Lyapunov Optimization

WANG, Qipeng

Harbin Institute of Technology

1160801008@stu.hit.edu.cn

1 Introduction

在论文“Thompson Sampling for Combinatorial Semi-bandits with Sleeping Arms and Long-Term Fairness Constrains”中 Fairness 那一部分，应用到了 Lyapunov Optimization. 这篇 Notes 简单地记录了一下学到的相关的内容。

Lyapunov Optimization 是指：使用 Lyapunov Function 控制动态的系统。如果系统朝向 undesirable 的方向发展，那么 Lyapunov function 的数值就会变大，换句话说，如果 Lyapunov function 沿 x 轴负方向逼近 0，那么系统就会趋于稳定。

2 Lyapunov Optimization

在这一节介绍了随机优化问题如何使用 Lyapunov Optimization 来进行求解。

2.1 Optimization

在介绍随机优化问题之前，首先给出了简单的优化问题定义。

一个简单优化问题定义如下：在满足数个约束条件下最小化某个目标。数学上可表达为：

$$
\begin{aligned}
\mathrm{P}_1: \quad & \min_{x\in\mathbb{R}^n} f(x) \\
s.t. \quad & \mathrm{c}_i(x) \le 0, i = 1,2,....k \\
& \mathrm{h}_j(x) = 0, j = 1,2,....k
\end{aligned}
$$

此问题可用拉格朗日乘子法进行求解。

2.2 Stochastic Optimization

随机优化比简单优化问题多了随机性，即每一时刻都会有随机事件产生，面对随机事件采取的策略也要随时变化。

记第 t 个时隙所产生的随机事件为：

$$W(t) \triangleq \{w_1(t), w_2(t).....w_n(t)\} \in \Omega_n$$

，其中 Ω_n 是随机事件集合。设 $\forall i \in \{1,...n\}, w_i(t)$ 为独立同分布。记录系统在每个时隙内所采用的策略为 $\alpha(t) \triangleq \{\alpha_1(t),...\alpha_n\} \in \mathrm{A}^m$，其中 A^m 是决策集合。

在第 t 个时隙，对于想要优化的目标函数 p_t 可以按某种已知的方式生成对应的数值：

$$p_t = P(w(t), \alpha(t)),$$

其中，$P(.)$ 就是一个确定的函数除了优化的目标，我们还需要关注系统中一系列变量 $y_k(t)$ 也会受到决策的影响，即：

$$y_k(t) = Y_k(w(t), \alpha(t)), k \in \{1, ...k\}$$

其中，$Y_k(.)$ 均为确定函数。

在随机优化问题中，优化目标是原始目标在 time-average 的情况下的表现。即：一个随机优化问题是指在一段时间内最小化 time average 的目标函数，该目标函数受到随机事件和采取的控制策略的影响，此外还需满足若干个约束条件，这些约束条件也会受到随机事件和采取的策略的影响。可以在数学上表示为：

$$\mathrm{P}_2: \quad \min_{\forall t, \alpha(t) \in \mathrm{A}^m} \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} E[p(t)]$$

$$s.t. \quad \lim_{T \to \infty} \frac{1}{T} \sum_{t=0}^{T-1} E[y_k(t)] \le 0, k \in \{1, ...K\}$$

接下来我们讨论如何构造 virtual queue。

2.3 构造虚拟队列

构造虚拟队列是为了将长期的约束条件所采取的操作分解到每个时隙内进行。因为操作是要落到每个时隙内的。这样，在每个时隙内，我们可以直接优化 p_t 。

因此，对于一个约束条件 $y_k(t)$ 定义的初始值为 0 的队列 $Q_k(t)$ ：

$$Q_k(t+1) = \max\{Q_k(t) + y_k(t), 0\}, k \in \{1, ...K\}.$$

$Q_k(t)$ 并不是第 k 个队列本身，而是第 k 个 queue's backlog. 队列只是概念，而队列的 backlog 才是一个数值。由于在约束条件中我们希望 time average of $y_k(t)$ 不大于 0，因此我们总是希望能处理的大于新抵达的。

由于 $Q_k(t+1) = \max\{Q_k(t) + y_k(t), 0\}, k \in \{1, ...K\}.$ ，因此可以得到：

$$Q_k(t+1) \ge Q_k(t) + y_k(t), k \in \{1, ...K\}.$$

即：

$$\sum_{t=0}^{T-1} y_k(t) \le Q_k(T) - Q_k(0), k \in \{1,...K\}.$$

两边同时取期望得：

$$\frac{1}{T}\sum_{t=0}^{T-1} E[y_k(t)] \le \frac{E[Q_k(T)]}{T}, k \in \{1,...K\}.$$

为了让优化问题中的约束条件成立，我们可以让：

$$\lim_{T\to\infty} \frac{E[Q_k(T)]}{T} = 0, k \in \{1,...K\}.$$

在 Micheal J Neely 书中这个被称为 mean rate stable。而更为严格的稳定是：

$$\exists \delta \le 0, \lim_{T\to\infty} \frac{E[Q_k(T)]}{T} \le \delta, k \in \{1,...K\}.$$

之后，随机优化问题可以转化为：

$$\begin{aligned} \mathrm{P}_3: \quad & \min_{\forall t, \alpha(t)\in \mathrm{A}^m} \quad \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} E[p(t)] \\ s.t. \quad & \lim_{T\to\infty} \frac{E[Q_k(T)]}{T} = 0, k \in \{1,....K\} \end{aligned}$$

2.4 解决方案

2.4.1 Drift-plus-penalty Expression

Lyapunov Function 是对当前时间片内所有 queues' backlog 的标量非负的描述。定义全体队列为一个矢量：

$$\Theta(t) \triangleq [Q_1(t),....Q_K(t)]$$

Lyapunov Function 定义为：

$$L(\Theta(t)) \triangleq \frac{1}{2}\sum_{k=1}^{K} Q_k^2(t)$$

严格定义为：

$$L(\Theta(t)) \triangleq \frac{1}{2}\sum_{k=1}^{K} a_k(t) Q_k^2(t)$$

其中，$a_k(t)$ 是每个队列的权重。

之后，我们可以用 $\Delta(\Theta(t)) \triangleq L(\Theta(t+1)) - L(\Theta(t))$ 表示从时隙 t 到 $t+1$ 的 queue's backlog 的增长量，称为 Lyapunov Drift。

由队列的定义可知：

$$Q^2{}_k(t+1) \le (Q_k(t)+y_k(t))^2, k \in \{1,...K\}.$$

对 K 个队列进行累加得到：

$$\frac{1}{2}\sum_{k=1}^{K} Q_k^2(t+1) \le \frac{1}{2}\sum_{k=1}^{K} Q^2{}_k(t) + \frac{1}{2}\sum_{k=1}^{K} y_k^2(t) + \sum_{k=1}^{K} Q_k(t)y_k(t), k \in \{1,...K\}.$$

因此有：

$$\begin{aligned}\Delta(\Theta(t)) &\triangleq \frac{1}{2}\sum_{k=1}^{K} Q_k^2(t+1) - \frac{1}{2}\sum_{k=1}^{K} Q_k^2(t) \\ &\le \frac{1}{2}\sum_{k=1}^{K} y_k^2(t) + \sum_{k=1}^{K} Q_k(t)y_k(t) \\ &\le B + \sum_{k=1}^{K} Q_k(t)y_k(t)\end{aligned}$$

注意，此处并不是在所有情况下都成立，而是在大多数实际情况中这一点都能满足。

2.4.2 Drift-plus-penalty Algorithm

在引入队列后可以在控制队列稳定的情况下直接优化 p_t，一个很直观的方式为在每个时隙内求解 $\Delta(\Theta(t))+V.p_t$ 的最小值，其中 V 是一个权重值，用来 balance 两者。因此，这个随机优化问题可以转为：

$$\begin{aligned}\mathrm{P}_4: \quad & \min_{\forall t, \alpha(t)\in \mathrm{A}^m} \ \mathrm{E}[\Delta(\Theta(t))+V.p_t] \\ s.t. \quad & \lim_{T\to\infty} \frac{E[Q_k(T)]}{T} = 0, k \in \{1,....K\}\end{aligned}$$

而 drift-plus-penalty algorithm 就是通过求解这个问题的近似最优解来求解 P3. 接下来的问题是如何求解 $\Delta(\Theta(t))+V.p_t$ 的最小值。对其进行放缩（放缩的理由是因为原来涉及到了 *t+1* 时刻的值）：

$$\Delta(\Theta(t))+V.p_t \le B + V.p_t + \sum_{k=1}^{K} Q_k(t)y_k(t)$$

因此可以直接求解上式右端的最小值，但是，求到的解并不是原始问题的最优解！！

Algorithm　Drift-plus-penalty Algorithm

1：　在时隙 t，观察所有随机事件 $w(t)$ 及队列储备量 $\Theta(t)$

2：　求解得到最优控制策略：

$$\alpha^*(t) = \arg\min_{\forall t, \alpha(t) \in \mathrm{A}^m} \mathrm{E}[\mathrm{B} + V.p_t + \sum_{k=1}^{K} Q_k(t) y_k(\mathrm{t})]$$

3：更新：

$$Q_k(t+1) = \max\{Q_k(t) + y_k(t), 0\}, k \in \{1, \ldots K\}.$$

$$t = t + 1$$

3 Case Study

分析完理论部分，在这一部分，选用了 Yuyi Mao 的“Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices”这篇论文来进一步加深对 Lyapunov Optimization 的理解。（由于在读论文的时候手写的笔记，在电脑整理的时候为了节约时间并没有手动输入每个公式，因此这一部分排版不好。）

这篇文章考虑了一个移动设备以及一个 MEC server，该移动设备又 energy harvesting 的部分。在每个时隙，移动设备可以选择本地计算或者 offload to MEC。

首先可以得到的优化问题为：

$$\mathcal{P}_1: \min_{\boldsymbol{I}^t, \boldsymbol{f}^t, p^t, e^t} \lim_{T \to \infty} \frac{1}{T} \mathbb{E}\left[\sum_{t=0}^{T-1} \mathrm{cost}^t\right]$$

$$\text{s.t.} \quad (1), (6), (8), (12)$$

$$I_{\mathrm{m}}^t + I_{\mathrm{s}}^t \le \zeta^t, t \in \mathcal{T} \tag{13}$$

$$\mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) \le E_{\max}, t \in \mathcal{T} \tag{14}$$

$$0 \le p^t \le p_{\mathrm{tx}}^{\max} \cdot \mathbf{1}\left(I_{\mathrm{s}}^t = 1\right), t \in \mathcal{T} \tag{15}$$

$$0 \le f_w^t \le f_{\max}^{\mathrm{CPU}} \cdot \mathbf{1}\left(I_{\mathrm{m}}^t = 1\right), w = 1, \cdots, W, t \in \mathcal{T}, \tag{16}$$

$$I_{\mathrm{m}}^t, I_{\mathrm{s}}^t, I_{\mathrm{d}}^t \in \{0, 1\}, t \in \mathcal{T}, \tag{17}$$

其中的限制条件（1）

$$I_{\mathrm{m}}^t + I_{\mathrm{s}}^t + I_{\mathrm{d}}^t = 1, t \in \mathcal{T}. \tag{1}$$

表明计算任务要么在本地执行，要么被 offload to MEC 或者 drop 掉，

限制条件（6）

$$0 \leq e^t \leq E_H^t, t \in \mathcal{T}, \tag{6}$$

表明每一时刻所 harvest 到的能量。

限制条件（8）

$$\mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) \leq B^t < +\infty, t \in \mathcal{T}. \tag{8}$$

表明能量消耗的限制，其中：

$$\mathcal{E}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) = I_{\mathrm{m}}^t E_{\mathrm{mobile}}^t + I_{\mathrm{s}}^t E_{\mathrm{server}}^t,$$

限制条件（12）

$$\mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) \leq \tau_d, t \in \mathcal{T}. \tag{12}$$

表明时延要求，其中：

$$\mathcal{D}\left(\boldsymbol{I}^t, \boldsymbol{f}^t, p^t\right) = \mathbf{1}\left(\zeta^t = 1\right) \cdot \left(I_{\mathrm{m}}^t D_{\mathrm{mobile}}^t + I_{\mathrm{s}}^t D_{\mathrm{server}}^t\right)$$

此外，限制条件（13）为：如果没有计算任务，移动端和 MEC 都不会执行任务。（14）为电池放电时输出的能量不能大于设定值。（15）和（16）分别是允许传输的最大能量和最大 CPU 频率的限制。

问题P1实际上是一个MDP（马尔可夫决策过程）问题，可以使用relative value iteration 和 the linear programming reformulation approach 来解决，但是对于这个问题我们需要用无穷的状态来表示，因此算法的速度以及储存最优策略问题是一个很大的 challenge。

而基于 Lyapunov Optimization 的卸载决策算法可以解决这个问题。

Algorithm 1 The LODCO Algorithm

1: At the beginning of time slot t, obtain the task request indicator ζ^t, virtual energy queue length $\tilde{B}^t$, harvestable energy E_H^t, and channel gain h^t.

2: Decide $e^t, \boldsymbol{I}^t$, f^t and p^t by solving the following deterministic problem:

$$\min_{\boldsymbol{I}^t, p^t, f^t, e^t} \tilde{B}^t \left[e^t - \mathcal{E}\left(\boldsymbol{I}^t, f^t, p^t\right)\right] + V\left[\mathcal{D}\left(\boldsymbol{I}^t, f^t, p^t\right) + \phi \cdot \mathbf{1}\left(\zeta^t = 1, I_{\mathrm{d}}^t = 1\right)\right]$$

$$\text{s.t.} \quad (1), (6), (12) - (18).$$

3: Update the virtual energy queue according to (9) and Definition 2.

4: Set $t = t + 1$.

4 Conclusion

在做 Online learning 中的 fairness 问题时采用了 Lyapunov Optimization，借此学习了一下 Lyapunov function 以及按照思路阅读了一篇文章，相似的文章也可以将 Notes 中提到的理论带入自己的模型即可。

Notes 中存在的问题敬请指正。