

一种针对注水问题求解的优化算法研究及其 C++ 实现

58122315 田昱锟

东南大学人工智能学院，南京 210000

东南大学“新一代人工智能技术与交叉应用”教育部重点实验室

邮箱: 213221787@seu.edu.cn

摘要—在各式各样的凸优化问题中，注水问题（Water Filling Problem）是其中较为经典的一个，也是日常生活中应用较为广泛的一个。其在功率分配、信道条件优化等场景中都有重要应用。注水问题常转化为拉格朗日对偶问题（Lagrange Dual Problem）进行求解。本文将简要介绍如何运用这一方法求解注水问题，主要工作包括：利用凸优化理论找出注水问题的解；对其中重点步骤进行分析和绘图解释；提出在较高精度要求下求解该问题的一种有效算法并给出该算法的一个 C++ 实现。实现部分主要由两个模块组成：随机参数生成模块和注水问题求解模块。本项目详细源代码和补充材料均已开源在作者 Github 上：https://github.com/TTiannaiTT/2023_Fall_Optimization-WaterFilling。

I. 实验目标及要求

A. 实验目标

理解课本 P236 例题 5.2，并编程实现。

例 5.2 注水。考虑如下凸优化问题

$$\begin{aligned} & \text{minimize} && -\sum_{i=1}^n \log(\alpha_i + x_i) \\ & \text{subject to} && x \succeq 0, \quad \mathbf{1}^T x = 1, \end{aligned}$$

B. 实验要求

- 1、实验数据需自行随机生成，数据具体要求及范围可参考给出的实现。
- 2、使用 C++ 独立编写程序，严禁抄袭，发现一律实验 0 分处理！
- 3、绘图作为附加 5 分（当然最后不会超过实验总分 100 分），可以不绘制。

II. 工作简介

注水问题是一种信道功率分配问题，其在日常生活中有重要应用。运用凸优化理论中的拉格朗日对偶方法对其进行求解是一种被广泛使用的注水问题求解方法。本文将简要介绍这一方法的基本内容，展示如何运用这一方法求解注水问题，并对其中的核心步骤进行严格的数学证明；得出该问题的解的一般形式之后，本文将提出一种在较高精度要求下进行该问题的求解的算法，并给出一种该算法的 C++ 实现；最后，本文将对结果进行分析，并绘图进行可视化的展示与讨论。

所以，本文的主要工作包括：

(1) 利用凸优化理论找出注水问题的解, 并对其中的核心步骤进行严格的数学证明。

(2) 给出一个在较高精度要求下求解该问题的算法, 并提供一个 C++ 实现。(本项目详细原代码和补充支持材料均已开源在作者 github 上)。

(3) 对算法设计和执行中重点步骤进行分析和可视化绘图的解释与讨论。

III. 背景及相关工作

1) 注水问题: 注水问题本身是一种信道功率分配问题, 根据某种优化准则和信道状况对发送功率进行自适应分配, 一般而言, 主要是信道状况较好的时候多分配功率, 状况较差的时候少分配功率, 从而达到最大化功率的效果。问题形式一般为:

$$\begin{aligned} \max_{P_1, \dots, P_K} \quad & \sum_{k=1}^K \alpha_k \log(L_k + P_k) \\ \text{s.t.} \quad & \sum_{k=1}^K P_k = P \\ & P_k \geq 0 \end{aligned}$$

本实验主要求解其简化形式的问题:

例 5.2 注水。考虑如下凸优化问题

$$\begin{aligned} \text{minimize} \quad & -\sum_{i=1}^n \log(\alpha_i + x_i) \\ \text{subject to} \quad & x \succeq 0, \quad \mathbf{1}^T x = 1, \end{aligned}$$

2) 拉格朗日对偶问题: 在一个优化问题中, 原始问题通常会带有很多约束条件, 这样直接求解原始问题往往是很困难的, 于是考虑将原始问题转化为它的对偶问题, 通过求解它的对偶问题来得到原始问题的解。对于原问题:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{s.t.} \quad & f_i(x) \leq 0, i = 1, 2, \dots, m \\ & h_j(x) = 0, j = 1, 2, \dots, p \end{aligned}$$

其拉格朗日函数:

$$L(x, \lambda, v) = f_0(x) + \sum_{i=1}^m \lambda_i f_i(x) + \sum_{j=1}^p v_j h_j(x)$$

其拉格朗日对偶函数:

$$g(\lambda, v) = \inf_x L(x, \lambda, v)$$

由此, 对偶性 (Duality) 是凸优化问题的核心内容。对于强对偶问题, 对偶问题的最优解与原问题的最优解是相同的, 在注水问题中, 可以证明, 注水问题转化对偶问题的最优解就是原注水问题的最优解。(详见 IV.A 节)

IV. 实验方案

A. 理论分析

1) 对偶问题最优解的有效性: 根据凸优化理论, 满足 Slater 条件的优化问题为强对偶问题。不难证明, 注水问题为凸优化问题, 且存在至少一个可行解满足严格不等式条件, 故注水问题满足 Slater 条件, 所以可以通过求解拉格朗日对偶函数求出原注水问题的最优解, 并可以使用 KKT 条件求出该问题的解。

2) 利用 KKT 条件求出注水问题解的形式: 首先写出对于原问题及其对偶问题的 KKT 条件: 对不等式约束 $\mathbf{x}^* \succeq 0$ 引入 Lagrange 乘子 $\boldsymbol{\lambda}^* \in \mathbb{R}^n$, 对等式约束 $\mathbf{1}^T \mathbf{x}^* = 1$ 引入乘子 $\nu^* \in \mathbb{R}$, 可以得到如下的 KKT 条件:

$$\begin{aligned} \mathbf{x}^* &\succeq 0, \\ \mathbf{1}^T \mathbf{x}^* &= 1, \\ \boldsymbol{\lambda}^* &\succeq 0, \\ \lambda_i^* x_i^* &= 0, \quad i = 1, \dots, n, \\ -1/(\alpha_i + x_i^*) - \lambda_i^* + \nu^* &= 0, \quad i = 1, \dots, n. \end{aligned} \tag{1}$$

注意到 λ^* 可以消去, 得:

$$\mathbf{x}^* \succcurlyeq 0, \quad (2a)$$

$$\mathbf{1}^T \mathbf{x}^* = 1 \quad (2b)$$

$$x_i^*(\nu^* - 1/(\alpha_i + x_i^*)) = 0, i = 1, \dots, n, \quad (2c)$$

$$\nu^* - 1/(\alpha_i + x_i^*) \geq 0, i = 1, \dots, n. \quad (2d)$$

经化简可得方程组:

$$\mathbf{x}_i^*(\nu^* - 1/(\alpha_i + x_i^*)) = 0, \quad (3a)$$

$$\nu^* \succcurlyeq 1/(\alpha_i + x_i^*) \quad (3b)$$

观察上式, 结合上述 KKT 条件, 可以看出: 如果 $\nu^* < 1/\alpha_i$, 那么只有当 $x_i^* > 0$ 时式(2d)才可能成立, 而由式(2c)可知 $\nu^* = 1/(\alpha_i + x_i^*)$, 所以, 当 $\nu^* < 1/\alpha_i$ 时, 有 $x_i^* = 1/\nu^* - \alpha_i$ 。如果 $\nu^* \geq 1/\alpha_i$, 且 $x_i^* > 0$, 那么 $\nu^* \geq 1/\alpha_i > 1/(\alpha_i + x_i^*)$, 这违背了互补松弛条件。因此当 $\nu^* \geq 1/\alpha_i$ 时, 必有 $x_i^* = 0$ 。所以可得如下最优解:

$$x_i^* = \begin{cases} 1/\nu^* - \alpha_i & \nu^* < 1/\alpha_i \\ 0 & \nu^* \geq 1/\alpha_i \end{cases} \quad (4)$$

即:

$$x_i^* = \max\{0, 1/\nu^* - \alpha_i\}. \quad (5)$$

再由等式约束(2b)可得:

$$\sum_{i=1}^n \max\{0, 1/\nu^* - \alpha_i\} = 1. \quad (6)$$

式(6)即为注水问题求解的最终转化问题, 下面将提出求解该式子的一般算法, 并给出运行该算法的一种 C++ 实现。

B. 算法设计

根据上述注水问题的解, 下面提出一种可求解较高精度要求下的注水问题的算法:

Algorithm 1 Water flooding algorithm

Input: α for parameters, ε for precision, k for dimension

Init: $Min = 0.0, Max = 1 + 1/k, init = (Min + Max)/2$

function WATERFILLING

while result-1 $> \varepsilon$ **do**

$1/\nu \leftarrow init$

for $i = 1$ to k **do**

$volume \leftarrow \max\{1/\nu - \alpha_i, 0\}$

end for

$result \leftarrow \sum_{i=1}^k x_i$

if $result > 1$ **then**

$Max = init$

$init = (init + Min)/2$

end if

if $result < 1$ **then**

$Min = init$

$init = (init + Max)/2$

end if

end while

end function

Output: x^*

该算法通过使用类似于二分查找的方法迅速缩小变量可能的取值范围, 在 α 的维度较大的时候仍然拥有较快的搜索速度, 是一种相当优越的求解算法。同时通过每次与设定精度进行比较进行上下界更新, 该算法得以在给定精度要求条件下运行。

C. 代码实现

```
1 void waterfilling() {
2     double Init = 0.5;
3     double Min = 0.0;
```

```

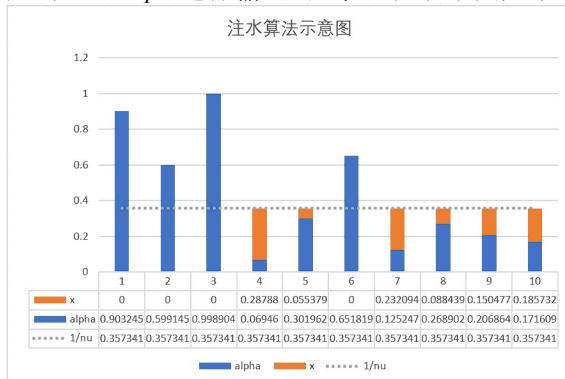
4      double Max = 1 + double(1) / dimension;
5      double result = 0;
6
7      while (abs(result-1)>precision){
8          fill(nu, nu + dimension, Init);
9          for (int i = 0; i < dimension; i++) {
10             x[i] = max(nu[i] - alpha[i], 0.0);
11
12         }
13         result = 0;
14         for (auto& i : x)
15             result += i;
16
17         if (result > 1) {
18             Max = Init;
19             Init = (Init + Min) / 2;
20         }
21         else if (result < 1) {
22             Min = Init;
23             Init = (Init + Max) / 2;
24         }
25     }
26 }

```

V. 实验过程

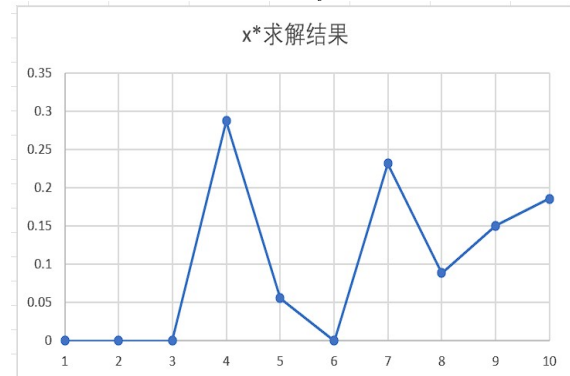
整体实验流程：

- (1) 随机生成一组参数值 α ，要求 $\alpha_i \in (0,1)$;
- (2) 按照 IV.B 节所述算法，进行注水求解，输入问题的维度、要求的精度和 (1) 中生成的 α ，算法返回一组满足要求的 x^* ;
- (3) 根据 x^* 的结果，计算最优值 p^* ，然后对 α, x^*, p^* 进行输出。注水过程效果图如下：



从图中不难看出，最终对于 x^* 的求解类似于一个注水过程， x_i^* 的值即为每个维

度上的 $\max\{0, 1/\nu^* - \alpha_i\}$ ，所以形象地解释为向一个凹凸不平的水箱中注水，要求注水的总量为 1，求最终达到的高度。每个位置对应的注水量即为 x_i^* ，结果示意图如下：



VI. 实验结果及分析

A. 实验结果

下面将给出三组不同随机生成的 α 的对应的结果截图：

```

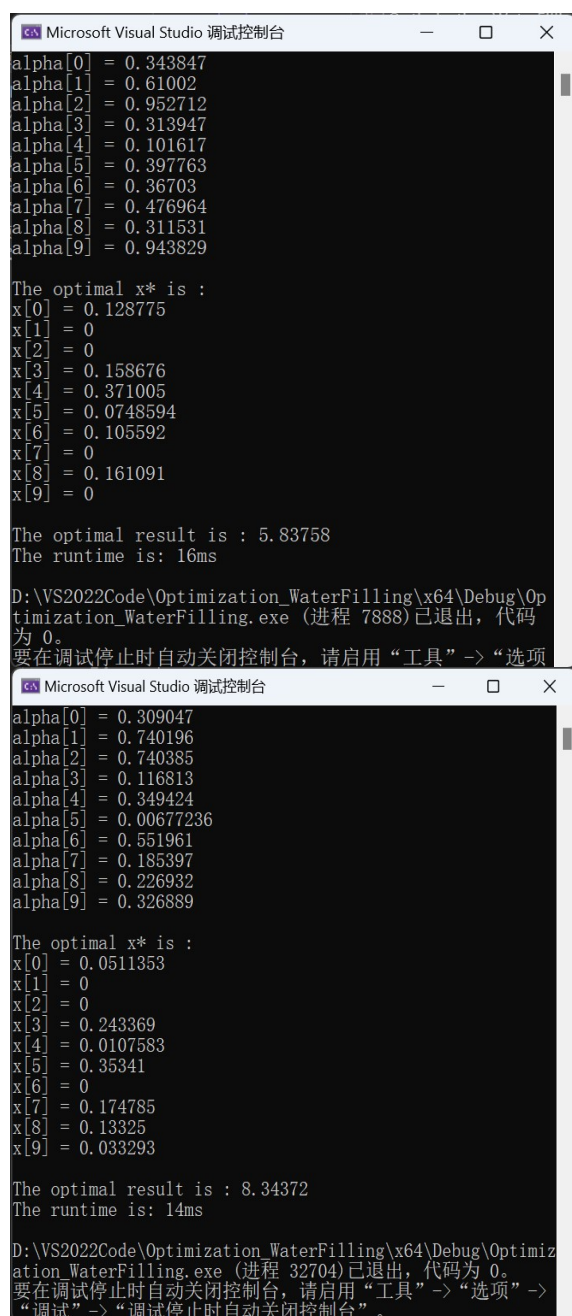
Microsoft Visual Studio 调试控制台
alpha[0] = 0.814382
alpha[1] = 0.18451
alpha[2] = 0.309774
alpha[3] = 0.296245
alpha[4] = 0.928579
alpha[5] = 0.0305757
alpha[6] = 0.0240995
alpha[7] = 0.120854
alpha[8] = 0.991982
alpha[9] = 0.318105

The optimal x* is :
x[0] = 0
x[1] = 0.141799
x[2] = 0.016535
x[3] = 0.030064
x[4] = 0
x[5] = 0.295733
x[6] = 0.302209
x[7] = 0.205455
x[8] = 0
x[9] = 0.00820408

The optimal result is : 8.12685
The runtime is: 18ms

D:\VS2022Code\Optimization_WaterFilling\x64\Debug\Optimization_WaterFilling.exe (进程 11588)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。

```



```
Microsoft Visual Studio 调试控制台
alpha[0] = 0.343847
alpha[1] = 0.61002
alpha[2] = 0.952712
alpha[3] = 0.313947
alpha[4] = 0.101617
alpha[5] = 0.397763
alpha[6] = 0.36703
alpha[7] = 0.476964
alpha[8] = 0.311531
alpha[9] = 0.943829

The optimal x* is :
x[0] = 0.128775
x[1] = 0
x[2] = 0
x[3] = 0.158676
x[4] = 0.371005
x[5] = 0.0748594
x[6] = 0.105592
x[7] = 0
x[8] = 0.161091
x[9] = 0

The optimal result is : 5.83758
The runtime is: 16ms

D:\VS2022Code\Optimization_WaterFilling\x64\Debug\Optimization_WaterFilling.exe (进程 7888)已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”

Microsoft Visual Studio 调试控制台
alpha[0] = 0.309047
alpha[1] = 0.740196
alpha[2] = 0.740385
alpha[3] = 0.116813
alpha[4] = 0.349424
alpha[5] = 0.00677236
alpha[6] = 0.551961
alpha[7] = 0.185397
alpha[8] = 0.226932
alpha[9] = 0.326889

The optimal x* is :
x[0] = 0.0511353
x[1] = 0
x[2] = 0
x[3] = 0.243369
x[4] = 0.0107583
x[5] = 0.35341
x[6] = 0
x[7] = 0.174785
x[8] = 0.13325
x[9] = 0.033293

The optimal result is : 8.34372
The runtime is: 14ms

D:\VS2022Code\Optimization_WaterFilling\x64\Debug\Optimization_WaterFilling.exe (进程 32704)已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
```

B. 实验结果分析

由上图可见实验结果, 经检验, 算法运行正常, 求解结果正确。

C. 算法分析

经大量检验, 算法运行时间基本落于 10-25ms 之内, 根据随机生成的 α 的数据求解难度略有不同, 但总体运行非常稳定, 且在较高精度要求下仍能在较短时间内给出一个符合精度要求的结果, 算法具备较强的优越性。

VII. 实验结论

本文主要工作包括: 利用凸优化理论找出注水问题的解; 对其中重点步骤进行分析和绘图解释; 提出在较高精度要求下求解该问题的一种有效算法并给出该算法的一个 C++ 实现。从理论上来看, 本文从优化理论出发, 严格证明了注水问题的解的数学形式, 并提出了一种用于在较高精度要求下通过计算机仿真进行求解的算法。从结果上来看, 本文所采用的算法在较高精度要求下仍然具备较好的运行效率和较低的计算复杂度, C++ 的实现方案也较为简洁, 容易部署使用。