Multitasking: Efficient Optimal Planning for Bandit Superprocesses

Dylan Hadfield-Menell and Stuart Russell

Abstract

A bandit superprocess is a decision problem composed from multiple independent Markov decision processes (MDPs), coupled only by the constraint that, at each time step, the agent may act in only one of the MDPs. Multitasking problems of this kind are ubiquitous in the real world, yet very little is known about them from a computational viewpoint, beyond the observation that optimal policies for the superprocess may prescribe actions that would be suboptimal for an MDP considered in isolation. (This observation implies that many applications of sequential decision analysis in practice are technically incorrect, since the decision problem being solved is often part of a larger, unstated bandit superprocess.) The paper summarizes the state-of-the-art in the theory of bandit superprocesses and contributes a novel upper bound on the global value function of a bandit superprocess, defined in terms of a direct relaxation of the arms. The bound is equivalent to an existing bound (the Whittle integral), but is defined constructively, as the value of a related multi-armed bandit. We provide a new method to compute this bound and derive the first practical algorithm to select optimal actions in bandit superprocesses. The algorithm operates by repeatedly establishing dominance relations between actions using upper and lower bounds on action values. Experiments indicate that the algorithm's run-time compares very favorably to other possible algorithms designed for more general factored MDPs.

1 INTRODUCTION

Multitasking is no doubt an activity familiar to the reader: one faces several decision problems but can act on only one (or perhaps a bounded number) at a time. Such problems are ubiquitous for individuals, corporations, armies, and governments.

Multitasking problems are expressed by the class of *bandit superprocesses* [Nash, 1973] or BSPs. A k-armed BSP M consists of k *independent* Markov decision processes $M_1, \ldots, M_k$; the MDPs are coupled by a common discount factor and by the constraint that at each time step the agent can act in only one MDP.

Since the MDPs are independent of each other, one might imagine that the optimal policy for M is obtained by solving each MDP—turning the BSP into a multi-armed bandit—and then using Gittins indices to choose a sequence of arms. In fact, *the optimal policy for a BSP may include actions that are suboptimal from the point of view of the constituent MDP in which they are taken*. The reason for this is that the availability of other MDPs in which to act changes the balance between short-term and long-term rewards in a component MDP; in fact, it tends to lead to greedier behavior in each MDP because aiming for long-term reward in one MDP would delay rewards in all the other MDPs. The globally and locally optimal policies necessarily coincide only when the discount factor is 1.

Hence, in addition to their general importance in the real world, a second reason to study multitasking problems is that they undermine an assumption implicit in practical applications of sequential decision analysis: the assumption that an optimal solution for the user's decision problem is optimal for a user who faces multiple decision problems.

Despite these considerations, there has been remarkably little research on bandit superprocesses and almost none in AI. Section 2 summarizes what is known. Obviously, there are connections to multi-armed bandits (MABs), which are a special case of BSPs in which each arm only allows one choice of action rather than several. Unfortunately, the index theorems that simplify the computation of optimal MAB policies are not valid for BSPs. There are also strong connections—hitherto unexplored—between BSPs and *sums of games* as studied in combinatorial game theory [Conway, 2000]. The principal question we address in this paper is whether an algorithm exists that is substantially more efficient than applying a standard MDP solver to the "cross-product" MDP obtained by combining the states spaces of the constituent MDPs.

The sole known case where planning for a BSP is provably more efficient is the case where a *dominating* policy exists: if a single policy is optimal across the family of *retirement processes* associated with each arm then a BSP can be reduced to an equivalent MAB [Whittle, 1980]. (Retirement processes provide a measure of how the optimal policy depends on context and are defined in Section 3.) However, this condition is seldom satisfied in practice.

This paper provides three contributions. First, we give a concise survey of the bandit superprocess literature tailored to the AI community. Second, we provide a novel upper bound on the global value function of a BSP. For a BSP, M, with arms $\{M_1, \dots, M_k\}$ we relax each arm to obtain an MAB $M' = \{M'_1, \dots, M'_k\}$ by adding actions to ensure that dominating policies exists. We show that this upper bound is equivalent to the *Whittle integral*, an existing upper bound for BSPs [Brown and Smith, 2013]. Because our bound is defined in terms of an explicit relaxation, it provides insight into the nature of the bound and opens an avenue to extend this work to more general MDPs. Finally, we describe a practical computational approach for solving BSPs: we derive a simple method for computing the Whittle integral upper bound that can use an arbitrary MDP solver as a black box; then we combine this upper bound with a lower bound to derive an efficient algorithm for ϵ-optimal decision making in a BSP. We present empirical results to show that it substantially improves over more general optimal factored MDP algorithms; we use it to compute provably optimal actions for problems with upwards of 10^{30} states in the full state space.

2 RELATED WORK

Robbins [1952] provided the first formulation of multi-armed bandits (MABs) in their modern form. The famous Gittins index theorem [Gittins, 1979] showed that optimal MAB policies are obtained by ranking the arms according to an index function defined on each seperately. An immediate corollary is that optimal decision making in an MAB is *linear* in the number of arms. Problems with this property are said to be *indexable*.

Bandit superprocesses (BSPs) were introduced by Nash [1973] as a generalization of multi-armed bandits to study allocation of resources among research projects. Whittle [1980] provided an alternate proof of the Gittins index theorem that extends to bandit superprocesses with dominating policies. His proof utilized a construction called the *Whittle integral*, which allows one to compute the value of a composite state in an MAB. Glazebrook [1982] provides a proof that bandit superprocesses are *not* indexable in general. Glazebrook [1993] considers bandit superprocesses where the arms can exert limited influence on each other and shows a result analogous to Whittle's.

Brown and Smith [2013] were the first to identify the signifigance of the Whittle integral for sequential decision making. They developed a version of policy iteration to compute it, and recognized that it upper bounds the value function of a BSP,[1] but did not derive an algorithm for solving BSPs. Our work provides two further contributions regarding the Whittle integral: a short proof that it is an upper bound and a simpler algorithm to compute it.

Within the AI community, there has been limited study of loosely coupled Markov decision processes. Singh and Cohn [1998] consider optimal solutions to *simultaneous MDPs*, where an agent can take actions in a number of MDPs. Their formulation is more general than ours, as it is possible to act in multiple MDPs at once. They derive bounds for this problem and give an algorithm that combines a form of real-time dynamic programming with pruning steps to remove provably suboptimal actions. However, their bounds are substantially looser than ours, and our experiments show that the corresponding algorithm can be impractical for simple BSPs with many arms.

Meuleau et al. [1998] examine MDPs that are coupled by constraints on the use of shared resources. BSPs fit within this class if we view the restriction to a single MDP at a time as a constraint on the agent's attention. Interestingly, their heuristics are also defined in terms of a parameterized value function for the component MDPs—it would be interesting to attempt to generalize the approaches considered here to their problem domain. In this work, we leverage the particular structure of our resource to compute optimal solutions; Meuleau et al. construct a heuristic policy.

3 TECHNICAL BACKGROUND

MARKOV DECISION PROCESSES

Definition 1. *(Markov Decision Process [Puterman, 2009]) A (finite-state, discounted) MDP, M, is a tuple $M = \langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$. $\mathcal{S}$ is a set of states. $\mathcal{A}$ is a set of actions. $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ is a function that assigns probability to state transitions for each state–action pair. R is a (bounded) reward function that maps state–action pairs to (positive) rewards $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^+$. $\gamma \in [0, 1)$ is a discount factor.*

A solution to M is a policy, π, that maps states to actions. The value of a state, s, under π is the sum of expected discounted rewards received by starting in s and selecting ac-

[1] This result first appears in the literature as an intermediate step in a larger proof from Whittle [1980]. It went unnoticed or unappreciated in the intervening 30+ years.

tions according to π:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | s_0 = s, \pi\right].$$

The optimal policy, π^*, maximizes this value. In the above definition, and the ones that follow, we use superscripts to indicate dependence on the agent's policy. To simplify notation, we will omit these superscripts when the policy referred to is the optimal policy (e.g., $V(s) = V^{\pi^*}(s)$). The Q-function for the state–action pair, (s, a), is the value of taking a in s and selecting future actions according to π^*.

RETIREMENT PROCESSES

Given an MDP, Whittle defines a family of optimal stopping problems:

Definition 2. *(Retirement Process [Whittle, 1980]) Let M be an MDP. For $\rho \geq 0$, the retirement process for M with retirement reward, ρ, is an MDP, M_ρ, with a single additional state, s_R, and action, a_R. a_R transitions deterministically to s_R and receives reward ρ. s_R is a sink state that accrues zero reward.*

We refer to a decision to select a_R as a decision to retire. We denote the retirement process value function as a function of a state and retirement reward, $V(s, \rho)$. We let the optimal policy for retirement reward ρ be π^*_ρ. We write the set of states where the policy, π, retires as τ^π_ρ. We use $\rho_-(\pi)$ and $\rho_+(\pi)$ to denote the interval of retirement rewards so that π is optimal:

$$\rho' \in [\rho_+(\pi), \rho_+(\pi)] \Rightarrow V^\pi(s, \rho) = V(s, \rho).$$

We adopt the convention from the MAB/BSP literature and abuse notation to denote the (random) number of steps prior to retirement as $\tau^\pi_\rho(s)$. For $s' \in \tau^\pi_\rho$ we let $P_{retire}(s'|s, \rho, \pi)$ be the probability that s' is the first state in τ^π_ρ the agent will reach given that it is in state s and executes policy π. We denote the expected discounted reward accrued prior to retirement, starting in s, as $R^\pi_\rho(s)$. In regions of retirement reward where the optimal policy and stopping rule do not change, $\frac{\partial V}{\partial \rho}(s, \rho)$ is defined and is equal to the expected value of the discount parameter at retirement. This allows us to write the following expression for the retirement process value function:

$$V(s, \rho) = R_\rho(s) + \mathbb{E}[\gamma^{\tau_\rho(s)}]\rho. \tag{1}$$

$V(s, \rho)$ is piecewise linear in ρ. Figure 1 shows $V(s, \rho)$ for the example BSP in Ex. 1. A policy that is optimal for every setting of ρ is called a *dominating policy*.

Definition 3. *(Dominating Policy) Let π be a policy for an MDP, M. π is a* dominating policy *iff*

$$\forall \rho \geq 0 \, \forall s \notin \tau_\rho \;\; \pi(s) = \pi^*_\rho(s).$$

MULTI-ARMED BANDITS AND BANDIT SUPERPROCESSES

Multi-armed bandits (MAB) are a restricted class of MDPs that have received extensive study. Of particular interest are the form of the optimal policy and its utility in modelling "exploration vs exploitation" trade-offs.

An MAB consists of a set of Markov reward processes (MRPs), where an MRP is an MDP with a single action. Each MRP is referred to as an *arm* of the problem. At each time-step, the agent selects an arm, that arm transitions according to its transition distribution, and the agent receives the corresponding reward. We adopt a summation notation to indicate the combination of several MRPs into an MAB. For example, if X and Y are MRPs and Z is the MAB with arms X and Y, we will write $Z = X + Y$.

A famous result is that the optimal policy for any MAB is an *index policy* [Gittins, 1979]. Each state, s_i, in the individual arms is assigned an index, I_{s_i}. In joint state $s = \{s_1, \ldots, s_k\}$, the optimal action is to select $\text{argmax}_i \, I_{s_i}$. For MAB arm M_i in state s_i, the index is defined as the value of retirement reward such that the agent is indifferent between immediate retirement and following the optimal stopping policy [Whittle, 1980]:

$$I_{s_i} = \min_\rho \{\rho \, ; \, V_i(s_i, \rho) = \rho\}.$$

This means that, in a sense, context for a multi-armed bandit can always be summarized by a single number.

To model a multi-tasking problem, we consider a generalization of multi-armed bandits: bandit superprocesses (BSPs). Bandit superprocesses allow arms that are arbitrary MDPs (and so are a generalization of MABs); at each time step, the agent selects both an arm *and* an action to take within that arm.

Definition 4. *(Bandit Superprocess [Nash, 1973]) Given k MDPs, $\{M_i = \langle \mathcal{S}_i, \mathcal{A}_i, T_i, R_i, \gamma \rangle\}$, we define*

$$M = \sum_i M_i = \langle \mathcal{S}, \, \mathcal{A}, \, T, \, R, \, \gamma \, \rangle$$

to be the bandit superprocess with arms $\{M_i\}$. $\mathcal{S} = \times_i \mathcal{S}_i$ and $\mathcal{A} = \cup_i \mathcal{A}_i$. The transition distribution is stationary for arms that are not selected and follows the identical reward and transition distributions for the selected arm.

Naturally, this makes planning more difficult. To see how, consider the following proposal:

Conjecture 1. *Let $X = \langle \mathcal{S}_X, \mathcal{A}_X, T_X, R_X, \gamma \rangle$ be an MDP. Define Y similarly and let $Z = X + Y$ be the bandit superprocess that is their sum. Let $a \in \mathcal{A}_X$, $s_X \in \mathcal{S}_X$ be a state–action pair from X. Suppose that this transition is suboptimal in every retirement process: $\forall \, \rho \, \geq \, 0, \, V_X(s_X, \, \rho) \, > \, Q_X((s_X, \, a), \, \rho)$. Then a is* *suboptimal for any state in Z with s_X as one of the components:*

$$\forall s_Y \in \mathcal{S}_Y, V_Z(\{s_X, s_Y\}) > Q_Z(\{s_X, s_Y\}, a).$$

This conjecture essentially proposes that state–action pairs can be safely ignored if they are suboptimal for all settings of a constant alternative. Unfortunately, as the following example illustrates, this is not the case.

Example 1. *Define X to be a deterministic reward chain and Y to be an initial choice between three reward chains (y_0, y_1, y_2) defined as follows:*

$$X = 28, 28, 28, 28, 28, 28, 0, 0 \ldots$$
$$Y = \begin{cases} y_0 = & 100, 100, 100, 0, 0, 0, 0 \ldots \\ y_1 = & 99, 99, 99, 1.4, 1.4, 1.4 \ldots \\ y_3 = & 28, 28, 28, \ldots \end{cases}$$

The retirement process value function for each reward chain can be computed by simulating the Gittins index policy:[2]

$$V(y_0, \rho) = \max\left\{100\left(\sum_{t=0}^{2} \gamma^t\right) + \gamma^3 \rho, \rho\right\}$$
$$V(y_1, \rho) = \max\left\{99\left(\sum_{t=0}^{2} \gamma^t\right) + \gamma^3 \max\left\{\frac{1.4}{1-\gamma}, \rho\right\}, \rho\right\}$$
$$V(y_2, \rho) = \max\left\{\frac{28}{1-\gamma}, \rho\right\}.$$

If we let $\gamma = .9$, then we have

$$\forall 280 > \rho \geq 0 \max\{V(y_0,\ \rho),\ V(y_2,\ \rho)\}\ >\ V(y_1,\ \rho).$$

Thus, any policy for a retirement process retirement process derived from Y that initially selects y_1 is suboptimal. We can apply the same strategy to compute the value of each combination of reward streams:

$$V(y_0 + X) = 100\left(\sum_{t=0}^{2} \gamma^t\right) + \gamma^3\left(28 \sum_{t'=0}^{5} \gamma^{t'}\right)$$
$$V(y_1 + X) = 99\left(\sum_{t=0}^{2} \gamma^t\right) + \gamma^3\left(28 \sum_{t'=0}^{5} \gamma^{t'} + \gamma^6 \frac{1.4}{1-\gamma}\right)$$
$$V(y_2 + X) = \frac{28}{1-\gamma}$$

From this we can see that the optimal policy for the BSP that combines these MDPs, $Z = X + Y$, initially collects reward from y_1. This contradicts Conjecture 1.

4 AN UPPER BOUND FOR BANDIT SUPERPROCESSES

In this section we show a bound on the value function of a bandit superprocess. We derive this bound by adding actions to a BSP so that it is equivalent to an MAB. The value function of our relaxed BSP is equivalent to the Whittle integral bound derived in Brown and Smith [2013] and so it yields insight into that computation. We begin by defining the Whittle integral and show some basic results in order to motivate our relaxation.

Definition 5. *(Whittle Integral [Brown and Smith, 2013]) Let M be a BSP. Let i index the arms of M. For any state, $s = \{s_i\}$, and $\rho \geq 0$, the Whittle integral of s is defined as*

$$\hat{V}(s, \rho) = I - \int_{x=\rho}^{I} dx \prod_i \frac{\partial V_i}{\partial \rho}(s_i, x). \tag{2}$$

Where $I \geq \max_i I_{s_i}$.

When the arms of a BSP admit a dominating policy the Whittle integral is equal to the value function:

Theorem 1. *(Whittle Condition [Whittle, 1980]) Let M be a k-armed BSP with components $\{M_i\}$ and state space $\mathcal{S}$. If each M_i has a dominating policy, then*

$$\forall s \in \mathcal{S}, \forall \rho \geq 0, \hat{V}(s, \rho) = V(s, \rho).$$

MRPs have a single action per state, so the Whittle condition is trivially satisfied for all multi-armed bandits. $V(s, 0) = V(s)$, so $\hat{V}(s, 0)$ provides an efficient method to compute the value of an MAB. For BSPs an arm that satisfies the Whittle condition can be replaced with an MRP that selects actions according to π^*.

The formula in Eq. 2 lends itself to a straightforward implementation, but is very challenging to interpret. We show below that this is equivalent to evaluating the retirement process value function for a single arm with a set of rewards determined by the other arms[3]. We refer to this set of rewards as the *critical points* of those arms.

Definition 6. *(Critical Points of an MDP) Let M be a Markov decision process. The critical points of M, $\mathcal{C}(M) = \{\rho_i\}$, are the values of retirement reward such that the optimal stopping rule or policy changes.*

These are points where there is a discontinuity in $\frac{\partial V}{\partial \rho}$. We let

$$\Delta^M(s, \rho) = \lim_{\delta \to 0} \frac{\partial V_M}{\partial \rho}(s, \rho + \delta) - \frac{\partial V_M}{\partial \rho}(s, \rho - \delta)$$

be the size of the corresponding discontinuities. This is equivalent to the expected increase in $\mathbb{E}\left[\gamma^\tau\right]$ under the new stopping rule. Theorem 2 shows that Δ and $\mathcal{C}$ characterize the interaction between arms of an MAB.

[2] Here we use the fact that the Gittins index of a non-increasing reward sequence is equal to the instantaneous reward.

[3] Our result is a small extension on a related result in Brown and Smith [2013]; it is primarily included to provide intuition.

Theorem 2. *Let X, Y be Markov reward processes. Let $Z = X + Y$ be the 2-armed bandit. $\forall s = \{s_X, s_Y\} \in \mathcal{S}_Z$,*

$$V_Z(s) = \sum_{\rho \in \mathcal{C}(Y)} V_X(s_X, \rho)\Delta^Y(s_Y, \rho). \qquad (3)$$

Proof. (sketch) As a first step, we follow the steps in Whittle [1980] and integrate Equation 2 by parts. This expresses $\hat{V}$ as the following integral:

$$\hat{V}_Z(s) = \int_{\rho'=\rho}^{I_{s_X}} V_X(s_X, \rho') \frac{\partial^2 V_Y(s_Y, \rho')}{\partial \rho^2} d\rho'.$$

V_Y is piecewise linear with respect to ρ so $\frac{\partial^2 V_Y}{\partial \rho^2}$ is a weighted sum of delta functions centered at V_Y's kinks. $\mathcal{C}(Y)$ and Δ^Y respectively characterize these kinks and weights. Thus, $\hat{V}_Z(s)$ is equal to the rhs of Eq. 3. X and Y are MRPs, so an appeal to Theorem 1 shows the result. □

A similar property holds for arbitrary k-armed bandits. This shows that we can summarize the context for an arm as a collection of weighted retirement rewards. Turning to BSPs, this lends insight into the approximation the Whittle integral introduces.

To see how, we reconsider Ex. 1. Recall that this example consists of the BSP $Z = X + Y$ and that Y is a choice between three reward chains, $\{y_0, y_1, y_2\}$. Theorem 2 allows us to write the gap between the Whittle integral upper bound and the value of selecting y_1 as

$$\sum_{\rho \in \mathcal{C}(X)} \left(\max_i V_{y_i}(s_i, \rho) - V_{y_1}(s_1, \rho) \right) \Delta^X(s_X, \rho)$$

Figure 1 shows the retirement process value functions for this example. While the retirement process value of y_1 is always less than that of either y_0 or y_2, it is close enough to their maximum that choosing y_1 essentially achieves the upper bound. The Whittle integral for Z is a weighted combination of distinct retirement process value functions (V_{y_0}, V_{y_2}) but, in reality, the agent will be forced to pick a single policy of the two.

DOMINATED RELAXATION OF AN MDP

In this section, we introduce our primary theoretical result: a relaxation for the arms of a BSP so that a dominating policy exists. This reduces the BSP to an MAB whose value upper bounds the value of states in the BSP. We can show that the Whittle integral computes the value of states in this MAB and arrive at a straightforward proof that the Whittle integral is an upper bound. We call the result the *dominated relaxation* of an MDP. Before providing the details, we illustrate the main ideas with an example. The relaxed MDP is actually a *Semi-MDP* (SMDP): a generalization of an MDP where each action, a, has a duration, $\delta(a) \in \mathbb{R}_+$.

Example 2. *Let MDP M be an initial choice between MRPs:*

$$M = \begin{cases} 15, 0, 0, 0 \ldots \\ 10, 10, 10, 3, 3, 3, \ldots \end{cases}.$$

Let T (B) be the top (bottom) MRP. Let π_T (π_B) be the policy that selects T (B) in s_0. We can relax M with the addition of a single durative action, a', that transitions from the sink state in T to the sink state B. We set $\delta(a') = 2$. We take ρ such that $V^{\pi_T}(s_0, \rho) = V^{\pi_B}(s_0, \rho)$ and set the rewards associated with a' to be

$$R_\rho^{\pi_B}(s_0) - R_\rho^{\pi_T}(s_0) = 10 \sum_0^2 \gamma^t - 15 = 12.1 .$$

With this change, at low settings of retirement reward, the agent is indifferent between a policy that opts for the top chain then selects a' and the bottom reward chain. For high settings of retirement reward, the optimal policy retires immediately or retires after collecting the reward of 15. This SMDP satisfies the Whittle condition and can be replaced by an MRP in any bandit superprocess. Fig. 1 (c) shows an illustration of the state space and the introduced action.

In this example, we connected the state where π_T retires to the state where π_B retires. This lets the agent collect short-term and long-term rewards with the same policy. To do this in general, we introduce multiple copies of the state space, one for each policy that is optimal for some ρ.

Definition 7. *(Dominated Relaxation of an MDP) Let M be an MDP with discount factor γ and state space $\mathcal{S}$. Let s be a state in M. The dominated relaxation of M for s, $M_D(s)$, is a semi-Markov decision process that fixes s as an initial state. Let $\{\pi_i\}$ be the policies that are optimal for some ρ: $\{\pi^*_\rho | \rho \in \mathcal{C}(M)\}$. This sequence is ordered so that $\rho_-(\pi_i)$ is increasing in i[4].*

For each i, we introduce a copy of the state space, $\mathcal{S}_i$, where the agent is restricted to following π_i. Let s'_i be the analogue of s' in $\mathcal{S}_i$. For $s'_i \in \tau_{\rho_-(\pi_i)}$, we introduce a single durative action, a_i, that takes the agent from $\mathcal{S}_i$ to $\mathcal{S}_{i-1}$ and characterize it as follows:

- $R(s'_i) = R_{\rho_+(\pi_{i-1})}(s) - R_{\rho_-(\pi_i)}(s)$
- $T(s'_i, a_i, s''_{i-1}) = P_{retire}(s'' | \pi_{i-1}, \rho_+(\pi_{i-1}), s)$
- $\delta(a_i) = \log_\gamma \mathbb{E}\left[\gamma^{\tau_{\rho_+}^{\pi_{i-1}}(s)}\right] - \log_\gamma \mathbb{E}\left[\gamma^{\tau_{\rho_-}^{\pi_i}(s)}\right]$

For each i, we introduce an action that transitions from s to s_i with $\delta = 0$. Let $V_D(s)$ represent the value of s in $M_D(s)$.

Theorem 3. *Let M be an MDP with state space $\mathcal{S}$. The following statements are true for $s \in \mathcal{S}$ and $\rho \geq 0$:*

[4] Recall that $[\rho_-(\pi), \rho_+(\pi)]$ is the interval of retirement rewards where π is optimal.

1. *$M_D(s)$ satisfies the Whittle condition.*

2. *$V_D(s, \rho) = \hat{V}(s, \rho)$.*

3. *$\hat{V}(s, \rho) \geq V(s, \rho)$*

Proof. See supplementary materials □

5 AN ϵ-OPTIMAL ALGORITHM FOR BANDIT SUPERPROCESSES

In this section, we present two algorithms related to BSPs. The first is a novel algorithm to compute $V(s, \rho)$ that can use any method to solve the underlying MDP. The second is an efficient algorithm to compute optimal actions for a BSP. Our approach, *Branch-and-Bound Value Iteration* (BBVI), uses Whittle integrals to compute upper and lower bounds on value. Then, we apply a modified Branch-and-Bound search to find provably optimal actions.

COMPUTING A RETIREMENT PROCESS VALUE FUNCTION

Before we present BBVI, we give a simple algorithm to compute a retirement process value function that uses an (arbitrary) MDP solver as a black box. Brown and Smith [2013] describe an algorithm to compute retirement process value functions, but their approach requires custom implementation of a modified simplex algorithm. Our approach is simple and based on an algorithm that initially appeared in the solutions manual for Russell and Norvig [2010].

Our goal is to identify each component of $V(\cdot, \rho)$, so our output will be a list of critical points and the associated slopes. First, we will need the following result.

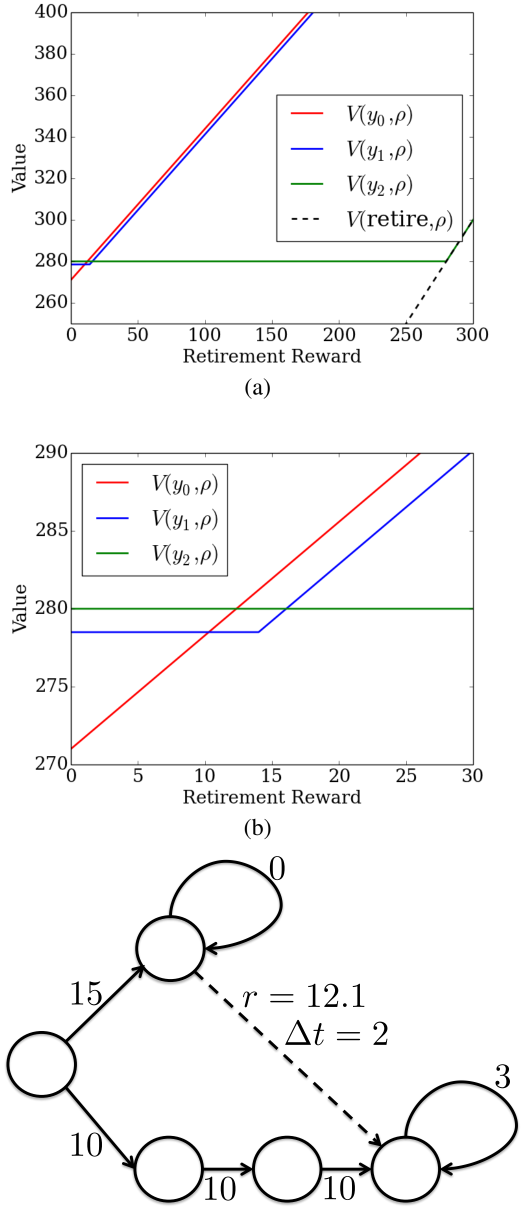
Lemma 1. *Let M be an MDP with state space $\mathcal{S}$. Let $\rho_0 < \rho_1$. Consider a retirement reward in the interior of this interval, $\rho \in (\rho_0, \rho_1)$. If, $\forall s \in \mathcal{S}$*

$$V(s, \rho) = V(s, \rho_0) + \frac{V(s, \rho_1) - V(s, \rho_0)}{\rho_1 - \rho_0}(\rho - \rho_0), \quad (4)$$

then there is at least one policy and stopping rule that is optimal for every $\rho' \in [\rho_0, \rho_1]$. The expected value of the discount factor at retirement is the slope between those points:

$$\mathbb{E}\left[\gamma^{\tau^*(s)}\right] = \frac{V(s, \rho_1) - V(s, \rho_0)}{\rho_1 - \rho_0}. \quad (5)$$

Proof. This follows from the form of Equation 1 and the fact that $V(s, \rho)$ is piecewise linear, increasing, and convex. □



(a)

(b)

(c)

Figure 1: (a-b) Retirement process value functions for the reward chains in Ex. 1. The slope of the lines is the expected value of the discount parameter at retirement: $\mathbb{E}\left[\gamma^{\tau}\right]$. Flat sections indicate regions of retirement rewards where retirement is always suboptimal. The kink in the green curve at $\rho = 280$ indicates that it has become optimal to retire immediately. The kink in the black curve at $\rho = 14$ increases the slope to $\gamma^6 < 1$; for $\rho > 14$, it is optimal to retire after collecting a prefix of the reward stream from y_2. (c) Dominated relaxation of the MDP in Ex. 2. We add additional durative actions to the state space to ensure that a dominating policy exists. Note that the original MDP does not admit a dominating policy.

This test allows us to implement a binary search over retirement rewards. Our approach relies on a given MDP solution method, SOLVE, that returns the vector of values for a given MDP. We initialize an interval that contains all critical points and make a call to SOLVE to compute the retirement process value at each endpoint. Then we call SOLVE to compute the retirement process value for the midpoint of our interval and apply the test in Eq. 4. If it succeeds, we return the lower endpoint and the corresponding slope. If this does not, we sub-divide our interval and recurse. We can concatenate the results to get the breakpoints and slopes for the retirement process value function over this interval. Algorithm 1 shows pseudocode for this algorithm.

Algorithm 1 Computing an MDP's Critical Points

```
Define: CRITICALPOINTS($M[, \rho^-, \rho^+, V^-, V^+]$)
Input: MDP, $M$; Interval of retirement values, $\rho^-, \rho^+$;
Values at interval endpoints, $V^-, V^+$
if $\rho^-, \rho^+$ are not set then
    $\rho^- \leftarrow 0$
    $\rho^+ \leftarrow \frac{\text{maxReward}(M)}{1-\gamma}$
    $V^- \leftarrow$ SOLVE($M_{\rho^-}$)
    $V^+ \leftarrow \rho^+$ ## Retirement is initially optimal
end if
$\rho \leftarrow \frac{\rho^+ - \rho^-}{2}$
$V \leftarrow$ SOLVE($M_\rho$)
if $|V - \left(V^- + (\rho - \rho^-)\frac{V^+ - V^-}{\rho^+ - \rho^-}\right)| < \epsilon$ then
    return $[\rho^-], \left[\frac{V^+ - V^-}{\rho^+ - \rho^-}\right]$
else
    $pts^-, slope^- \leftarrow$CRITICALPOINTS($M, \rho^-, \rho, V^-, V$)
    $pts^+, slope^+ \leftarrow$CRITICALPOINTS($M, \rho, \rho^+, V, V^+$)
    ## Merge adjacent intervals with the same slope
    if $slope^-[-1] = slope^+[0]$ then
        del $pts^+[0]$,del $slope^+[0]$
    end if
    return $pts^- \cup pts^+, slopes^- \cup slopes^+$
end if
```

BRANCH-AND-BOUND VALUE ITERATION

Now, we present a practical algorithm to compute optimal actions in a bandit superprocess. While a BSP is not indexable, we would like to be able to plan efficiently when the arms are 'close' to indexable—when there are only a few states where the optimal policy changes in response to the opportunity cost of delayed rewards in other arms.

Our approach is based on envelope dynamic programming methods to solve MDPs: we compute value estimates for a given initial state by solving a dynamic program defined over a reachable subset of the state space [Gardiol, Natalia H and Kaelbling, Leslie P, 2003, Hansen and Zilberstein, 2001]. The primary difference between our approach and standard envelope methods is that we use dynamic programming on our envelope to update an upper bound *and* a lower bound on the value. This is useful in a BSP because the Whittle integral allows efficient calculation of both bounds.

Our goal is to compute the optimal action for a given state s in a BSP, $M = \langle M_1, \ldots, M_K \rangle$. We can obtain an upper bound on $V(s)$ with a Whittle integral. We use a Whittle integral for the MAB that solves each arm independently to compute a lower bound. Our algorithm maintains upper and lower bounds on the Q-function for each action that could be executed from s. We write these bounds as $Q^+(s, a)$ and $Q^-(s, a)$ respectively. If there is a pair of actions a, a' such that $Q^+(s, a') < Q^-(s, a) + \epsilon$, then we can conclude that a' is at least ϵ-suboptimal. If this test removes all but a single action, then we have found a ϵ-optimal action and return it.

In the event that more than one action remains, we do a partial expansion of the state space around s. We keep track of a set of expanded states, $\mathcal{E}$, and a set of boundary states, $\mathcal{B}$. States in the boundary set, $\mathcal{B}$, are states that some expanded state can transition to but have not yet been added to $\mathcal{E}$. We can use these states to update the bounds on $Q(s, \cdot)$ by solving a related MDP. This is formalized in the following theorem.

Theorem 4. *Let M be an MDP with state space $\mathcal{S}$ and action space $\mathcal{A}$. Let $\mathcal{S}' \subseteq \mathcal{S}$ where $\mathcal{S}' = \mathcal{B} \cup \mathcal{E}$ and $\forall\, s \in \mathcal{E}, T(s, a, s') \neq 0 \Rightarrow s' \in \mathcal{S}'$. Let M^+ be an MDP with state space $\mathcal{S}' \cup \{\alpha\}$. M^+'s transition distribution and rewards are identical for states in $\mathcal{E}$ (expanded states). Each state $s \in \mathcal{B}$ transitions deterministically to α and receives a reward that is an upper bound on $V_M(s)$. α is a sink state that accrues 0 reward. Then the value of a state in M^+ is an upper bound on its value in M:*

$$V_{M^+}(s) \geq V_M(s).$$

Proof. See supplementary materials. □

It is straightforward to show that an updated lower bound can be computed by fixing boundary states to have value equal to a lower bound. Our approach alternates between pruning actions based on dominance relations, adding states to $\mathcal{E}$, and computing the value of an MDP defined over $\mathcal{E}$. We interleave value iteration with a branch-and-bound search, so we call it Branch-and-Bound Value Iteration (BBVI). Algorithm 2 shows pseudocode to implement this method. At termination, the action we return is guaranteed to be at most ϵ-suboptimal.

Theorem 5. *Let M be a BSP and let s be a state in M. Let a be the action returned by $BBVI(M, s, \epsilon)$.*

$$V(s) - Q(s, a) < \epsilon.$$

Proof. See supplementary materials. □

Large regions of the state space are irrelevant to our objective so we use a heuristic to guide node expansion. Our heuristic is a measure of the sensitivity of values at the root to change in the upper and lower bounds of unexpanded nodes. Similar heuristics have been applied in many settings [Rivest, 1987, Smith and Simmons, 2004]. In BBVI, these values are the expected discounted visitation rates in the bounding MDPs and are computed as the dual variables from a value iteration LP. The backup procedure is slower than node expansions, so we perform backups in batches. In between backups and heuristic computations, we approximate the state space as a tree and push new states onto the agenda with their parent's heuristic value weighted by the transition probability.

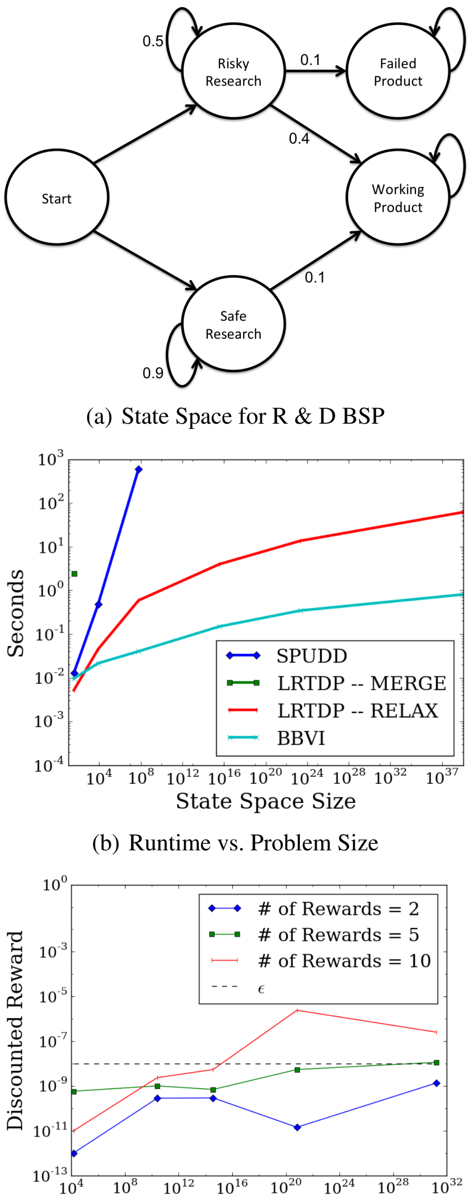
Algorithm 2 Branch-and-Bound Value Iteration

```
Define: $\text{BBVI}(\langle M_0, \ldots, M_K \rangle, s_0, \epsilon)$
Input: BSP arms, $M_k$; Initial state, $s_0$; Tolerance, $\epsilon$
# Lower bound $M$ by fixing a policy for each arm
$LB_k \leftarrow$toMRP$(M_k)$
Compute critical points for $M_k, LB_k$
Compute bounds on $Q(s_0, \cdot)$ with Whittle integrals for $M$ and $LB$.
# Keep track of expanded region of state space
$\mathcal{E} \leftarrow \emptyset$
# Keep track of states at boundary of $M^+, M^-$
$\mathcal{B} \leftarrow \{s_0\}$
$a^* \leftarrow \underset{a}{\text{argmax}} Q^-(s_0, a)$
while $\exists a' \neq a^*\ s.t\ Q^+(s_0, a') \geq Q^-(s_0, a^*) + \epsilon$ do
    $s \leftarrow$pop$(\mathcal{B})$
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{s\}$
    for $a \in \mathcal{A}$ do
        for $s' \in successors(s)$ do
            Compute upper and lower bounds for $Q(s', \cdot)$
            $\mathcal{B} \leftarrow \mathcal{B} \cup \{s'\}$
        end for
    end for
    $Q^+ \leftarrow \text{SOLVE}(\text{BOUNDMDP}(\mathcal{E}, \mathcal{B}, Q^+))$
    $Q^- \leftarrow \text{SOLVE}(\text{BOUNDMDP}(\mathcal{E}, \mathcal{B}, Q^-))$
    $a^* \leftarrow \underset{a}{\text{argmax}} Q^-(s_0, a)$
end while
return $\epsilon$-optimal action $a^*$
```

6 EMPRIRICAL EVALUATION

We implemented BBVI in Python and use the linear programming solver Gurobi to compute value functions. Our experiments were run on an Intel i7 with 16 GB of RAM. In our first experiments, we compare the scalability of BBVI with that of standard MDP algorithms. In particular we compare the following algorithms:

- SPUDD: the baseline factored MDP algorithm in the 2011 IPPC [Hoey et al., 1999].



(a) State Space for R & D BSP

(b) Runtime vs. Problem Size

(c) Solution Quality vs. Problem Size

Figure 2: (a) the state space for an arm of a R & D BSP. All states accrue 0 reward except for the 'Working Product' state, which collects a fixed reward per time step. (b) shows runtime (log-scale) vs problem size for BBVI and several alternatives on an R & D BSP. SPUDD [Hoey et al., 1999] and LRTDP (with the heuristic from Singh and Cohn [1998]) scale poorly compared with algorithms that leverage the Whittle integral. BBVI's additional improvement over LRTDP stems from the use of an efficient, exact lower bound to check convergence. (c) BBVI's bound on suboptimality after 10,000 node expasions (measured in units of discounted reward). Changing the number of rewards in each arm allows us to measure BBVI's performance as arms become more sensitive to context.

- LRTDP(Merge): Labeled Real-Time Dynamic Programming [Bonet and Geffner, 2003] with the upper bound from Singh and Cohn [1998] as a heuristic.
- LRTDP(WI): LRTDP with the Whittle integral as an upper bound.
- BBVI: Branch-and-Bound Value Iteration.

We evaluate performance on a simple BSP designed to model allocation of research resources to product research and development. Each arm in this problem corresponds to a potential product that our agent could sell in the market. However, before we sell a product we must devote effort to R&D. This can be done in one of two ways, a safe research approach that is slow but reliable, and a risky approach that is fast, but runs a risk of producing a defective product.

After at least one product has been researched, the agent can opt to spend a round to produce and sell that product or continue research on a different project. Because taking a product to market does not change the state of the BSP, it is straightforward to show that any stationary policy (including π^*) will only ever produce a single product. We define a distribution over these problems by selecting a random market value for each product uniformly from [0, 1] and random durations for the safe and risky research strategies (although we ensure that safe research is at least 3x as slow as risky research). Figure 2 (a) shows the state space for an example arm with typical parameters.

Figure 2 (b) shows the results of our comparison. We used a timeout of 1000s and set a memory limit of 4 GB. We ran our experiments in an online setting and running times reflect the amount of time to select an optimal action from scratch. While this is a natural setting for LRTDP and BBVI, it is admittedly a little unfair to SPUDD (which computes a full policy). This is mitigated by the fact that BBVI's improvement over SPUDD, which is a little under 5 orders of magnitude with 10^8 states, is such that an agent would need to be planning over an immense horizon before SPUDD presents a reasonable alternative.

The factored MDP bounds from Singh and Cohn [1998] are quite ill-suited to this problem. This is because its upper bound (the sum of the independent value for each arm) is very loose and it essentially forces LRTDP to enumerate the state space. In contrast, LRTDP performs quite well when it has good heuristic information. However, its performance degrades faster than that of BBVI, because it does not effectively leverage a lower bound on value. Even for very large problem instances, BBVI computes optimal solutions in well under a second.

BBVI's performance in the R&D domain is largely explained by the structure of the arms and the short horizon within each arm. Although there may be a large number of arms, BBVI reduces each individual arm to an equivalent MRP after a small number of node expansions. This means that running time is essentially linear in the number of arms. Note that, in this domain, the solution that solves each MDP independently and executes the corresponding index policy will just opt for conservative research on the most valuable option. This policy is essentially always suboptimal: as long as there are reasonable alternatives it is usually a good idea to try some risky projects.

Our next experiment uses a synthetic domain to explore the performance of BBVI as the structure and number of arms changes. In this BSP, each arm is a 20x20 grid world. The actions in this world correspond to moving in the 4 cardinal directions. Rewards are 0 everywhere except at randomly chosen locations and after receiving a reward, the agent transitions to a random location.

In this experiment we vary the number of rewards that are available in each arm. Adding rewards will typically cause the optimal policy for an arm to be more sensitive to context. This means that the dominated relaxation will produce a looser upper bound, because it will need to add more actions. We evaluate this difficultly by measuring the convergence criterion of BBVI after 10,000 node expansions. Figure 2 (c) shows the results of this experiment. We can see that BBVI's solution quality decreases with the number of states, but the primary variation comes from changes in the properties of the arms.

7 CONCLUSION AND FUTURE WORK

In summary, we presented a model of highly decoupled decision making: bandit superprocesses. A BSP presents an agent with the opportunity to act in one of several Markov decision processes. The key constraint is that the agent can only act in a single process at a time. We provided a summary of BSP research, including an upper bound for the value function of a BSP in the form of the Whittle integral. We derived an equivalent relaxation and thus gave a novel proof that the Whittle integral is an upper bound. Finally we presented and evaluated algorithmic solutions for this class of decision problems.

In future work, we plan to extend these ideas in three directions. The first is algorithmic improvements for BSPs. A potential direction here is to leverage factored dynamics in the bounding MDPs themselves. A search over sequences of MDPs that only consider states where the current policy stops could be more efficient. Next, also plan to explore generalizations of the bounds used for BSPs to more general cases of global resource constraints (e.g., those considered in Meuleau et al. [1998]). Finally, the similarity to sums of games studied in Conway [2000] suggests an unexplored connection between BSPs and combinatorial game theory.

References

Blai Bonet and Hector Geffner. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *IEEE Conference on Automated Planning and Scheduling*, pages 12–21, 2003.

David B. Brown and James E. Smith. Optimal sequential exploration: Bandits, clairvoyants, and wildcats. *Operations Research*, 61(3):644–665, 2013.

John H. Conway. *On Numbers and Games*. CRC Press, 2000.

Gardiol, Natalia H and Kaelbling, Leslie P. Envelope-based planning in relational MDPs. In *Advances in Neural Information Processing Systems*, page None, 2003.

John C. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 148–177, 1979.

Kevin D. Glazebrook. On a sufficient condition for superprocesses due to Whittle. *Journal of Applied Probability*, pages 99–110, 1982.

Kevin D. Glazebrook. Indices for families of competing Markov decision processes with influence. *The Annals of Applied Probability*, pages 1013–1032, 1993.

Eric A Hansen and Shlomo Zilberstein. Lao: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 129(1):35–62, 2001.

Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In *Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 279–288. Morgan Kaufmann Publishers Inc., 1999.

Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. Solving very large weakly coupled Markov decision processes. In *Fifteenth National Conference on Artificial Intelligence*, pages 165–172, 1998.

Peter Nash. *Optimal allocation of Resources Between Research Projects.* PhD thesis, University of Cambridge, 1973.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2009.

Ronald L. Rivest. Game tree searching by min/max approximation. *Artificial Intelligence*, 34(1):77–96, 1987.

Herbert Robbins. Some aspects of the sequential design of experiments. In *Herbert Robbins Selected Papers*, pages 169–177. Springer, 1952.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* 3 edition, 2010. Excercise 17.5.

Satinder Singh and David Cohn. How to dynamically merge Markov decision processes. *Advances in Neural Information Processing Systems*, (10):1057–1063, 1998.

Trey Smith and Reid Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 520–527. AUAI Press, 2004.

Peter Whittle. Multi-armed bandits and the Gittins index. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 143–149, 1980.