Sequential Resource Allocation Under Uncertainty: An Index Policy Approach

Weici Hu
Adviser: Peter Frazier

Cornell ORIE

July 6, 2017

Problem Setup

We consider an MDP $(\mathbb{S}^K, \mathbb{A}^K, \mathbb{P}^{\cdot}, R)$ that consists of K identical sub-processes $(\mathbb{S}, \mathbb{A}, P^{\cdot}, r)$, specifically,

- Time horizon $T < \infty$.
- State space $\mathbb{S}^K$ is the cross-product of K $\mathbb{S}$. $\mathbb{S}$ is assumed finite.
- Action space $\mathbb{A}^K$ is the cross-product of K $\mathbb{A}$. $\mathbb{A} = \{0, 1\}$.
- Reward $R_t(\mathbf{s}, \mathbf{a}) = \sum_{x=1}^{K} r_t(s_x, a_x)$, $1 \leq t \leq T$, is additive of the reward of individual sub-processes.
- Transition probability $\mathbb{P}^{\mathbf{a}}(\mathbf{s}', \mathbf{s}) = \prod_{x=1}^{K} P^{a_x}(s'_x, s_x)$.

Problem Setup Con't

- A Markov policy $\boldsymbol{\pi} : \mathbb{S}^K \times \mathbb{A}^K \times \{1, ..., T\} \to [0, 1]$, with $\boldsymbol{\pi}(\mathbf{s}, \mathbf{a}, t) = P(\mathbf{a}|\mathbf{S}_t = \mathbf{s})$ (Our decision).
We require $\sum_{\mathbf{a}\in\mathbb{A}^K} \boldsymbol{\pi}(\mathbf{s}, \mathbf{a}, t) = 1$. $\forall \mathbf{s} \in \mathbb{S}^K, \forall 1 \le t \le T$.
- Objective

$$
\begin{aligned}
\underset{\boldsymbol{\pi}\in\boldsymbol{\Pi}}{\text{maximize}} \quad & \mathbb{E}^{\boldsymbol{\pi}}\left[\sum_{t=1}^{T} R_t(\mathbf{S}_t, \mathbf{A}_t)\right] \\
\text{subject to} \quad & P^{\boldsymbol{\pi}}(|\mathbf{A}_t| = m_t) = 1, \quad \forall 1 \le t \le T.
\end{aligned} \tag{1}
$$

Difficulty: Optimal solutions are computationally infeasible

Optimal solutions of (1) can be obtained with Bellman optimality equations.

But it requires $O(|\mathbb{S}|^K|\mathbb{A}|^K T)$ time complexity and $O(|\mathbb{S}|^K|\mathbb{A}|^K T)$ storage complexity.

The complexity grow exponentially with the number of sub-processes K, and becomes computationally infeasible for large K.

Past attempts

Pre-computations: 1. Optimal Lagrange Multiplier of (1)

Relax the original problem (1) to

$$\begin{aligned} \underset{\boldsymbol{\pi}\in\boldsymbol{\Pi}}{\text{maximize}} \quad & \mathbb{E}^{\boldsymbol{\pi}}\left[\sum_{t=1}^{T} R_t(\mathbf{S}_t,\mathbf{A}_t)\right] \\ \text{subject to} \quad & \mathbb{E}^{\boldsymbol{\pi}}(|\mathbf{A}_t|) = m_t, \quad \forall 1 \le t \le T. \end{aligned} \tag{2}$$

The Lagrangian relaxation of (2)

$$P(\boldsymbol{\lambda}) = \max_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \mathbb{E}^{\boldsymbol{\pi}}\left[\sum_{t=1}^{T} R_t(\mathbf{S}_t,\mathbf{A}_t)\right] - \sum_{t=1}^{T}\lambda_t\left(\mathbb{E}^{\boldsymbol{\pi}}[|\mathbf{A}_t|] - m_t\right). \tag{3}$$

Pre-computations: 1. Optimal Lagrange Multiplier of (1)

Decomposition of the Lagrangian relaxation

$$P(\boldsymbol{\lambda}) = KQ(\boldsymbol{\lambda}) + \sum_t \lambda_t m_t, \tag{4}$$

where

$$Q(\boldsymbol{\lambda}) = \max_{\pi \in \Pi} \mathbb{E}^{\pi} \left[\sum_{t=1}^{T} r_t(S_t, A_t) - \lambda_t A_t \right], \tag{5}$$

is the objective function for sub-process $(\mathbb{S}, \mathbb{A}, P^{\cdot}, r)$. Definition of policy π is similar to $\boldsymbol{\pi}$, with $\pi(s, a, t) = P(a | S_t = s)$.

Remark: Problem (5) can be solved using the Bellman recursion with complexity $O(|\mathbb{S}||\mathbb{A}|T)$, hence it is computationally feasible.

Pre-computations: 2. Occupation Measure ρ^*

The Optimal Policy Puts Samples Where They Help Most

Literature Review

Related work for MCS in simulation scenario:

- Frequentist work: Paulson (1952) and Dunnett (1955)(one-stage procedure, normal sampling); Dudewicz and Dalal (1983), Bofinger and Lewis (1992), Damerdji and Nakayama (1996) (Two stage procedures, more general sampling distributions)
- Similar work: Xie and Frazier (2013)(Fully sequential in a Bayesian setting)
    - We consider a finite horizon, rather than a geometric horizon and infinite horizon.
    - We allow cost per sample to be optional.
    - We look at allocating parallel simulation resources

Literature Review

Related work for crowdsourcing scenario:

- RayKar et al(2010), Whitehill et al(2009)(static inference); Thanh et al(2013)(static budget allocation); Karger et al(2013)(dynamic assignment of tasks but require large worker budget)
- Similar work: Chen et al(2013)(dynamic allocation under Bayesian framework with optimal policy in the form of a DP)
  - We offer an upper bound in addition to a heuristic
  - We consider a continuous time horizon and a M/M/k queue to model the flow of workers.

Problem set-up for simulation scenario

- k alternatives
- m parallel simulating resources per time step
- N time horizon
- θ_x is the underlying true performance of alternative x, $x \in \{1, ..., k\}$
- d_x the known threshold for alternative x
- $z_{n,x}$ is the number of simulation resources to use on alternative x at time step n. This is our allocation decision. $\sum_x z_{n,x} \leq m$
- After time step N, for each alternative x, we decide whether $\theta_x > d_x$ based on the past results of simulation
- We obtain a reward $R = \sum_x R_x$. R_x can be a 0-1 reward or linear reward.

Goal:Find an allocation of simulation resources to best support the decision at time N

Problem set-up for crowdsourcing scenario

- k labeling tasks
- N total workers
- T time horizon
- M/M/k queue: workers come in with rate r, and complete their job with rate μ.
- θ_x is the underlying likelihood for a task to have a positive label.
- d_x the known threshold for alternative x
- $z_{l,x} \in \{0,1\}$ indicates whether the l^{th} worker is assigned to task x.
- After worker budget N has been exhausted or time T has been reached, for each alternative x, we decide whether whether the true label is positive or negative based on a 1-0 reward.

Goal:Find an allocation of workers to best support the final decision on true labels

We use a Bayesian approach

Use the simulation-scenario as an example:

- $Y_{n,x}$ is the number of successes observed after we do $z_{n,x}$ simulations on alternative x at time n

$$Y_{n,x} | \theta_x, z_{n,x} \sim \mathrm{Binomial}(z_{n,x}, \theta_x)$$

- We use Beta as a conjugate prior θ_x

$$\theta_x \sim \mathrm{Beta}(\alpha_{0,x}, \beta_{0,x}).$$

$$\theta_x | z_{1,x}, Y_{1,x}, \ldots, z_{n,x}, Y_{n,x} \sim \mathrm{Beta}(\alpha_{n,x}, \beta_{n,x}).$$

The Bayesian approach for the crowdsourcing scenario works in similar manner.

We use a Bayesian approach

The allocation problem under Bayesian framework is

$$\sup_{\pi} \mathbb{E}^{\pi}\left[R \middle| \text{prior}\right] \tag{6}$$

Where a policy π is a mapping from histories onto allocations of resources with:

- $\mathbf{z}_n = (z_{n,1}, \ldots, z_{n,k}) \in \mathbb{N}^k$ satisfying

  $$\sum_{x=1}^{k} z_{n,x} \leq m$$

  for simulation scenario
- $\mathbf{z}_l = (z_{l,1}, ..., z_{l,k}) \in \mathbb{N}^k$ satisfying

  $$\sum_{x=1}^{k} z_{l,x} \leq 1$$

  for crowdsourcing scenario

Dynamic Programming gives an optimal solution

We formulate the simulation problem as a dynamic program with

- state at time n, $\mathbf{S}_n = (s_{n,1}, \ldots, s_{n,k})$ = posterior parameters of all the alternatives
- value function $V_n(\mathbf{S}_n)$ = the maximum expected total reward to be obtained from time step n onward given the current state $\mathbf{S}_n$.
- The optimal value is $V_0(\mathbf{S}_0) = (6)$
- The optimal policy π^* is the sequence of $\mathbf{z}_1^*, \ldots, \mathbf{z}_N^*$ that achieves the maximum in Bellman's recursion

Dynamic Programming gives an optimal solution

Similarly, we formulate the crowdsourcing problem as a dynamic program.

- Possible transitions: a worker comes into the system; a worker completes a task and leaves the system.
- $\mathbf{S}_n = (\boldsymbol{\alpha}, \boldsymbol{\beta}, t, \mathbf{w}, l)$, where n denotes n^{th} transition, $\boldsymbol{\alpha}, \boldsymbol{\beta}$ are posterior parameters, t is the time the n^{th} transition happens, $\mathbf{w}$ is the number of workers working on each task x, l is the total number of workers have arrived.
- Inter-transition time $\Delta_n \sim \text{Exp}(\mu|\mathbf{w}| + r)$
- The optimal value is $V_0(\mathbf{S}_0) = (6)$
- The optimal policy π^* is the sequence of $\mathbf{z}_1^*, \ldots, \mathbf{z}_N^*$ that achieves the maximum in Bellman's recursion

Problem: Dynamic Programming is computationally infeasible

- In the simulation scenario:
  - The number of states in state space at time n is $O((mn)^k)$.
  - Memory scales exponentially in k.
  - Computation scales exponentially in k.
  - E.g., $m = k = 8$, at time step $N = 5$, there are $2.35426 * 10^{12}$ states.
- Even more complex in crowdsourcing scenario.

Curse of Dimensionality!

Our approach: Upper bound+heuristics

- We first calculate an upper bound to the original problem.
- We then propose an index-based heuristic policy.
- We use the upper bound to measure the performance of the heuristic policy.

Forming an upper bound

Step 1 in forming an upper bound: Lagrangian Relaxation

- Original set of feasible policies:

$$\Pi = \{\pi = (\mathbf{z}_1, ..., \mathbf{z}_N) : \sum_{x=1}^{k} z_{n,x} \leq m\}$$

- Relaxed set of policies:

$$\Pi_1 = \{\pi = (\mathbf{z}_1, ..., \mathbf{z}_N) : \text{no constraint}\}$$

- Define $V_0^{\boldsymbol{\lambda}}(\mathbf{S}_0) = \sup_{\pi \in \Pi_1} \mathbb{E}^{\pi}[R - \sum_{n=1}^{N}(\lambda_n(\sum_{x=1}^{k} z_{n,x} - m))]$

Lemma (1)

For $\boldsymbol{\lambda} \geq 0$, we have

$$V_0^{\boldsymbol{\lambda}}(\mathbf{S}_0) \geq V_0(\mathbf{S}_0)$$

Step 2 in forming an upper bound: Decompose the relaxed DP to single-alternative DPs

- Define
  $V^{\boldsymbol{\lambda}}_{0,x}(S_{0,x}) = \sup_{\pi^{(x)} \in \Pi^{(x)}} \mathbb{E}^{\pi^{(x)}} \left[R_x - \sum_{n=1}^{N} \lambda_n z_{n,x} \middle| S_{0,x} \right], \forall x,$
  where
  $\Pi^{(x)} = \{\pi^{(x)} = (z_{1,x}, z_{2,x}, ..., z_{N,x}) : z_{n,x} \leq m, \forall 1 \leq n \leq N\}$

Lemma (2)

For any $\boldsymbol{\lambda} > \mathbf{0}$, let $V^{\boldsymbol{\lambda}}_{0,x}(S_{0,x})$ be the value of an single-alternative MCS problem on x,

$$V^{\boldsymbol{\lambda}}_0(\mathbf{S}_0) = \sum_{x=1}^{k} V^{\boldsymbol{\lambda}}_{0,x}(S_{0,x}) + m \sum_{n=1}^{N} \lambda_n, \tag{7}$$

- Solving single-alternative MCS problems by dynamic programming is computationally feasible.

Now we have a computable upper bound

- Lemma 1 and 2 hold for any $\boldsymbol{\lambda} \geq 0$, hence $\sum_{x=1}^{k} V_{0,x}^{\boldsymbol{\lambda}}(S_{0,x}) + m \sum_{n=1}^{N} \lambda_n$ forms an upper bound to MCS problem for any given $\boldsymbol{\lambda}$

Theorem

An upper bound on the optimal value of the original MCS problem is

$$\inf_{\boldsymbol{\lambda} \geq \mathbf{0}} \left[\sum_{x=1}^{k} V_{0,x}^{\boldsymbol{\lambda}}(S_{0,x}) + m \sum_{n=1}^{N} \lambda_n \right] \tag{8}$$

Compute the tightest possible upper bound

- $V_0^{\lambda}(\mathbf{S}_0)$ is a convex function about $\boldsymbol{\lambda}$.
- Use first-order gradient descent
- Subgradient at $\boldsymbol{\lambda}$:

$$(-\mathbb{E}^{\pi_x^*(\boldsymbol{\lambda})}[z_{n,x}|S_{0,x}] : n = 1, ..., N)$$

  where $\pi_x^*(\boldsymbol{\lambda})$ is the optimal policy for single-task MCS problem with cost $\boldsymbol{\lambda}$.
- Subgradient can be computed recursively.

This analysis also inspires this index policy

For simulation scenario:

- Let $\bar{\boldsymbol{\lambda}}$ achieves the infimum in (3).

- Let $z^{\boldsymbol{\lambda}}_{n,x}(S_{n-1,x})$ be the number of samples taken under an optimal single-alternative policy given $\boldsymbol{\lambda}$, breaking ties arbitrarily.

At each time step $n = 1, \ldots, N$,

1. Let $\beta^* = \inf \left\{ \beta \geq 0 : \sum_x z^{\beta\bar{\boldsymbol{\lambda}}}_{n,x}(S_{n-1,x}) \leq m \right\}$.
2. Set $\boldsymbol{\lambda}^* = \beta^* \boldsymbol{\lambda}$.
3. Let $\mathbf{z}_n = \{z^{\boldsymbol{\lambda}^*}_{n,x}, x = 1, ..., k\}$, breaking ties arbitrarily between different allocations $\mathbf{z}_n$ that satisfy this constraint.

This analysis also inspires this index policy

For crowdsourcing scenario:

While $\ell < N$:

1. For each $x \in \{1, \ldots, k\}$, compute $\beta_x^* = \inf\{\beta \geq 0 : z_{\ell,x}^{\beta\bar{\boldsymbol{\lambda}}} = 1\}$.
2. Let $x_* = \arg\max_x \beta_x^*$. Break tie arbitrarily.
3. Assign task x_* to the ℓ^{th} worker.

Numerical experiment

For each $k = 2, 4, 8, 16$

- $m = k$
- $d_x = 0.2, \forall x \in \{1, ..., k\}$
- $\mathbf{S}_0 = (\boldsymbol{\alpha}_0, \boldsymbol{\beta}_0) = (1, 1)^k$
- □ – Upper bound
- — 95% confidence interval with index policy, based on 10000 replications
- **—** 95% confidence interval with equal allocation policy, based on 50000 replications

Numerical experiment

Numerical experiment

For each $k = 4, 8, 12, 16$

- $m = 1.25k$
- $d_x = 0.5, \forall x \in \{1, ..., k\}$
- $(\alpha_0, \beta_0) = (1, 1)^k$
- $\square$ – Upper bound
- – 95% confidence interval with index policy, based on 10000 replications
- — 95% confidence interval with equal allocation policy, based on 50000 replications
- - - 95% confidence interval with optimistic knowledge gradient policy (Chen et al 2013), based on 10000 replications.

Numerical experiment

Conjecture: Index policy is asymptotically optimal

Conjecture: The index policy perform asymptotically close to an optimal policy as k tends to infinity while keeping the number of resources to task ratio (m/k or N/k) constant.

Future Work

- Prove the conjecture.
- Conduct numerical experiment on a larger scale
- Conduct experiment using index policy using Amazon Mechanic Turk in real time.
- Apply the same method to ranking & selection problems.

Publication strategy

- Accepted: "Parallel Bayesian policy for finite-horizon multiple comparisons with a known standard". Proceedings of the 2014 Winter Simulation Conference 2014.
- Rejected: "Bayes-Optimal Effort Allocation in Crowdsourcing: Bounds and Index Policies", ICML 2015.

Thank you!