, \pi(\mathbf{x})) p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}), \top) + \widehat{p}(\bot|\mathbf{x}, \pi(\mathbf{x})) p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}), \bot) \right) \widetilde{D}^{(t-1)}(\mathbf{x})}{\sum_{\mathbf{x}} \left(p(\top|\mathbf{x}, \pi(\mathbf{x})) p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}), \top) + p(\bot|\mathbf{x}, \pi(\mathbf{x})) p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}), \bot) \right) \widetilde{D}^{(t-1)}(\mathbf{x})} \\
&\leq \max_{\mathbf{x}'} \ln \max_{\mathbf{x}, \phi \in \{\top, \bot\}} \frac{\widehat{p}(\phi|\mathbf{x}, \pi(\mathbf{x})) p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}), \phi) \widetilde{D}^{(t-1)}(\mathbf{x})}{p(\phi|\mathbf{x}, \pi(\mathbf{x})) p(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}), \phi) \widetilde{D}^{(t-1)}(\mathbf{x})} \\
&= \max_{\mathbf{x}'} \ln \max_{\mathbf{x}, \phi \in \{\top, \bot\}} \frac{\widehat{p}(\phi|\mathbf{x}, \pi(\mathbf{x}))}{p(\phi|\mathbf{x}, \pi(\mathbf{x}))} \\
&= \ln \max_{\mathbf{x}} \max \left(\frac{\delta(\mathbf{x}, \pi(\mathbf{x}))}{\sigma(\mathbf{x}, \pi(\mathbf{x}))}, \frac{1 - \delta(\mathbf{x}, \pi(\mathbf{x}))}{1 - \sigma(\mathbf{x}, \pi(\mathbf{x}))} \right).
\end{aligned}
$$

□

We call the l.h.s. of the inequality in Prop. 4 the *log-relative error* of approximation δ, or $LRE(\delta)$.[14] We immediately have:

Corollary 5. *If the MDP mixes with rate at least α for any policy π, then*

$$KL(D^{(t)} \| \widetilde{D}^{(t)}) \leq LRE(\delta)/\alpha.$$

Typical MDPs have many static variables as discussed above. To apply the Boyen-Koller definition of mixing, we can view each set of static features as inducing a distinct MDP (like "context" in contextual bandits); it is only within these sub-processes that we require mixing. Hence, the relevant mixing rate for our purposes is that of the "sub-MDP" with the smallest mixing rate. (See the discussion above for why we solve the problem as a single joint MDP, despite the non-communicating nature of these "sub-MDPs.")

This formulation allows us to bound the error in the value function associated with any policy when evaluated using our approximate dynamics.

Theorem 6. *Let π be an arbitrary deterministic policy and $D^{(0)}$ an initial state distribution. Let V_π be the value function of π under the true dynamics T_π and $\widetilde{V}_\pi$ under the approximate dynamics $\widehat{T}_\pi$. Then*

$$\max_{\mathbf{x}} |V_\pi(\mathbf{x}) - \widetilde{V}_\pi(\mathbf{x})| \leq \frac{1}{1-\gamma} \sqrt{(2 \ln 2) \frac{1}{\alpha} LRE(\delta)} \max_{\mathbf{x}} R(\mathbf{x}, \pi(\mathbf{x})).$$

[14] We can remove the dependence on π by maximizing over actions $\mathbf{a}$.

Proof. The expected value of π under the initial distribution $D^{(0)}$ satisfies $\langle V_\pi, D^{(0)}\rangle = \langle D^*_\pi, R_\pi\rangle$, where $R_\pi(\mathbf{x}) = R(\mathbf{x}, \pi(\mathbf{x}))$, and

$$D^*_\pi = \sum_{t=0}^{\infty} \gamma^t T^t_\pi D^{(0)}$$

is the discounted total visitation frequency over $\mathbf{X}$ induced by π. We define $\widetilde{D}^*_\pi$ similarly and note that $\langle \widetilde{V}_\pi, D^0\rangle = \langle \widetilde{D}^*_\pi, R_\pi\rangle$.

Recalling that $\|D - D'\|_1 \leq \sqrt{(2\ln 2) KL(D\|D')}$, we have

$$\|\widetilde{D}^*_\pi - D^*_\pi\|_1 = \left\| \sum_{t=0}^{\infty} \gamma^t \widehat{T}^t_\pi D^{(0)} - \sum_{t=0}^{\infty} \gamma^t T^t_\pi D^{(0)} \right\|_1 \leq \sum_{t=0}^{\infty} \gamma^t \|\widehat{T}^t_\pi D^{(0)} - T^t_\pi D^{(0)}\|_1 \leq \frac{1}{1-\gamma}\sqrt{(2\ln 2)\frac{1}{\alpha} LRE(\delta)}.$$

Therefore:

$$\begin{aligned}
|\langle (V_\pi - \widetilde{V}_\pi), D^0\rangle| &= |\langle D^*_\pi, R_\pi\rangle - \langle \widetilde{D}^*_\pi, R_\pi\rangle| \\
&= |\langle (D^*_\pi - \widetilde{D}^*_\pi), R_\pi\rangle| \\
&\leq \|(D^*_\pi - \widetilde{D}^*_\pi) R_\pi\|_1 \\
&\leq \|D^*_\pi - \widetilde{D}^*_\pi\|_1 \|R_\pi\|_\infty \\
&\leq \frac{1}{1-\gamma}\sqrt{(2\ln 2)\frac{1}{\alpha} LRE(\delta)}\ \|R_\pi\|_\infty,
\end{aligned}$$

(where the next-to-last inequality follows from the Hölder Inequality). This holds for any D^0, hence applying the above with a degenerate distribution $\delta_x(\mathbf{x})$ for every individual state x yields the bound. □

Assuming no additional value function approximation, this limits the error between the value of the optimal policy under the true dynamics and the policy purported to be optimal under the approximate dynamics to be at most twice this bound.

The LRE term also suggests a mechanism for finding optimal constant probabilities for any given set of bands or intervals. Furthermore, we can obtain an LRE bound of arbitrary precision $\varepsilon > 0$ with $O(\frac{1}{\varepsilon})$ (*a priori* chosen) bands.

Theorem 7. *A bounded log-relative error for the logistic regression (assuming features with finite domains) can be achieved with $O(\frac{1}{\epsilon}\|\mathbf{u}\|_1)$ intervals in logit space, where $\mathbf{u}$ is the logistic regression weight vector.*

Proof. To demonstrate this, we partition logit space into intervals of equal size. This is possible as we only approximate the sigmoid on the finite range between the upper and lower bounds on the logits (i.e., $(\mathbf{x}, \mathbf{a})$ pairs according to the regression model). The range's length is $O(\|\mathbf{u}\|_1)$ (this is a very crude upper bound; in examples of interest, the dynamic range is roughly 2).

Suppose we split this range into intervals of length 2ϵ and take the middle of each interval as its piecewise-constant value. For an arbitrary interval $[\ell, u]$, denote its by s_0 (meaning $\delta(s) = \sigma(s_0)$) and its endpoints $s_0 - \epsilon, s_0 + \epsilon$. Note that both terms in the LRE are monotone in the actual logit value $s \in [\ell, u]$: $\frac{\delta(s_0)}{\sigma(s)}$ is monotone decreasing whereas $\frac{1-\sigma(s_0)}{1-\sigma(s)}$ is monotone increasing when s is

non-negative (and vice versa when it is negative). As a consequence, for $s \geq 0$ LRE in the interval can be rewritten as:

$$\ln \max_{s_0-\epsilon \leq s \leq s_0+\epsilon} \max\left(\frac{\delta(s)}{\sigma(s)}, \frac{1-\delta(s)}{1-\sigma(s)}\right) = \max\left(\ln \frac{\sigma(s_0)}{\sigma(s_0-\epsilon)}, \ln \frac{1-\sigma(s_0)}{1-\sigma(s_0+\epsilon)}\right).$$

For negative s, the ϵ switches sign. Now, we can show that both terms are $O(\epsilon)$. Let us consider the case $s \geq 0$ (the argument for $s \leq 0$ is analogous):

$$\begin{aligned}
\ln \frac{\sigma(s_0)}{\sigma(s_0-\epsilon)} &= \ln(1+e^{-s_0+\epsilon}) - \ln(1+e^{-s_0}) \\
&= \ln e^{\epsilon}(e^{-\epsilon}+e^{-s_0}) - \ln(1+e^{-s_0}) \\
&= \epsilon + \ln(e^{-\epsilon}+e^{-s_0}) - \ln(1+e^{-s_0}) \\
&\leq \epsilon + \ln(e^{0}+e^{-s_0}) - \ln(1+e^{-s_0}) \\
&= \epsilon + \ln(1+e^{-s_0}) - \ln(1+e^{-s_0}) \\
&= \epsilon.
\end{aligned}$$

Similarly,

$$\begin{aligned}
\ln \frac{1-\sigma(s_0)}{1-\sigma(s_0+\epsilon)} &= \ln \frac{e^{-s_0}}{1+e^{-s_0}} - \ln \frac{e^{-s_0-\epsilon}}{1+e^{-s_0-\epsilon}} \\
&= \ln e^{-s_0} - \ln e^{-s_0-\epsilon} + \ln(1+e^{-s_0-\epsilon}) - \ln(1+e^{-s_0}) \\
&= \epsilon + \ln(1+e^{-s_0-\epsilon}) - \ln(1+e^{-s_0}) \\
&\leq \epsilon,
\end{aligned}$$

since $1+e^{-s_0-\epsilon} \leq 1+e^{-s_0}$ as $s_0, \epsilon \geq 0$. □

This bound holds for a uniform quantization with the constant probability chosen at the midpoint of the interval. One could go one step further and ask, given an interval $[a, b]$, which constant probability minimizes uniform LRE. It turns out this is not hard to compute:

Theorem 8. *Given an interval $[a, b]$ in logit space, the value $\sigma(x)$ with $x = \ln \frac{e^{a+b}+e^b}{1+e^b}$ minimizes the log-relative error over the interval.*

Proof. Recall that

$$\ln \max_{a \leq s \leq b} \max\left(\frac{\delta(s)}{\sigma(s)}, \frac{1-\delta(s)}{1-\sigma(s)}\right) = \max\left(\ln \frac{\sigma(x)}{\sigma(a)}, \ln \frac{1-\sigma(x)}{1-\sigma(b)}\right).$$

We want an x^* that minimizes this quantity. Given that the error is a maximum of two monotone terms (one increasing and one decreasing), the its minimum lies at the point where they meet. Hence s_0^* is the solution of the equation:

$$\ln \frac{\sigma(x)}{\sigma(a)} = \ln \frac{1-\sigma(x)}{1-\sigma(b)}.$$

Solving for x (e.g., in a computer algebra system) yields the desired result. □

5 Empirical Analysis

We test the effectiveness of the logistic MDP framework, focusing on the scalability of the ALP-APPROX algorithm using a targeted advertising domain and models of various sizes. We also test the approach on rather small models so that we can explicitly determine the approximation quality of ALP-APPROX relative to the optimal ALP solution and the optimal MDP.

We briefly outline the basic experimental set up. Our focus is on a single binary user response variable ϕ—does the user click on the displayed ad—resulting in pCTR models that predict the probability (or rate) of click-throughs. We construct logistic regression pCTR models of various sizes using test data derived from the serving of a specific source of display ads to users, and the user responses to those ads (namely, clicks). The data set consists of data drawn over a period of one day with approximately 300M training examples, and is characterized by roughly 50 features (i.e., random variables with highly variable domain sizes) per example. We do not describe the features in detail here, but simply note that they consist of features reflecting characteristics of the user being shown the ad (including both static user features and summaries of past behavior), characteristics of the ad itself, and some that involve the "intersection" of the user and the ad (e.g., has the user been exposed to an ad from this campaign within a certain past time frame). The first set of features are *state features*, while the second and third sets are *action features* (since "intersection" features cannot be determined without knowing the ad to be shown).[15] The data, and our models, are constructed using a sparse binarized representation; but we also discuss the natural variable representation when discussing model size. Models are learned using a highly scalable logistic regression platform (trained using gradient descent with parallel boosting) [12].

From this model, we construct a logistic MDP with a single binary response variable ϕ at the core of the logistic transition model. Each (natural, not binarized) variable $X^{(t)}$ at time t is independent of all other variables $Y^{(t)}$ given the previous state and ϕ, and depends on a small number of previous time variables. The transition functions are derived from the semantics of the features themselves. In particular, each feature is either static or a bucketized count of some event over some recent horizon, hence the transition functions are, respectively, identity functions or a simple Bernoulli function reflecting the odds of staying in the same bucket or progressing to the next bucket. The MDP reward function assigns 1 to a (true, not predicted) click. Hence our sequential objective is to maximize the expected discounted *cumulative click-through rate (CCTR)*, in contrast to the usual myopic objective.[16]

We produce several models of various sizes:

- Tiny: this has two (natural) state variables (with domain sizes six and 42) and one action variable (with domain size seven). It gives rise to a one-hot encoding with 55 sparse binarized features, of which 48 are state features and seven are action features. The sizes of the induced state and action spaces are 252 and seven, respectively.

[15] Technically, most "intersection" features could replaced and modeled using more complex user and action features. But it is usually much easier to evaluate these at ad serving time using intersection features.

[16] Incorporating bid predictions so that we maximize social welfare or expected ad revenue is straightforward, but we ignore this extension here.

- Small: six state variables, four action variables; 86 sparse binarized features (71 state, 15 action); 158,760 states, 126 actions.
- Medium: 11 state variables, eight action variables; 421 sparse binarized features (251 state, 170 action); approximately 2^{43} states, 220M actions.
- Large: 12 state variables, 11 action variables; 2854 sparse binarized features (2630 state, 224 action); approximately 2^{54} states, 2^{39} actions.

We tested the ALP variant, ALP-APPROX, using basis functions that represent all domain values of the natural variables; thus, we have one basis function per sparse binarized state feature in the domain in question (thus, for instance, Small uses 71 basis functions while Large uses 2854). We run the ALP-APPROX algorithm using GLOP[17] to solve the master LP, and SCIP[18] to solve the Boolean subproblems. We solve the subproblems using various numbers of bands (or subintervals) of logit space to assess the impact of coarser and finer discretizations on computation time and solution quality.

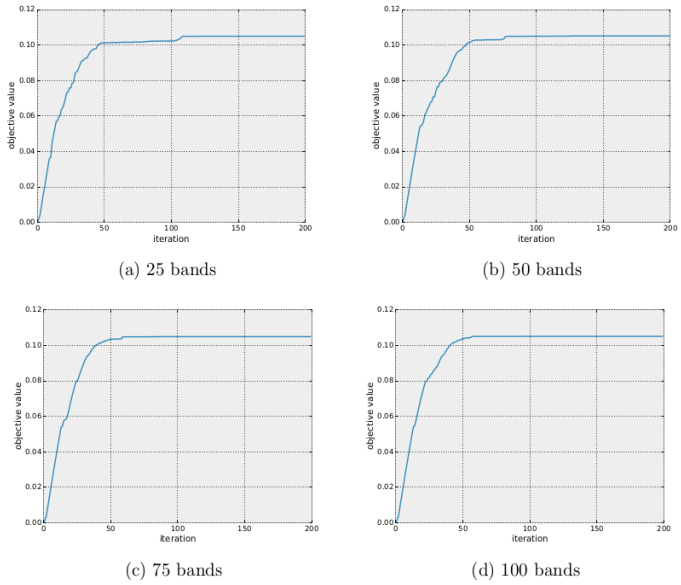
We first explore the convergence of the objective value w.r.t. the number of constraint generation iterations. Figs. 4, 5, 6 and 12(a) show performance for Tiny, Small, Medium and Large instances, respectively. We see that convergence to the maximum value occurs in very few CG iterations relative to the size of state-action space. For Small and Medium we also observe that the number of bands (i.e., preciseness of the PWC sigmoid approximation) matters little above 25 bands—performance with 50, 75 or 100 bands is not appreciably better than 25. In Tiny, we note that the move to 75 or 100 bands does offer a slight improvement between iterations 50 and 100.

Figs. 7, 8, 9 and 12(b) show how the degree of maximum constraint violation evolves with CG iterations. The approximated maximum violation is quite noisy, especially in the Medium and Large instances; but the true degree of violation of the constraint so-identified (shown in blue) is much smoother, suggesting that the approximation is still identifying good candidate constraints for the master. The relatively smooth increase in objective value discussed above is further evidence of the quality of the identified constraints. Not surprisingly, increasing the number of bands smooths the degree of approximate violation by reducing the (general tendency for) overestimation relative to the true violation. In the Small and Medium problems, a large increase in the number of bands (e.g., to 200 or 400) "smooths" or reduces the error in the approximated violation considerably. Despite this, performance in the master is relatively unaffected.

Figs. 10, 11 and 12(c) show computation per CG iteration. Note that at each CG iteration, the subproblems associated with each band are solved in sequence; a parallel implementation would allow, in principle, each to be solved independently in parallel, leading to a roughly a factor k speedup for subproblem optimization with k bands—and recall, since the subproblems are BOPs, this dominates master LP solution time. For Medium (25 bands) and Large we indicate the expected solution time for a parallel implementation. In Medium we observe a roughly linear increase (at rate slightly below 1.0) in computation time with the number of bands (consistent with the observation above about subproblem domination). Given the relatively slight impact on overall solution quality, this suggests

[17] See `https://developers.google.com/optimization/`.

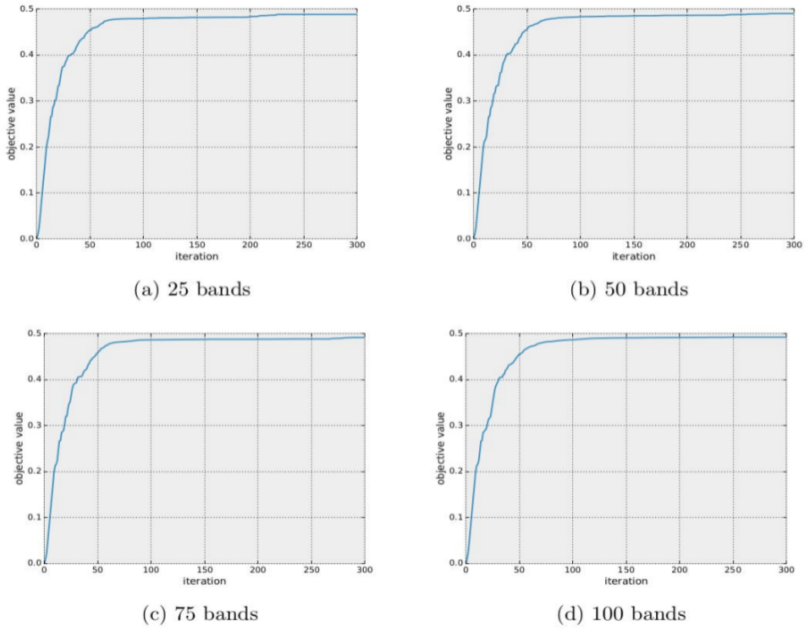
[18] See `http://scip.zib.de/`.



(a) 25 bands (b) 50 bands

(c) 75 bands (d) 100 bands

Fig. 4: Tiny MDP (55 sparse binarized features): Objective value per CG iteration (varying number of bands).



(a) 25 bands (b) 50 bands

(c) 75 bands (d) 100 bands

Fig. 5: Small MDP (86 sparse binarized features): Objective value per CG iteration (varying number of bands).

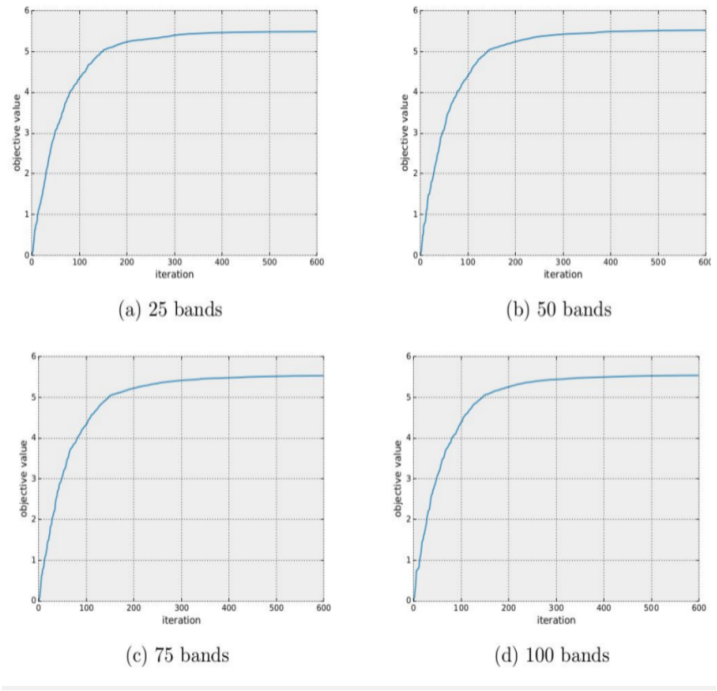


Fig. 6: Medium MDP (421 sparse binarized features): Objective value per CG iteration (varying number of bands).

that using a (relatively) coarse discretization provides a better rate of improvement in solution quality per unit time (CPU or wall clock).

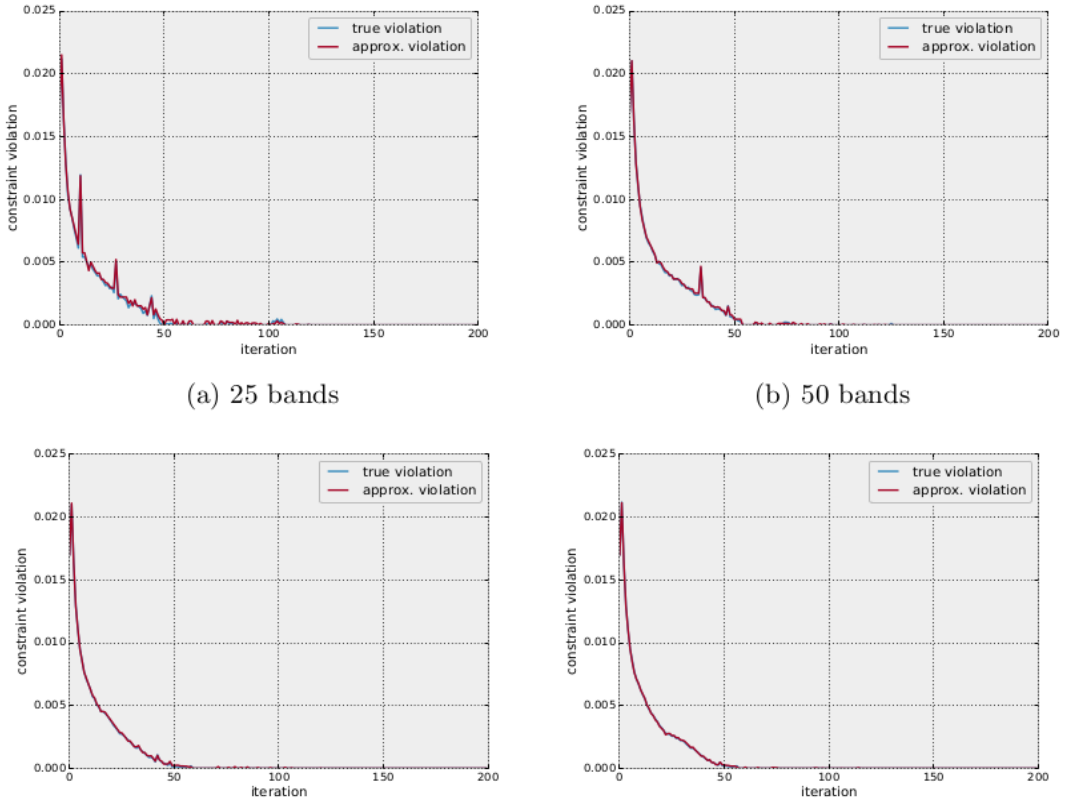
Finally, we perform a simple empirical comparison of the cumulative reward generated by the MDP-optimized policy to that obtained using myopic optimization. Using the Medium domain, we generate trials by sampling 50 random ads from the data set as the eligible action set,[19] a random initial state from the data set, and then measure reward accrued over trajectories of length 60 by the myopic and ALP policies. We sample 40 eligible sets and 2500 initial states for each set, giving 100K trials. The ALP policy shows a small improvement of about 0.52% in value over the myopic policy.[20]

6 Enhancements and Extensions

We outline several enhancements and extensions of the basic methods described above.

[19]This reflects the fact that for any impression only a small set of ads are considered eligible for various reasons (e.g., targeting, budget throttling, campaign expiration, etc.).

[20]The feature subset used in this synthetic model gives a model with low pCTR and very limited user state. As a result, the actions have a very small influence on user state. We observed a mean reward (std. dev.) per trajectory as follows: myopic: 0.75325 (1.0690); ALP: 0.75718 (1.0713). We do not find this difference to be statistically significant.



(a) 25 bands (b) 50 bands

(c) 75 bands (d) 100 bands

Fig. 7: Tiny MDP (55 sparse binarized features): Degree of maximum constraint violation (approximate red, exact, blue) per CG iteration (varying number of bands).

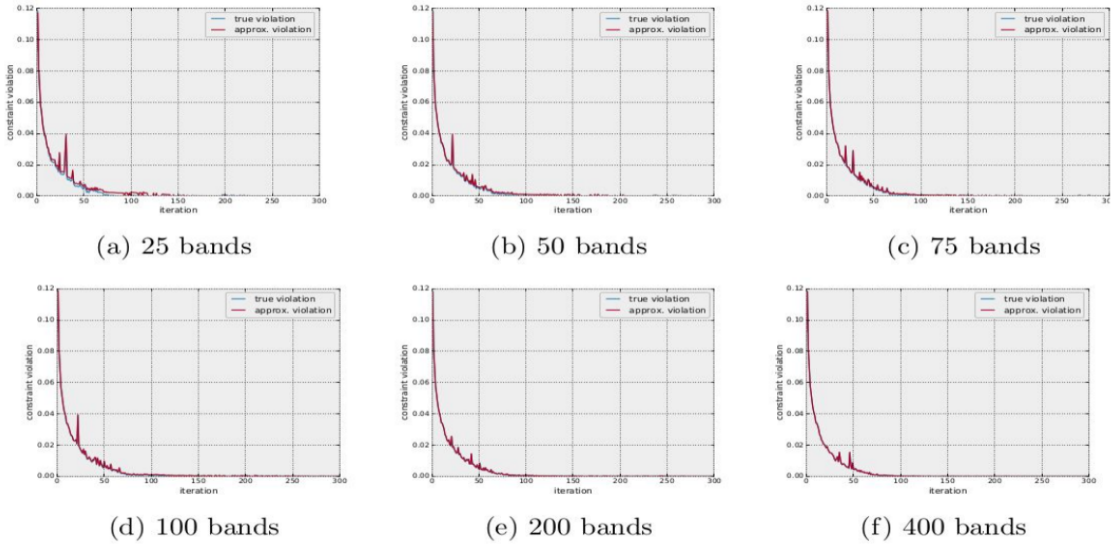


Fig. 8: Small MDP (86 sparse binarized features): Degree of maximum constraint violation (approximate red, exact, blue) per CG iteration (varying number of bands).

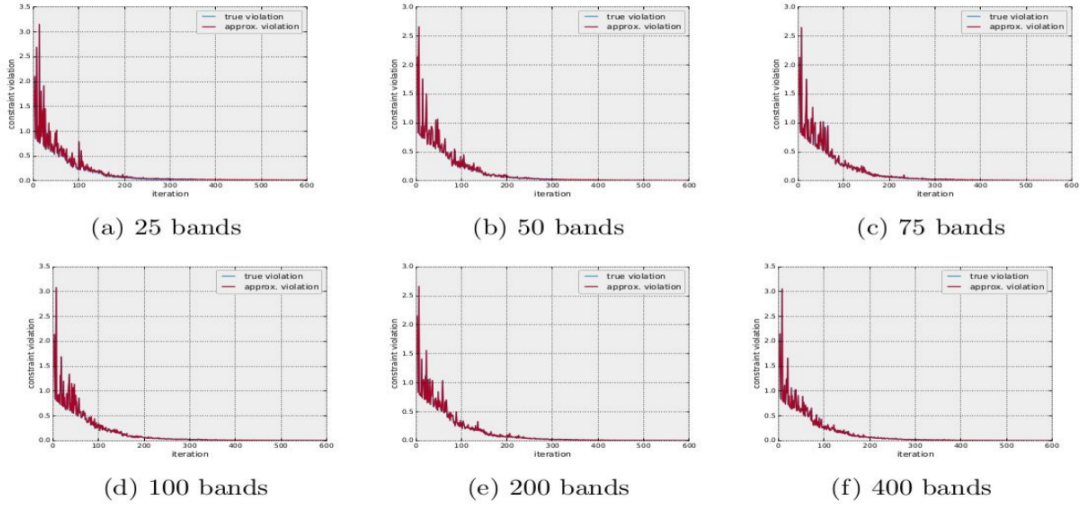


Fig. 9: Medium MDP (421 sparse binarized features): Degree of maximum constraint violation (approximate red, exact, blue) per CG iteration (varying number of bands).

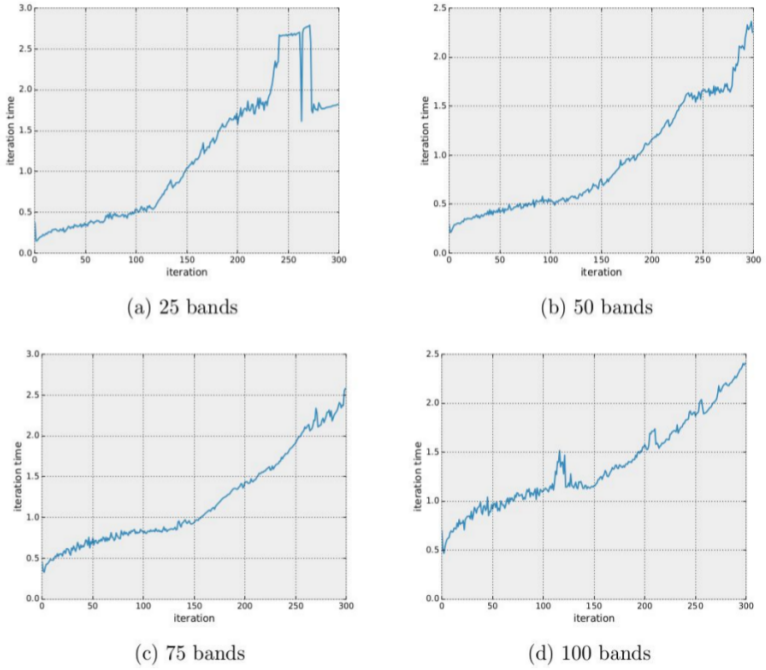


Fig. 10: Small MDP (86 sparse binarized features): Computation time per CG iteration (varying number of bands).

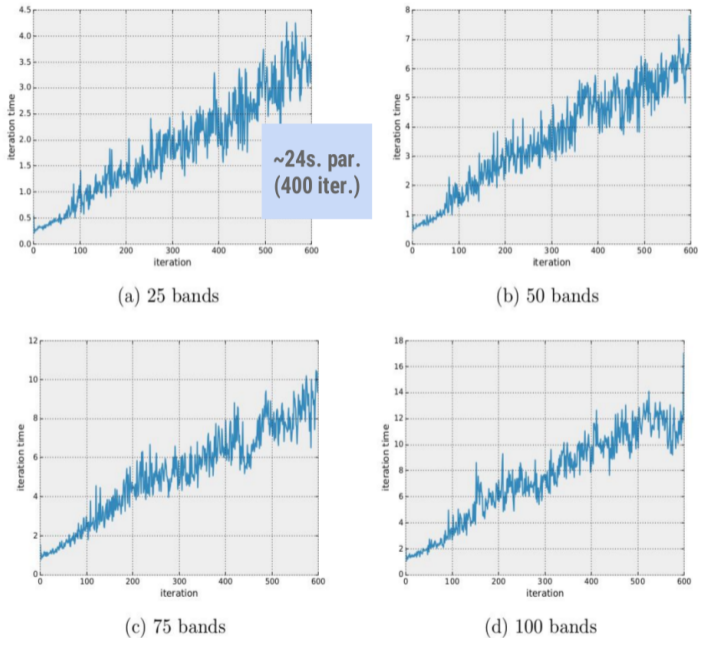


Fig. 11: Medium MDP (421 sparse binarized features): Computation time per CG iteration (varying number of bands).

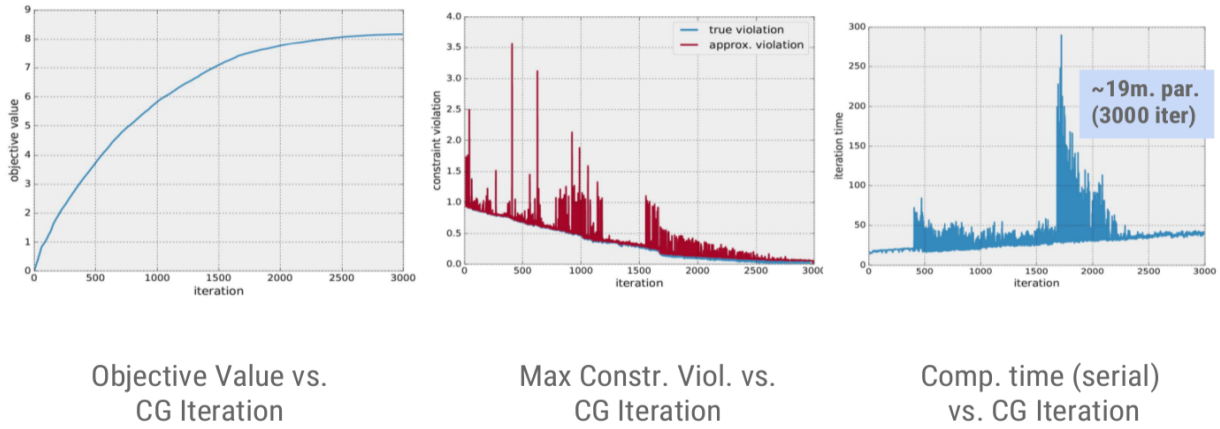


Fig. 12: Large MDP (2854 sparse binarized features), solved with 50 bands. (a) Objective value, (b) approximate and exact max constraint violation, and (c) computation time per CG iteration.

6.1 Cross-Product Features

In many practical uses of logistic regression, non-linear interactions among variables are captured by "crossing" features, that is, using the cross-product of the domains of two or more variables as an input to the response predictor.[21] Incorporating response models with crosses into the specification of a logistic MDP requires no special consideration, since it is simply part of the response variable dynamics and does not impact the core transition dynamics.

Within our constraint generation procedure, the presence of crosses is easily handled. No changes are required to the master LP formulation. In the generation subproblem, we treat the crossed variables as if they were additional variables, adding indicators for all elements of the crossed domain (which are then used exactly as base-level variables in the encoding). To ensure logical consistency in the instantiation of state and action variables, we impose standard constraints that relate the instantiation of the crossed variables to their corresponding base variables. For instance, if two variables X and Y are crossed, the indicator I_{x_i,y_j} for a specific instantiation (x_i, y_j) is related to the indicators for the base variables X and Y by constraints $I_{x_i,y_j} \leq I_{x_i}$ and $I_{x_i,y_j} \leq I_{y_j}$.

6.2 Relaxation of CG Optimization

Our ALPCG approaches are very scalable, able to handle logistic MDPs with very large state and action spaces. However, the main impediment to further scalability is the requirement to solve very large BOPs using either general MIP solvers or more specialized Boolean solvers. The theoretical intractability of constrained Boolean optimization can be addressed in a general way by using standard relaxations. We briefly outline several approaches of this type here.

[21] The use of crosses in the response model is analogous to using basis functions (in our value function representation) that incorporate multiple variables.

6.2.1 Relaxations in ALPCG

We first consider improving the efficiency of the solvers for the constraint generation problem (see Eq. 29-30 and Eq. 31-32) of the unified form,[22]

$$\max_{\mathbf{x},\mathbf{a}} \quad \bar{\sigma} h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \bar{\sigma}) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \tag{43}$$

$$\textit{s.t.} \quad f_\ell \leq f(\mathbf{x}, \mathbf{a}) \leq f_u \tag{44}$$

$$h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) \geq h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \qquad \text{[optional]} \; . \tag{45}$$

Some of the difficulty in this problem stems from the constraints (the hardness of the unconstrained problem depends on the form of h, but the constrained problem might be hard even when the unconstrained one is tractable). To improve tractability, we propose to use Lagrangian relaxation to incorporate constraints into the objective function. In particular, we define the functions:

$$g_\ell(\mathbf{x}, \mathbf{a}) = \begin{cases} -\infty & f(\mathbf{x}, \mathbf{a}) < f_\ell \\ 0 & \text{ow} \end{cases} \qquad g_u(\mathbf{x}, \mathbf{a}) = \begin{cases} -\infty & f(\mathbf{x}, \mathbf{a}) > f_u \\ 0 & \text{ow} \end{cases} \; .$$

With these we can rewrite the above problem as:

$$\begin{aligned} \max_{\mathbf{x},\mathbf{a}} \quad & \bar{\sigma} h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \bar{\sigma}) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) + g_\ell(\mathbf{x}^\ell, \mathbf{a}^\ell) + g_u(\mathbf{x}^u, \mathbf{a}^u) \\ \textit{s.t.} \quad & \mathbf{x} = \mathbf{x}^\ell, \mathbf{a} = \mathbf{a}^\ell \text{ and } \mathbf{x} = \mathbf{x}^u, \mathbf{a} = \mathbf{a}^u \text{ and } h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) \geq h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) \; . \end{aligned}$$

We then formulate the Lagrangian for this problem and obtain the dual,

$$\min_{\nu \geq 0, \lambda^\ell, \lambda^u, \mu^\ell, \mu^u} \quad \max_{\mathbf{x},\mathbf{a}} \quad \Big[(\bar{\sigma} + \nu) h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) + (1 - \bar{\sigma} - \nu) h(\mathbf{x}, \mathbf{a}, \bot; \mathbf{w}) + (\lambda^\ell + \lambda^u) \cdot \mathbf{x} + (\mu^\ell + \mu^u) \cdot \mathbf{a} \Big] \tag{46}$$

$$+ \quad \max_{(\mathbf{x}^\ell, \mathbf{a}^\ell) : f(\mathbf{x}^\ell, \mathbf{a}^\ell) \geq f_\ell} \quad \left[\lambda^\ell \cdot \mathbf{x}^\ell + \mu^\ell \cdot \mathbf{a}^\ell \right] \tag{47}$$

$$+ \quad \max_{(\mathbf{x}^u, \mathbf{a}^u) : f(\mathbf{x}^u, \mathbf{a}^u) \leq f_u} \quad \left[\lambda^u \cdot \mathbf{x}^u + \mu^u \cdot \mathbf{a}^u \right] \; . \tag{48}$$

Notice that the objective in the first "subproblem" Eq. 46 is very similar to that in Eq. 43, differing only in its constants and an additional linear term. The hardness of this subproblem depends on the structure of h. When h decomposes as: $h(\mathbf{x}, \mathbf{a}, \top; \mathbf{w}) = \sum_c h_c((\mathbf{x}, \mathbf{a})_c, \top; \mathbf{w})$ (and similarly for $\bot$), where $(\mathbf{x}, \mathbf{a})_c$ are subsets of state-action variables, then the subproblem is equivalent to computing the *maximum a posteriori (MAP)* configuration in a *Markov random field (MRF)*. This can be done efficiently in some special cases and can be approximated otherwise, for example, using linear programming relaxations.

The other two subproblems, Eq. 47 and 48, are a generalization of the *knapsack* problem, whenever f is a linear function in $(\mathbf{x}, \mathbf{a})$. In that case, u_i serves as the weight of an item, and λ_i (or μ_i) serves as its value in the knapsack problem. In general, we can have negative weights and costs in this formulation. An easy case is when either the weight or the cost are negative, but not both. If

[22] Other constraints can be added similarly (e.g., one-hot constraints and standard logical consistency constraints, which are typically used for the features in Section 6.1).

the weight is negative and the cost positive, then we can safely set x_i (or a_i) to 1 since we can only gain. On the other hand, if the weight is positive and the cost negative, we can set x_i (or a_i) to 0. This has the effect of reducing the number of variables in the knapsack problem, which leaves the positive-positive or negative-negative weight-cost variables. Those can be handled using standard approximations. For example, a simple greedy bang-per-buck algorithm gives a 2-approximation, and there is also an FPTAS which obtains a $(1+\epsilon)$-approximation.

Also note that we can incorporate one-hot and logical consistency constraints, used in Section 6.1, directly into the knapsack subproblems. In this context it means that we have a generalized knapsack problem, where we need to choose exactly one item in each subset corresponding to the one-hot scope.

Finally, to optimize the above program, we optimize the dual variables λ and ν using subgradient steps, where subgradients are computed by solving the subproblems Eq. 46-48.

6.2.2 Partition-Free Constraint Generation

We can re-write the constraint generation problem Eq. 24 without the need to partition state-action space into bands as follows:

$$\max_{\mathbf{x},\mathbf{a}} \ \sigma\left(f(\mathbf{x},\mathbf{a})\right) h(\mathbf{x},\mathbf{a},\top;\mathbf{w}) + \sigma\left(-f(\mathbf{x},\mathbf{a})\right) h(\mathbf{x},\mathbf{a},\bot;\mathbf{w}) \ ,$$

or equivalently,

$$\max_{\mathbf{x},\mathbf{a},y} \ \sigma\left(y\right) h(\mathbf{x},\mathbf{a},\top;\mathbf{w}) + \sigma\left(-y\right) h(\mathbf{x},\mathbf{a},\bot;\mathbf{w}) \qquad \text{s.t. } y = f(\mathbf{x},\mathbf{a}), \ \ y \in \mathbb{R}.$$

We propose two optimization schemes to obtain an approximate solution for this "partition-free" but now non-linear formulation.

One way to approximate this is by alternating optimization over $(\mathbf{x},\mathbf{a})$, and y. We set y to some initial value, then we maximize w.r.t. $(\mathbf{x},\mathbf{a})$ while holding y fixed. Then set $y = f(\mathbf{x},\mathbf{a})$ and reiterate. This process, however, is very likely to get stuck at a local optimum quite quickly. A simple alternative approach uses the Lagrangian relaxation. Applying this technique as above, we obtain the saddle-point problem:

$$\min_{\lambda}\max_{\mathbf{x},\mathbf{a},y} \ \sigma(y)h(\mathbf{x},\mathbf{a},\top;\mathbf{w}) + \sigma(-y)h(\mathbf{x},\mathbf{a},\bot;\mathbf{w}) - \lambda f(\mathbf{x},\mathbf{a}) + \lambda y \ .$$

To optimize this problem, we can use primal-dual alternating optimization:

1: Initialize $\lambda, \mathbf{x}, \mathbf{a}, y$
2: **for** $t = 1, \ldots, T$ **do**
3: $\quad y^{(t+1)} = y^{(t)} + \eta_t \nabla_y^{(t)}$
4: $\quad (\mathbf{x},\mathbf{a})^{(t+1)} = \text{argmax}_{\mathbf{x},\mathbf{a}} \ h_{y^{(t+1)}}(\mathbf{x},\mathbf{a};\mathbf{w}) - \lambda^{(t)} f(\mathbf{x},\mathbf{a})$
5: $\quad \lambda^{(t+1)} = \lambda^{(t)} - \bar{\eta}_t(y^{(t+1)} - f((\mathbf{x},\mathbf{a})^{(t+1)}))$
6: **end for**

---

where η_t and $\bar{\eta}_t$ are step sizes, and the gradient is:

$$\nabla_y^{(t)} = (h((\mathbf{x},\mathbf{a})^{(t)},\top;\mathbf{w}) - h((\mathbf{x},\mathbf{a})^{(t)},\bot;\mathbf{w}))\frac{e^{-y^{(t)}}}{(1+e^{-y^{(t)}})^2} + \lambda^{(t)} \ .$$

Analyzing the convergence properties of this algorithm is left as future work.

6.3 Multiple Response Variables

In many MDPs for sequential user modeling we may have multiple response variables that are relevant to our objective. For instance, in advertising domains, in addition to click prediction, we may have models that predict various types of conversion events conditional on a click. Similar models arise in generic recommendation tasks, where objectives may involve several different desirable user behaviors (clicks, browsing, purchase, installation, product engagement, revenue, etc.). In such domains, we will have multiple myopic predictive models of user responses, many of which will involve probabilistic prediction using logistic regression.

The techniques we have developed can, in principle, be applied directly to such models. With k logistic response variables, the same discretization techniques above can be used over the k-dimensional prediction space. With the PWC approximation, the generalization from a 1-D grid to a k-D grid is straightforward. The exact search procedure can similarly be modified to handle k dimensions as follows:

- Intervals being searched are now k-dimensional hypercubes, with upper and lower bounds on each of the k response probabilities (or more precisely, their corresponding logits).
- The two suboptimizations for a given interval corresponding to the restriction of $(\mathbf{x}, \mathbf{a})$ pairs to $H^+_{\mathbf{w}}$ and $H^i_{\mathbf{w}}$, respectively, is generalized to account for all combinations of response constraints. More specifically, let vector $\tau \in \{\top, \bot\}^k$ denote a truth assignment to the k response variables (assuming some fixed ordering of the variables). Then for each such τ, we define $H^\tau_{\mathbf{w}}$ as the set of $(\mathbf{x}, \mathbf{a})$ pairs s.t. $h(\mathbf{x}, \mathbf{a}, \tau; \mathbf{w}) \geq h(\mathbf{x}, \mathbf{a}, \tau'; \mathbf{w})$ for all $\tau' \in \{\top, \bot\}^k$. The search is conducted for all 2^k such conditions.
- Termination conditions for any interval are the same as in the 1-D case. Interval splits can be done in a binary fashion (i.e., one dimension at a time), or using multi-way splits over multiple dimensions. Several natural heuristics can be used to determine the choice of dimension(s) for splitting.

The disadvantage of these natural extensions of our one-response-variable algorithms is the exponential increase in complexity with the number of response variables. However, for many user modeling problems we expect that no more than a small handful of responses will be modeled, ensuring that this straightforward approach remains tractable.

There are other domains in which the number of response variables is quite large. One example is the wildfire domain used in the ICAPS 2014 International Probabilistic Planning Competition (IPPC) [46]. In this domain, a forest is divided into cells, and the probability of a fire triggering in one cell is given by a logistic function that depends on the state of all neighboring cells. As such, we have one response variable per cell in the domain. Except for very small problems, such a logistic MDP is unlikely to be feasible using the naive grid-based approach. Other approaches to high-dimensional logistic response variables remain an interesting avenue for future research.

6.4 Extensions to Non-linear Models

Extending the ALP formulation to handle both non-linear response models and non-linear value function approximators. Both are the subject of on-going investigation and we leave details to future

work. But to illustrate the generality of our approach and the prospects for application to non-linear predictors, we consider the case of *wide-and-deep* models for recommendation [16].[23]

In a recommendation context, our aim is to predict a user response (e.g., click or install of a recommended app) given a set of relevant user, item, context and historical interaction features. Wide-and-deep models comprise: a "wide" component, in which certain features (including crosses) are used as inputs to a (final) logistic output unit (exactly as above); and a "deep" component, in which certain features are passed through a DNN with several hidden layers (often with ReLU activations), whose final outputs are also inputs to the final logistic output.[24]

Since wide-and-deep models of this type predict the probability of user responses using a logistic output, both of our approaches to constraint generation (exact search or PWC approximation) can be applied as long as the final input to the logistic function can be encoded in a linear fashion as a function of state-action pairs $(\mathbf{x}, \mathbf{a})$. For various types of activation functions, the output of a (trained) neural network can be encoded within a MIP; this is true, in particular, of ReLU activation functions (generally, one or two Boolean indicator variables are needed per hidden unit) [3]. Thus our approach can be applied directly to response models specified by deep ReLU networks as well as wide-and-deep networks. The computational feasibility of this approach remains to be investigated.

7 Concluding Remarks

We have proposed *logistic MDPs* as a new model for capturing dynamical systems in which state transitions are, in part, governed by a logistic response variable, but in which state dynamics is otherwise factored as is common in DBN models. The model is especially appropriate for modeling user interactions with online advertising and recommender systems, and can leverage both the structure and parameterization of state-of-the-art logistic prediction models commonly used in these domains. We have developed an extension of the usual constraint generation procedure for approximate linear programming that handles the two conceptual and computational bottlenecks introduced by augmented DBNs with a logistic response variable, namely, the breakdown of extreme conditional independence usually present in DBNs, and the induced non-linearity of the dynamics.

We developed two variants of the constraint generation procedure. ALP-SEARCH is an exact procedure that uses linear mixed-integer programming to find maximally violated constraints, but also supports approximation using simple thresholds on degree of violation. ALP-APPROX is an approximation based on a piecewise-constant approximation of the logistic response function. In both cases, the impact of the approximation on solution quality can be bounded. Experiments on large MDPs derived from real-world user response data show that our ALP-APPROX are feasible for approximate policy optimization using generic LP and MIP solvers.

A number of interesting directions remain to be explored. Further experimentation to test the scalability and robustness of the method in a wider variety of domains is of clear interest. A direct comparison of this model-based approach to model-free RL on the same domains should also provide

[23] An open-source implementation of this learning framework and corresponding APIs can be found at `http://tensorflow.org`.

[24] Categorical features in the deep component may be first passed through an embedding layer to create a dense representation to allow suitable generalization.

some valuable insights. With respect to the many extensions of the method we outlined in Section 6, two of the more interesting are: exploring the computational effectiveness and solution quality offered by the LP relaxations proposed for the constraint generation procedure; and further developing and testing the generalization of ALPCG to DNN response models and non-linear value function approximation. Finally, the idea of compiling LP constraints into a compact form [23, 35] rather than relying on constraint generation is an intriguing prospect.

Acknowledgements

Thanks to Ross Anderson for many insightful comments and suggestions provided on this work and to the anonymous reviewers of the conference (IJCAI-17) version of this paper for their valuable comments.

References

[1] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *Journal of Machine Learning Research*, 15(1):1111–1133, 2014.

[2] Kareem Amin, Michael Kearns, Peter Key, and Anton Schwaighofer. Budget optimization for sponsored search: Censored learning in MDPs. In *Proceedings of the Twenty-eighth Conference on Uncertainty in Artificial Intelligence (UAI-12)*, pages 543–553, Catalina, CA, 2012.

[3] Ross Anderson. Optimizing a neural networks forward step with integer programming. Personal Communication, 2016.

[4] Nikolay Archak, Vahab Mirrokni, and S. Muthukrishnan. Budget optimization for online campaigns with positive carryover effects. In *Proceedings of the Eighth International Workshop on Internet and Network Economics (WINE-12)*, pages 86–99, Liverpool, 2012.

[5] Nikolay Archak, Vahab S. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. In *Proceedings of the Nineteenth International World Wide Web Conference (WWW 2010)*, pages 31–40, Raleigh, NC, 2010.

[6] Dimitri P. Bertsekas and John. N. Tsitsiklis. *Neuro-dynamic Programming*. Athena, Belmont, MA, 1996.

[7] Tobias Blask, Burkhardt Funk, and Reinhard Schulte. To bid or not to bid? investigating retailbrand keyword performance in sponsored search advertising. In *International Joint Conference on E-Business and Telecommunications: Revised Selected Papers*, volume 314, pages 129–140. Springer, 2012.

[8] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[9] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1104–1111, Montreal, 1995.

[10] Craig Boutilier and Tyler Lu. Budget allocation using weakly coupled, constrained Markov decision processes. In *Proceedings of the Thirty-second Conference on Uncertainty in Artificial Intelligence (UAI-16)*, pages 52–61, New York, 2016.

[11] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 33–42, Madison, WI, 1998.

[12] T. Chandra, E. Ie, K. Goldman, T. L. Llinares, J. McFadden, F. Pereira, J. Redstone, T. Shaked, and Y. Singer. Sibyl: A system for large scale machine learning. 2010.

[13] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):61:1–34, 2015.

[14] Olivier Chapelle and Ya Zhang. A dynamic Bayesian network click model for web search ranking. In *Proceedings of the Eighteenth International World Wide Web Conference (WWW-09)*, pages 1–10, Madrid, 2009.

[15] Moses Charikar, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. On targeting Markov segments. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC-99)*, pages 99–108, Atlanta, 1999.

[16] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, Boston, 2016.

[17] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, Boston, 2016.

[18] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, 2003.

[19] Daniela Pucci De Farias and Benjamin Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[20] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

[21] Jon Feldmann, S. Muthukrishnan, Martin Pál, and Cliff Stein. Budget optimization in search-based advertising auctions. In *Proceedings of the Eighth ACM Conference on Electronic Commerce (EC'07)*, pages 40–49, San Diego, 2007.

[22] Thore Graepel, Joaquin Q. Candela, Thomas Borchert, and Ralf Herbrich. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *Proceedings of the Twenty-seventh International Conference on Machine Learning (ICML-10)*, pages 13–20, Haifa, Israel, 2010.

[23] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[24] Ruining He and Julian McAuley. Fusing similarity models with Markov chains for sparse sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM-16)*, Barcelona, 2016.

[25] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 279–288, Stockholm, 1999.

[26] Henning Hohnhold, Deirdre O'Brien, and Diane Tang. Focusing on the long-term: It's good for users and business. In *Proceedings of the Twenty-first ACM International Conference on Knowledge Discovery and Data Mining (KDD-15)*, pages 1849–1858, Sydney, 2015.

[27] Kristian Kersting, Martijn Van Otterlo, and Luc De Raedt. Bellman goes relational. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML-04)*, pages 59–66, Banff, Alta., 2004.

[28] Ting Li, Ning Liu, Jun Yan, Gang Wang, Fengshan Bai, and Zheng Chen. A Markov chain model for integrating behavioral targeting into contextual advertising. In *Proceedings of the Third International Workshop on Data Mining and Audience Intelligence for Advertising (ADKDD-09)*, pages 1–9, Paris, 2009.

[29] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. A convolutional click prediction model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM-14)*, pages 1743–1746, Melbourne, 2015.

[30] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Arnar Mar Wattenberg, Martin Hrafnkelsson, Tom Boulos, and Jeremy Kubica. Ad click prediction: A view from the trenches. In *Proceedings of the Nineteenth ACM International Conference on Knowledge Discovery and Data Mining (KDD-13)*, pages 1222–1230, Chicago, 2013.

[31] Biswanath Panda, Joshua S Herbach, Sugato Basu, and Roberto J. Bayardo. Mapreduce and its application to massively parallel learning of decision tree ensembles. In Ron Bekkerman, Mikhail Bilenko, and John Langford, editors, *Scaling Up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press, 2011.

[32] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

[33] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized Markov chains for next-basket recommendation. In *Proceedings of the 19th International World Wide Web Conference (WWW-10)*, pages 811–820, Raleigh, NC, 2010.

[34] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *Proceedings of the Sixteenth International World Wide Web Conference (WWW-07)*, pages 521–530, Calgary, AB, 2007.

[35] Philipp Robbel, Frans A. Oliehoek, and Mykel J. Kochenderfer. Exploiting anonymity in approximate linear programming: Scaling to large multiagent mdps. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2537–2543, Phoenix, 2016.

[36] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. A hidden Markov model for collaborative filtering. *Management Information Systems Quarterly*, 36(4), 2012.

[37] Scott Sanner and Craig Boutilier. Practical solution techniques for first-order MDPs. *Artificial Intelligence*, 173(5):748–788, 2009.

[38] Dale Schuurmans and Relu Patrascu. Direct value approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14 (NIPS-2001)*, pages 1579–1586, Vancouver, 2001.

[39] Guy Shani, David Heckerman, and Ronen I. Brafman. An MDP-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005.

[40] David Silver, Leonard Newnham, David Barker, Suzanne Weller, and Jason McFall. Concurrent reinforcement learning from customer interactions. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 924–932, Atlanta, 2013.

[41] Robert St-Aubin, Jesse Hoey, and Craig Boutilier. APRICODD: Approximate policy construction using decision diagrams. In *Advances in Neural Information Processing Systems 13 (NIPS-2000)*, pages 1089–1095, Denver, 2000.

[42] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. Usage-based web recommendations: A reinforcement learning approach. In *Proceedings of the First ACM Conference on Recommender Systems (RecSys07)*, pages 113–120, Minneapolis, 2007. ACM.

[43] Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 17–22, Boston, 2016.

[44] Georgios Theocharous, Philip S. Thomas, and Mohammad Ghavamzadeh. Personalized ad recommendation systems for life-time value optimization with guarantees. In *Proceedings of the Twenty-fourth International Joint Conference on Artificial Intelligence (IJCAI-15)*, pages 1806–1812, Buenos Aires, 2015.

[45] Ilya Trofimov, Anna Kornetova, and Valery Topinskiy. Using boosted trees for click-through rate prediction for sponsored search. In *Proceedings of the Sixth International Workshop on Data Mining for Online Advertising and Internet Economy*, page 2, 2012.

[46] Mauro Vallati, Lukas Chrpa, Marek Grześ, Thomas Leo McCluskey, Mark Roberts, and Scott Sanner. The 2014 international planning competition: Progress and trends. *AI Magazine*, 36(3):90–98, 2015.

[47] Chenggang Wang, Saket Joshi, and Roni Khardon. First-order decision diagrams for relational MDPs. *Journal of Artificial Intelligence Research*, 31:431–472, 2008.

[48] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J. Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM-17)*, Cambridge, UK, 2017. To appear.

[49] Yinyu Ye. The simplex and policy-iteration methods are strongly polynomial for the Markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36(4):593–603, 2011.