Parallel Bayesian Policies for Multiple Comparisons with a Known Standard

Weici Hu
Peter Frazier

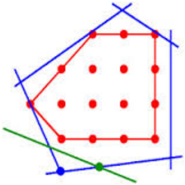
Operations Research & Information Engineering (ORIE)
Cornell University

CORS/INFORMS International Conference, Montreal
June 14, 2015

In MCS, we use simulation to determine which alternatives perform better than a threshold

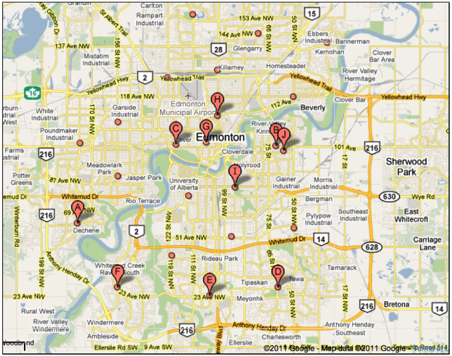
- MCS = multiple comparisons with a known standard
- An MCS problem involves k projects/systems and has a known threshold value d
- It uses simulation/experiment to decide which among the k projects perform better than the threshold

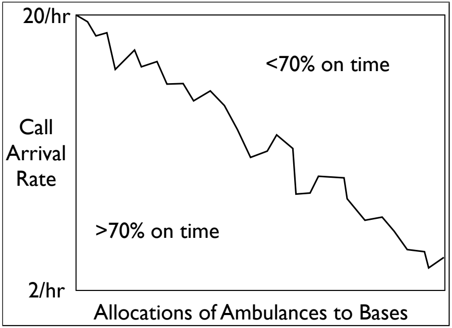
MCS problems appears in crowdsourced image labeling



MCS appears in Ambulance Positioning

We must allocate ambulances across 11 bases in the city of Edmonton. Which allocations satisfy mandated minimums for percentage of calls answered in time, under a variety of different possible call arrival rates?



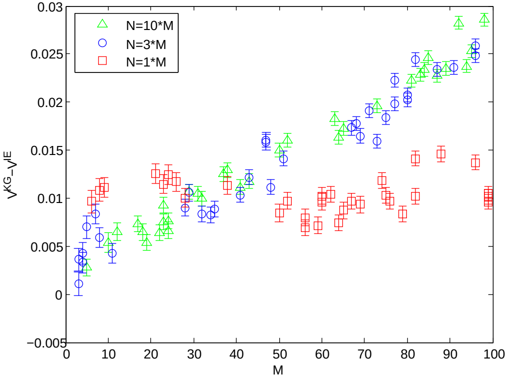


[Thanks to Shane Henderson and Matt Maxwell for providing the ambulance simulation]

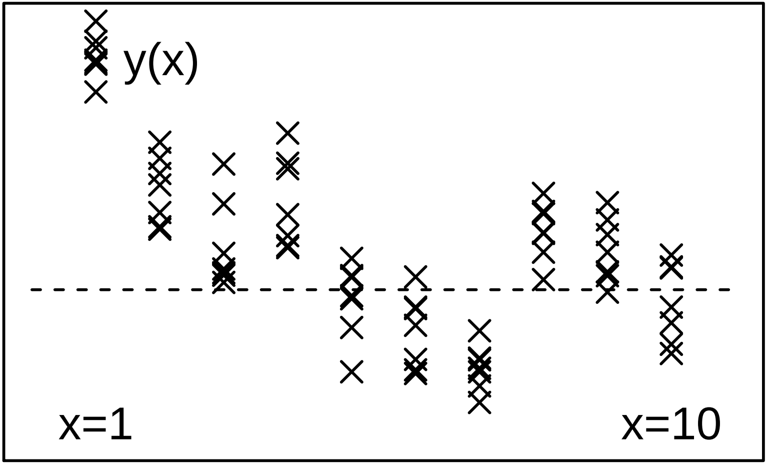
MCS appears in Algorithm Development

A researcher who develops a new algorithm would like to know:

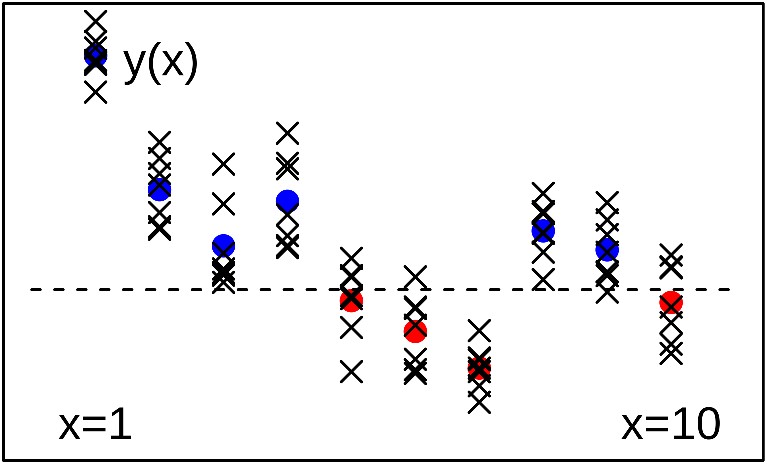
- In which problem settings is average-case performance better with Algorithm A than with Algorithm B?



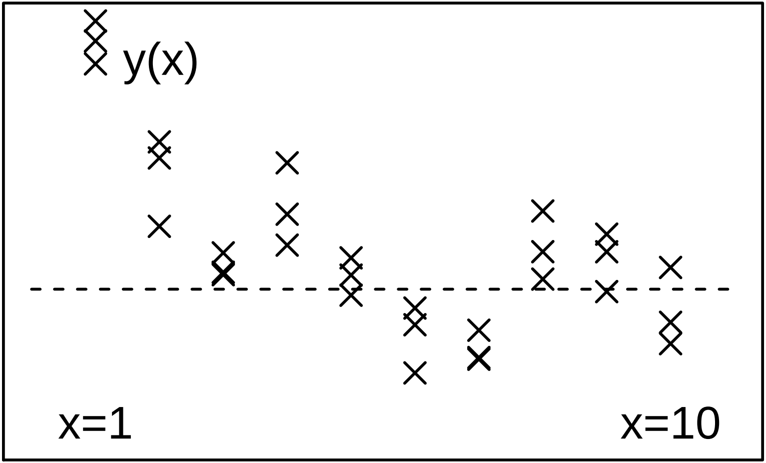
Given Samples, Estimating the Level Set is Well-Understood



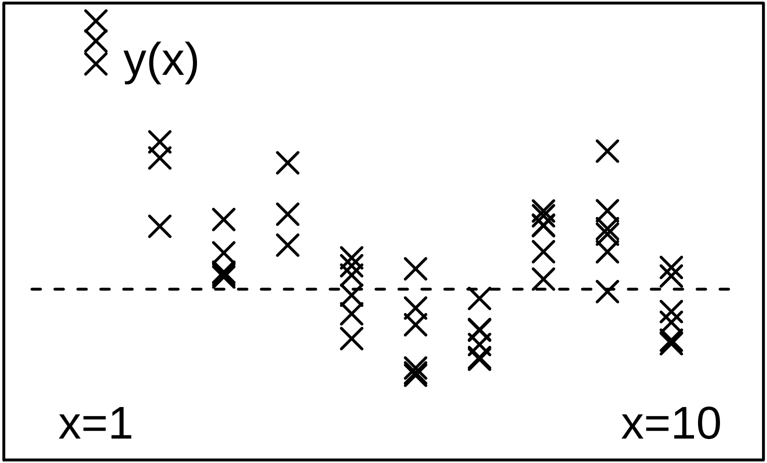
Given Samples, Estimating the Level Set is Well-Understood



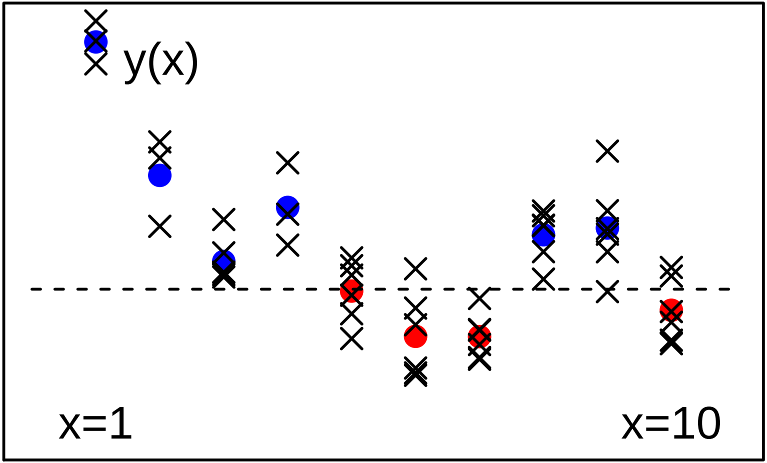
The Optimal Policy Puts Samples Where They Help Most



The Optimal Policy Puts Samples Where They Help Most



The Optimal Policy Puts Samples Where They Help Most



Comparison to previous work on the MCS problem

- We differ from most of the previous work on MCS by using a Bayesian setting, and/or focusing on the allocation problem.
- Comparing to a similar work done by [Xie & Frazier 2013],
  - We consider a finite horizon, which is more realistic than a geometric horizon and infinite horizon studied in [Xie & Frazier 2013]
  - We allow cost per sample to be optional.
  - We look at allocating parallel simulation resources

Goal: Allocate budget to best support classification

- k alternatives
- N time periods
- m parallel computing resources. Each one can simulate 1 alternative in 1 time period.
- θ_x is the underlying true performance of alternative x.
- We will observe iid $\mathrm{Bernoulli}(\theta_x)$ samples from alternative x.
- Our goal is to determine whether each θ_x is larger or smaller than some known threshold d_x.

Goal: Allocate budget to best support classification

- c_x optional monetary cost to simulate alternative x once. $c_x = 0$ for simplicity in this talk.
- $z_{n,x}$ is the number of simulation resources to use on alternative x. It is the decision to be made at each step. $\sum_x z_{n,x} \leq m$
- After we finish sampling, we decide whether to classify each alternative x as above the threshold or below.
- If we decide it is above, we get reward $\theta_x - d_x$.
- If we decide it is below, we get reward $d_x - \theta_x$.

Goal:Allocate our samples to best support our final classification of alternatives

We use a Bayesian approach

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

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

- We use Beta as a conjugate prior θ_x

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

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

- A policy π is a mapping from histories onto allocations of simulation resources $\mathbf{z}_n = (z_{n,1}, \ldots, z_{n,k}) \in \mathbb{N}^k$ satisfying

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

- The MCS problem under Bayesian framework is

$$\max_{\pi} \mathbb{E}^{\pi} \left[\sum_{x=1}^{k} | \frac{\alpha_{N,x}}{\alpha_{N,x} + \beta_{N,x}} - d_x | \middle| \boldsymbol{\alpha}_0, \boldsymbol{\beta}_0 \right] \quad (1)$$

Dynamic programming gives an optimal solution

We formulate the MCS problem as a dynamic program with

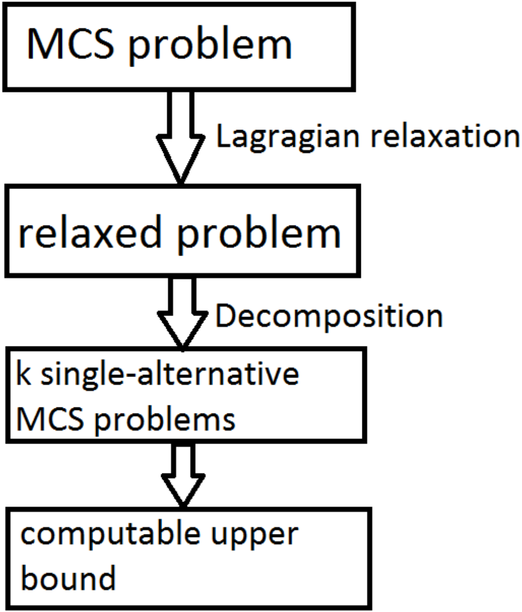
- state at time n, $\mathbf{S}_n = (s_{n,1}, \ldots, s_{n,k})$ = posterior parameters of all the alternatives
- value function $V_n(\mathbf{S}_n)$ = the maximum expected total reward to be obtained from time step n onward given the current state $\mathbf{S}_n$.
- The optimal value is $V_0(\mathbf{S}_0)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

Problem: This dynamic program is computationally infeasible

- 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.

Curse of Dimensionality!

Solution: instead we form an upper bound



Step 1 in forming an upper bound: Relax the original DP

- We perform a Lagrangian relaxation:
  We replace the hard constraint

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

by a linear penality

$$\lambda_n(\sum_{x=1}^{k} z_{n,x} - m)$$

where $\boldsymbol{\lambda} = \{\lambda_1, ..., \lambda_N\} \geq \mathbf{0}$

Lemma (1)

For $\boldsymbol{\lambda} \geq 0$, let $V_0^{\boldsymbol{\lambda}}(\mathbf{S}_0)$ be the optimal value of the relaxed problem, 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

- We define an MCS problem on a single alternative x, for each $x \in \{1, \ldots, k\}$:
  - Up to m resources can be used at each time step
  - The additional cost per sample is λ_n at time n.
- Solving single-alternative MCS problems by dynamic programming is computationally feasible.

Lemma (2)

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

$$V^{\boldsymbol{\lambda}}_0(\mathbf{S}_0) = \sum_{x=1}^{k} V^{\boldsymbol{\lambda}}_{0,x}(S_{0,x}) + m \sum_{n=1}^{N} \lambda_n, \tag{2}$$

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

$$\inf_{\boldsymbol{\lambda} \geq \mathbf{0}} \left[\sum_{x=1}^{k} V_{0,x}^{\boldsymbol{\lambda}}(S_{0,x}) + m \sum_{n=1}^{N} \lambda_n \right] \tag{3}$$

- Computing method:
  first order convex optimization

This analysis also inspires this index policy

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

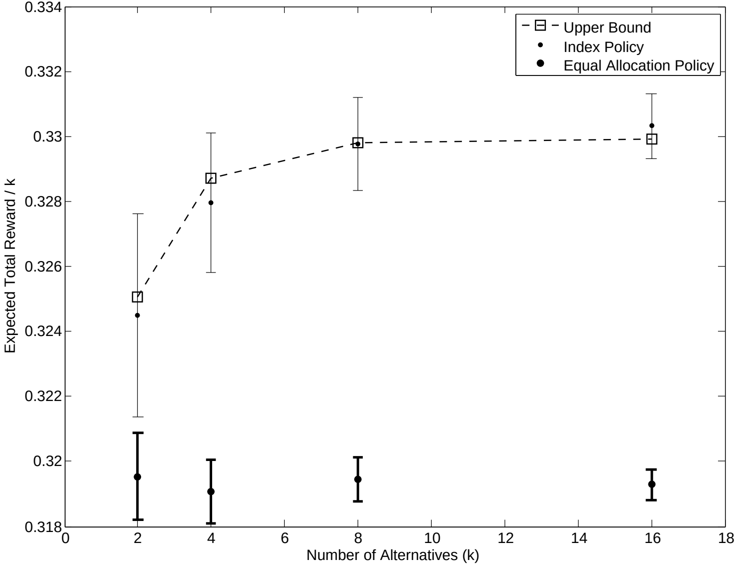
1. Let $z^{\boldsymbol{\lambda}}_{n,x}(S_{n-1,x})$ be the number of samples taken under an optimal single-alternative policy given $\boldsymbol{\lambda}$, breaking ties arbitrarily.
2. Let $\lambda^* = \inf\left\{\lambda : \sum_x z^{\boldsymbol{\lambda}}_{n,x}(S_{n-1,x}) \leq m, \boldsymbol{\lambda} = \lambda\mathbf{e}\right\}$.
3. Set $\boldsymbol{\lambda}^* = \lambda^*\mathbf{e}$.
4. Let $\mathbf{z}_n = \{z^{\boldsymbol{\lambda}^*}_{n,x}, x = 1, ..., k\}$ so as to satisfy $\sum_{x=1}^{k} z^{\boldsymbol{\lambda}^*}_{n,x} \leq m$, breaking ties arbitrarily between different allocations $\mathbf{z}_n$ that satisfy this constraint.

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



Conclusion and Future Work

- We create a computationally feasible upper bound on the value of finite-horizon MCS porblem which allows parallel computing resources.
- We create an index-based policy in the settings studied.
- Conjecture: Index policy becomes optimal policy as m and k increases to infinity

Thank you!