Optimal Waterfilling Policy for Energy Harvesting Wireless Networks

Sasank Chilamkurthy; Advisor: Prof. Abhay Karandikar

Department of Electrical Engineering, Indian Institute of Technology Bombay

The Big Question

For energy harvesting transmitter, how do we allocate power for slots to maximise throughput?

What is WaterFilling Policy

It is a optimal power allocation scheme if we have k independent Gaussian channels with different noise levels in parallel with common power constraint $\bar{P}$. This is a model for fading channel for k time slots each having independent fading states.

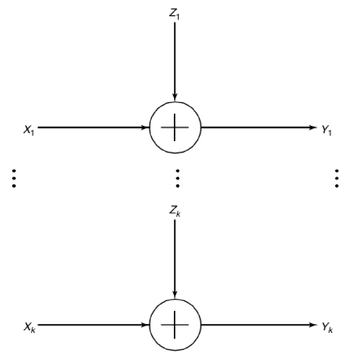

Figure 1: Parallel Gaussian Channels

- If we allocate a power P_i to ith channel which has AWGN noise level of N_i, Then the capacity of ith channel is
$$R_i = 1/2 \log(1 + P_i/N_i)$$
- Average Power over k channels is constrained by P. i.e, $\frac{\sum_i P_i}{k} \leq \bar{P}$
- We want to maximize $\sum_i R_i$ subject to constraints $\sum_i P_i \leq k\bar{P}$ and $P_i \geq 0$
- This can be solved by Lagrangian multipliers. Solution of this optimization problem turns out to be
$$P_i = (\nu - N_i)^+$$
where $f^+ = \max(f, 0)$ and ν is chosen such that $\sum_i (\nu - N_i)^+ = P_i$. This ν is called *Water Level*.
- If ith channel corresponds to ith fading state with fading coefficient F_i, then $N_i = N/F_i^2$ where N is SNR of the channel when fading coefficient is 1.

What is a Energy Harvesting Wireless Network

This is modeled as standard fading channel
$$Y_i = F_i X_i + N_i$$
with the following extra conditions

- Transmitter Node has a rechargable battery. It gets recharged using energy harvested from the sorroundings. It has a maximum capacity E_{max}
- Ammount of battery recharge is not known before hand and varies randomly due to environmental fluctuations
- Suppose in ith slot, E_i is energy available in the battery, P_i is energy consumed for transmission and K_i is recharge from the harvesting,
$$E_{i+1} = \min(E_i - P_i + K_i, E_{max})$$
- K_i is iid random variables which can be modelled to depend on current battery charge. We used $K_i = U_i(1 - e^{\alpha(E_n - E_{max})})$ where U_i is a uniform random variable.
- Note that P_i has to be at most E_i. This is what makes power allocation harder. If ith slot is really good and if we do not have enough energy in the battery, we'll have to use suboptimal power for transmission.
- We assume that transmitter has full CSI at ith slot and it has both channel fading characteristics and recharge characteristics available.

Our Problem

Suppose that the an energy harvesting tranmitter node has full CSI, energy recharge characterstics and fading state characterstics available, which water level of the waterfilling policy does it have to chose to maximise average expected throughput $1/N \sum_i^N \mathbb{E}[\log(1 + P_i F_i^2/N_0)]$

Simulation Methodolgy

- For each waterlevel ν, N slots will be simulated.
- t such independent sample paths for system are generated and are averaged. This is an approximation for expected value over randomness involved in battery rechanrge.

Simulation Results

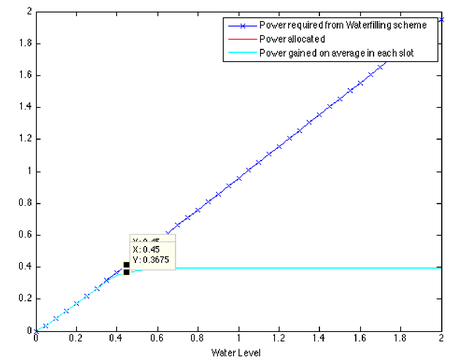

Figure 2: Expected power values of different quantities vs. `waterlevel` (ν)

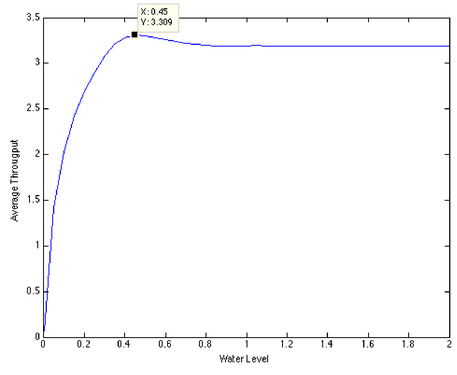

Figure 3: Expected throughput for a given `waterlevel` (ν)

Infinitesimal Perturbation Analysis

To maximize R, we do independent simulations of the system, $\hat{X}_1, \hat{X}_2 \ldots, \hat{X}_N$ and $\mathring{X}_1, \mathring{X}_2 \ldots, \mathring{X}_N$ corresponding to $\nu(n) + \delta/2$ and $\nu(n) - \delta/2$ and perform the iterate
$$\nu(n+1) = \nu(n) - a(n) \left(\frac{R(\hat{X}_1, \hat{X}_2 \ldots, \hat{X}_N) - R(\mathring{X}_1, \mathring{X}_2 \ldots, \mathring{X}_N)}{\delta} \right)$$

Algorithm to Find Best Waterfilling Scheme

We use *Infinitesimal Perturbation Analysis (IPA)* scheme to find optimal waterlevel ν.

To increase stability of the algorithm, we simulate chain `times` number of times and take average gradient.

```
% maximize R with respect to waterlevel
n = 1;
waterlevel = 0.2; % initial guess
delta = 0.05;
diff = 1;

% Iteration
while diff > 1e-7

w = [waterlevel-delta/2 waterlevel+delta/2];
avgGradient = 0;
times = 2;

% Compute Average gradient
for i = 1:times

[R, ~] = simulateChain(w,parameters);
avgGradient = avgGradient+(R(2)-R(1))/delta;

end

avgGradient = avgGradient/times;

a_n = 1/(10*n); % Step Size

%Stochastic Gradient Scheme
waterlevel = waterlevel + a_n*avgGradient;

diff = abs(a_n*(R(1)-R(2))/delta);
n = n+1;

end
```

This algorithm gives following results when parameters from Figure 3 are used:

```
waterlevel = 0.4597
diff = 9.0463e-08
1124 iterations
```

Future Work

Although running of the algorithm has been verified by simulations, to theoretically prove the convergence of the above algorithm we need to prove the quasi-concaveness of expected average throughput. This is being worked on.