Budget Allocation using Weakly Coupled, Constrained Markov Decision Processes[*]

Craig Boutilier
Google
cboutilier@google.com

Tyler Lu
Google
tylerlu@google.com

May 6, 2016

Abstract

We consider the problem of budget (or other resource) allocation in sequential decision problems involving a large number of concurrently running sub-processes, whose only interaction is through their gradual consumption of budget (or the resource in question). We use the case of an advertiser interacting with a large population of target customers as a primary motivation. We consider two complementary problems that comprise our key contributions.

Our first contribution addresses the problem of computing MDP value functions as a function of the available budget. In contrast to standard *constrained MDPs*—which find optimal value functions given a *fixed* expected budget constraint—our aim is to assess the tradeoff between expected budget spent and the value attained when using that budget optimally. We show that optimal value functions are concave in budget. More importantly, in the finite-horizon case, we show there are a finite number of *useful* budget levels. This gives rise to piecewise-linear, concave value functions (piecewise-constant if we restrict to deterministic policies) with an representation that can be computed readily via dynamic programming. This representation also supports natural approximations. Our model not only allows the assessment of budget/value tradeoffs (e.g., to find the "sweet spot" in spend), but plays an important role in the allocation of budget across competing subprocesses.

Our second contribution is a method for constructing a policy that prescribes the *joint* policy to be taken across all sub-processes given the joint state of the system, subject to a global budget constraint. We cast the problem as a *weakly coupled MDP* in which budget is allocated online to the individual subprocesses based on its observed (initial) state and the subprocess-specific value function. We show that the budget allocation process can be cast as a *multi-item, multiple-choice knapsack problem (MCKP)*, which admits an efficient greedy algorithm to determine optimal allocations. We also discuss the possibility of online, per-stage *re-allocation* of budget to adaptively satisfy *strict* rather than expected budget constraints.

[*]This paper is an extended version of a paper by the same title to appear in the *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence (UAI-16), New York, 2016.*

1 Introduction

Markov decision processes (MDPs) have found wide application throughout AI and provide firm foundations for decision-theoretic planning [10] and reinforcement learning [27]. In many domains, however, actions have costs or otherwise consume limited resources—in such cases, optimal policies must be constructed subject to resource constraints, a problem often best formulated using *constrained MDPs (CMDPs)* [2]. The complexity of these problems is exacerbated in domains where multiple independent or semi-independent MDPs must be controlled jointly, since the state and action spaces for the joint process typically consists of the cross-product of those from the individual subprocesses [32].

A key setting where both of these characteristics arise is online advertising. A number of models of online user engagement have been proposed that treat user behavior as a Markov process [15, 7]. Archak *et al.* [6], in particular, propose a constrained MDP model for the optimal allocation of an advertiser's budget over an extended horizon that captures the sequential, cumulative, stochastic effect of multiple advertising touch points on a user's behavior. Their model assumes that a fixed budget is predetermined for each user, and focuses on how to optimally advertise to a user, subject to that budget constraint. However, this formulation is not well-suited to advertisers who (a) have a global budget constraint (rather than a per-user constraint); and (b) are free to allocate that budget differentially and *adaptively* across users of different types, or users in different states of the underlying MDP. We instead model this as a joint optimization problem in which a *joint policy* across all users must be constructed. In particular, we consider the *optimal, global, dynamic allocation of budget* across all users in the target market, removing the need to pre-specify a per-user budget.

Using this problem as motivation, we consider a natural decomposition of this type of resource allocation problem as a *weakly coupled MDP* [32]. In this model, each "local" MDP (corresponding to a customer or customer type) is solved separately, and the solutions composed into a joint policy. Within this setting, our first contribution relates to the solution of the local MDPs themselves. One standard approach to dealing with resource constraints, such as budget limits, is CMDPs [2], in which actions consume one or more resources (e.g., budget, power) and optimal policies are sought that use no more than some pre-defined level of each resource in expectation. CMDPs have been applied in online advertising (see above), robotics, and other domains [12, 19, 18]. While CMDPs are valuable models, one weakness is their required *a priori* specification of fixed "budget" or limit on each resource—in our example, this would be, say, a daily monetary budget cap on ad spend for each customer or across a specific ad network [7, 6, 4]. However, in many cases, determining a suitable budget depends on the tradeoff between ad spend and expected value. In our motivating example, in particular, the budget to be allocated to customers of a specific type, or in a specific MDP state, must be traded off against that allocated to other types or states.

We propose a model called *budgeted MDPs (BMDPs)* in which, like CMDPs, actions consume resources and can only be taken when the required resources are available. Unlike CMDPs, we do not place an *a priori* limit on resources, but instead solve the *for all possible resource levels*, allowing one to explore the tradeoff between (optimal) expected value and allocated resources. Note that we could formulate this problem by embedding the available resource levels in the state space, creating a "hybrid state" with one continuous dimension per resource. However, typically one *chooses* a budget (or appropriate resource level) when initiating a policy—in our approach, *after* solving the budgeted MDP to assess value/budget tradeoffs. As a consequence, we instead treat the budget as a separate parameter of the value function (VF) and policy, and analyze the structure of optimal VFs w.r.t. budget. Specifically, for finite state and action BMDPs, we show that for any fixed state s: (a) the optimal VF $V(s, b)$ is concave in available budget b; and (b) the optimal VF for any finite horizon is piecewise-linear and concave (PWLC) in budget, and is defined by a finite set of *useful* resource levels (the VF is piecewise constant if policies are deterministic). We present a dynamic programming (DP) algorithm to compute this PWLC VF representation, one that readily supports approximation. This approach is well-suited not only to multi-task problems, like our ad domain, but more generally to any constrained MDP where the appropriate level of resource to allocate depends on the value obtainable (e.g., the "sweet spot" in spend) and may not be easily determined *a priori*. Using the VF to inform the selection of an appropriate budget, one automatically determines the optimal policy for that (expected) spend level—as if one had solved the corresponding CMDP for that budget.

Our second contribution is a method for piecing together the solutions of the individual BMDPs to determine a joint policy, subject to a global resource/budget constraint. Since the MDP is *weakly coupled* [32]—specifically, the individual customer MDPs evolve independently, linked only through the consumption of shared budget—our primary aim is to determine an appropriate allocation of budget to each customer, which is turn dictates the optimal policy for that customer. We show that, given the form of BMDP optimal value functions, the budget allocation problem can be formulated as a *multiple-choice knapsack problem (MCKP)* [40], for which a straightforward greedy optimization can be used to construct the optimal online allocation of budget. We also discuss circumstances in which the dynamic, per-stage *reallocation* of budget may be valuable—in such cases, the offline solution of the underlying BMDPs and the greedy nature of the online allocation algorithm admit fast, real-time response.

The paper is organized as follows. We discuss related work on ad budget optimization and constrained and weakly coupled MDPs in Sec. 2. We outline our basic constrained weakly coupled MDP model in Sec. 3. In Sec. 4, we introduce *budgeted MDPs (BMDPs)* as a model for the local (e.g., single-customer) MDPs that make up the weakly coupled model. Unlike CMDPs, the aim is to determine VFs and policies *as a function of available budget*. In this section we describe the PWLC structure of optimal VFs for BMDPs; introduce a DP algorithm for computing VFs and policies; describe methods for approximation and provide an analysis of approximation error; describe how to compute the variance in expected spend induced by the optimal policy; and provide empirical results on both synthetic MDPs and MDPs derived from the data of a large advertiser. In Sec. 5, we describe how to exploit BMDP solutions to allocate a global budget to multiple independent local MDPs, linked only by their budget consumption. We formulate the problem as a multiple-choice knapsack problem (MCKP) and describe a simple greedy algorithm for optimal budget allocation. We show that this way of exploiting BMDP solutions provides much better results than adopting a static or fixed per-user budget. We also propose a *dynamic budget reallocation* method that reallocates budget as each CMDP process evolves. This reduces variance in the global spend, ensures guaranteed budget constraint satisfaction rather than expected satisfaction and, in settings with tightly constrained budgets, can offer better expected value through budget redistribution.

2 Background and Related Work

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

2.1 Budget Optimization in Online Advertising

Optimal allocation of advertising budgets is addressed at length in the marketing literature. Traditional concerns include how to allocate budget to different segments of the target market [41], and exploiting the different stochastic responses predicted from such segments [26]. Budget allocation is often modeled as the control of a continuous dynamical system [21], where audience response (e.g., purchases) to control actions (e.g., ad spending) are used to justify policies such as *pulsing*. User behavior is also sometimes modeled as a discrete Markov process (reflecting consumer demand, ad response, ad carryover effects, etc.) [12, 34].

A variety of mechanisms to support online advertising have been studied (and successfully deployed), including auctions for sponsored search [44, 20]. In this context, advertiser budget optimization is critical to maximizing impact and ROI in keyword auctions, leading to a variety of proposals for optimal bidding strategies under various models and constraints, including budget constraints [8, 23, 33, 24, 17, 28]. Even more expressive methods (e.g., for entire ad campaigns) have been proposed for both search and display advertising in online settings [11, 29].

By contrast, relatively little work has considered budget optimization in an online setting that is sensitive to user behavior as it *extends over time*. 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.* [6] tackle a related problem, offering a somewhat different model, but still assuming Markovian user behavior (here 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 an *a priori* fixed budget *for that user*. It is this model we adopt for our approach below, though we do not fix a per-user budget. Interestingly, the same authors [7] 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 [6]). Amin *et al.* [4] develop an MDP model of the budget optimization process itself, formulating the problem facing the online advertiser—who is uncertain about the winning bid distribution—as one of learning under censored observations. Unlike the models above (and unlike our work), this work does not model Markovian *user* behavior.

Li *et al.* [30] also consider Markov models in behavioral and contextual advertising. More recent work has considered the application of reinforcement learning methods to determine appropriate advertising and offer strategies to large populations of customers, learning optimal policies from data generated by online interactions with users [38, 43]. These methods are similar in their underlying user modeling assumptions, but attempt to learn policies from data and do not account for budget constraints or tradeoffs, whereas we assume an underlying transition model is available and optimize policies relative to these budget tradeoffs.

2.2 Constrained MDPs

An important extension of the standard MDP model is offered by *constrained MDPs (CMDPs)* [2], in which actions consume one or more resources (e.g., monetary budget, power) and optimal policies are sought that use no more than some pre-defined level of each resource in expectation. CMDPs have been applied in online advertising, mobile robotics, and other domains within AI and OR [12, 19, 18], and techniques that exploit the structure of CMDPs for effective solution have received some attention [2, 19, 18]. We briefly define MDPs and the basic (one-dimensional) constrained extension.

2.2.1 Markov Decision Processes

A finite state and action *Markov decision process (MDP)* [36] is given by: a finite state space S and action space A; a stochastic transition model P, with $P(i, a, j) = p^a_{ij}$ denoting the probability of a transition from state i to j when action a is taken; and a bounded, real-valued reward function $R(i, a) = r^a_i$ denoting the immediate (expected) reward of taking action a at state i. We decompose reward as $r^a_i = U(i, a) - C(i, a) = u^a_i - c^a_i$, where *cost function* C reflects action costs and *utility function* U reflects the desirability of the state-action pair.[1]

We define the *reachable set* for state-action pair i, a to be $S^a_i = \{j \in S | p^a_{ij} > 0\}$ and the *maximum out-degree* of an MDP to be $d = \max_{i\in S, a\in A} |S^a_i|$. A *stationary, deterministic* policy π specifies an action $\pi(i) \in A$ to be taken in any state i. A *randomized* policy π specifies a distribution $\pi(i) \in \Delta(A)$ over which action to be taken at state i.

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

$$V^*(i) = \max_{a\in A} r^a_i + \gamma \sum_{j\in S} p^a_{ij} V^*(j),$$

while the optimal policy π^* satisfies:

$$\pi^*(i) = \operatorname*{argmax}_{a\in A} r^a_i + \gamma \sum_{j\in S} p^a_{ij} V^*(j).$$

It is sometimes convenient to use *Q-functions*:

$$Q^*(i, a) = r^a_i + \gamma \sum_{j\in S} p^a_{ij} V^*(j),$$

and to write $V^*(i) = \max_{a\in A} Q^*(i, a)$. The optimal value function and policy can be computed using dynamic programming algorithms like value or policy iteration, or using natural linear programming formulations [36, 10].

It is also useful to specify the value associated with an arbitrary policy π:

$$V^\pi(i) = r^{\pi(i)}_i + \gamma \sum_{j\in S} p^{\pi(i)}_{ij} V^\pi(j).$$

[1] U is often independent of a.

If π is randomized, $r_i^{\pi(i)}$ and $p_{ij}^{\pi(i)}$ denote expected reward and transition probability with respect to $a \sim \pi(i)$.

2.2.2 Constrained MDPs

Constrained MDPs (CMDPs) extend the classic MDP model by introducing constraints on the resources that can be used to implement the target policy [2]. For ease of exposition, we outline CMDPs using a one-dimensional cost model, but note that the general CMDP model is typically specified using multiple resource types with constraints. A (finite state and action) CMDP consists of an MDP specified as above, along with a *budget constraint* B that limits the *total expected discounted cost* of allowable policies. For example, if the cost component c_i^a of our reward function reflects the monetary cost of taking action a in state i, the budget constraint B requires that any policy have an expected discounted monetary cost of no more than B. The model, of course, applies to the use of other limited resources (e.g., power, CPU, even time).

Satisfying the budget constraint *in expectation* is quite standard in the CMDP literature. This approach is suitable in domains where the policy may be executed many times in a variety of circumstances (and the budget is fungible over these instances). It is also appropriate when some flexibility exists to exceed the budget. Our motivating ad scenario is such a domain. In other settings, where the budget constraint is *strict*, expected budget satisfaction may be unsuitable—our model below easily accommodates strict budgets as well. However, we focus on expected constraint satisfaction in this work. In some domains, it may be useful not to discount costs in the budget constraint. This can be problematic in some CMDP formulations, though it is easily handled in our approach, as we discuss below.

Some CMDP models use costs as a constraint on behavior only, and do not factor costs into the reward function. In settings where actions have monetary costs (such as the ad domain), one typically assumes commensurability of C and U and deducts action costs from the expected return (e.g., revenue from conversions). In other cases, costs (e.g., power available) are not commensurable with return and simply serve as a constraint. Everything that follows applies whether the reward R incorporates cost C or only depends on utility U.

Since assessing expected cost requires some initial distribution α over states—where α_i denotes the probability that the decision process begins at state i—we take this as input as well. The *expected discounted cost* of policy π at state i is given by:

$$C^\pi(i) = c_i^{\pi(i)} + \gamma \sum_{j \in S} p_{ij}^{\pi(i)} C^\pi(j).$$

This cost can be discounted at a different rate than the reward function if desired, and can even be undiscounted in finite horizon problems (or in settings with absorbing states, where total cost can be bounded). We consider both discounted and undiscounted budgets in below.

The *optimal stationary policy* for a CMDP is given by:

$$\operatorname*{argmax}_{\pi} \ \alpha_i V^\pi(i) \ \text{ s.t. } \ \alpha_i C^\pi(i) \leq B.$$

The optimal solution (both the policy and VF) can be computed in polynomial time using a linear programming formulation—see Altman [2] for an overview of key results regarding CMDPs. We note that there always exists an optimal stationary policy for a CMDP, but unlike in the case of unconstrained MDPs, there may be no deterministic optimal policy (i.e., any optimal policy must be stochastic). We also note that the policy is not generally *uniformly optimal*; in other words, the optimal policy varies with the initial distribution α. This stands in contrast to unconstrained MDPs, for which there is always a uniformly optimal stationary, deterministic policy.

2.2.3 Drawbacks of CMDPs

While CMDPs are valuable models, one weakness is their required *a priori* specification of fixed "budget" or limit on each resource. For example, in an MDP model of sequential online advertising, an advertiser might wish to specify a daily monetary budget cap on ad spend for each customer or across a specific ad network [7, 5, 4]. However, *the determination of a suitable budget* depends on the tradeoff between ad spend and expected value. More generally, determining the level of resources to allocate to a specific problem depends on the solution quality that can be achieved with that allocation. For instance, before solving the relevant CMDPs, an ad campaign manager cannot generally know that (say) an expected spend of \$10 per user yields an expected return of \$20, while a spend of \$15 has a return of \$28. The latter offers greater absolute return while the former has a better return on investment (ROI). Assessing the tradeoff so a suitable budget can be set is not typically viable *a priori*.

Moreover, in many certain circumstances, limited resources must be shared across *multiple tasks*, each represented by a distinct MDP [32, 39, 9]. In such a case, determining the value attainable in each MDP for *any* specific level of resource allocation is crucial to assessing the tradeoffs *across* the task-specific MDPs. In our ad domain, for instance, we may have two classes of users, some early in the sales/search funnel, and others at states later in the funnel: determining how to interact with each class given an overall budget constraint requires assessing the tradeoff between expected value derived from each class given different class-specific engagement policies and their budget requirements. The methods we introduce, by constructing optimal policies for *any budget made available*, readily support the assessment of such tradeoffs.

A second property of CMDPs that is sometimes problematic is their reliance on randomized policies. The following simple example illustrates why CMDPs generally require randomized (stationary) policies, and why such policies may be undesirable. Consider a trivial one-state, two-action deterministic CMDP with actions a and b, where $C(a) = 1, C(b) = 0, U(a) = 10, U(b) = 1$, and $\gamma = 0.9$. Suppose our discounted resource/budget limit is 1.9. If action a is to be taken in a stationary policy, the policy must be randomized, otherwise the policy will consume more than the allowable expected budget. Indeed, the optimal stationary policy takes action a with probability 0.19 (b with 0.81) and has an expected value of $27.1 = \frac{0.19(10)+0.81(1)}{1-\gamma}$. The *only* feasible deterministic stationary policy takes action b, with a suboptimal expected value of $10 = \frac{1}{1-\gamma}$. However, the same value 27.1 *can* be achieved with a deterministic policy as long as we allow *nonstationarity*—by taking action a twice, then repeating action b thereafter. However, if we allow the policy to depend on the budget, then we need not rely on nonstationarity. Instead we can simply allow that a be taken if enough discounted budget is available. In this case, a will be taken twice before the budget is exhausted. This shows that—while imposing determinism on a policy that is stationary *relative to the original state space* can lead to lower expected value than a randomized policy[2]—allowing policy choice to depend on the budget remaining gives rise to much more natural stationary deterministic policies. In our model, stochastic policies are needed only to deal with budget feasibility, not to handle artificial stationarity requirements.

Finally, the reliance of CMDP solutions on the initial distribution is also a drawback in certain circumstances. For instance, suppose an advertiser wishes to deploy a policy for engaging customers online, but the initial state distribution (i.e., expected starting point of the engagement) depends on the source of customer traffic. Having a value function and policy that is easily adaptable to different initial distributions is often more desirable than computing a separate policy for each initial distribution. The techniques we outline below can be used for this purpose as well.

2.3 Weakly Coupled MDPs

The decomposition of MDPs into independent or semi-independent processes can often be used to mitigate the curse of dimensionality by reducing the size of the MDP state and action space. Challenges lie in discovering a suitable decomposition structure and in determining how best to use the sub-process solutions to construct a (generally approximate) global solution.

There have been a number of approaches developed that have this flavor, in both standard MDPs and *decentralized (DEC)* MDPs and POMDPs [39, 39, 1, 42, 31, 19, 45, 3, 35, 14]. The approach most related to ours is the decomposition method for *weakly-coupled MDPs* of Meuleau *et al.* [32]. In their model, a joint fully observable MDP is comprised of a number of almost completely independent subprocesses, each itself a fully observable MDP (a *local* MDP). Each MDP reflects the task or objectives of a specific agent, but the local policies themselves require resources, both consumable (e.g., fuel or money, that, once used in one local policy, are unavailable for future use in that or any other local MDP), and non-consumable (e.g., equipment, that can be reused multiple times and swapped across local MDPs, but can only be used in one local MDP at a time). The method of Meuleau *et al.* [32] solves the local MDPs independently to produce local value functions and policies that are parameterized by the resources available. these local value functions then inform a simple greedy algorithm that assigns resources to each local MDP. Finally, unconsumed resources at each stage of the process are reassigned depending on the updated observed state of the local MDPs.

Our approach to budget decomposition is similar, but has some key differences. We use the more standard *expected budget* constraint, rather than a guaranteed budget constraint to determine the optimal policy—this requires the development of new DP techniques (as opposed to the use of standard value or policy iteration [32]). We also show that our resource (budget) allocation algorithm is optimal. Our dynamic budget reallocation scheme is derived from the reallocation mechanism of Meuleau *et al.* [32].

[2]Indeed, computing the optimal deterministic policy is computationally more difficult than the polytime problem of computing an optimal randomized policy—it is NP-complete [22] and can be formulated as a MIP [18].

3 MDPs for Large-scale Budget Allocation

We begin by outlining a simple MDP model for engagement with users from a large, heterogeneous population, using a sequence of potentially costly actions. We use direct-response advertising as our main motivating example, though our techniques apply to control of any large, distributed collection of fully observable MDPs where actions consume limited resources. We note that our model abstracts away a number of factors that arise in realistic ad domains (e.g., partial or intermittent observability of user responses, hidden user state, coarse-grained control and lagging effects, incentives and game-theoretic interactions) in order to focus on the essence of budget allocation.

We assume an *advertiser* has a fixed budget with which to influence a target population of *users* though the use of various *advertising actions*. Each action taken by the advertiser has a specific cost and is targeted to a specific user (e.g., a textual or graphical ad in response to a search query, an in-app or web page graphical/video ad, direct mail to a household).[3]

Users respond to advertising actions stochastically, and in a way that may depend not only on user features (e.g., demographics), but also on past actions to which they have been subjected (e.g., previous ad exposures or solicitations) and past responses (e.g., click through or purchase behavior). Following Archak *et al.* [6], we model this situation as an MDP in the following fashion. We assume a finite set of *users* $i \leq M$. Generally speaking, users may be segmented into *types* that reflect static, observable characteristics that influence their response to advertising. For ease of exposition, we assume all users have the same type; but the extension to multiple types is straightforward, and aspects of the model requiring modification for multiple types will be indicated as they arise.

We assume a finite set S of *user states* $j \leq N$. At any time, user i is some state $s[i] \in S$. Let $\mathbf{S} = S^M$ be the *joint state space*, with the joint state of all users denoted $\mathbf{s} = \langle s[1], \ldots, s[M] \rangle \in \mathbf{S}$. Intuitively, the user state captures all relevant user characteristics and history that could alter her response to an advertising action. S could be relatively small or may be quite large in certain contexts (e.g., the most recent keyword search on which the advertiser bid, or some sufficient statistics summarizing historical interactions and user responses). A finite set A of *advertising actions* is available: for ease of presentation, we assume that any $a \in A$ can be applied to a user in any state (subject to cost constraints, discussed below). Accommodating state-dependent constraints on the feasible action set is straightforward (as in any MDP model). At each stage, the advertiser selects an action $a[i]$ to apply to user i. Letting $\mathbf{A} = A^M$, a joint action is $\mathbf{a} = \langle a[1], \ldots, a[M] \rangle \in \mathbf{A}$.[4]

The stochastic response of a user to an action is captured by the *transition model* $P : S \times A \to \Delta(S)$, with $P(s, a, t)$ denoting the probability that a user in state s moves to state t when subjected to action a.[5] The *reward model* $R(s, a)$ reflects the costs and payoffs when an advertiser applies action a to a user in state s. We decompose reward as $R(s, a) = U(s) - C(s, a)$: *cost model* C reflects action costs (e.g., cost of placing an ad, or potential annoyance factor and associated lost revenue); and *utility function* U reflects benefits/payoffs (e.g., revenue from sales, value of brand exposure, etc.).

[3]We can also apply the "same" action to multiple users simultaneously in broadcast fashion by assuming the set of users so targeted can be identified. If the costs of such simultaneous actions are not linear, our decomposition below is only approximate.

[4]If we have multiple user types with distinct MDPs, the joint state and action spaces are defined analogously.

[5]If the response depends on user type, we define separate transition models for each type. Embedding the user type in the state is not needed, since the state sets for different types are non-communicating.

The advertiser has a maximum *(global) budget* B that can be spent over the planning horizon on all (potential or existing) users. We assume that the population of users is either known, or new users enter the system according to a known distribution over initial states. Because user transitions are stochastic, and because they may enter the system at different times, different users will occupy a variety of states at any point in time. The question we address: how should we engage with users at different states? Hence, how should we "allocate" our budget to such users?

The optimal policy is defined w.r.t. the *joint constrained MDP*, with state space $\mathbf{S} = S^M$, action set $\mathbf{A} = A^M$, a (budget-independent) transition model, and cost, utility and reward functions defined as follows:

$$P(\mathbf{s}, \mathbf{a}, \mathbf{t}) = \prod_{i \le M} P(s[i], a[i], t[i]); \quad U(\mathbf{s}) = \sum_{i \le M} U(s[i]);$$
$$C(\mathbf{s}, \mathbf{a}) = \sum_{i \le M} C(s[i], a[i]); \quad R(\mathbf{s}, \mathbf{a}) = U(\mathbf{s}) - C(\mathbf{s}, \mathbf{a}).$$

The joint transition model reflects the natural independence of user transitions (i.e., states/actions taken for one user don't influences the transitions of another). Costs and utilities are additive across users.

Our aim is to find a policy, i.e., mapping from joint states to actions, that maximizes expected discounted reward subject to the budget constraint: in expectation, the policy should spend no more than B. The optimal solution to this joint CMDP can, in principle, be found by linear programming or dynamic programming as discussed above. However, the state and action spaces of the joint MDP is extremely large, exponential in the number of active users (i.e., $O(N^M)$ states). There are some simplifications that can be made that exploit symmetries. For example, the optimal action at state s does not depend on the "identities" of users in a specific state $s \in S$; since all users in s are identical, the state can be simplified by considering only the number of users in each state, reducing state space size to $\binom{M+N-1}{N-1}$, which can be much more manageable if M is small. Still an MDP of this size will generally be intractable.

Since the MDP is weakly coupled in the sense described above [32], we would like to take advantage of this fact and solve the individual component MDPs in such a way that their solutions can be effectively "pieced together" to form an approximately optimal solution to the joint problem. We tackle this problem in two stages: we first show how to solve the "customer-level" subprocess MDPs to derive optimal value functions and policies as a *function of expected budget*. This allows us to consider the precise tradeoff between expected budget consumption and expected value. This is critical to the second stage of the process, in which we use these value functions to optimally allocate budget online to users in different states of their corresponding MDPs.

4 Solving Budgeted MDPs

We first formulate *budgeted MDPs*, a variant of CMDPs in which budgets (or other resources) are modeled so that decisions can be made readily about the level of budget to provide. The budget is implicitly treated as a component of the state, and value functions and policies are constructed that vary with both the state *and available budget*. We first outline the model, and then develop key intuitions regarding VF structure by considering deterministic policies. We move on to discuss stochastic policies—where randomization is much more natural than in the CMDP model—and show that these give rise to value functions that have computationally useful structural properties. This structure can be exploited in a simple dynamic programming (DP) algorithm, and gives rise to natural approximations. We conclude the section with some computational experiments to show the efficacy of our DP method.

4.1 The Budgeted MDP Model

A *(finite, one-dimensional) budgeted Markov decision process (BMDP)* $M = \langle S, A, P, U, C, B_{\max}, \gamma \rangle$ has the same components (states, actions, transitions, utility, cost, discount factor) as a CMDP, but without a budget constraint. We allow an optional constant $B_{\max}$ that reflects some plausible upper bound on the useful or available budget that can be exploited at any point in the decision process.[6] We abuse notation and write $U(i) = u_i$ for the terminal utility at state i if no action is taken (e.g., end of the planning horizon). Finally, we assume that, for each state i, there is some action a s.t. $c_i^a = 0$. This ensures we can define a proper policy when no budget is available.[7]

As in CMDPs, an optimal policy is sought which maximizes expected cumulative discounted reward over some horizon of interest, but which consumes no more than some allocated budget $b \leq B_{\max}$ in expectation. Unlike CMDPs, however, we want an optimal policy that is a *function* of b, rather than a single optimal policy for a fixed budget B.

4.2 Deterministic Policies

We begin with an analysis of deterministic policies for finite-horizon problems, which provide useful intuitions upon which we build below.

We first define the *optimal deterministic t-stage-to-go value function*. For all $i \in S, b \leq B_{\max}$, define $V_D^0(i,b) = u_i$, and:

$$V_D^t(i,b) = \max_{\substack{a \in A \\ \mathbf{b} \in \mathbb{R}_+^n}} r_i^a + \gamma \sum_{j \leq n} p_{ij}^a V_D^{t-1}(j, b_j) \tag{1}$$

$$\text{subj. to } c_i^a + \gamma \sum_{j \leq n} p_{ij}^a b_j \leq b. \tag{2}$$

The optimal policy $\pi_D^t(i,b)$ is given by substituting $\operatorname{argmax}_{a \in A}$ for $\max_{a \in A}$ in Eq. 1. The value function reflects the fact that taking action a at state i consumes c_i^a of the available budget b, while the optimal budget consumption at each reachable next state must be such that the *expected* budget used subsequently is no more than the budget $b - c_i^a$ remaining after taking a.[8] Notice that we discount the

[6]For finite-horizon problems, or MDPs with zero-cost absorbing states, we can forego $B_{\max}$ in our algorithms. In infinite-horizon problems, $B_{\max}$ can also be ignored if costs are discounted. However, since we wish to also consider the undiscounted cost case, we allow specification of such a cap.

[7]If the process ends when budget is exhausted, this can be modeled by a zero-cost "terminating" action that moves the process to a zero-cost absorbing state.

[8]While a constraint on expected budget is typical in CMDPs, the variant in which the budget cap must be respected in every realization of the MDP dynamics can also be defined easily, and may be more suitable when budgets are strict, or for resources other than monetary budget. Strict budgets can easily be handled by DP [32].

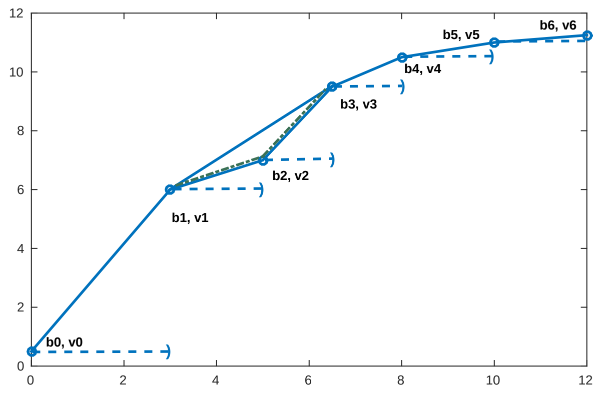


Fig. 1: An example value function $V_D(i, b)$ for a fixed state i. The deterministically useful budget levels are shown (the points (b, v) along with: (a) the induced piecewise constant VF $V_D(i, b)$ (the dashed line) for a deterministic policy; and (b) the PWLC VF (the solid line) for a randomized policy. Notice that randomizing between b_0, b_1 offers greater expected value for any expected spend $b \in (b_0, b_1)$ than deterministically using b_0; and that randomizing between b_1 and b_3 with expected spend b_2 dominates the deterministic policy value at b_2 (see dashed-dotted lines).

expected spend as is standard in CMDPs; but removing the discount factor γ from the constraint Eq. 2 is sensible in settings where budgets are strict and are allocated *a priori* once values are determined.[9]

It is easy to see that $V_D^t(i, b)$ is monotone non-decreasing in b. While the optimal VF in the BMDP formulation involves a continuous dimension b, it has a concise finite representation. For a fixed stage-to-go t and state i, define budget $0 \leq b \leq B_{\max}$ to be *(deterministically) useful* iff $V_D^t(i, b) > V_D^t(i, b - \varepsilon)$ for all $\varepsilon > 0$, and *useless* otherwise. We observe that:

Proposition 1. *For any finite t and state i, $V_D^t(i, \cdot)$ has a finite number of deterministically useful budget levels.*

Prop. 1 implies that $V_D^t(i, \cdot)$ is *piecewise constant* and monotone, with $V_D^t(i, b) = V_D^t(i, b_k^{i,t})$ for all $b \in [b_k^{i,t}, b_{k+1}^{i,t})$, for each $k \leq M$. We write

$$[\langle b_0^{i,t}, v_0^{i,t} \rangle, \langle b_1^{i,t}, v_1^{i,t} \rangle, \ldots \langle b_M^{i,t}, v_M^{i,t} \rangle]$$

to denote this piecewise constant function. Fig. 1 illustrates the piecewise constant form of the VF.

We describe an algorithm to compute these useful budgets, which will be instructive before considering a proof of Prop. 1. Let $b_0^{i,t} = 0 < b_1^{i,t} < \ldots < b_M^{i,t}$ be the useful budget levels for (i, t). We can compute the useful budgets for each state-stage pair—hence the optimal VF and policy—using a simple DP algorithm. Suppose we have a piecewise constant representation of V_D^{t-1}. For

[9]Without discounted costs, the definition of useful budgets in infinite horizon problems requires some care, but is straightforward. Our experiments below use undiscounted budgeted constraints.

V_D^0 the representation is trivial. For ease of exposition, assume each state j has $M+1$ useful levels $b_0^{j,t-1}, \ldots, b_M^{j,t-1}$. We compute the useful budgets and values $V_D^t(i,b)$ for each state i as follows:

- Let $\sigma : S_i^a \to [M]$ be an assignment of each reachable state $j \in S_i^a$ to a useful budget level $b_\sigma^{j,t-1}(j)$ with $t-1$ stages-to-go. Let Σ be the set of such mappings. $\sigma(j)$ is the (index of) budget that will be consumed if we reach j after taking action a; we sometimes informally refer to the selected budget itself as $\sigma(j)$. Implicitly, σ dictates the $t-1$ stage-to-go policy by selecting a budget level at each next state.[10]
- Let the *potentially useful budget levels* for (i,t,a) be:

$$\widetilde{B}_a^{i,t} = \{c_i^a + \sum_{j \in S_i^a} p_{ij}^a b_{\sigma(j)}^{j,t-1} \mid \sigma \in \Sigma\} \cap \{b \leq B_{\max}\}.$$

  Note that each entry $b_k^{i,t} \in \widetilde{B}_a^{i,t}$ is determined by some mapping σ. Let the corresponding expected value associated with $b_k^{i,t}$ be $v_k^{i,t} = r_i^a + \gamma \sum_j p_{ij}^a v_{\sigma(j)}^{j,t-1}$.
- Assume a reindexing of the entries in $\widetilde{B}_a^{i,t}$ so that the budget levels are ascending (ties broken arbitrarily). The *useful budget levels* for (i,t,a) are:

$$B_a^{i,t} = \{b_k^{i,t} \in \widetilde{B}_a^{i,t} : \nexists k' < k \text{ s.t. } v_{k'}^{i,t} \geq v_k^{i,t}\}.$$

  In other words, any potentially useful budget level that is weakly dominated w.r.t. value by some smaller (lower-indexed) level is discarded. The useful budget levels and corresponding values represent the Q-function $Q^t(i,a,b)$.
- Let the *potentially useful budget levels* for (i,t) be

$$\widetilde{B}^{i,t} = \cup_{a \in A} B_a^{i,t},$$

  and let $B^{i,t}$ be the *useful budget levels*, obtained by pruning $\widetilde{B}^{i,t}$ in the same fashion as above. The useful budget levels and corresponding values represent the VF $V^t(i,b)$.

We provide a formal proof of Prop. 1 in the appendix, based on the algorithm above. Informally, it is easy to see that $B^{i,t}$ contains all and only useful budget levels for (i,t) if this fact holds for all $B^{j,t-1}$ at the previous stage. First, note that $\widetilde{B}_a^{i,t}$ contains all useful budget levels if action a is taken at stage t. If not, then there would be some $b \notin \widetilde{B}_a^{i,t}$ that admitted a different expected value after a was executed; however, a simple inductive argument, and the fact that Σ includes all useful ways of exploiting future budgets, show this cannot hold. The pruning phase (i.e., transformation of $\widetilde{B}_a^{i,t}$ to $B_a^{i,t}$) prunes all and only useless levels. Similar reasoning applies to $\widetilde{B}^{i,t}$ and $B^{i,t}$.

While finite, the number of useful budget levels can grow exponentially in the horizon:

Proposition 2. *For any finite t and state i, $V_D^t(i,\cdot)$ has at most $O((|A|^d)^t)$ useful budget levels, where d is the maximum out-degree of the underlying MDP.*

[10]For example, if i has two reachable states j_1 and j_2 under action a, then $\sigma(j_1) = b_1$ and $\sigma(j_2) = b_2$ means that after action a is taken at i, budget b_1 will be made available if we transition to j_1 (and the policy subsequently executed will be optimal for that expected budget), and b_2 will be made available if we reach j_2.

This is easy to see as a consequence of the proof above (a proof sketch is provided in the appendix).[11] Such a bound is only reached when no potentially useful budgets are dominated. Still this motivates the desire to approximate this set of useful budgets, as we discuss in Sec. 4.4.

4.3 Stochastic Policies

Stochastic policies can offer greater value than deterministic policies due to their inherent flexibility, and thus have a rather different structure. To illustrate, suppose budget b is available at state-stage pair (i, t), where b lies strictly between two adjacent deterministically useful levels, $b_k^{i,t} < b < b_{k+1}^{i,t}$. If forced to choose an action and spend level deterministically, the excess budget greater than $b_k^{i,t}$ provides no additional value. A stochastic policy, by contrast, allows one to randomly spend $b_{k+1}^{i,t}$ with probability $p = \frac{b - b_k^{i,t}}{b_{k+1}^{i,t} - b_k^{i,t}}$ and $b_k^{i,t}$ with probability $1 - p$, providing greater expected value from (expected) spend b than the piecewise constant value offered by the optimal deterministic policy, which only allows the "use" of $b_k^{i,t}$ (see Fig. 1). Apart from randomizing the spend at state i, this stochastic choice also randomizes the choice of action when the optimal actions associated with $b_k^{i,t}$ and $b_{k+1}^{i,t}$ differ.

It is not hard to see that the optimal value function under such "single-stage randomized" policies at (i, t) is given by the *convex hull* of the deterministically useful budget levels for (i, t) (see Fig. 1). Given useful budget levels $B^{i,t} = \{b_0^{i,t} = 0 < b_1^{i,t} < \ldots < b_M^{i,t}\}$, we say $b_k^{i,t}$ is *dominated* if there are two bracketing budget levels $b_{k^-}^{i,t}, b_{k^+}^{i,t}$ $(k^- < k < k^+)$ such that some convex combination of their values dominates that of $b_k^{i,t}$, i.e., $(1 - p)v_{k^-}^{i,t} + pv_{k^+}^{i,t} > v_k^{i,t}$. The convex hull is *piecewise linear and concave (PWLC)* and monotone, comprising the PWL function formed by the set of *non-dominated* points.

We now show that this PWLC structure is preserved by Bellman backups, and can be computed effectively in two stages: first, a simple *greedy algorithm* is used to assign budget incrementally to reachable next states, thus computing a PWLC representation of the Q-functions for each action a; and second, we compute the backed up VF by taking the convex hull of the union of these Q-functions. Thus we will show that any finite-horizon (stochastically) optimal VF for a BMDP has a finite PWLC representation.

Computing Q-functions Assume that $V^{t-1}(j, \cdot)$ is PWLC for all $j \in S$, and has the form:

$$[\langle b_0^{j,t-1}, v_0^{j,t-1} \rangle, \ldots, \langle b_M^{j,t-1}, v_M^{j,t-1} \rangle],$$

where each $b_k^{j,t-1}$ is non-dominated (for ease of exposition, we assume each j has $M + 1$ such non-dominated levels). Let $B(S_a^i) = \cup_{j \in S_a^i} B^{j,t-1}$, and re-index the elements of $B(S_a^i)$ in decreasing order of their *bang-per-buck ratio (BpB)*:

$$BpB(b_k^{j,t-1}) = \frac{v_k^{j,t-1} - v_{k-1}^{j,t-1}}{b_k^{j,t-1} - b_{k-1}^{j,t-1}} = \frac{\Delta v_k^{j,t-1}}{\Delta b_k^{j,t-1}}.$$

[11]Note that this exponential growth occurs in a fashion similar to an analogous phenomenon in POMDP value functions, where the number of so-called α-vectors in the PWL representation of the value function grows exponentially with the horizon due to a mapping from possible observations at each state to a next-stage policy tree.

(For $k = 0$, we leave BpB undefined, since $b_0^{j,t-1} = 0$ for all j.) This expresses the (per-unit budget) increase in value associated with increasing the available budget at $(j, t-1)$ from $b_{k-1}^{j,t-1}$ to $b_k^{j,t-1}$. Let $j(m), k(m), \Delta b(m), \Delta v(m)$ denote, resp., the state j, the useful budget index for j, the budget increment, and the value increment associated with the mth element of (the sorted set) $B(S_a^i)$. Let $M_i^* = |S_a^i|M$ denote the size $B(S_a^i)$.

We first consider the Q-function $Q^t(i, a, b)$ for a fixed action a as a function of budget b. We derive the budget allocation that optimizes *future value* with $t-1$ stages to go—$Q^t(i, a, b)$ is obtained by discounting the future value and adding immediate reward r_i^a. We begin with an informal description of the method to illustrate the underlying intuitions.

For any available budget b at state i, we must consider the optimal allocation of budget to the next states $j \in S_a^i$ so that the expected spend doesn't exceed $b' = b - c_i^a$. The minimum budget that allows a to be taken is c_i^a: at this level, no budget is available at any next state and expected future value is given by $\sum_j p_{ij}^a v_0^{j,t-1}$.

The first units of budget *beyond* c_i^a are most effectively allocated to the state $j(1)$—i.e., the first state in $B(S_a^i)$—since it has the highest BpB for its first useful budget level. Indeed, allocating budget up to $\Delta b(1) = b_{k(1)}^{j(1),t-1}$ to $j(1)$ gives an expected (future) value improvement (relative to no budget) of $\Delta v(1) = \Delta v_{k(1)}^{j(1),t-1}$ with probability $p_{i,j(1)}^a$, and has an expected spend of $p_{i,j(1)}^a \Delta b(1)$. This is, by definition of $B(S_a^i)$, the greatest expected (future) value attainable at i for any $b' \le p_{ij}^a b_{k(1)}^{j(1),t-1}$.

The next units of budget should be allocated to $j(2)$: the first $\Delta b(2)$ units so allocated provide a return of $BpB(b_{k(2)}^{j(2),t-1})$ per unit; but the expected spend and expected return occur with probability $p_{i,j(2)}^a$, hence the expected allocation at i is $p_{i,j(2)}^a \Delta b(2)$. Any additional budget available at i should continue to be allocated to next states $j \in S_a^i$ in "increments" of $p_{i,j(m)}^a \Delta b(m)$ in decreasing order of the corresponding BpB ratios. The (future component of a) Q-function constructed from two reachable next states is illustrated in Fig. 2: it is easily seen that this corresponds to a simple sorted merge of the linear segments of all reachable-state VFs, with each segment scaled by the transition probability of its next state.

A simple inductive argument shows that this simple greedy allocation of budget determines the optimal Q-function for action a. Furthermore, the Q-function is PWLC and monotone, with $M|S_i^a|+1$ useful budget levels and values. More formally, we define:

Definition 1. Let V^{t-1} be a VF such that, for all $j \in S$, $V^{t-1}(j, b)$ is bounded, monotone and PWLC in b with a finite number of budget points. Define:

- the minimal (i.e., 0th) useful budget for action a at state i to be $b_{a,0}^{i,t} = c_i^a$, which gives value $v_{a,0}^{i,t} = r_i^a + \gamma \sum_{j \in S_i^a} p_{ij}^a v_0^{j,t-1}$;
- the mth useful budget, for $0 < m \le M_i^*$, to be $b_{a,m}^{i,t} = \sum_{\ell \le m} p_{i,j(\ell)}^a \Delta b(\ell)$ which gives value $v_{a,m}^{i,t} = r_i^a + \gamma \sum_{\ell \le m} p_{i,j(\ell)}^a \Delta v(\ell)$.

For any $0 < m \le M_i^*$, and budget level b falling between two consecutive useful budgets $b_{a,m-1}^{i,t} <$

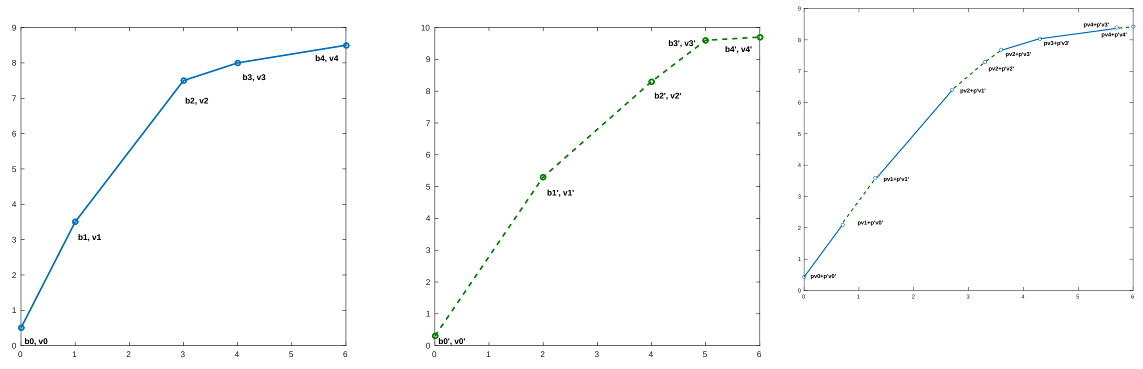


Fig. 2: (a) VF for state j; (b) VF for state j'; (c) Q-function (future component) for action that reaches j with prob. p and j' with $p' = 1 - p$.

$b < b^{i,t}_{a,m}$, let $p^{i,t,a}_b = \frac{b - b^{i,t}_{a,m-1}}{b^{i,t}_{a,m} - b^{i,t}_{a,m-1}}$. Define the Q-function for action a at stage t as follows:

$$Q^t(i,a,b) = \begin{cases} \text{undefined} & \text{if } b < b^{i,t}_{a,0} \\ v^{i,t}_{a,m} & \text{if } b = b^{i,t}_{a,m}, \text{ for } 0 \le m \le M^*_i \\ p^{i,t,a}_b v^{i,t}_{a,m} + (1 - p^{i,t,a}_b) v^{i,t}_{a,m-1} & \text{if } b^{i,t}_{a,m=1} < b < b^{i,t}_{a,m}, \text{ for } 0 < m \le M^*_i \\ v^{i,t}_{a,M^*_i} & \text{if } b > b^{i,t}_{a,M^*_i}. \end{cases}$$

We show that this Q-function indeed reflects the optimal value that can be extracted from any action a at the given budget level b:

Theorem 3. *The $Q^t(i,a,b)$ defined in Defn. 1 gives the maximal expected value that can be extracted when taking action a at state i with available budget b with future value given by V^{t-1}.*

We provide a proof of Thm. 3 in the appendix.

Computing Value Functions Given the PWLC representation of the Q-functions, we can construct a similar PWLC representation of the value function $V^t(i,b) = \max_a Q^t(i,a,b)$. Intuitively, we simply need to take the union of the points that determine each Q-function, and remove any dominated points (much like in the case of the move from deterministic to stochastic policies). More precisely, assuming a fixed state-stage (i,t), let Q_a be the set of budget-value points in a's Q-function, and let $Q_* = \cup_a Q_a$ be the union of of these points—we also annotate each point with the action from which is was derived, so each annotated point has the form (b,v,a). We say that (b,v,a) is *dominated* in Q_* if there are two (different) points $(b_1,v_1,a_1),(b_2,v_2,a_2) \in Q_*$ such that $b_1 \le b \le b_2$ and $(1-\alpha)v_1 + \alpha v_2 > v$, where $\alpha = \frac{b-b_1}{b_2-b_1}$. Removing all dominated points from Q_* leaves the set of points that form the useful budget levels in the PWLC representation of $V^t(i,\cdot)$. In other words, we form the convex hull of Q_*. Intuitively, it is easy to see that no dominated point (b,v,a) is useful in a stochastic policy, since a greater value can be attained, using the same expected budget, by α-mixing between actions a_1 and a_2 (and spending the corresponding expected budgets).

The construction above shows that the finite-horizon VF for any fixed state is monotone, PWLC in budget:

Theorem 4. *For any finite t and state i, $V^t(i,b)$ is piecewise linear, concave and monotone in b.*

We omit the proof, as it is analogous that that describing the union of Q-functions in the deterministic case.

The PWLC representation of $V^t(i,\cdot)$ can be constructed using a variety of convex hull methods. In our context, a basic *Graham scan* [25] is especially appropriate, since the budget points in the Q-functions are maintained in sorted order. A Graham scan inserts these points in order of increasing budget from the set of Q-functions into the PWLC representation of $V^t(i,\cdot)$. With the insertion of the next point (b_{new}, v_{new}), its BpB ratio is computed w.r.t. to the last budget point (b_{last}, v_{last}) in the VF prior to this insertion. If this new BpB is at least that of last budget point (i.e., $\frac{v_{new}-v_{last}}{b_{new}-b_{last}} \geq \frac{v_{last}-v_{2ndlast}}{b_{last}-b_{2ndlast}}$), then we delete (b_{last}, v_{last}) from VF. We repeat this process until the "new" last point remains undeleted. The complexity of this procedure is $O(|Q_*|\log|Q_*|)$.

As in the case of deterministic policies, the number of useful budget levels can grow exponentially in the horizon. In fact, the number of stochastically useful budget levels can be no greater than the number of deterministically useful levels, since the set of potentially useful budgets and actions over which randomization occurs is identical to the deterministic case; but it can be significantly less in practice because stochastic domination can prune many more points. We immediately have:

Proposition 5. *For any finite t and state i, $V^t(i,\cdot)$ has at most $O((|A|^d)^t)$ useful budget levels, where d is the maximum out-degree of the underlying MDP.*

For infinite horizon problems, we may no longer have a finite number of useful budget levels. However, the VF remains concave:

Theorem 6. *For any state i, the optimal infinite-horizon value function $V(i,b)$ is concave and monotone in b.*

This can be shown using a simple mixing argument (see proof in the appendix). Standard MDP approximation bounds apply when using the finite-horizon approximation V^t to approximate the infinite-horizon VF. Let the Bellman error of V^t be $\max_{i,b\leq B_{\max}} V^t(i,b) - V^{t-1}(i,b)$. Then we have:

Observation 7. *Let V^t be the optimal t-stage VF for a BMDP have Bellman error ε. Then*

$$\max_{i,b\leq B_{\max}} V^*(i,b) - V^t(i,b) \leq \frac{\varepsilon}{1-\gamma}.$$

We note that the computation of Q-functions and VFs can be distributed rather easily for large BMDPs by distributing the computation of V^t for different states to different compute nodes. Communication is needed of the final VFs only between consecutive stages.

4.4 Value Function Approximation

The complexity of determining the optimal VF is largely determined by the number of useful budget points (i.e., the number of segments in the PWLC representation)—this is clearly evident from Prop. 5 and the algorithm outlined above. Fortunately, the VF can be approximated in a natural fashion by removing non-dominated points that are "close" to lying within the convex hull of the remaining points. For instance, in Fig. 2(c), deletion of the second and third points, $(pb_1 + p'b'_0, pv_1 + p'v'_0)$ and $(pb_1 + p'b'_1, pv_1 + p'v'_1)$, would result in a simpler Q-function with a single segment from $(pb_0 + p'b'_0, pv_0 + p'v'_0)$ to $(pb_2 + p'b'_1, pv_2 + p'v'_1)$ replacing the three segments in the true Q-function. It would however, closely approximate the true Q-function since the slopes (BpBs) of all deleted segments are nearly identical. Similarly, deleting point $(pb_2 + p'b'_2, pv_2 + p'v'_2)$ would induce very little approximation error because it is very close to the following point $(pb_2 + p'b'_3, pv_2 + p'v'_3)$ (even if the adjacent BpBs differ significantly).

A simple pruning criterion can be added to the insertion step of the Graham scan used to construct the convex hull. Without approximation, when inserting (b_{new}, v_{new}) with BpB β_{new}, the last point in the current VF—say, (b_k, v_k) with BpB β_k—is deleted if $\beta_{new} \geq \beta_k$. To approximate the set, we can instead delete this point if $\beta_{new} \geq \beta_k - \varepsilon$, with a given approximation tolerance ε. Assuming max-norm error, which is the standard (and most useful) for MDP applications, this pruning introduces a (one-shot) error in the VF of at most $\varepsilon(b_k - b_{k-1})$. In the recursive application of this pruning rule, error accumulates at most additively: if a single insertion step results in s pruning steps, error is at most $s\varepsilon(b_k - b_{k-s})$, and exact maximum error can be computed easily. Since the error introduced also depends on the length $b_k - b_{k-s}$ of the segment being pruned (i.e., the difference in the budget endpoints), we can also prune if the budget points are close. Thus we distinguish *slope* pruning and *length* pruning: both criteria can be used, or pruning be dictated by the product, i.e., terminated when this product bound reaches some threshold τ.

Once pruning is halted at a specific insertion step, any pruning at *subsequent* insertion steps introduces *independent* error, since the VF is exact at any unpruned point. The overall error is bounded by the maximum of the per-insertion-step introduced errors. If a maximum pruning tolerance of τ is used while creating the VF, then total error in the finite-horizon VF can be bounded using standard MDP approximation bounds:

Observation 8. *Let V^t be the optimal t-stage VF for a BMDP, and $\widetilde{V}^t$ be the approximation produced with a maximum pruning tolerance of τ. Then*

$$\max_{i, b \leq B_{\max}} V^t(i,b) - \widetilde{V}^t(i,b) \leq \frac{(1-\gamma^t)\tau}{1-\gamma}.$$

In practice, it is often useful to use aggressive pruning early in the DP process to get a crude approximation of the VF—essentially getting the VF in the right neighborhood while minimizing computation by dramatically reducing the number of segments maintained in the PWLC VF representation—followed by pruning with a much lower error tolerance for the final few iterations. This exploits the contraction properties of Bellman backups to discount the error/impact of the early aggressive pruning stages. For example, if V^t is approximated using an aggressive pruning tolerance τ_{high} for the first k stages and a much lower tolerance τ_{low} for the final $t-k$ stages, the bound in Obs. 8 becomes $\frac{(\tau_{high} - \tau_{low})(1-\gamma^k) + \tau_{low}(1-\gamma^t)}{1-\gamma}$. As we show empirically in Sec. 4.7, this scheme can reduce computation time significantly with little introduction of error.

4.5 Policy Execution

The optimal value function V^* and policy π^* for a BMDP allow one to easily assess the tradeoff between the expected budget one is willing to spend and the expected value one obtains by implementing the corresponding optimal policy. For instance, given an initial state i_0, the graph of $V^*(i_0, b)$ can be used to visually determine the "sweet spot" in spend (see Fig. 4 for examples). The PWLC representation of the VF can also be used directly to determine the maximum spend that achieves a certain ROI. Finally, the VFs can be used to assess the budget-value tradeoff across different MDPs, or different states of the same MDP (e.g., as occupied by different users). We return to this last point in the next section.

Suppose we use the optimal value function V^* to determine the ideal level of (expected) spend b_0 at some initial state of the MDP i_0, with an expected return of $V^*(i_0, b_0)$. One method to determine how to act is to use the BMDP model as a true *constrained MDP*, solving the CMDP with a fixed budget of b_0 (and initial distribution $\Pr(i_0) = 1$), and implementing the resulting policy. However, this is wasteful since the BMDP solution embeds the optimal solution to this specific CMDP in its solution.[12]

The *execution* of the BMDP policy π^* is somewhat subtle however. Policy execution is not simply a matter of taking the action distribution $\pi^*(i, b)$ whenever the current state is i and b remains of the initial budget (as would be typical of a stationary MDP policy). This is so because the optimal value $V^*(i_0, b_0)$ is determined by a policy whose *expected spend* is b_0—and achieving this value sometimes requires spending more than b_0. Specifically, the optimization in Eqns. 1 and 2 (whether with deterministic or stochastic policies) generally assigns some future states a *greater* budget than that actually remaining (i.e., greater than $b_0 - c_{i_0}^a$) after taking the action a drawn from distribution $\pi^*(i_0, b_0)$, and some future states less. It is only the *expected spend* that must not exceed the remaining budget. If the next state j is one whose "assigned budget" b_j differs from $b_0 - c_{i_0}^a$, whether greater or less, then we must execute action $\pi^*(j, b_j)$ in order to ensure we achieve the expected value of $V^*(i_0, b_0)$.

Implementing this non-stationary policy requires that we maintain (at least, indirectly) some form of "history" to identify the action-determining budget level at any state we could reach at each stage of policy execution. Fortunately, this history is Markovian. Specifically, at the final stage of our DP algorithm, when we compute $V^*(i, b)$ for each useful budget level b, we not only record its expected value, but also record the mapping of the remaining budget $b - c_i^a$ to next states S_i^a. Thus for any state i and any useful budget level b (with optimal action a), we record the budget assignment $\mathbf{b}$ to S_i^a that maximizes expected future value in Eqns. 1 and 2 (again, this is only required at the final Bellman backup for stationary policies).

We initiate policy execution, given starting state i_0 and initial expected budget b_0 in the usual way, by determining the optimal action distribution $\pi^*(i_0, b_0)$—generally, this randomizes the choice of expected budget b between two useful budget levels, and the corresponding action a. However, we also note the choice of next-state budget assignment b_j to each $j \in S_i^a$—if the next state reached is j, we act as if budget b_j is available, even if that exceeds the true remaining budget. The next

[12]This is an informal claim, unless the CMDP solution method admits non-stationary policies, since the optimal BMDP policy will be non-stationary w.r.t. states (though not state-budget pairs).

action chosen proceeds in the same way, executing $\pi^*(j, b_j)$. Thus the only history that needs to be accounted for when selecting an action at state j is the budget assignment to j from the state at the previous stage.

4.6 Variance in Spend

As noted above, the optimal policy $\pi^*(i, b)$ that attains value $V^*(i, b)$ satisfies the budget constraint b in expectation; but depending on the realization of state transitions, it may in fact spend more or less than b. If we execute the policy many times, possibly initiating it a variety of different states (as in our ad domain), such variance in will be unproblematic; but when executed in one or a small number of instances, the actual variance in spend can be an important determinant in the budget allocation process.

The variance in actual spend b associated with executing $\pi(i, b)$ can, in fact, be computed within the DP algorithm while the optimal VF is being constructed. Specifically, we can readily compute the variance associated with the useful budget points determined at each stage of the DP algorithm (for both Q-functions and VFs) and propagate these through Bellman backups. We describe the recursion here briefly, and note that we need only compute variance at useful budget points (since variance at intermediate points is derivable from these).

Beginning with 1-stage-to-go, we note that there is no variance in spend at the *useful* budget points b in $V^1(i, \cdot)$ (for any state i). Of course, for any budget point b that lies strictly between useful points $b_1 < b < b_2$, the optimal 1-stage to go policy will randomize spend with probabilities $p, 1-p$ (see Sec. 4.3 for the definition of the mixing probabilities), and hence has variance $\sigma^2_{i,b,1} = p \cdot (b_1)^2 + (1-p) \cdot (b_2)^2 - b^2$. This latter variance term is useful if acting in a 1-stage process, but is especially so when computing variance for the 2-stage VF.

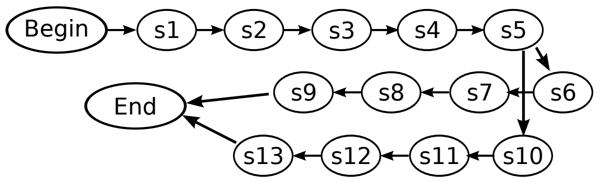
Suppose we have recorded the variance $\sigma^2_{j,b,t-1}$ for the optimal t-stage-to-go VF V^{t-1} for each state j and for each useful budget point b in $V^{t-1}(j, \cdot)$. As above, this allows us to compute the variance for each intermediate budget point as well. Recall that any useful budget point b in $V^t(i, \cdot)$ is given by an action a with cost c_i^a and a mapping associating budget levels b_j to each reachable next state j. The spend at stage $t-1$ is thus a mixture random variable, with each expected spend level b_j realized with probability p^a_{ij}, and the spend b_j itself subject to its own variance.[13] Thus the variance of spend, whose expectation is $C_{i,t,b} = c_i^a + \sum_{j \in S_i^a} p^a_{ij} b_j$, is that of a mixture distribution:

$$\sigma^2_{i,b,t} = \sum_{j \in S_i^a} p^a_{ij}((b_j)^2 + \sigma^2_{j,b_j,t-1}) - (C_{i,t,b} - c_i^a)^2.$$

4.7 Empirical Evaluation

We evaluate the effectiveness of our dynamic programming method on a selection of MDPs, and measure the impact of approximating the PWLC value functions.

[13]We note that spend, hence the associated variance, at all $j \in S_i^a$ will lie at one of j's useful budget levels, with the exception of at most one such j, for which the simple computation above for variance at an intermediate budget point may be needed.



Search: s1: unint; s2: general int; s3: search1, s4: search2, s5:search3
Advertiser: s6: interest1; s7: interest2; s8: interest3, s9: conversion
Compt'r: s10: interest1; s11: interest2; s12: interest3, s13: conversion

Fig. 3: A Synthetic Ad MDP. Note that the "Begin" state reflects the beginning of a customer's search process (and transitions from it can be viewed as encoding a prior distribution over interest states). From an advertiser's perspective, the MDP for a specific customer can begin at any state of this process depending on when that customer became a target (when they entered the ad network, when a targeted campaign began, etc.).

Synthetic Ad MDP We begin with evaluation of a small synthetic MDP that reflects the influence of ads on product search behavior. The small size of the MDP allows a detailed examination of its optimal VF and policy structure. The MDP M_{synth}, depicted in Fig. 3, consists of 15 states reflecting various levels of customer interest in an advertiser's product (or those of its competitors), and five actions comprising various levels of advertising intensity (e.g., reflecting ad position or ad format), including a zero-cost "no-op" (no advertising).[14] It is assumed that competitor behavior is stationary. Transitions representing "nominal" (stochastic) progress through the search funnel are shown in the figure, with other stochastic transitions omitted for clarity. More intense ad actions are more costly, but increase the probability of progressing in the funnel and developing specific interest in the advertiser's product (as opposed to those of competitors), and decrease the odds of regression or abandoning one's purchase intent (moving to the zero-cost absorbing/terminal state).

We solve the corresponding BMDP using a discount rate of 0.975 and a horizon of 50. We use four methods with different degrees of approximation: exact (no pruning); mild pruning (slope and length pruning set to 0.01); aggressive pruning (both set to 0.05); and hybrid (mild pruning for 45 iterations followed by exact computation for the final five stages). The following table shows the average (and minimum, maximum) number of segments in the PWLC VF over the 15 MDP states[15]—it is the number of segments that determines the (representational and computational) complexity of the VFs—and computation time under each regime. For the regimes involving approximation, we also show the maximum absolute and relative errors induced by the approximation (and the optimal value at which the maximum error occurs). For calibration, we note that the optimal VF has an average maximum value of 56.5 and an average maximum useful budget of 15.2 (excluding the terminal states).

	No pruning	Mild	Aggressive	Mild then No
Segments	3066 (0–5075)	18.3 (0–47)	10.4 (0–26)	480.8 (0–877)
Max. Err.	—	4.84 (26.61)	4.84 (26.61)	0.21 (58.77)
Max. Rel. Err.	—	40.9% (4.24)	48.7% (1.54)	2.3% (0.55)
CPU Time (s.)	1055.4	17.54	10.36	28.67

[14]See Archak *et al.* [6] for further motivation for this type of model.

[15]The minimum is always 0 due to the absorbing and conversion states.

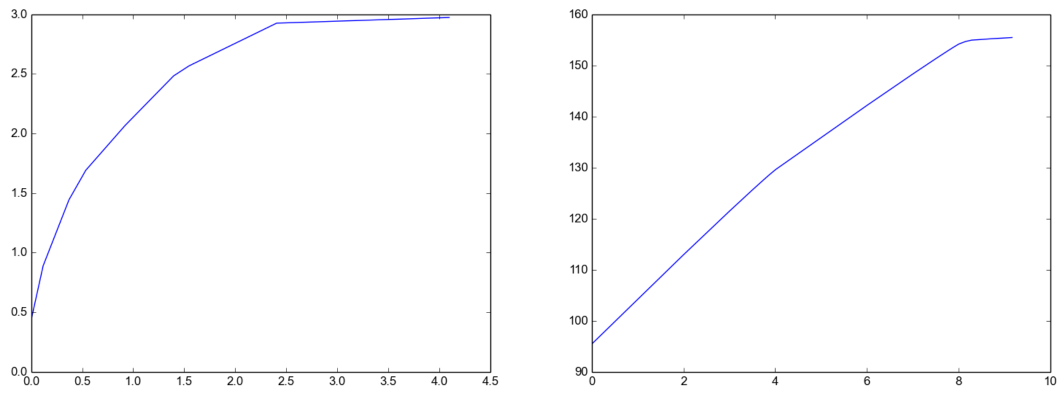


Fig. 4: PWLC VFs for (a) state s_1; (b) state s_8.

We see that the optimal VF has a large number of segments per state (over 3000 on average, and as many as 5000), but can be approximated quite well with very few segments. With aggressive and mild pruning, the VF is very compact, but has reasonably large maximum error (4.84, which is 18% rel. error at the point at which it occurs); the maximum relative error is also significant in each case (from 41–49%), though it occurs at points with low value (hence corresponds to small abs. error). The hybrid scheme that prunes during the first stages of DP, but then stops pruning for the last few iterations, works very well—by exploiting the contraction properties of the MDP, we see that the error associated with the initial pruning is almost entirely overcome by a final few iterations with no approximation. This reduces the number of segments in the final VF by an order of magnitude, and the computation time by nearly two orders of magnitude, while introducing very small maximum absolute error (0.21, or about 0.36%) and maximum relative error (2.3%, or about 0.0013).[16]

Fig. 4 illustrates the form of the PWLC VFs for two representative states: s_1, which occurs early in the search funnel; and s_8, a state indicating maximal interest in the advertiser's product prior to potential conversion. Recall that these VF curves, plotted for initial states or initial distributions, make it possible to make budget-value tradeoffs and assess ROI prior to making actual allocations of budget.

We note that the variance associated with the actual spend induced by the policy associated with a specific budget level is quite high. This is to be expected given the inherent uncertainty in customer behavior in this model.[17] For instance, while maximal useful budget at s_1 is 4.09, should a customer reach s_5, the maximal useful budget increases dramatically to 22.31. We discuss the impact of this variance in further detail in the next section.

Advertiser MDPs We next consider two MDPs, of size ranging from roughly 1500 to half a million states, with these states to derived from the contact data of a large advertiser. The data consists of

[16]CPU time is measured using a simple prototype implementation written in Python, executed on a Intel(R) Xeon(R) 3.5GHz CPU E5-1650 with 32Gb of RAM.

[17]For example, e.g., the probability of a customer moving to an abandon state from any other state is at least 0.05 under *any action* (and up to 0.175 in several non-conversion states), at which point any allocated budget would not be used.

sequences of cross-channel advertising touch points with specific users—each touch point is labeled with a specific "channel" or contact event (e.g., display ad, email, paid search ad, direct navigation to web site) or type of activity on the advertiser's web site (including "transactions" or "conversions"). There are a total of 28 event types in a data set of 3.6M trajectories comprising 10 million events.

From this data, we learn several *variable-order Markov chain (VOMC) models* [13, 37, 16] using simple maximum likelihood estimation (MLE). These VOMCs predict the transitions induced by the advertiser's contact policy, based on a small sufficient history involving up to 10 of the preceding events. The sufficient histories comprising each VOMC form the state space of a Markov chain over histories (i.e., the states of the MDP are "event trajectories" of varying lengths). From these we derive action-conditional transition models by substituting the end points of these sufficient histories with one of a small number of "controllable" events and using the VOMC model for these altered histories for action-conditional transition predictions. In all cases, we consider four actions (no-op, email, paid search, display ad). We consider two different models: the first is a single VOMC model with a maximum state-order of 10 and an average state-order of five. This induces an MDP with 481,582 states and actions. The second is a two-component mixture of much smaller VOMC models (with a maximum state-order of three). We experiment with the first component, which has 1469 states.

We solve the corresponding budgeted MDPs for both models using the same horizon (50) and discount factor (0.975) as above. The table below shows pruning level, number of segments, and computation time for the 1469-state MDP using the same pruning strategies as above.[18]

	No pruning (1469)	Mild (1469)	Aggr. (1469)	Mild then No (1469)
Segments	251.5 (74–359)	234.2 (77–342)	25.6 (5–39)	76.84 (18–321)
Max. Err.	—	5.13 (171.56)	28.88 (169.33)	3.94 (167.61)
Max. Rel. Err.	—	2.99% (171.56)	12.32% (169.33)	2.35% (167.61)
CPU Time (s.)	19918.9	10672.5	1451.8	2390.0

As above, hybrid pruning proves to be most effective, reducing computation time by an order of magnitude, while introducing a maximum relative error of only 2.35%. For the large model (481K states), the BMDP was solved only with aggressive pruning (slope 2.0, length 0.1), and averaged about 11.67 segments per state and 1168s. per DP iteration.

5 Budget Allocation across MDPs

We now consider the problem facing an advertiser with a (roughly) fixed budget and a specific set of target customers, each of whom occupies the state of some underlying MDP. As discussed above, while this could be viewed as a large joint MDP, the per-user MDPs exhibit *almost* complete independence: given a specific joint action $\mathbf{a}$, the transitions and rewards of one user have no impact on—and

[18]Given the large branching factor and small transition probabilities in these MDPs, the number of segments without pruning blows up quickly—indeed, after as few as 12 DP steps, many states have many thousands of segments with length on the order 10^{-6} (i.e., making inconsequential budget distinctions on the order of millionths of dollars); so the "no pruning" optimal benchmark in fact uses slope/length pruning of $0.001/0.0001$ for 20 iterations and $0.01/0.001$ for 30. Aggressive is slope/length = $0.2/0.01$; mild is $0.02/0.001$.

provide no information about—those of another. Indeed, the sub-MDPs are linked only by the budget constraint: the action taken in one sub-MDP may limit the ability to take an action in another (concurrently or in the future) by consuming budget. In this sense, the problem is a *weakly coupled MDP* [32]. As a consequence, we apply the same general decomposition method of [32] to solve the joint MDP: we solve the sub-MDPs offline, and compose their solutions online. However, because of differences in the structure of the sub-MDP solutions, we handle the online composition phase differently.

5.1 Offline Decomposition

Our approach to decomposition is straightforward. We first solve each *single-user sub-MDP*, $\langle S, A, P, U, C, \gamma \rangle$, as a budgeted MDP with a some natural per-user maximum budget B_u.[19] For ease of exposition, we will proceed as if there is only one user type, so each user behaves (e.g., responds to ads) according to the same dynamics; this means there is only one user MDP, with one user distinguished from another only by the MDP state they occupy. In such a case, there is only a single BMDP to solve.[20]

The solution of the BMDP gives an optimal single-user policy π and value function V spanning all $s \in S$ and $b \leq B_u$. Both π and V indicate the action choice and value *as a function of the budget available to be spent (in expectation) on that user alone*. We take advantage of this when solving the joint MDP below. Indeed, it is for this reason that we do not solve the user MDP as a constrained MDP: such an approach would fail to indicate the value of allocating *varying* levels of budget to users in different states.

5.2 The Budget Allocation Problem

Suppose the initial (or current) state of the joint MDP with M customers is $\mathbf{s} = \langle s[1], \cdots s[M] \rangle$ and our available budget is B. A natural way of exploiting the offline solution of the BMDP is to allocate some portion $b[i]$ of the budget B to each customer $i \leq M$ to optimize the sum of expected values $v[i]$ obtained from each i given optimal engagement with i given an expected budget of $b[i]$. Specifically, define the *budget allocation problem (BAP)* to be:

$$\max_{b[i]:i\leq M} \sum_{i\leq M} V(s[i], b[i]) \text{ subj. to } \sum_{i\leq M} b[i] \leq B, \tag{3}$$

where V is the optimal value function for the underlying BMDP.

Intuitively, the BAP determines an allocation $\mathbf{b}^* = \langle b^*[1], \ldots, b^*[M] \rangle$ that maximizes the expected value of *committing* a specific (expected) budget $b^*[i]$ to customer i. If we implement the corresponding optimal policy for each such user, then by linearity of expectation, we obtain the expected value given by the optimal solution to the budget allocation problem, and have an expected spend no greater than B:

[19] B_u, the maximum profitable spending level for a single user, is much less than the global budget B and can be determined using a variety of techniques. This upper bound can omitted if future budget spend is discounted in the budget constraint, since the BMDP solution will find natural caps regardless.

[20] More broadly, if there a multiple user types, there will be a single BMDP per type—the solution does not depend on the number of customers. The extension of our algorithm below to multiple types is straightforward.

Observation 9. *Let V be the optimal VF and π the optimal policy for the user BMDP. Let $\mathbf{b}$ be the optimal solution to the budget allocation problem. Then the joint (non-stationary) policy $\overline{\pi}(\mathbf{s}) = \prod_i \pi(s[i], b[i])$ has expected value $\sum_i V(s[i], b[i])$ and expected spend $\sum_{i \leq M} b[i] \leq B$.*

We cannot ensure that this policy is truly optimal for the joint MDP, since this decomposed policy does not admit *recourse* in the execution of one user's MDP that depends on the realized outcome of some other user's MDP. For instance, consider a trivial two-user joint MDP, with one user beginning in state s_1, the other in s_2. Suppose the optimal allocation b_1, b_2 exhausts the global budget B, but does not hit the maximum useful budget in s_2 (i.e., b_2 would have been greater if b had been larger). Suppose user 1 reaches a state s', prior to consuming all of b_1, in which the optimal policy (in the single-user MDP) spends nothing further. The expected value of the BAP solution does not consider that the unused budget could profitably be applied to user 2. With no ability to reallocate, should user 2 be in a state where this additional budget would prove useful, persisting with original allocation of b_2 is suboptimal from the perspective of the joint MDP.

For MDPs with large numbers of customers, or where the variance in the spend induced by the local (per-user) policy is low, the impact of this form of suboptimality will be small. However, as we discuss below, *repeated online reallocation of budget* can largely overcome this potential suboptimality in practice, even in joint MDPs with a small number of users. In our example above, reallocating the remaining budget from user 1 to user 2 would improve the quality of the (implicit) joint policy.[21]

We now discuss the optimization of Eq. 3, the budget allocation problem. This can be viewed as a (multi-item variant of the) *multiple-choice knapsack problem (MCKP)* [40]. In the classic MCKP, we are given M classes of items, with each class i containing n_i items. Each item j has a profit p_j and weight w_j, and we have a knapsack with capacity C. The goal is to select a set of items, one from each class, with maximum profit, such that the total weight does not exceed the capacity of the knapsack.

The budget allocation problem can be viewed similarly. Let B^*_j be the finite set of (stochastically) useful budget levels at state j in the PWLC representation of $V(j, \cdot)$.[22] We assume that the points $\beta^0_j, \cdots, \beta^L_j$ in B^*_j are indexed in increasing order (i.e., $\beta^k_j < \beta^{k+1}_k$). (To reduce the the number of indices we write as if all useful budget sets have the same size m.) For any user i, let $B^*[i] = B^*_{s[i]}$ and $\beta_{ik} = \beta^k_{s[i]}$.

We first consider the *useful-budget assignment problem (UBAP)* in which each user i must be assigned a useful budget level from the discrete set $B^*[i]$ (we will relax this constraint below). The UBAP is exactly an MCKP: each user i can be viewed as an item class for whom exactly one budget level must be chosen from the set of items $B^*[i]$ in that class. The "weight" of an item $b \in B^*[i]$ is b (i.e., the amount of the budget it consumes); and the profit of assigning b to i is $V(s[i], b)$. The capacity is the global budget B, so total "weight" (i.e., total budget assigned) cannot exceed B. We can also view this as a *multi-item* version of MCKP since we have multiple "copies" of the same class—specifically, all users in state j have exactly the same items (i.e., weights/budget levels and profits/value $V(j, \cdot)$) in their corresponding classes.

[21]While this reallocation can only improve overall quality, it still does not ensure optimality—the optimal *joint* policy may well induce a different behavior w.r.t. user 2 even prior to this reallocation in anticipation of this (stochastic) occurrence.

[22]We restrict attention to finite B^*_j (in infinite horizon cases, we use a finite horizon, possibly pruned, approximation).

MCKP can be solved in pseudo-polynomial time by dynamic programming, but can also be formulated as an integer program (IP) and solved directly using IP solvers. Treating each user as a distinct item class, let x_{ik} be a binary variable indicating that user i has been allocated the kth useful budget level β_{ik} from the set of useful budget levels $B^*[i]$. The MCKP IP for the budget allocation problem is:

$$\max_{x_{ik}} \quad \sum_{i\leq M}\sum_{k\leq L} V(s[i],\beta_{ik})x_{ik} \tag{4}$$

$$\text{subject to} \quad \sum_{i\leq M}\sum_{k\leq L} \beta_{ik}x_{ik} \leq B \tag{5}$$

$$\sum_{k\leq L} x_{ik} = 1, \quad \forall i \leq M \tag{6}$$

$$x_{ik} \in \{0,1\} \tag{7}$$

This IP is quite large, requiring $O(ML)$ integer variables (where L bounds the number of useful budget levels for any state). The problem can be simplified considerably by treating all users in the same state identically, reducing the number of integer variables to $O(NL)$ by multiplying the value of a budget allocation to a state by the number of users in that state. In the UBAP (where we insist on assigning useful budgets), this incurs a potential loss of optimality, but this can be overcome as we discuss below.[23]

Fortunately, the linear programming (LP) relaxation of this IP has considerable structure, allowing approximately optimal solutions to the discrete UBAP to be found in polynomial time with a simple greedy algorithm. More importantly, this relaxation induces *optimal* solutions to the original BAP (Eq. 3). Specifically, the LP relaxation of UBAP can readily be viewed as allowing the stochastic choice of discrete budget allocations to users (or states if we aggregate users that lie within the same state). This is equivalent (w.r.t. expected value) to assigning a user some arbitrary (non-useful) budget point equal to the expected budget given the randomized discrete assignment. Indeed, a minor adaptation of the analysis of Sinha and Zoltners [40]—and their greedy algorithm for the LP relaxation of MCKP—to our BAP shows that any randomization will only occur between two consecutive useful budget levels, exactly as in the optimal policies for BMDPs.

Consider the LP relaxation of the IP Eq. 4 and its optimal solution, which gives a (potentially fractional) assignment of budget levels to user i in state $s[i] = j$. We claim that this optimal allocation to any user i is either integral (i.e., assigns a useful budget level to i), or is a mixture of two *consecutive* budget levels, in which case the expected budget b_i and expected value v_i corresponds to a point on the convex hull of the useful points. In other words, it lies on the PWLC VF $V(i,\cdot)$. This implies that the optimal solution of the LP relaxation of the discrete UBAP is an optimal solution to the true BAP.

Proposition 10. *The optimal solution to the LP relaxation of the IP formulation Eq. 4 of UBAP is such that, for each i: (a) $x_{ik} = 1$ for one value of k; or (b) there is some k such that only $x_{ik}, x_{i,k+1} > 0$ (i.e., only two budget levels are allocated, and they must be consecutive).*

[23]The suboptimality arises because, at the "edges" of the global budget, the optimal UBAP solution generally needs to treat some users in a class differently if we do not have enough budget to treat them all the same. This happens in at most one class, as we discuss below.

The proof of this proposition follows very closely the analysis of Sinha and Zoltners [40] for general MCKP (but we provide a proof sketch in the appendix for completeness). Intuitively, this must be the case since the integral allocation of a single budget point, or the fractional allocation of (exactly) two consecutive budget points, to a user gives an expected value that lies on the PWLC VF. Any non-degenerate mixture involving any two non-consecutive points (including any mixture of three or more points) with the same expected budget consumption has an expected value dominated by the VF, due to its "strict" concavity across budget points. As a consequence, we immediately obtain:

Corollary 11. *The optimal solution to the LP relaxation of UBAP is an optimal solution to the budget allocation problem BAP.*

The structure of the LP relaxation of MCKP is very valuable. Sinha and Zoltners [40] show that a simple greedy algorithm can be used to solve the relaxation, hence the same greedy algorithm can be used to solve the relaxation of UBAP, Eq. 4, and hence our original BAP. We briefly outline the algorithm, which we call *greedy budget allocation (GAB)*.

For each state s_j, and each $1 \leq k \leq L$, we define the *bang-for-buck ratio* exactly as we did above when constructing VFs:

$$BpB_{jk} = \frac{V(s_j, \beta_{jk}) - V(s_j, \beta_{jk-1})}{\beta_{jk} - \beta_{jk-1}}.$$

As before, this expresses the per-unit budget increase in value associated with increasing a user's (discrete, useful) budget from $\beta_{j,k-1}$ to β_{jk}. GBA assigns budget incrementally to each user in the order given by the BpB ratio. Initially each i in state $s[i] = s_j$ is assigned a budget of $\beta_{j0} = 0$, and the remaining budget is set to B (our budget constraint). At any stage of GBA, let b denote the unallocated budget, let β_{j,k_i} be i's current allocation and i's current ratio to be $BpB[i] = BpB_{jk_i+1}$. Let i^* be any *best user*, with maximum ratio $BpB[i]$. If sufficient budget remains, we increase i's allocation from β_{j,k_i} to β_{j,k_i+1}, then update both i's ratio and the remaining budget (decrease it by $\beta_{j,k_i+1} - \beta_{j,k_i}$). We continue until the remaining budget is less than $\beta_{j,k_i+1} - \beta_{j,k_i}$. At that point, we allocate the remaining budget "fractionally" to the best user i, assigning it the lower amount β_{j,k_i} with probability p and the higher amount β_{j,k_i+1} with probability $1 - p$. Setting p to be $\frac{(\beta_{j,k_i+1} - \beta_{j,k_i}) - b}{(\beta_{j,k_i+1} - \beta_{j,k_i})}$ ensures that the "expected" budget allocation to i is $\beta_{j,k_i} + b$.

It is not hard to show that GBA will find the optimal solution to the LP relaxation of the IP in Eq. 4 [40]. More precisely, for any user i in state s_j, apart from the one allocated fractionally at the last stage, we set $x_{ik} = 1$ iff β_{jk} was the final allocation to i. For the user allocated at the last stage, we set $x_{ik} = p$ and $x_{ik+1} = 1 - p$. Thus we see the optimal solution is purely integral except for (possibly) two fractional variables consisting of positive assignment to two adjacent budget levels for a single user.

As mentioned above, we can optimize GBA by aggregating all users that are in the same state and increasing all of their budgets at once. If the remaining budget is insufficient, we do this for the greatest number of users from that class who can be increased fully to the next budget level (chosen arbitrarily), and the fractionally allocate one last user via the process above.[24] Fig. 5 details GBA, exploiting this aggregation.

[24]In terms of expected value, it is equivalent to fractionally assign all users in this class in a way that satisfies the expected budget constraint. The approach described simply involves less randomization.

```
1: Input: Budget $B$; Value function $V$; Non-dominated budget set $B^*_j = \{\beta_{j0}, \ldots, \beta_{jL}\}, j \le N$; Bang-per-
   buck ratios $R_{jk}, j \le N, k \le L$; User count $C_j, j \le N$;
2: $b \leftarrow B$                                                ▷ remaining budget
3: $k_j \leftarrow 0\ , \forall j \le N$      ▷ Index of budget level allocated to all users in state $j$; initially allocated $\beta_{j0} = 0$.
4: $R_j \leftarrow R_{jk_j+1}, \forall j \le N$          ▷ Bang-per-buck ratio of users in state $j$ at current allocation.
5: $J \leftarrow \text{argmax}_j\, R_j$                                  ▷ State with largest BBP.
6: while $b \ge C_J(\beta_{Jk_J+1} - \beta_{Jk_J})$ do
7:     $b \leftarrow b - C_J(\beta_{Jk_J+1} - \beta_{Jk_J})$                  ▷ decrease remaining budget
8:     $k_J \leftarrow k_J + 1$                     ▷ increase budget allocation to every user in state J
9:     $R_J \leftarrow R_{Jk_J+1}$                                   ▷ Update $J$'s BBP.
10:    $J \leftarrow \text{argmax}_j\, R_j$                            ▷ Update state with largest BBP.
11: end while
12: Assign budget $\beta_{jk_j}$ to all users in state $j$, for all $j \ne J$
13: $m \leftarrow \lfloor b/(\beta_{Jk_J+1} - \beta_{Jk_J}) \rfloor$       ▷ $m = 0$ if insufficient budget to move some user to next budget level.
14: Assign budget $\beta_{Jk_J+1}$ to $m$ arbitrary users in state $J$.
15: $b \leftarrow b - m(\beta_{Jk_J+1} - \beta_{Jk_J})$
16: Assign to an $m + $ 1st arbitrary user in state J budget $\beta_{Jk_J+1}$ with probability $b/(\beta_{Jk_J+1} - \beta_{Jk_J})$, and
    budget $\beta_{Jk_J}$ with probability $(\beta_{Jk_J+1} - \beta_{Jk_J} - b)/(\beta_{Jk_J+1} - \beta_{Jk_J})$    ▷ If $b = 0$, then $\beta_{Jk_J}$ is assigned
    with probability 1.
17: Assign budget $\beta_{Jk_J}$ to the remaining $C_J - m - 1$ users in state $J$.
```

Fig. 5: The greedy budget allocation (GBA) algorithm.

The GBA algorithm can easily be extended to account for the stochastic arrival of customers of various types, or at various states. Given a global budget constraint, some of the budget should be "allocated" and set aside to account for new arrivals. We defer details to a longer version of the paper.

5.3 Empirical Evaluation

We test the effectiveness of GBA on the synthetic and advertiser-based BMDPs described in Sec. 4.7. We assess the expected spend and expected value of the resulting "implied" joint policies, as well as the variance in spend and how often it exceeds the global budget (since budget constraints are satisfied in expectation only).

We consider two ways of implementing policies induced by the solution of BAP. The first joint policy is the *BMDP policy*, where once the BAP allocation $b[i]$ is made to each user i, we implement the corresponding BMDP policy starting at state $s[i]$—recall that this ensures that the *expected spend* does not exceed $b[i]$, but doesn't guarantee this constraint won't be violated for any specific user. The second is the *static user budget policy (SUB)*: once the BAP allocation $b[i]$ is made to each user i, we implement the first action a in the BMDP policy at $s[i]$; but once we move to the next state, we take the optimal action as if we only had that user's *remaining* budget, essentially ignoring the next state budget mapping from the BMDP policy, and limiting ourselves to using i's "actual" budget $b[i] - c(a)$ at the next state. The remaining budget is updated at each stage to reflect the cost of the action just taken. Apart from the risk of some *small* overspending due to $c(a)$, this policy will recalibrate the actual spend to minimize the odds of overspending $b[i]$.

One important value of solving BMDPs and using GBA to allocate budget is that it allows one assess the tradeoffs when allocating budget to different users in different states (or in different MDPs).

Without the budget-dependent value function, these tradeoffs can only be made heuristically. An alternative allocation scheme when the BMDP has not been solved is to simply apportion the global budget equally across all users, an approach we dub *uniform budget allocation (UBA)*. Once the budget is allocated, one could in principle solve the induced *constrained MDP* (one per occupied user-state) and implement that policy. We can simulate this process by using UBA and then implementing the BMDP policy given this uniform allocation.

Synthetic Ad MDP We begin with the synthetic MDP, adopting a basic setup with 1000 customers starting in the initial state s_0. The following table shows the expected value obtained by three different policies (we explain DBRA below) for four different global budgets:

Total Budget	BMDP Value	DBRA Value	SUB Value
1000	8209.9	8578.8 (830.5)	4106 (707)
2000	10,905	11,019 (964)	4429 (825)
5000	15,692	15,658 (1239)	5270 (830.5)
10,000	18,110	17,942 (—)	6329 (1159)

The BMDP value shown is the true expected value of executing the optimal BMDP policy for each user. The SUB values are estimated by executing the policy for all 1000 users, averaged over 100 trials (with sample std. dev. also shown). Using the optimal BMDP policy has a considerable advantage over selecting a static budget per user and not allowing that budget to be exceeded, yielding 2-3 times the return.[25]

The expected value under BMDP also show a clear pattern of diminishing return with increasing budget. Indeed, our MCKP-based GBA method allows one to *rapidly* assess the value of *optimally* using different global budgets to find the "sweet spot" in spend. Fig. 6 illustrates the budget-value tradeoff (or *sweet spot curve*) for this MDP at 16 budget points (from 0 to just above the maximally useful budget). With the BMDP solution in hand, generating this tradeoff curve requires just 16 runs of GBA.[26]

As discussed earlier, the optimal BMDP policy for this MDP has consider variance in spend. In one run of 1000 customers at initial state s_0, each given an initial budget allocation of \$10 (just under s_0's maximal useful budget of 11.77), the average spend of 10.012 is very close to the expected spend; but the sample standard deviation is 15.24. Indeed, with an initial budget of \$10/user at s_0, the empirical odds that the spend on any single user exceeds the budget is 32.7%, and the odds of exceeding it by at least 50% is 28.7%. This alone explains the poor performance of the SUB allocation policy—by "capping" the spend per user, rather than allowing it to vary around a target expected value, SUB forces the abandonment of the optimal BMDP policy if the required actions have spends that push on over the cap.

Even across a large user population, because of the inherent variance in the spend associated with the optimal BMDP policy, there is always some risk that the global budget constraint will be exceeded. This risk is greater when the number of customers being addressed is small and the variance in the optimal policy is large, and is much less so for large populations. But even in the synthetic BMDP

[25]Note that the expected values are discounted and incorporate the actual budget spent (i.e., reflect reward less cost).

[26]The greedy algorithm averages 1.47ms. to compute the optimal allocation at each budget point.

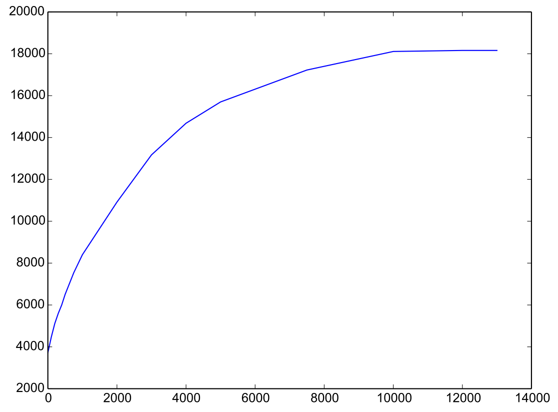


Fig. 6: A plot of global budget (x-axis) and expected value (y-axis) tradeoffs (1000 users). The flat tail demonstrates that the maximal useful budget has been reached.

results above with 1000 users, the global budget is exceeded in 13 of 30 trials, in 5 instances by over 3%, with the largest overspend exceeding the budget by 11.7%.

Dynamic Budget Reallocation One way to alleviate the risk of overspending is through *dynamic budget reallocation (DBRA)*. Rather than committing to the optimal BMDP policy for each user given their initial budget allocation (as SUB does), we reallocate the remaining budget at each stage relative to the current state of each user. More precisely, at each stage of the process, given the current joint state $\mathbf{s} = \langle s[1], \ldots, s[M] \rangle$ and remaining budget B, we: we: (a) solve the BAP to determine an allocation $b[i], i \leq M$ given $(\mathbf{s}, B)$; (b) execute the action $\pi(s[i], b[i])$ prescribed by the optimal BMDP policy for each i, incurring the cost of each such action; and (c) observe the resulting state $\mathbf{s}'$ and remaining budget B' and repeat. This approach closely follows that of Meuleau *et al.* [32].

This approach has the advantage of (virtually) guaranteeing that the global budget B will not be overspent (if the BMDP policy randomizes, there may be a small chance of some small budget violation). It also has the benefit of implementing a form of recourse—should the initial budget allocated to some user no longer be useful due to, say, a transition to an unprofitable state, that budget can be effectively reallocated to a more profitable user (conversely, budget can be reclaimed for a user who makes an unlikely transition to a profitable state). The DBRA column in the table above shows the performance (over 100 trials) of this strategy. In each trial the global budget constraint is satisfied, yet the average return matches or exceeds the expected value of the BMDP policy, which attains its value at the (typically small) risk of violating the budget constraint. Indeed, in the case of the very constrained budget (1000), DBRA appears to offer the additional advantage of reallocating budget to more promising customers over time. It is unsurprising that reallocation helps most when budgets are tightly constrained, since many customers will be lacking the budget needed to exact a high value in the initial allocation, but can benefit from the budgets gradually freed-up from others.

We also compare GBA to uniform budget allocation (UBA), which seems to be the most reasonable alternative if one does not have access to the BMDP solution. We do the comparison on the same MDP, but with 1000 customers uniformly spread among the 12 non-terminal, non-conversion states

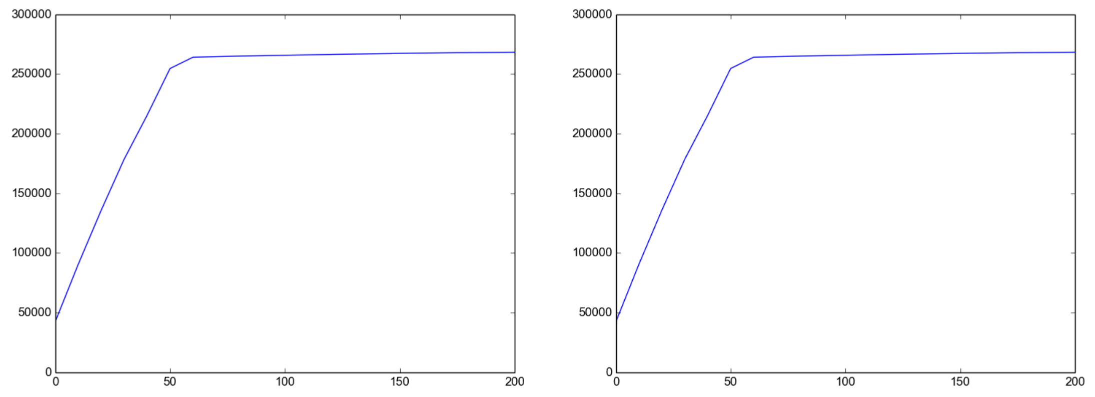


Fig. 7: Budget-value tradeoffs for (a) 451K-state BMDP; (b) 1469-state BMDP. Both are for 1000 customers evenly distributed over the 50 states with the largest value gaps.

(the methods will obviously be identical in the setting above, where all customers begin the process in the same state). The table below shows (exact) expected value of GBA and UBA (where the optimal BMDP policy is then implemented given the corresponding budget allocations) for several global budgets.

Total Budget	GBA Value	UBA Value
1000	39818.6	36997.2
2000	44559.5	40311.8
5000	53177.7	47142.4
10,000	58356.8	53773.8

We see that using the optimal BMDP solution to make budget tradeoffs among customers in different states offers significantly greater value than using a uniform scheme.

Advertiser MDPs We also examine the results of applying GBA on the advertiser-based MDPs described above. We first use GBA to derive "sweet spot" curves for both advertiser MDPs. In each case, we assume 1000 customers, with 20 customers each entering the process in the 50 states with the largest "value spans" (i.e., difference in expected value at the state given the minimal and maximal useful budget). Fig. 7 shows the budget-value tradeoff for both MDPs.

These MDPs model behavior that is quite random (i.e., not influenced very strongly by the actions).[27] As a consequence, once the GBA algorithm is run, there is not a great difference between implementing the BMDP policy, or implementing the SUB policy (as discussed above) with users at these states. The table below shows results for the large, 451K-state BMDP for two fairly constrained budget levels (DBRA, SUB are averaged over 50 trials, BMDP and UBA values are exact expected values).

[27] Indeed, since a large fraction of the states (about 75%) have no-ops as optimal actions at any budget level, maximal useful budgets tend to be relatively small.

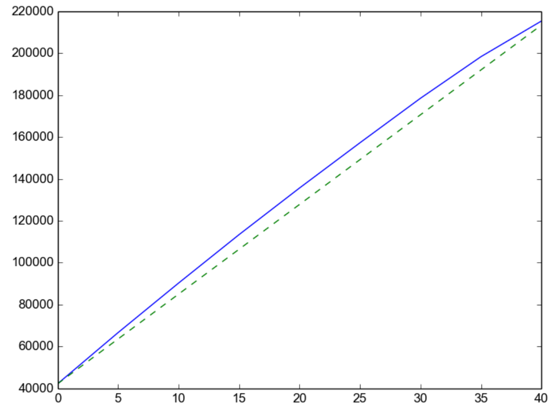


Fig. 8: Difference in expected value obtained by GBA (blue solid line) vs. UBA (green dashed line); 481K states, 1000 customers.

Total Budget	BMDP Value	DBRA Value	SUB Value	UBA Value
15	113358	99236 (3060)	112879 (1451)	106373
25	157228	142047 (3060)	157442 (2589)	149175

Neither BMDP or SUB exhibits much variance in spend and both give similar expected values; and the SUB policy rarely overspends (e.g., the maximum overspend for SUB with $B = 25$ is 0.16%). Variance tends to be greater when budgets are tighter. Among the 50 BMDP trials, 14 instances exceed the global budget, though only four instances exceed the spend by more than 4.0% (and one does so by 8.6%). As above, the dynamic reallocation scheme can eliminate the risk of overspending, but has little effect in this instance (indeed, we see it has a negative impact on expected value).

The table also shows that GBA offers greater expected value than UBA (which is one of the only viable options if the BMDP has not been solved). Fig. 8 shows the expected value of both GBA and UBA for a range of reasonably constrained global budgets. GBA exceeds UBA by up to 6.5% in this budget range.

6 Concluding Remarks

We have addressed the problem of budget (or other resource) allocation in MDPs in a way that allows budget-value tradeoffs to be addressed effectively. Our first contribution was the formulation of *budgeted MDPs*, an alternative view of constrained MDPs that allows value to be derived as a function of available budget (as opposed to fixing a specific budget constraint). We characterized the structure of optimal VFs and developed a DP algorithm that exploits the PWLC form of these VFs. We showed how this method can be approximated and how the results can be used to discover the sweet spot in the tradeoff between value and expected budget used (or other resources).

Our second contribution was a method for exploiting the solution to a BMDP to allocating budget across a variety of independently operating processes. Casting the problem as a multi-item, multiple-choice knapsack problem, we showed how a simple greedy algorithm can be used to rapidly optimally allocate budget in a "committed" fashion. We also demonstrated that dynamic reallocation of budget over time can reduce the risk of exceeding the expected budget (and could in some circumstances offer a form of recourse that improves overall return).

The extension of our methods to account for dynamic user populations is straightforward, but warrants empirical investigation. Future work includes the further study of and experimentation with our algorithms on richer models of user behavior. We are also interested in extending our models to partially observable settings (e.g., where user type estimation based on behavioral observations is needed).

References

[1] Daniel Adelman and Adam J. Mersereau. Relaxations of weakly coupled stochastic dynamic programs. *Operations Research*, 56(3):712–727, 2008.

[2] Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall, London, 1999.

[3] Eitan Altman, Konstantin Avrachenkov, Nicolas Bonneau, Merouane Debbah, Rachid El-Azouzi, and Daniel Sadoc Menasche. Constrained cost-coupled stochastic games with independent state processes. *Operations Research Letters*, 36(2):160–164, 2008.

[4] 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.

[5] Nikolay Archak, Vahab Mirrokni, and S. Muthukrishnan. Budget optimization for online advertising campaigns with carryover effects. In *Proceedings of the Sixth Workshop on Ad Auctions*, Cambridge, MA, 2010.

[6] 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.

[7] Nikolay Archak, Vahab S. Mirrokni, and S. Muthukrishnan. Mining advertiser-specific user behavior using adfactors. In *Proceedings of the 19th International Conference on World Wide Web (WWW 2010)*, pages 31–40, 2010.

[8] Christian Borgs, Jennifer Chayes, Omid Etesami, Nicole Immorlica, Kamal Jain, and Mohammad Mahdian. Bid optimization in online advertisement auctions. In *Workshop on Sponsored Search Auctions*, 2006.

[9] Craig Boutilier, Ronen I. Brafman, and Christopher Geib. Prioritized goal decomposition of Markov decision processes: Toward a synthesis of classical and decision theoretic planning. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 1156–1162, Nagoya, 1997.

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

[11] Craig Boutilier, Tuomas Sandholm, David C. Parkes, and William E. Walsh. Expressive banner ad auctions and model-based online optimization for clearing. In *Proceedings of the Twenty-third AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 30–37, Chicago, 2008.

[12] Bart J. Bronnenberg. Advertising frequency decisions in a discrete Markov process under a budget constraint. *Journal of Marketing Research*, 35(3):399–406, 1998.

[13] Peter Bühlmann and Abraham J. Wyner. Variable-length Markov chains. *The Annals of Statistics*, 27(2):480–513, 1999.

[14] David Castanon. Approximate dynamic programming for sensor management. In *Proceedings of the 36th IEEE Conference on Decision and Control*, pages 1202–1207, San Diego, 1997.

[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] Flavio Chierichetti, Ravi Kumar, Prabhakar Raghavan, and Tamas Sarlos. Are web users really Markovian? In *Proceedings of the 21st International World Wide Web Conference*, pages 609–618, Lyon, 2012.

[17] Bhaskar DasGupta and S. Muthukrishnan. Stochastic budget optimization in internet advertising. *Algorithmica*, 65(3):634–661, 2013.

[18] Dmitri Dolgov and Edmund Durfee. Stationary deterministic policies for constrained mdps with multiple rewards, costs, and discount factors. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1326–1331, Edinburgh, 2005.

[19] Dmitri A. Dolgov and Edmund H. Durfee. Optimal resource allocation and policy formulation in loosely-coupled markov decision processes. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004)*, pages 315–324, Whistler, BC, 2004.

[20] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[21] Gustav Feichtinger, Richard F. Hartl, and Suresh P. Sethi. Dynamic optimal control models in advertising: Recent developments. *Management Science*, 40(2):195–226, 1994.

[22] Eugene A. Feinberg. Constrained discounted Markov decision processes and Hamiltonian cycles. *Mathematics of Operations Research*, 25(1):130–140, 2000.

[23] 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.

[24] Arpita Ghosh, Preston McAfee, Kishore Papineni, and Sergei Vassilvitskii. Bidding for representative allocations for display advertising. In *Fifth Workshop on Internet and Network Economics*, pages 208–219, 2009.

[25] Ronald L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1(4):132–133, 1972.

[26] Duncan M. Holthausen, Jr. and Gert Assmus. Advertising budget allocation under uncertainty. *Management Science*, 28(5):487–499, 1982.

[27] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[28] Chinmay Karande, Aranyak Mehta, and Ramakrishnan Srikant. Optimizing budget constrained spend in search advertising. In *Sixth ACM International Conference on Web Search and Data Mining (WSDM-13)*, pages 697–706, Rome, 2013.

[29] Sébastien Lahaie, David C. Parkes, and David M. Pennock. An expressive auction design for online display advertising. In *Proceedings of the Twenty-third AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 108–113, Chicago, 2008.

[30] 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.

[31] Francisco S. Melo and Manuela Veloso. Decentralized MDPs with sparse interactions. *Artificial Intelligence*, 175(11):1757–1789, 2011.

[32] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas Dean, and Craig Boutilier. Solving very large weakly coupled Markov decision processes. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 165–172, Madison, WI, 1998.

[33] S. Muthukrishnan, Martin Pál, and Zoya Svitkina. Stochastic models for budget optimization in search-based advertising. In *Internet and Network Economics*, number 4858 in Lecture Notes in CS, pages 131–142. Springer, 2007.

[34] Pieter Otter. On dynamic selection of households for direct marketing based on Markov chain models with memory. *Marketing Letters*, 18(1):73–84, 1981.

[35] Scott Proper and Prasad Tadepalli. Solving multiagent assignment Markov decision processes. In *Proceedings of the Eighth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-09)*, pages 681–688, Budapest, 2009.

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

[37] Jorma Rissanen. A universal data compression system. *IEEE Transactions on Information Theory*, 29(5):656–664, 1983.

[38] 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.

[39] Satinder P. Singh and David Cohn. How to dynamically merge Markov decision processes. In *Advances in Neural Information Processing Systems 10*, pages 1057–1063. MIT Press, Cambridge, 1998.

[40] Prabhakant Sinha and Andris A. Zoltners. The multiple-choice knapsack problem. *Operations Research*, 27(3):503–515, 1979.

[41] Jack Z. Sissors and Roger B. Baron. *Advertising Media Planning (6th Ed.)*. McGraw-Hill, Chicago, 2002.

[42] Matthijs T. J. Spaan and Francisco S. Melo. Interaction-driven Markov games for decentralized multiagent planning under uncertainty. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-08)*, pages 525–532, Estoril, Portugal, 2008.

[43] 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.

[44] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

[45] Stefan J. Witwicki and Edmund H. Durfee. Influence-based policy abstraction for weakly-coupled DEC-POMDPs. In *Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS-10)*, pages 185–192, Toronto, 2010.

A Appendix: Proofs of Main Results

We provide proofs of the main results in this appendix.

Proposition 1. *For any finite t and state i, $V^t_D(i, \cdot)$ has a finite number of useful budget levels.*

Proof. We refer to Section 4.2, including the algorithm for computing useful budget levels and piece-wise constant VFs, for relevant definitions. The proof is a simple inductive argument that, for any

$i \in S$, $V_D^t(i,b)$ is piecewise constant in b with finite number of useful budget levels whenever V_D^{t-1} is. As a base case, it is easy to see that this holds for $V_D^0(i,b) = u_i$ for all b.

Assume this holds for V_D^{t-1}, and for ease of exposition, assume that each state j has $M+1$ useful levels $b_0^{j,t-1}, \ldots, b_M^{j,t-1}$. We first show that $Q_D^t(i,a,b)$ is finite and piecewise constant in $b \geq c_i^a$. Let Σ_i^a be the set of mappings from S_i^a into $\{0,1,\ldots,M\}$, where $\sigma(j) = m$ denotes the optimal use of (expected) budget level $b_m^{j,t-1}$ if state j is reached with $t-1$ stages to go after taking action a at state i. This gives an expected value of $v_m^{j,t-1}$. Hence the expected value of doing a followed by implementing σ is $EV(a,\sigma) = r_i^a + \gamma \sum_j p_{ij}^a v_{\sigma(j)}^{j,t-1}$ and has expected cost $EC(a,\sigma) = c_i^a + \gamma \sum_j p_{ij}^a b_{\sigma(j)}^{j,t-1}$.

We claim that the (potentially) useful budget levels for (i,a,t) are only those of the form $EC(a,\sigma)$. Suppose to the contrary that some b, distinct from these, is useful for (i,a,t); note that $b \geq c_i^a$ since action a cannot be taken otherwise. This means that there is an expected spend vector $\mathbf{b} = (b_j)_{j \in S_i^a}$ at stage $t-1$ such that: (a) the total expected spend $EC(a,\mathbf{b}) = c_i^a + \gamma \sum_j p_{ij}^a b_j \leq b$ and expected value is $EV(a,\mathbf{b}) = r_i^a + \gamma \sum_j p_{ij}^a V_D^{t-1}(j)$; and (b) there is no σ with $EC(a,\sigma) \leq b$ and $EV(a,\sigma) \geq EV(\mathbf{b})$. Let σ' be the "tightest lower bound" on $\mathbf{b}$, i.e., the unique $\sigma' \in \Sigma_i^a$ s.t., for all j, $b_{\sigma'(j)}^{j,t-1} \leq b_j$ and $b_{\sigma'(j)+1}^{j,t-1} > b_j$ (if $b_j \geq b_M^{j,t-1}$, we waive the second condition). Since V^{t-1} is piecewise constant, it follows immediately that $EV(a,\sigma') = EV(a,\mathbf{b})$ at state i. But $EC(a,\mathbf{b}) \geq EC(a,\sigma')$ at i, contradicting the existence of a distinct useful budget level.

The correctness of the algorithm for pruning dominated budget levels in the representation of $Q^t(i,a,b)$ is easily seen: the budget level for mapping σ is pruned iff there is some σ' where $EC(a,\sigma') \leq EC(a,\sigma)$ and $EV(a,\sigma') \geq EV(a,\sigma)$. Since the highest-valued such mapping σ' can be realized after taking a with lower expected cost than σ, the Q-value for a at budget level b is given by the mapping with highest value with cost $EC(a,\sigma') \leq b$.

Since $V^t(i,b) = \max_a Q^t(a,i,b)$, the finiteness of useful budget levels for $V^t(i,\cdot)$, and its piecewise constant nature, follows immediately from the finiteness of the action set A. And the correctness of the pruning phase of the algorithm holds by the same reasoning given above for pruning the Q-function.

□

Proposition 2. *For any finite t and state i, $V_D^t(i,\cdot)$ has at most $O((|A|^d)^t)$ useful budget levels, where d is the maximum out-degree of the underlying MDP.*

Proof. Since action choice at any state i can be conditioned on the budget, potentially different decisions can be made at state i at each stage-to-go $h \leq t$ within the horizon t. Thus the set of all t-stage non-stationary policies determines all possible behavioral choices, and each non-stationary policy has a unique expected cost (i.e., determines a potentially useful budget level) for (i,t). The number of non-stationary policies of depth t rooted at state i is $|A|^D$ where D is the number of distinct decision points in a depth t policy tree; and $D = \sum_{h=1}^t d^h = \frac{d^{t+1}}{d-1} - 1 = O(d^t)$. The result follows.

□

Theorem 3. *The $Q^t(i,a,b)$ defined in Defn. 1 gives the maximal expected value that can be extracted when taking action a at state i with available budget b when future value given by V^{t-1}.*

Proof. Let $\mathbf{b}$ be the allocation of budget $b - c_i^a$ after taking a at i (i.e., budget remaining) to next states S_i^a as determined by the greedy allocation outlined in Defn. 1. If the greedy algorithm determines a suboptimal Q-value for (i, a, b), there must be an allocation $\mathbf{b}' \neq \mathbf{b}$ s.t. $\sum p_{ij}^a b_j' \leq b - c_i^a$, where $EV(\mathbf{b}') > EV(\mathbf{b})$. This means $b_j < b_j'$ and $b_k > b_k'$ for some $j, k \in S_i^a$. For ease of exposition, assume that the allocations differ only on these two next states (generalizing to multiple next states is straightforward). We can assume w.l.o.g. that $\sum_j p_{ij}^a b_j = \sum_j p_{ij}^a b_j'$, hence that $\delta(k) = b_k - b_k'$ and $\delta(j) = b_j' - b_j$ satisfy $\delta(j) = \frac{p_{ik}^a}{p_{ij}^a}\delta(k)$ (i.e., we obtain $\mathbf{b}'$ by shifting budget from state k to state j at rate $\frac{p_{ik}^a}{p_{ij}^a}\delta(k)$).

Recall that $B(S_i^a)$ denotes the (sorted) set of budget points over all successor states of i in V^{t-1}. Let m_j denote the index of the segment of $V^{t-1}(j, b_j)$ in which b_j lies, and similarly for m_k; i.e, $b_{m_j-1}^{j,t-1} < b_j \leq b_{m_j}^{j,t-1}$ and $b_{m_k-1}^{k,t-1} < b_k \leq b_{m_k}^{k,t-1}$. Notice that the greedy algorithm must allocate budget to at least one of j or k that satisfies the upper bound with equality (only one state in S_i^a can have budget different than one of its useful budget points). We prove the result for the case of $b_j = b_{m_j}^{j,t-1}$ and $b_k = b_{m_k}^{k,t-1}$; the arguments when either b_j or b_k differs from the end point of its segment is analogous.

Suppose $\delta(j) \leq b_{m_j}^{j,t-1} - b_{m_j-1}^{j,t-1}$ and $\delta(k) \leq b_{m_k+1}^{k,t-1} - b_{m_k}^{k,t-1}$. Then the loss in future value by removing $\delta(j)$ from the budget available at j at stage $t-1$ is

$$p_{ij}^a \cdot BpB(b_{m_j-1}^{j,t-1}) \cdot \delta(j),$$

while the gain from adding $\delta(k)$ to k's future budget is

$$p_{ik}^a \cdot BpB(b_{m_k}^{k,t-1}) \cdot \delta(k).$$

Since segment m_{j-1} was added to $\mathbf{b}$ by the greedy algorithm, but m_k was not, we know $BpB(b_{m_j-1}^{j,t-1}) \geq BpB(b_{m_k}^{k,t-1})$.

The difference in expected value between $\mathbf{b}$ and $\mathbf{b}'$ is solely due to the lesser allocation to state j (in $\mathbf{b}$ relative to $\mathbf{b}'$), Loss($\delta(j)$), and the greater allocation to k, Gain($\delta(k)$). Thus:

$$\begin{aligned}
\text{Loss}(\delta(j)) &= p_{ij}^a \cdot BpB(b_{m_j-1}^{j,t-1}) \cdot \delta(j) \\
&\geq p_{ij}^a \cdot BpB(b_{m_k}^{k,t-1}) \cdot \delta(j) \\
&= p_{ij}^a \cdot BpB(b_{m_k}^{k,t-1}) \cdot \frac{p_{ik}^a}{p_{ij}^a} \cdot \delta(k) \\
&= BpB(b_{m_k}^{k,t-1}) \cdot p_{ik}^a \cdot \delta(k) \\
&= \text{Gain}(\delta(k)).
\end{aligned}$$

This contradicts the claim that $EV(\mathbf{b}') > EV(\mathbf{b})$.

If $\delta(j)$ reduces the j's budget below the preceding budget point, the loss per unit budget decreases is even greater due to the PWLC nature of the value function; similarly, if $\delta(k)$ increases k's budget greater than the next budget point, the per-unit gain is even less.

□

Theorem 4. *For any finite t and state i, $V^t(i,b)$ is piecewise linear, concave and monotone in b.*

Proof. The proof is analogous to that for deterministic policies. $\square$

Proposition 5. *For any finite t and state i, $V^t_D(i,\cdot)$ has at most $O((|A|^d)^t)$ useful budget levels, where d is the maximum out-degree of the underlying MDP.*

Proof. This follows directly from Prop. 2. $\square$

Theorem 6. *For any state i, the optimal infinite-horizon value function $V(i,b)$ is concave and monotone in b.*

Proof. We provide a simple proof sketch. For monotonicity, suppose by way of contradiction that, for some i and $b' > b$, we have $V(i,b) > V(i,b')$. Since any policy that realizes $V(i,b)$ with an expected spend of b can be implemented at state i with budget b', this is not possible.

For concavity, suppose by way of contradiction that, for some i, there exist three budget points $b_1 < b_2 < b_3$, with $b_2 = \alpha b_1 + (1-\alpha)b_3$ for some $\alpha \in (0,1)$, s.t. $V(i,b_2) < \alpha V(i,b_1) + (1-\alpha)V(i,b_3)$. Let $V(i,b_1), V(i,b_3)$ be realized by policies π_1, π_3, respectively. Since the policy that mixes these two policies with probabilities α and $(1-\alpha)$, respectively, has expected value greater than $V(i,b_2)$ but uses an expected budget of b_2, this contradicts the hypothesized non-concavity. $\square$

Proposition 10. *The optimal solution to the LP relaxation of the IP formulation Eq. 4 of the budget allocation problem is such that, for each i: (a) $x_{ik} = 1$ for one value of k; or (b) there is some k such that only $x_{ik}, x_{i,k+1} > 0$ (i.e., only two budget levels are allocated, and they must be consecutive).*

Proof. The proof follows that of [40]) and is omitted. $\square$