Maximally Informative Underwater Sensor Placement

Jefferson Huang

Assistant Professor
Operations Research Department
Naval Postgraduate School

Naval Surface Warfare Center, Crane Division

Crane, IN

20 October, 2022

Joint work with Robert Bassett (NPS) and LT Erik Vargas (USN)

About Me

Recent Academic History:

- Applied Math PhD from Stony Brook University (2016)
- Postdoc in Cornell Operations Research (OR) Department (2016-2018)
- Assistant Prof. in NPS OR Department (2018-Present)

Main Research Interests:

- Markov Decision Processes (**MDPs**)
- Dynamic Resource Allocation Problems
- Defense Logistics

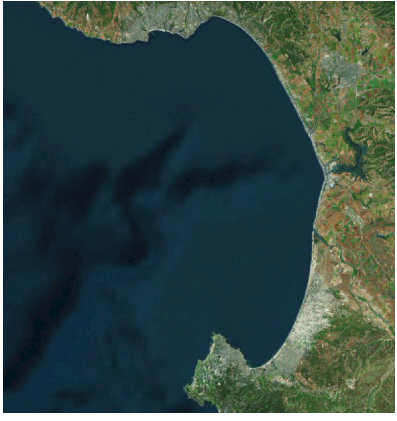
Some Recent Thesis Topics:

- Optimizing Supply Blocks for Expeditionary Units (Capt N.C. Anthony, USMC, June 2021)
- Application of Generative Adversarial Networks to Predicting and Manipulating an Adversary's Behaviors for Mobile Networked Control Systems (LT K. E. Plunkett, USN, March 2023; co-advising with Rudy Yoshida, NPS)
- Approximate Dynamic Programming Methods for the Dynamic Airlift Routing Problem (LCDR A.J. Cooper, USN, March 2023)
- Fleet-Informed Workload Forecasting for the DDNV Material Processing Center (LT A. P. Davidson, USN, September 2023)

For more, see https://faculty.nps.edu/jefferson.huang/

Sensing Correlated Numerical Signals

Example: Acoustic signals in the Monterey Bay.



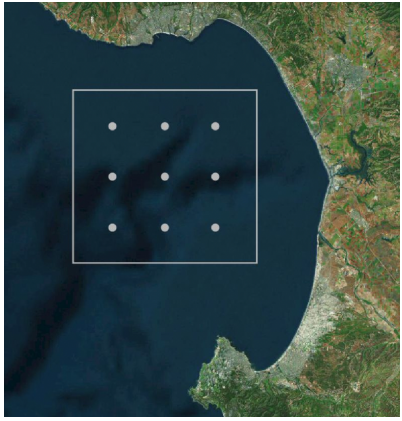
General Problem

Where should data be collected, in order to maximize the amount of "information" gleaned about the entire area of operations (AO)?

- i.e., in order to best infer the numerical signal across the entire AO, **where should sensors be placed?**

Sensor Placement Problem

Example: Placing hydrophones to detect vessels.



Suppose the AO has been discretized into a *finite* number of **locations** (e.g., grid squares). Let:

$$n = \text{number of locations}$$

- The numerical signal X_i at each location i is a random variable.

Only a subset of these locations can be monitored (e.g., with sensors). Let:

$$k = \text{number of locations that can be monitored}$$

Problem

Which k out of the n locations should be **selected**?

Sensor Placement Problem (Continued)

Each selection has an associated **utility**. For each subset $A \subseteq \{1, \ldots, n\} =: [n]$ of the locations, let:

$u(A) =$ utility of monitoring the locations in A

Optimization Problem

$$\begin{aligned} \text{maximize} \quad & u(A) \\ \text{subject to} \quad & A \subseteq [n] \\ & |A| = k \end{aligned}$$

What should $u(A)$ be?

Examples: $u(A)$ could measure:

- how well the AO is covered by the sensors; e.g., [Zhao, Yoshida, Cheung & Haws 2013], [Craparo & Karatas 2019]
- the uncertainty (e.g., entropy) associated with the locations in A; e.g., [Ko, Lee & Queyranne 1995], [Chen, Fampa & Lee 2022]
- **how informative the measurements in A are about the remaining locations**; e.g., [Caselton & Zidek 1984], [Krause, Singh & Gusterin 2008]

Utility Function

Use the **mutual information** $u(A) = \mathrm{MI}(A)$ between the sensed and un-sensed locations.

Gaussian Signals

Suppose $X_1, \ldots, X_n$ are jointly Gaussian, where

$$\mathbf{\Sigma}_A = \text{covariance matrix for locations } i \in A \subseteq [n]$$

Fact

$$\mathrm{MI}(A) = \frac{1}{2} \cdot \ln\left(\frac{\det \mathbf{\Sigma}_A \cdot \mathbf{\Sigma}_{[n]\setminus A}}{\det \mathbf{\Sigma}_{[n]}}\right), \quad A \subseteq [n]$$

So, in the Gaussian case, our optimization problem is equivalent to:

$$\begin{aligned} \text{maximize} \quad & \ln(\det \mathbf{\Sigma}_A) + \ln(\det \mathbf{\Sigma}_{[n]\setminus A}) \\ \text{subject to} \quad & A \subseteq [n] \\ & |A| = k \end{aligned} \tag{P}$$

Theorem [Bassett, H & Vargas 2022]

The following **semidefinite program** can be viewed as a relaxation of the discrete optimization problem (P):

$$\begin{aligned} \text{maximize} \quad & \ln\det\left(\mathbf{\Sigma}_{[n]} \odot \frac{\mathbf{X}+1}{2}\right) \\ \text{subject to} \quad & \mathrm{diag}(\mathbf{X}) = 1 \\ & \mathbf{X} \in \mathrm{PSD}_n \end{aligned}$$

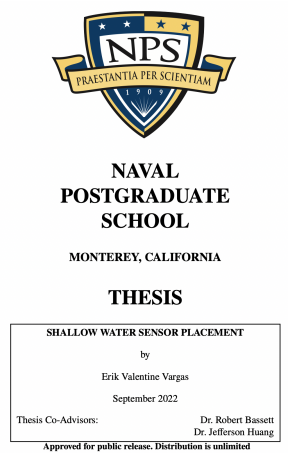
where PSD_n is the set of all symmetric positive semidefinite $n \times n$ matrices.

- Inspired by Goemans-Williamson relaxation of the Max Cut Problem [Goemans & Williamson 1995].

The relaxation can be solved efficiently, and be used in the context of a branch and bound algorithm for (P) [Bassett, H & Vargas 2022].

Summary: Maximally Informative Underwater Sensor Placement

- We consider a sensor placement problem formulated as a mutual information maximization problem.
- We propose a new efficiently solvable relaxation of the problem.
- The relaxation can be used in an effective branch & bound algorithm for the original sensor placement problem.



NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

SHALLOW WATER SENSOR PLACEMENT

by

Erik Valentine Vargas

September 2022

Thesis Co-Advisors: Dr. Robert Bassett
Dr. Jefferson Huang

Approved for public release. Distribution is unlimited

Broader Research Interests:

Markov Decision Processes

MDPs Model (Stochastic) Sequential Decision Problems

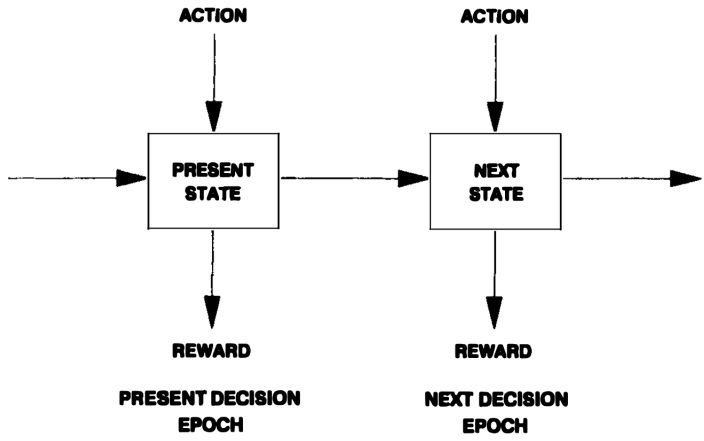


Figure 1.1.1 Symbolic representation of a sequential decision problem.

Source: M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2005.

Potential defense-related applications abound...

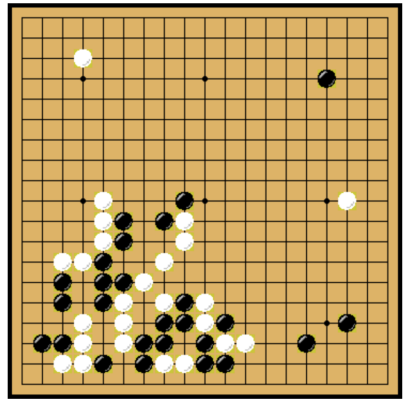
Recent Survey: M. Rempel & J. Cai, A review of approximate dynamic programming applications within military operations research, *Operations Research Perspectives* 8, 2021.

...but computing good solutions at scale is notoriously difficult.

It suffers from what Bellman called "the **curse of dimensionality**,*" meaning that its computational requirements grow exponentially with the number of state variables . . .*

. . . but it is still far more efficient and more widely applicable than any other general method.

Sutton & Barto, *Reinforcement Learning: An Introduction*, 2018 (p. 14)



http://norvig.com/atoms.html

The MDP Model

A Markov decision process (MDP) is defined by

- a **state set** $\mathbb{X}$,
- sets of feasible **actions** $A(x)$ for each state $x \in \mathbb{X}$,
- **one-step rewards** $r(x, a)$ for each state $x \in \mathbb{X}$ and action $a \in A(x)$, and
- **transition probabilities** $p(y|x, a)$ for $x, y \in \mathbb{X}$ and $a \in A(x)$.

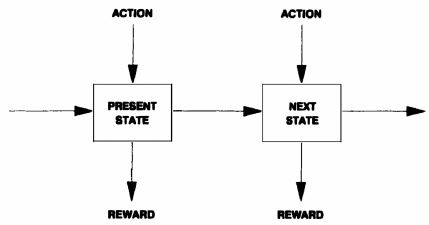


Figure 1.1.1 Symbolic representation of a sequential decision problem.

Source: M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 2005.

Objective: Find an *optimal policy* ϕ that, for each state $x \in \mathbb{X}$, prescribes an action $\phi(x) \in A(x)$ to take.

Finding Optimal Policies

- It suffices to solve some associated **optimality equations** (aka. "Bellman equations")
- Solving the optimality equations provides the **value function**, which indicates which states are more valuable to be in than others.
- The value function can then be used to derive an **optimal policy**.

Example: When the objective is to maximize the expected total reward that is earned, the optimality equations can be viewed as a functional equation

$$v = T(v), \qquad v : \mathbb{X} \to \mathbb{R}$$

where T is a non-linear "optimality operator". Given a solution v to the optimality equation, an optimal policy is

$$\Phi^*(x) = \arg\max_{a \in A(x)} \left[r(x,a) + \sum_{y \in \mathbb{X}} v(y) p(y|x,a) \right]$$

Algorithms for Computing Optimal Policies

There are two main algorithmic paradigms:

Value Iteration: Iteratively approximate the value function $v : \mathbb{X} \to \mathbb{R}$.

Example: Start with an initial guess v_0, and iterate the optimality operator T:

$$v_k = T(v_{k-1}), \qquad k = 1, 2, \ldots$$

Policy Iteration: Iteratively approximate the optimal policy ϕ^*.

Example: Start with an initial policy ϕ_0, and iteratively improve it by identifying actions to switch to.

Note: There is a close connection between policy iteration and applying the simplex method to an associated linear program. Policy iteration can also be viewed as applying Newton's method to finding a root of $T(v) - v$. (See e.g., Puterman (2005) for details.)

Dealing with Computational Intractability

There are two main algorithmic paradigms:

Value Function Approximation: Search within a structured classes of value functions.

Examples:

- Piecewise-constant functions (e.g., via state aggregation)
- Parameterized functions (e.g., linear in hand-selected features, neural networks, . . .)

Policy Function Approximation: Search within structured classes of policies.

Examples:

- Piecewise-constant policies (e.g., via state aggregation)
- Parameterized policies (e.g., linear in hand-selected features, neural networks, . . .)

Reinforcement Learning: Dealing with Unknown/Complex System Dynamics

Idea: Use experience from interacting with an environment (or a simulation model of it) to learn good value or policy function approximations.

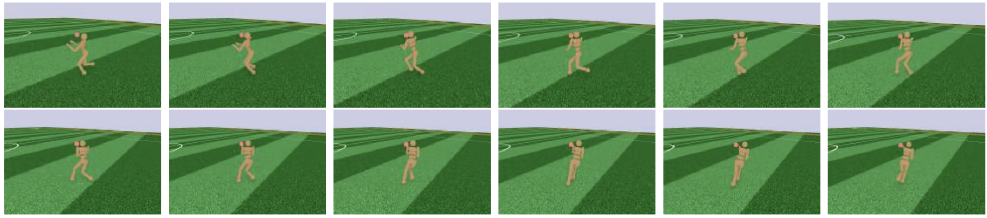


Figure 5: Still frames of the policy learned from RoboschoolHumanoidFlagrun. In the first six frames, the robot runs towards a target. Then the position is randomly changed, and the robot turns and runs toward the new target.

Source: Schulman et al., Proximal policy optimization algorithms, `arXiv:1707.06347v2`, 2017

Reinforcement Learning: Dealing with Unknown/Complex System Dynamics

Idea: Use experience from interacting with an environment (or a simulation model of it) to learn good value or policy function approximations.

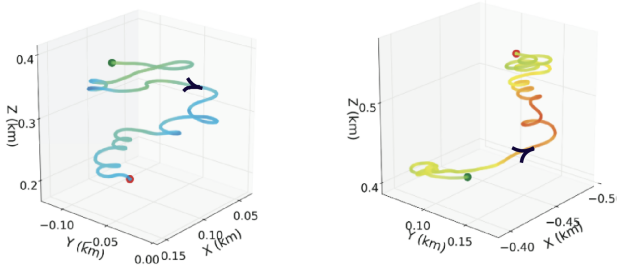


Figure 16.10: Sample thermal soaring trajectories, with arrows showing the direction of flight from the same starting point (note that the altitude scales are shifted). Left: before learning: the agent selects actions randomly and the glider descends. Right: after learning: the glider gains altitude by following a spiral trajectory. Adapted with permission from PNAS vol. 113(22), p. E4879, 2016, Reddy, Celani, Sejnowski, and Vergassola, Learning to Soar in Turbulent Environments.

Source:, Sutton & Barto, 2018 (p. 456)

Contact Information

Jefferson Huang, PhD

Assistant Professor
Operations Research Department
Naval Postgraduate School

✍ **Web:** http://faculty.nps.edu/jefferson.huang/
✉ **Email:** jefferson.huang@nps.edu