Lyapunov Optimization: An Introduction

Hai-Liang Zhao*
hliangzhao97@gmail.com

2018 年 11 月 21 日

目录

1 About

本文档从引入优化问题的标准定义开始, 循序渐进地介绍了如何将Lyapunov optimizaiton技术运用到随机优化问题的求解中. 作为一个完美主义者, 写作本文档可费了不少心思, 为了清晰地给出每一个结论是如何得到的, 数学推导部分在我看来显得过于啰嗦了. 但我相信这样的付出是有意义的. 本文中每一个出现的符号、脚注, 加粗、颜色和斜体都有着各自不同的含义.

`Wikipedia`关于Lyapunov optimization的词条能给我们一个整体的认知, 但细节上却存在诸多纰漏. 在此之前我本人反复阅读数十遍, 但依旧有许多细节问题没有搞清楚. 当倘若先暂停手头的工作直接去阅读Micheal J. Neely的著作*Stochastic Network Optimization with Application to Communication and Queueing Systems*, 时间上太不划算, 这也绝不是我做研究的方式. **优秀的研究者应当在应用中搞懂新的概念和方法, 而不是先搞懂所有涉及的概念再去思考应用, 因为**

*Hai-Liang Zhao is a pre-graduate student in Zhejiang University. Currently he is a fourth year undergraduate in School of Computer Science and Technology, Wuhan Univeristy of Technology.

概念和方法是无穷无尽的. 后来深入研究了自己方向内数篇将Lyapunov optimization运用的炉火纯青的论文, 总算把各类问题都搞清楚了.

本文档的目标读者是**想迅速将Lyapunov optimization技术投入到论文的写作中并能针对性地给出性能分析**的人. 我原本打算在本文档最后提供数个case studies来加深读者的印象, 但当我写完Lyapunov optimization的理论分析后, 就不愿意浪费笔墨和时间做这件事了ㄟ. **因为只要读者认真搞懂了本文档全部内容, 这些case studies完全不是问题**. 读者可自行阅读这些case studies对应的论文:

1. Yuyi Mao关于LODCO algorithm的论文: *Dynamic computation offloading for mobile-edge computing with energy harvesting devices*;
2. Xu Chen团队关于mobility at MEC的论文: *Follow Me at the Edge: Mobility-Aware Dynamic Service Placement for Mobile Edge Computing*;
3. Yuyi Mao关于non-binary task计算卸载的论文: *Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems*.

这些论文均可直接在ArXiv上下载.

写作此文档花费了3个晚上和4个下午, 在繁忙的十一月, 这样做似乎太过任性了ㄟ.

 The latest version can be found at my GitHub Page, please report any mistakes to `hliangzhao97@gmail.com`.

2 A Brief Introduction

Lyapunov Optimization是指: 使用Lyapunov function[1]来优化控制一个动态的系统. 系统在某一个特定的时间节点的状态(state)可以使用一个**多维向量(multi-dimentional vector)**来描述, 而Lyapunov function则是对这个多维向量所表达的状态的一个**非负的、标量的描述(a non-negative scalar measure)**, 我们常常采用'所有状态在各自权重下的平方和'来实行这一点. Lyapunov function被设计的初衷是: 如果系统的状态朝向一个不被期望的(undesirable)方向发展, 那么Lyapunov function的数值就会变大. 这样, 我们就可以通过将Lyapunov function沿着x轴的负方向逼近0使系统趋向于稳定.

3 How It Works

现在, 我们看看一个随机优化问题(stochatsic optimization problem)如何使用Lyapunov Optimization来求解, i.e., (i)为原始问题构造队列; (ii)给出使用Lyapunov优化技术得到的解; (iii)分析该解和原始问题的最优解之间的差距(optimality gap).

3.1 Optimization Problems

在给出随机优化问题(stochatsic optimization problem)的通用形式之前, 先给出**不包含随机事件的**、简单的优化问题(optimization problem)的定义.

[1]后文将会循序渐进地给出Lyapunov function以及相关概念的定义.

一个优化问题是：在满足数个约束的条件下最小化某一个目标(最大化某一个目标可以转化为最小化该目标和-1的乘积)．在数学上可以表达为

$$\mathcal{P}_1: \quad \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{3.1}$$

$$s.t. \quad c_i(\mathbf{x}) \leq 0, i = 1, 2, ..., k \tag{3.2}$$

$$h_j(\mathbf{x}) = 0, j = 1, 2, ..., l. \tag{3.3}$$

对于这样的约束最优化问题, 常常可以借助**拉格朗日对偶性(Lagrange duality)**将原始问题转换为对偶问题, 通过求解对偶问题得到原始问题的解, 这就是高等数学中所谓的 '拉格朗日乘子法', 以后会专门写作一篇monograph来讲解该方法.

3.2 Stochastic Optimization Problems

随机优化问题和优化问题相比, 多了**随机性**．随机优化问题所谓的'随机'是指每个时间片内有随机事件产生, 面对这个随机事件我们能够采取的策略也要随机应变才行.

不妨将第t个time slot内产生的所有随机事件标记为$\mathbf{w}(t) \triangleq [w_1(t), w_2(t), ..., w_n(t)] \in \Omega^n$, 假设$\forall i \in \{1, ..., n\}, w_i(t)$对每个time slot而言满足独立同分布(i.i.d.)[2], 其中Ω^n为随机事件集合．系统[3]在每个时间片内采取的策略, i.e., 动作(control actions)记为$\boldsymbol{\alpha}(t) \triangleq [\alpha_1(t), \alpha_2(t), ..., \alpha_m(t)] \in \mathcal{A}^m$, 其中$\mathcal{A}^m$为控制决策集合.

在第t个time slot, 根据当前发生的各个随机事件$\mathbf{w}(t)$, 系统采取的一系列的控制决策$\boldsymbol{\alpha}(t)$, 对于想要优化的目标函数$p(t)$而言, 会按照**某个已知的方式**(如latency的计算方法)生成对应的数值, i.e.,

$$p(t) = P(\mathbf{w}(t), \boldsymbol{\alpha}(t)), \tag{3.4}$$

其中$P(\cdot)$就是一个确定的函数．除了优化目标, 系统中的一系列变量$y_k(t)$(如功率的开销, 网络中可用的结点数量, 带宽子信道的个数)[4]也会受到我们所采取的控制决策的影响, 同样地, 可以描述为

$$y_k(t) = Y_k(\mathbf{w}(t), \boldsymbol{\alpha}(t)), k \in \{1, ..., K\}, \tag{3.5}$$

其中$\forall k \in \{1, ..., K\}, Y_k(\cdot)$均为确定函数.

除了随机性, 在随机优化问题中, 优化目标不再是简单的$f(\mathbf{x})$, 而是原始目标函数在time average下的表现．这是因为只有当我们把time average引入, 每个时间片下的表现才能**都**被纳入考虑中．现在, 我们给出随机优化问题的标准形式.

一个随机优化问题是：在一个时间被分片的时间域上最小化一个time average的目标函数，该目标函数在每个时间片的取值受到随机事件和采取的控制策略的影响．此外还需满足数个time average的约束条件，这些约束条件在每个时间片的取值亦受到随机事件和采取的控制策略的影响．在数学上可以表达为

$$\mathcal{P}_2: \min_{\forall t, \boldsymbol{\alpha}(t)\in\mathcal{A}^m} \lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[p(t)] \tag{3.6}$$

[2]其实, '满足独立同分布'这一条件是对现实问题的简化处理, Micheal J. Neely的书中论证过Lyapunov Optimization同样适用于non-i.i.d.的场景, 但求解会复杂很多, 本文档所涉及的drift-plus-penalty algorithm是无法处理这种情形的.

[3]这里的系统只是一个抽象的描述.

[4]其实, 这些变量可以是**任意变量**, 完全依赖于我们想要解决的问题. 在LODCO算法中, 这个除了latency之外还被影响到了的变量即是移动设备的电量.

$$s.t.\ \lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[y_k(t)]\le 0, k\in\{1,...,K\}. \tag{3.7}$$

其中$p(t)$由(3.4)得到，$y_k(t)$由(3.5)得到.

简单起见, 上述定义没有考虑等式约束.

为什么(3.6)和(3.7)中会出现期望运算(Expectation)？ 这是因为问题中存在随机变量的函数． 我们将服从独立同分布的随机事件$\mathbf{w}(t)$看成在第t个时间片内的随机变量$\mathbf{X}^n$，那么由(3.4)和(3.5)可知$p(t)$和$y_k(t)$是随机变量的函数值． 我们知道期望是定义在随机变量(或者随机变量的函数)上的, 因此期望运算很自然地被引入随机优化问题中.

现在, 我们已经形成了一个标准的随机优化问题, 接下来将会深入阐述**如何为这个问题的约束条件构造队列**. 在构造虚拟队列之前, 请回忆一下上次的讨论. 当时我谈到*我们会为那些对优化目标产生影响的变量定义队列*. 这是显然的, 因为一个随机优化问题的约束条件会缩小可行解空间(feasible solution space), 显然会影响到优化目标.

3.3 How to Construct Virtual Queues

为什么要为随机优化问题的约束条件定义队列? 这是因为**我们希望把满足这些长期的约束条件所应当采取的操作分解到每一个时间片内进行.**

毕竟, 操作$\boldsymbol{\alpha}(t)$是要落地到每一个时间片t内的. Lyapunov Optimization的思想就是: 将具有长期约束条件的某个长期优化指标(i.e., $\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[p(t)]$)分解到每一个时间片内进行. 这样, 在每一个时间片t内, 我们就可以在不破坏当前这个时间片需要遵循的约定(i.e., 控制队列保持稳定)的情况下直接优化$p(t)$.

因此, 对每一个约束条件$y_k(t)$定义一个对应的、初始值为0[5]的队列$Q_k(t)$:

$$Q_k(t+1)=\max\{Q_k(t)+y_k(t),0\}, k\in\{1,...,K\}. \tag{3.8}$$

其实$Q_k(t)$并非队列(the kth queue)本身, 而是*队列的储备量*(the kth queue's backlog). 队列只是一个概念, 而队列的储备量才是一个数值. 我们先前提到的'有进有出'作为增量出现在(3.8)中, 也就是$y_k(t)$, 为什么我们之前总是**希望'出的'(能处理的)尽可能大于'进的'(新抵达的)**？这是因为约束条件中, 我们希望time average of $y_k(t)$不大于0.

现在, 我们研究一下如何控制队列$Q_k(t)$来保证(3.7)恒成立. 由(3.8)可知

$$Q_k(t+1)\ge Q_k(t)+y_k(t), k\in\{1,...,K\}^{6}. \tag{3.9}$$

变形可得$y_k(t)\le Q_k(t+1)-Q_k(t)$, 对所有时间片执行累加操作, 即可得

$$\sum_{t=0}^{T-1}y_k(t)\le Q_k(T)-Q_k(0)=Q_k(T), k\in\{1,...,K\}. \tag{3.10}$$

两边同时取期望可得

$$\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[y_k(t)]\le\frac{\mathbb{E}[Q_k(T)]}{T}, k\in\{1,...,K\}. \tag{3.11}$$

这时想必大家会好奇**期望**这个运算为何如此来去自由～. 这是因为(3.8)-(3.10)只是中间的推导阶段, 不加上期望运算可以让整个过程简洁易懂. 但是不含期望运算的(3.11)是没有意义的.

[5]即$\forall k\in\{1,...,K\}, Q_k(0)=0$.

[6]这是因为$\max\{a,b\}\ge a$且$\max\{a,b\}\ge b$.

现在, 让我们观察一下(3.7)和(3.11). 为了让前者成立, 我们可以让

$$\lim_{T\to\infty}\frac{\mathbb{E}[Q_k(T)]}{T}=0, k\in\{1,...,K\} \tag{3.12}$$

成立, 也可以让

$$\exists\delta\le 0, \lim_{T\to\infty}\frac{\mathbb{E}[Q_k(T)]}{T}\le\delta, k\in\{1,...,K\} \tag{3.13}$$

成立. (3.12)和(3.13)都可以让约束条件(3.7)恒成立, 但是显然后者是**更为严格的稳定**. 在Micheal J. Neely的书中, (3.12)被称为*中等程度的稳定(mean rate stable)*.

Lyapunov Optimization常常采用(3.12). 在对稳定性更为苛刻的系统中, 显然(3.13)更适合.

有了队列的概念, 我们就可以形成新问题了.

新的随机优化问题可定义成

$$\mathcal{P}_3: \min_{\forall t,\boldsymbol{\alpha}(t)\in\mathcal{A}^m}\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}[p(t)] \tag{3.14}$$

$$s.t. \quad (3.12).$$

3.4 How to Solve the Problem

3.4.1 Drift-plus-penalty Expression

说了这么多, 终于轮到Lyapunov函数(Lyapunov function)登场了. 正如前文所述, Lyapunov function是对当前时间片内所有queues' backlog的一个标量的、非负的描述. 将队列全体用一个矢量来描述: $\boldsymbol{\Theta}(t)\triangleq[Q_1(t),...,Q_K(t)]$, 则Lyapunov function定义[7]为

$$L(\boldsymbol{\Theta}(t))\triangleq\frac{1}{2}\sum_{k=1}^{K}Q_k(t)^2. \tag{3.15}$$

有了Lyapunov function, 我们可以用$\Delta(\boldsymbol{\Theta}(t))\triangleq L(\boldsymbol{\Theta}(t+1))-L(\boldsymbol{\Theta}(t))$[8]表示从时间片$t$到时间片$(t+1)$**全体queues' backlog的增长量**, 称之为*Lyapunov漂移(Lyapunov drift)*.

由(3.9)可知

$$Q_k(t+1)^2\le(Q_k(t)+y_k(t))^2, k\in\{1,...,K\}. \tag{3.16}$$

为什么(3.16)成立? 接下来给出证明.

分类讨论. (i)若$Q_k(t)+y_k(t)\ge 0$, 则$Q_k(t+1)=Q_k(t)+y_k(t)$, 因此

$$Q_k(t+1)^2=(Q_k(t)+y_k(t))^2, k\in\{1,...,K\}. \tag{3.17}$$

(ii)若$Q_k(t)+y_k(t)<0$, 则$Q_k(t+1)=0>Q_k(t)+y_k(t)$, 因此

$$Q_k(t+1)^2=0<(Q_k(t)+y_k(t))^2, k\in\{1,...,K\}. \tag{3.18}$$

结合(3.17)和(3.18), 可知(3.16)成立. 虽然直观上会觉得(3.16)和(3.9)相悖.

在(3.16)的基础上对所有K个队列累加, 可得

$$\frac{1}{2}\sum_{k=1}^{K}Q_k(t+1)^2\le\frac{1}{2}\sum_{k=1}^{K}Q_k(t)^2+\frac{1}{2}\sum_{k=1}^{K}y_k(t)^2+\sum_{k=1}^{K}Q_k(t)y_k(t), \tag{3.19}$$

[7]仔细阅读Micheal J. Neely的书, 会发现Lyapunov函数的严格定义为$L(\boldsymbol{\Theta}(t))\triangleq\frac{1}{2}\sum_{k=1}^{K}w_kQ_k(t)^2$, $\{w_k\}_{k=1}^{K}$为各个队列的权重.

[8]$\Delta(\boldsymbol{\Theta}(t))$的标准定义是: $\Delta(\boldsymbol{\Theta}(t))\triangleq\mathbb{E}[L(\boldsymbol{\Theta}(t+1))-L(\boldsymbol{\Theta}(t))|\boldsymbol{\Theta}(t)]$. 此处从简处理.

因此有

$$\begin{aligned}\Delta(\mathbf{\Theta}(t)) &= \frac{1}{2}\sum_{k=1}^{K}Q_k(t+1)^2-\frac{1}{2}\sum_{k=1}^{K}Q_k(t)^2\\ &\le \frac{1}{2}\sum_{k=1}^{K}y_k(t)^2+\sum_{k=1}^{K}Q_k(t)y_k(t)\le B+\sum_{k=1}^{K}Q_k(t)y_k(t),\end{aligned}\tag{3.20}$$

其中B是一个常数. 为什么$\frac{1}{2}\sum_{k=1}^{K}y_k(t)^2$一定存在一个正常数上界？观察(3.5)，其实可以找到反例[9]. 虽然在绝大多数实际问题中这一点都能得以满足，但只要存在反例就不应该这样描述. 因此，对于这一点我们先留个疑问$^{\triangle}$.

3.4.2 Drift-plus-penalty Algorithm

先前我们说过，引入队列之后就可以在不破坏当前这个时间片需要遵循的约定(控制队列保持稳定)的情况下直接优化$p(t)$，那么，如何实现对相关约定的遵循呢？一个很直观的方式就是**在每一个时间片内求解$\Delta(\mathbf{\Theta}(t))+V\cdot p(t)$的最小值**，也就是同时求解Lyapunov漂移和$p(t)$的最小值，借助权重V来调节对二者的重视程度. 因此我们可以形成问题$\mathcal{P}_4$:

$$\mathcal{P}_4:\min_{\forall t,\boldsymbol{\alpha}(t)\in\mathcal{A}^m}\mathbb{E}[\Delta(\mathbf{\Theta}(t))+V\cdot p(t)|\mathbf{\Theta}(t)]\tag{3.21}$$
$$s.t.\quad (3.12).$$

接下来我们会看到drift-plus-penalty algorithm就是通过求解$\mathcal{P}_4$的**近似最优解**来求解$\mathcal{P}_3$的.

可是，在时间片t内怎么去求解$\Delta(\mathbf{\Theta}(t))+V\cdot p(t)$的最小值呢？这个优化目标中，只有获得$L(\mathbf{\Theta}(t+1))$才能表示出$\Delta(\mathbf{\Theta}(t))$，这意味着我们需要下一个时间片的信息，这样我们就无法在当前这个时间片内解决这个问题了. 仔细观察(3.20)，如果我们对$\Delta(\mathbf{\Theta}(t))+V\cdot p(t)$ 做一些放缩处理，i.e.,

$$\Delta(\mathbf{\Theta}(t))+V\cdot p(t)\le B+V\cdot p(t)+\sum_{k=1}^{K}Q_k(t)y_k(t),\tag{3.22}$$

然后求解(3.22)的右端的最小值，这样问题就得以解决. 代价是**我们求解的并非原始问题的最优解**[10]. 因此，我们形成问题$\mathcal{P}_5$:

$$\mathcal{P}_5:\min_{\forall t,\boldsymbol{\alpha}(t)\in\mathcal{A}^m}\mathbb{E}[B+V\cdot p(t)+\sum_{k=1}^{K}Q_k(t)y_k(t)|\mathbf{\Theta}(t)]\tag{3.23}$$
$$s.t.\quad (3.12).$$

现在终于可以引入传说中的drift-plus-penalty algorithm了. **请仔细观察该算法. 可以发现该算法求解的其实是去掉了约束条件(3.12)的$\mathcal{P}_5$.** 其实从引入Lyapunov漂移$\Delta(\mathbf{\Theta}(t))$开始，也就是对问题$\mathcal{P}_4$和$\mathcal{P}_5$，对约束(3.12)的处理就已经被浓缩进**最小化$\Delta(\mathbf{\Theta}(t))$**中了. 至于为什么能够这样做，这是讨论时杜老师提出的问题，现在我可以解答了◡̈. 其实这个问题是在质疑**drift-plus-penalty**算法的合理性，想要回答这个问题也很简单，我们只需证明**drift-plus-penalty**算法取得的解能够让(3.12)恒成立即可$^{\triangle}$. 对这个问题的解答会放到§4.1.

[9]例如对于某个来自random event set的$\mathbf{w}$和某个来自random action set的$\boldsymbol{\alpha}$, $Y_k(\mathbf{w},\boldsymbol{\alpha})\to-\infty$.

[10]关于这一点有必要作一说明. 其实一旦使用了 Lyapunov优化，一定求不到原始问题最优解，因为此处作了放缩处理. 但是对于新问题——求解$B+V\cdot p(t)+\sum_{k=1}^{K}Q_k(t)y_k(t)$的最小值而言，最优解是可以取得的.

Algorithm 1 Drift-plus-penalty algorithm

1: 在时间片t的开头, 观察发生的所有随机事件以及所有队列的储备量: $\mathbf{w}(t), \mathbf{\Theta}(t)$.

2: 通过求解如下问题确定最优的控制决策$\boldsymbol{\alpha}^*(t)$:

$$\boldsymbol{\alpha}^*(t) = \underset{\boldsymbol{\alpha}(t)\in\mathcal{A}^m}{\operatorname{argmin}} \mathbb{E}[B + V \cdot p(t) + \sum_{k=1}^{K} Q_k(t) y_k(t) | \mathbf{\Theta}(t)]. \tag{3.24}$$

3: $\forall k \in \{1, ..., K\}$, 根据(3.8)更新$Q_k(t)$.

4: $t \leftarrow t + 1$.

至此, Lyapunov优化的核心部分已经讲解完毕. 可以发现, drift-plus-penalty算法的确将一个从长期角度来衡量的随机优化问题的求解具体到了每一个时间片内. 除此之外, 该算法不仅不需要来自未来的系统状态信息, 甚至也不需要知道随机事件$\mathbf{w}(t)$的概率分布! 这意味该算法几乎不需要对环境有任何认知, 在强化学习领域, 这样的算法被称为**免模型学习**(model-free learning).

4 Performance Analysis

所谓的性能分析(performace analysis)分析的是**drift-plus-penalty算法取得的解和原始问题$\mathcal{P}_2$的最优解之间的差距(optimality gap).** 要执行性能分析, 首先要描述出$\mathcal{P}_2$的最优解. 接下来分别从目标函数和队列稳定程度两个角度深入分析optimality gap.

4.1 Average Penalty Analysis

首先定义$\mathcal{P}_2$的$\mathbf{w}$-*only*策略*(*$\mathbf{w}$-*only policy)*.

$\mathbf{w}$-*only policy.* 这个策略(which is stationary and random)在每个时间片t内仅仅依赖于观测到的$\mathbf{w}(t)$来选择最优控制动作$\boldsymbol{\alpha}^\dagger(t)$. i.e., 对于每一个可能的随机事件$\mathbf{w}(t) \in \Omega^n$, $\mathbf{w}$-only policy依据一个条件概率分布选取$\boldsymbol{\alpha}^\dagger(t)$:

$$\boldsymbol{\alpha}^\dagger(t) = \underset{\boldsymbol{\alpha}(t)\in\mathcal{A}^m}{\operatorname{argmax}} \Pr(\boldsymbol{\alpha}(t)|\mathbf{w}(t)). \tag{4.1}$$

由此可见, $\mathbf{w}$-*only policy*是一个并不依赖于队列状态(queues' backlog)的策略[11].

接下来给出最优$\mathbf{w}$-*only*策略*(optimal* $\mathbf{w}$-*only policy)*的定义.

最优$\mathbf{w}$-*only*策略是满足如下条件的一个$\mathbf{w}$-*only*策略——在每个时间片内, 最优$\mathbf{w}$-*only*策略采取的动作$\boldsymbol{\alpha}^\star(t)$满足:

$$P(\mathbf{w}(t), \boldsymbol{\alpha}^\star(t)) = p^\star, \tag{4.2}$$

$$Y_k(\mathbf{w}(t), \boldsymbol{\alpha}^\star(t)) \leq 0, k \in \{1, ..., K\}, \tag{4.3}$$

其中$p^\star$是$\mathcal{P}_2$的最优目标函数值, i.e.,

$$p^\star = \min\big(\lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}[p(t)]\big). \tag{4.4}$$

[11]这是显然的, 因为原始问题$\mathcal{P}_2$中根本没有队列的概念.

请注意$\boldsymbol{\alpha}^{\dagger}(t)$和$\boldsymbol{\alpha}^{\star}(t)$的区别.

假设问题$\mathcal{P}_2$是存在最优$\mathbf{w}$-*only*策略的. 这是一个至关重要的假设, 它至少[12]意味着原始问题$\mathcal{P}_2$是有最优解的. 如果没有最优解, 那么optimality gap也无从说起. 但就算有最优解, 倘若不知道这个最优解是如何构造出来的, 我们还是无法做optimality gap的分析. 而最优$\mathbf{w}$-*only*策略正是提供了最优解的一种构造方法. 实际上, Micheal J. Neely在书中论证过: **如果原始问题$\mathcal{P}_2$有解, 那么$\mathcal{P}_2$一定存在最优$\mathbf{w}$-*only*策略. 如无特别强调, 接下来的$\boldsymbol{\alpha}(t)$表示在每个时间片t执行drift-plus-penalty algorithm得到的策略, $\Delta(\boldsymbol{\Theta}(t))$和$p(t)$分别表示执行drift-plus-penalty算法得到的Lyapunov drift和penalty.**

现在开始分析optimality gap. 由(3.22)和(3.24)可得

$$\begin{aligned}\Delta(\boldsymbol{\Theta}(t))+V\cdot p(t) &\leq B+V\cdot p(t)+\sum_{k=1}^{K}Q_k(t)y_k(t)\\ &= B+V\cdot P(\mathbf{w}(t),\boldsymbol{\alpha}(t))+\sum_{k=1}^{K}Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}(t))\\ &\leq B+V\cdot P(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))+\sum_{k=1}^{K}Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t)).\end{aligned}\tag{4.5}$$

对(4.5)两边同时取期望[13], 得

$$\begin{aligned}&\mathbb{E}[\Delta(\boldsymbol{\Theta}(t))+V\cdot p(t)|\boldsymbol{\Theta}(t)]\\ \leq\ & \mathbb{E}[B+V\cdot P(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))+\sum_{k=1}^{K}Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))|\boldsymbol{\Theta}(t)] &(4.6)\\ =\ & B+V\cdot\mathbb{E}[P(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))|\boldsymbol{\Theta}(t)]+\sum_{k=1}^{K}\mathbb{E}[Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))|\boldsymbol{\Theta}(t)] &(4.7)\\ =\ & B+V\cdot\mathbb{E}[P(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))]+\sum_{k=1}^{K}\mathbb{E}[Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))|\boldsymbol{\Theta}(t)] &(4.8)\\ =\ & B+V\cdot\mathbb{E}[P(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))]+\sum_{k=1}^{K}\mathbb{E}[Q_k(t)|\boldsymbol{\Theta}(t)]\mathbb{E}[Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))|\boldsymbol{\Theta}(t)] &(4.9)\\ =\ & B+V\cdot\mathbb{E}[P(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))]+\sum_{k=1}^{K}\mathbb{E}[Q_k(t)|\boldsymbol{\Theta}(t)]\mathbb{E}[Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))] &(4.10)\\ \leq\ & B+V\cdot p^{\star}. &(4.11)\end{aligned}$$

上述推导中, 从(4.6)到(4.7)是因为期望的运算规则; 从(4.7)到(4.8)是因为最优$\mathbf{w}$-*only*策略不依赖于队列状态$\boldsymbol{\Theta}(t)$, 因而$P(\cdot)$与$\boldsymbol{\Theta}(t)$无关; 从(4.8)到(4.9)是因为$Q_k(t)$和$Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\star}(t))$相互独立[14], 因为最优$\mathbf{w}$-*only*策略与队列$Q_k(t)$无关; 从(4.9)到(4.10)是因为最优$\mathbf{w}$-*only*策略得到的$Q_k(\cdot)$与队列状态$\boldsymbol{\Theta}(t)$无关; 从(4.10)到(4.11)是因为(4.3).

[12]请体会此处'至少'的含义.

[13]注意$\Delta(\boldsymbol{\Theta}(t))$的严格定义(在脚注8给出), 因为本身就携带了条件期望, 因此$\mathbb{E}[\Delta(\boldsymbol{\Theta}(t))|\boldsymbol{\Theta}(t)]=\mathbb{E}[\Delta(\boldsymbol{\Theta}(t))]$.

[14]能否去掉条件期望中的条件,要看期望本身是否与条件相互独立.

将上述推导结论对时间片进行累加, 得到

$$
\begin{aligned}
(B+V\cdot p^{\star})T &\geq \sum_{t=0}^{T-1}\mathbb{E}[\Delta(\mathbf{\Theta}(t))+V\cdot p(t)|\mathbf{\Theta}(t)] && (4.12)\\
&= \mathbb{E}[L(\mathbf{\Theta}(T))-L(\mathbf{\Theta}(0))]+V\cdot\sum_{t=0}^{T-1}\mathbb{E}[p(t)|\mathbf{\Theta}(t)] && (4.13)\\
&= \mathbb{E}[L(\mathbf{\Theta}(T))]+V\cdot\sum_{t=0}^{T-1}\mathbb{E}[p(t)|\mathbf{\Theta}(t)] && (4.14)\\
&\geq V\cdot\sum_{t=0}^{T-1}\mathbb{E}[p(t)|\mathbf{\Theta}(t)] && (4.15)
\end{aligned}
$$

从(4.14)到(4.15)是因为$L(\mathbf{\Theta}(T))\geq 0$. **因此我们可以得到最终结论:**

$$\sum_{t=0}^{T-1}\mathbb{E}[p(t)|\mathbf{\Theta}(t)]\leq p^{\star}+\frac{B}{V}. \tag{4.16}$$

这个结论意味着什么呢? 这意味着**只要将V设置的足够大, 通过执行drift-plus-penalty算法得到的解可以无限逼近$\mathcal{P}_2$的最优解$p^{\star}$.** 那么我们常看到的$O(\frac{1}{V})$又是什么意思呢? 我们先来回顾一下《算法导论》中对于符号O的定义:

> **对于给定的函数$g(x)$, 用$O(g(x))$表示以下函数的集合:**
>
> $$O(g(x))\triangleq\{f(x)|\exists c,x_0\in\mathbb{R}^{+},\forall x\geq x_0,0\leq f(x)\leq cg(x)\}. \tag{4.17}$$

观察(4.16)的右端, 只有一个变量(可调节参数)V, 其余均为定值(常数). 因此, 对于(4.16), $g(V)=p^{\star}+\frac{B}{V}$, $\sum_{t=0}^{T-1}\mathbb{E}[p(t)|\mathbf{\Theta}(t)]$就可以用$O(g(V))=O(p^{\star}+\frac{B}{V})=O(\frac{1}{V})$来描述.

接下来我们证明为什么执行drift-plus-penalty算法得到的解能够让约束条件(3.12)恒成立. 假设[15]**存在一个w-*only*策略$\boldsymbol{\alpha}^{\bullet}(t)$(不强求是最优w-*only*策略))满足**

$$\exists\epsilon>0,\mathbb{E}[Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\bullet}(t)]\leq-\epsilon,k\in\{1,...,K\}. \tag{4.18}$$

此外, **我们还需要假设$P(\cdot)$是有界函数, i.e.,**

$$\exists p_{min},p_{max}\in\mathbb{R},\forall\mathbf{w}\in\Omega^{n},\boldsymbol{\alpha}\in\mathcal{A}^{m},p_{min}\leq P(\mathbf{w},\boldsymbol{\alpha})\leq p_{max}. \tag{4.19}$$

类似(4.5), 结合(3.22)与(3.24),我们可以得到

$$
\begin{aligned}
\Delta(\mathbf{\Theta}(t))+V\cdot p(t) &\leq B+V\cdot p(t)+\sum_{k=1}^{K}Q_k(t)y_k(t)\\
&= B+V\cdot P(\mathbf{w}(t),\boldsymbol{\alpha}(t))+\sum_{k=1}^{K}Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}(t))\\
&\leq B+V\cdot P(\mathbf{w}(t),\boldsymbol{\alpha}^{\bullet}(t))+\sum_{k=1}^{K}Q_k(t)Y_k(\mathbf{w}(t),\boldsymbol{\alpha}^{\bullet}(t)). && (4.20)
\end{aligned}
$$

[15]用棕色标示出的是假设, 他们是论证成立的前提. 在大多数实际问题中, 这些假设基本都能成立. 实际上我们在§3.4.1提出的疑问也是假设.

因此

$$\Delta(\mathbf{\Theta}(t)) + V \cdot p_{min} \leq B + V \cdot p_{max} + \sum_{k=1}^{K} Q_k(t) Y_k(\mathbf{w}(t), \boldsymbol{\alpha}^{\bullet}(t)). \tag{4.21}$$

对(4.21)两边同时取期望, 得到

$$\begin{aligned} & \mathbb{E}[\Delta(\mathbf{\Theta}(t))|\mathbf{\Theta}(t)] + V \cdot p_{min} \\ \leq \quad & B + V \cdot p_{max} + \sum_{k=1}^{K} \mathbb{E}[Q_k(t)|\mathbf{\Theta}(t)]\mathbb{E}[Y_k(\mathbf{w}(t), \boldsymbol{\alpha}^{\bullet}(t))] \qquad (4.22) \\ \leq \quad & B + V \cdot p_{max} - \epsilon \sum_{k=1}^{K} \mathbb{E}[Q_k(t)|\mathbf{\Theta}(t)] \qquad (4.23) \end{aligned}$$

所以可得

$$\mathbb{E}[L(\mathbf{\Theta}(t+1))] - \mathbb{E}[L(\mathbf{\Theta}(t))] \leq B' - \epsilon \sum_{k=1}^{K} \mathbb{E}[Q_k(t)|\mathbf{\Theta}(t)], \tag{4.24}$$

其中$B' \triangleq B + V(p_{max} - p_{min})$. 对所有时间片累加(4.24), 可得

$$\mathbb{E}[L(\mathbf{\Theta}(T))] - \mathbb{E}[L(\mathbf{\Theta}(0))] \leq B' \cdot T - \epsilon \sum_{t=0}^{T-1} \sum_{k=1}^{K} \mathbb{E}[Q_k(t)|\mathbf{\Theta}(t)], \tag{4.25}$$

根据(3.8)可知$Q_k(t) \geq 0$, 因此对于(4.25), 可以放缩得到

$$\mathbb{E}[L(\mathbf{\Theta}(T))] - \mathbb{E}[L(\mathbf{\Theta}(0))] \leq B' \cdot T. \tag{4.26}$$

将Lyapunov function的定义(3.15)带入上式可得

$$\frac{1}{2} \sum_{k=1}^{K} \mathbb{E}[Q_k(T)^2] \leq 0 + B' \cdot T. \tag{4.27}$$

由柯西不等式$\left(\sum_{i=1}^{n} x_i y_i\right)^2 \leq \left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i^2\right)$, 继续推导上式可得

$$\left(\sum_{k=1}^{K} \mathbb{E}[Q_k(T)]\right)^2 \leq K \sum_{k=1}^{K} \mathbb{E}[Q_k(T)^2] \leq 2KB' \cdot T. \tag{4.28}$$

我们已经十分接近结论了. 对上式两边开方并取极限得

$$\lim_{T\to\infty} \frac{\sum_{k=1}^{K} \mathbb{E}[Q_k(T)]}{T} \leq \lim_{T\to\infty} \sqrt{\frac{2KB'}{T}} = 0. \tag{4.29}$$

再次运用结论$Q_k(t) \geq 0$, 我们就成功解答了在§3.4.1提出的疑问——**所有队列都是*mean rate stable*:**

$$\forall k \in \{1, ..., K\}, \lim_{T\to\infty} \frac{\mathbb{E}[Q_k(T)]}{T} = 0. \tag{4.30}$$

这意味drift-plus-penalty的确能够求解原始问题$\mathcal{P}_2$.

4.2 Average Queue Size Analysis

本节分析执行drift-plus-penalty算法取得的queues' backlog**收敛到一个非正数的速度**.

对于§4.1推导得到的中间结论(4.25), 变形可得

$$\frac{1}{T} \sum_{t=0}^{T-1} \sum_{k=1}^{K} \mathbb{E}[Q_k(t)|\mathbf{\Theta}(t)] \leq \frac{B'}{\epsilon} + \mathbb{E}[L(\mathbf{\Theta}(0))] - \mathbb{E}[L(\mathbf{\Theta}(T))] \leq \frac{B + V(p_{max} - p_{min})}{\epsilon}. \tag{4.31}$$

上述过程运用了结论$\mathbb{E}[L(\mathbf{\Theta}(T))] \geq 0, \mathbb{E}[L(\mathbf{\Theta}(0))] = 0$.

现在我们已经得到了最终结论. 仔细观察(4.31), 它意味着什么呢? 同样地, 请回忆(4.17)对符号O的定义和$O(\frac{1}{V})$结论的分析过程. 这里V作为唯一可变参数决定了queues' backlog的大小, **它决定了原始问题$\mathcal{P}_2$的时均约束(3.7)收敛到一个非负数的速度.** 因此时均队列大小(time average queue size)可以用$O(V)$来衡量.

4.3 Trade-off by Tuning V

在§4.1和§4.2我们分别证明了time average penalty(optimization goal)和time average queue size(optimization constraints)分别是$O(\frac{1}{V})$和$O(V)$的. 显然, 参数V对于这二者有着完全相反的影响. 因此我们必须做出以下trade-off:

1. 获得更加逼近于原始问题$\mathcal{P}_2$最优解$p^\star$的解, 但需要一个更长的时间域才能让时均约束得以满足;

2. 在尽可能短的时间域内使时均约束得以满足, 但会牺牲目标函数值的优越性(只能取得一个相对较差的解).

实际上, 这就是算法执行时间(迭代次数)和算法效率(解的精度)之间的trade-off. 因为我们考虑的实际问题中不可能存在$T \to \infty$的系统, 因此算法的精度总会打一个折扣. 就这是为什么drift-plus-penalty algorithm常常被批评'只能取得近似最优解'的原因. 关于这个问题, 我已经在前文的脚注10中注明.

5 Conclusions

本文档给出drift-plus-penalty算法和理论分析后就不再续笔. 因为全文是针对一个普适的随机优化问题$\mathcal{P}_2$来分析的, 因此全部推导过程均具备普适性, 读者只需带入自己模型所定义问题即可.

虽然说从头到尾阅读Micheal J. Neely的书不现实, 但实际上我还是做了一些参考的工作的ㄟ. 如果我们认真阅读此书, 会发现书中有着严格得多的定义和详实的论证. Prof. Neely在本文档所阐述的基本模型的基础上, 又讨论了Lyapunov optimization(i)在存在真实队列的系统中的运用(而非从约束条件构造出来的虚拟队列); (ii)在随机凸优化问题中的具体实现; (iii)在不满足i.i.d.的随机事件的随机优化问题中的运用; (iv)在可变长时间片的时域系统中的运用; (v)在凸规划(convex programming)问题中的具体实现; (vi)在线性规划问题中的具体实现; ...

对于一类比较特殊的问题——*系统中存在**不收敛到非正数**且真实存在的队列*, 我在后续的工作中会专门写一篇monograph来分析.