2022-秋-《最优化方法(研)》考核作业

姓名：高伟
学号：2220321236
专业：电子信息
班级：研 2217

题目 有约束优化问题的各类约束处理方法及特点

1. 引言

优化问题是指在一些约束条件下，决定一些可选择的变量应该取何值，使目标函数达到最优的问题。优化问题在工程设计、数学规划、深度学习等领域有广泛的应用，而约束优化问题是指自变量取值受到一定限制的优化问题。约束优化问题的一般结构为[1]:

$$\begin{aligned} \min \quad & f(x) \\ sx \quad & h_i(x)=0, i\in M=\{1,2,\ldots,l\}, \\ & g_j(x)\geq 0, j\in N=\{1,2,\ldots,m\}. \end{aligned}$$

其中，$f: R^n \to R^n$为目标函数，可行域是$B=\{x\in R^n \mid h_i(x)=0(i\in M), g_j(x)\geq 0(j\in N)\}$，优化问题满足可行域 B 的向量 x 称为可行解。可行域中使得目标函数取得最小值的向量 x 称为问题 f 的最优解，当$M=N=\emptyset$时，该问题为无约束优化问题。

约束优化问题按照约束分类可以分为无约束优化问题、等式约束优化问题和不等式约束优化问题；按照目标数量可以分为单目标优化问题和多目标优化问题；按照变量性质可以分为数值优化问题和组合优化问题；按照目标函数可以分为线性规划问题、非线性规划问题和二次规划问题。[11]

约束优化问题是指在给定的目标函数和约束条件之间寻找最优解的问题。例如，我们可能想要在有限的预算和时间内完成一个工程项目，或者在满足一定的质量和安全标准下设计一个产品。这些都是约束优化问题的实例。

约束优化问题具有很高的理论意义和实际应用价值，因为它们广泛地存在于科学研究和工程实践中。在机器学习中，我们经常需要在训练数据和模型复杂度之间找到一个平衡点，以达到最佳的泛化能力；在控制理论中，我们需要在系统稳定性和性能之间设计合适的控制器。

然而，约束优化问题也具有很大的求解难度，因为它们涉及到多个相互影响、相互制约、甚至相互冲突的因素。如何平衡约束条件和目标函数之间的关系，以及如何有效地搜索可行解空间，是设计约束优化方法的关键挑战。

为了求解约束优化问题，人们提出了许多不同类型和技术的方法。如序列无约束优化方法，即把约束问题转化为一系列无约束问题，用这些无约束问题的极小值点去逼近约束极小值点；序列二次规划法，在迭代的每一步用一个二次函数逼近目标函数，用线性约束近似一般约束，构造一系列二次规划来逼近原问题；可行方向法，每次迭代找到可行下降方向，保证迭代始终处于可行域内；以及进化算法，这是一类比较新颖且具有潜力的方法，这是模拟自然界生物进化过程的一类基于群体搜索的随机优化算法。[2]

2. 序列无约束优化方法

对于无约束的优化问题，我们可以直接通过直接求解的方法得到最优解，这种方法称为解析法。序列无约束优化方法就是把约束优化问题转化为无约束优化问题，然后在利用无约束优化问题求解方法求解，用这些无约束问题的极小值点去逼近约束极小值点，如惩罚函数法、乘子罚函数法。本文以惩罚函数法为例进行简单介绍。

惩罚函数法的基本思想是根据约束条件的特点将其转化为某种惩罚函数加到目标函数中去，以而将约束优化问题转化为一系列的无约束优化问题来求解。这些无约束问题由原问题及罚函数，再加上惩罚因子组成；而且，这些无约束问题的解会收敛于所求问题的解。[3]

惩罚函数法的基本形式如下，首先定义约束问题:

$$\begin{aligned} \min \quad & f(x) \\ sx \quad & h_i(x)=0, i\in M=\{1,2,\ldots,l\}, \\ & g_j(x)\geq 0, j\in N=\{1,2,\ldots,m\}. \end{aligned}$$

然后定义该问题的罚函数：$\bar{P}(x)=\sum_{i=1}^{l} h_i^2(x)+\sum_{i=1}^{m}[\min\{0,g_i(x)\}]^2$，以及增广目标函数$P(x,\sigma)=f(x)+\sigma\bar{P}(x)$，其中$\sigma>0$称为罚因子，这样约束问题就转化为无约束优化问题：$\min\ P(x,\sigma)$，

下面给出外点法的算法步骤[3]:

步 0：给定初始点$x_0\in R^n$，终止误差$0\leq\epsilon\leq 1$，$\sigma_1>0$，$\gamma>1$，k=1；

步 1：x_{k-1}为初始点求解子问题$\min\ P(x,\sigma_k)$。令其极小点为x_k；

步 2：若$\sigma_k\bar{P}(x)\leq\epsilon$，停算，输出$x^*=x_k$以作为近似极小点；

步 3：令$\sigma_{k+1}=\gamma\sigma_k$，k=k+1，转步 1。

序列无约束优化方法是一种基于梯度的优化方法，它通过沿着目标函数下降最快的方向进行搜索，逐步逼近极小点序列无约束算法结构比较简单，可以直接调用无约束优化算法的通用程序，编程易于实现，该方法具有如下特点:

- 理论明确，程序简单；
- 由于梯度是函数的局部性质，所以整体收敛速度不快
- 搜索路线呈锯齿状，在远离极小点时逼近速度较快，接近极小点时速度较慢。
- 收敛速度与目标函数性质密切相关 。目标函数的等值线形成的椭圆簇越扁，迭代次数越多，搜索难以到达极小点；而同心圆，或椭圆簇对称轴，则一次搜索即可到达。

3. 序列二次规划法

序列二次规划法（SQP）是一种求解非线性约束优化问题的方法。它的基本思想是：在每次迭代中，利用泰勒展开将目标函数和约束函数在当前点附近近似为二次函数和线性函数；通过求解这个近似的二次规划子问题，得到一个搜索方向；通过一维搜索或信赖域方法，确定一个合

适的步长，更新当前点；重复上述过程，直到满足收敛准则。[9]

序列二次规划法的原理是利用目标函数和约束函数的近似，构造一个二次规划子问题，并求解该子问题得到搜索方向和步长，然后进行约束一维搜索，找到最优步长，并更新迭代点，直到满足收敛条件。具体来说，序列二次规划法的步骤如下：

- 首先利用泰勒展开将非线性优化问题的目标函数在迭代点处简化为二次函数，同时将约束函数简化为线性函数。
- 然后求解二次规划子问题，得到搜索方向和步长。可以使用一些成熟的快速求解方法，如 KKT 条件、梯度投影法等。
- 然后进行约束一维搜索，找到满足一定条件的最优步长。可以使用一些有效的搜索方法，如 Armijo 法、Goldstein 法等。
- 最后，判断收敛条件，如果满足则停止迭代，否则更新迭代点并重复上述步骤。收敛条件可以是目标函数值的变化、梯度范数的大小、迭代次数等。

序列二次规划法的算法如下：

步 0：给定初始点 x_0，收敛精度ϵ，令$k=0$。

步 1：在点 x_k简化原问题为二次规划子问题，利用牛顿法或拟牛顿法求解，得到增量 d_k 和拉格朗日乘子 λ_k。

步 2：通过一维搜索或信赖域方法，在方向d_k上确定一个合适的步长α_k，更新当前点为 $x_{k+1}=x_k+\alpha_k d_k$。

步 3：检查是否满足终止条件，如果是，则停止迭代；如果否，则令 $\mathrm{k}=\mathrm{k}+1$，返回第二步。

序列二次规划法具有以下特点：收敛性好，能够收敛到约束问题的 KKT 点。计算效率高，每次迭代只需要求解一个二次规划子问题，而不需要求解非线性方程组。边界搜索能力强，能够处理边界约束和不等式约束。需要计算目标函数和约束函数的一阶和二阶导数，对于复杂的非线性函数可能很困难或者不可行。对于 Hessian 矩阵的正定性要求较高，如果不正定则可能导致子问题无解或者无界。对于大规模的问题，子问题的求解可能很耗时或者内存不足。序列二次规划法一般适用于中小规模的非线性约束优化问题，尤其是目标函数和约束函数都有良好的光滑性和凸性。不适用于大规模、稀疏、非光滑或者非凸的非线性约束优化问题。

4. 可行方向法

可行方向法是一类直接处理约束优化问题的方法，不需要转化为无约束问题，其基本思想是要求每一步迭代产生的搜索方向不仅是对目标函数是下降方向，而且对约束函数来说是可行方向，及迭代点总是满足所有的约束条件，不同的可向方向法的主要区别是选取可行方向的策略不同[3]。

可行方向法的具体步骤如下：

- 给定一个初始可行点和一个容许误差。
- 在当前可行点处，用某种方法确定一个可行下降方向。
- 沿着该方向进行一维搜索，找到一个新的可行点。
- 判断是否满足终止条件，如果是，则停止迭代；否则，返回步骤 2。

不同的可行方向法主要区别在于如何选择可行下降方向。常用的方法有：通过求解一个线性规划来确定，如 Zoutendijk 法；利用投影矩阵直接构造，如梯度投影法；利用简约矩阵直接构造，如简约梯度法。

本文以线性约束的 Zoutendijk 可行方向法为例进行简单介绍：

首先确定约束问题：

$$\begin{aligned} \min \quad & f(x) \\ s.t. \quad & Ax \geq b \\ & Ex = e \end{aligned}$$

f(x)连续可微，A 是 m*n 矩阵，E 是 l*n 矩阵，$x \in \mathbb{R}^n, b \in \mathbb{R}^m, e \in \mathbb{R}^l$，可行方向 d 是 f(x)在$\bar{x}$处的下降方向的充要条件是$\nabla f(\bar{x})^T d < 0$。要寻找约束问题的可行点$\bar{x}$处的一个下降可行方向 d，可以通过求解线性规划问题得到：

$$\begin{aligned} \min \quad & \nabla f(\bar{x})^T d, \\ \text{s.t.} \quad & A_1 d \geq 0, \\ & Ed = 0, \\ & -1 \leq d_i \leq 1, i = 1, \cdots, n, \end{aligned}$$

确定可行方向后，x_k的后继点为x_{k+1}，$x_{k+1}=x_k+\alpha_k d_k$，步长α可以通过求解下面的以为搜索来解决，

$$\begin{aligned} &\min_{0 \leq \alpha \leq \bar{\alpha}} f(x_k + \alpha d_k), \\ &\bar{\alpha} = \max\{\alpha \mid x_k + \alpha d_k \in \mathcal{F}\}. \end{aligned}$$

综上，Zoutendijk 法的算法如下：

步 0：给定初始可行点x_0，终止误差$0 < \varepsilon_1 \ll 1$，$0 < \varepsilon_2 \ll 1$，令 k:=0。

步 1：在x_k 处，将不等式约束分为有效约束和非有效约束。

步 2:若x_k是可行域的一个内点（此时问题中没有等式约束，即 E=0，且$A_1=0$，且$\|\nabla f(\bar{x})^T d\| < \varepsilon_1$，停算，得到近似极小点$x_k$；否则，若$x_k$是可行域的一个内点，但$\|\nabla f(\bar{x})^T d\| > \varepsilon_1$，则取搜索方向$d_k = -\nabla f(\mathrm{x})^T d$，转步 5（用目标函数的负梯度方向作为搜索方向求步长，此时类似于无约束优化问题），若x_k不是可行域的一个内点，则转步 3。

步 3：求解线性规划问题，设求得最优解和最优值分别为d_s和z_k

步 4：若$|z_k| < \varepsilon_2$，停算，输出x_k 作为近似极小点，否则，以d_k作为搜索方向，转步 5。

步 5：首先由（10.13）和（10.14）计算α，然后作一维线搜索：$\begin{cases} \min & f(x_k + \alpha d_k), \\ \text{s.t.} & 0 \leq \alpha \leq \bar{\alpha}, \end{cases}$求得最优解$\alpha_k$。

步 6：置$x_{k+1} \coloneqq x_k + \alpha * x_k$，转步 1。

可行方向法是一种通过在可行域内直接搜索最优解的约束优化方法，它可以看作无约束优化下降算法的自然推广。可行方向法的核心是选择可行下降搜索方向和确定搜索步长。

可行方向法适用于一般的非线性约束优化问题，特别是当约束条件较多或者复杂时，可行方向法可以有效地减

少计算量和提高求解效率。在工程设计、运筹学、经济学等领域中都用一定的应用。

可行方向法的优点是可以避免求解拉格朗日对偶问题或者 KKT 条件，从而简化了计算过程。可以直接在非线性的设计空间进行搜索，不需要转换为线性规划问题。它也可以处理不等式约束和等式约束。可行方向法的缺点是需要保证每次迭代都能找到一个可行下降方向，否则可能导致迭代停滞或者无法收敛。它也需要选择合适的搜索方向和步长，这可能比较复杂和耗时。它也可能陷入局部最优解或无界解。

5. 进化约束优化方法

进化约束优化问题是一类非线性优化问题，需要在满足一些约束条件的同时，优化多个可能冲突的目标。进化算法是一种受自然启发的元启发式方法，可以有效地处理这类问题。[5]

这是一种模拟自然过程的全局优化方法，它用个体来代表待求解问题的解，将一定数量的不同个体组成种群。从初始种群开始，采用变异、交叉、选择等操作，模仿自然界的进化过程，引导种群进行进化，使得种群中的个体逐步接近问题的最优解。与传统优化算法相比，进化算法是一种基于群体的搜索技术,具有鲁棒性强、搜索效率高、不易陷入局部最优等特点，因此，它更适合于求解约束优化问题。然而,进化算法的本质是一种无约束的优化技术，必须将其与一定的约束处理机制相结合，构成约束优化进化算法，才能更好地求解复杂的约束优化问题[6]

下面是进化计算的一般步骤[7]

一般说，进化计算包含以下步骤：

一. 进化代数计数器初始化：t←0；

二. 随机产生初始种群 P(t)；

三. 评价群体 P(t)的适应度；

四. 个体重组操作：P′(t)←Recombination[P(t)]；

五. 个体变异操作：P′(t)←Mutation[P′(t)]；

六. 评价群体 P′(t)的适应度；

七. 个体选择、复制操作：P(t+1)←Reproduction[P(t)P′(t)]U；

八. 终止条件判断。若不满足终止条件，则：t←t+1，转到第四步，继续进 行进化操作过程。

若满足终止条件，则输出当前最优个体，算法结束。

现在有很多种进化约束优化问题算法，它们的核心是约束处理技术，也就是如何平衡约束条件和目标函数。根据约束处理技术的不同，可以将进化约束优化问题算法分为以下几类[6]：

- 惩罚函数法：将约束违反度转化为惩罚项，加到目标函数中
- 修正操作法：对不可行解进行修正，使其变为可行解
- 分离策略法：将可行解和不可行解分开处理，采用不同的选择、变异和交叉操作
- 多目标优化法：将约束违反度作为一个额外的目标，构造一个多目标优化问题

进化约束算法具有以下特点：能够处理复杂的非线性、多峰、离散等约束优化问题,对函数本身性质要求非常低，往往只要求目标函数值是可以计算的，不要求它具有连续性、可微性及其它解析性质；它是基于群体进化的算法，因此具有更强的鲁棒性和全局搜索能力，能够在全局范围内寻找最优解或近似最优解；对于非线性等式约束、离散等式约束问题的收敛性能不佳，难以找到可行解；需要大量的计算资源和时间，尤其是在高维空间中；需要设计合适的约束处理技术和参数设置，否则会影响算法效果。[10]

6. 总结

本文第一部分首先介绍优化问题及约束优化问题的相关概念，对约束处理的方法进行简单分类，第二部分介绍序列无约束优化方法，第三部分介绍序列二次规划法，第三部分介绍可行方向法,第四部分介绍进化约束优化方法，第五部分介绍进化约束优化方法，最后第六部分对本文的主要内容进行总结。

本文介绍的几类约束处理方法有各自的特点和适用范围，并没有一种通用且高效的方法。序列无约束优化方法中罚函数惩罚参数的选取比较困难；序列二次规划法在面对大规模问题耗时非常多；可行性法可以保证搜索过程中始终满足约束条件，但是可能存在收敛速度慢或陷入局部最优的问题。进化算法在求解问题时所消耗的计算资源也比较高。

约束优化问题是一类具有广泛应用价值的问题，它涉及到工程、经济、管理、生物等多个领域。随着实际问题的复杂性和规模的增加，传统的数学方法往往难以有效地求解约束优化问题，因此，近年来，基于启发式算法和机器学习的方法受到了越来越多的关注。未来约束优化算法的研究方向可能会在提高约束处理技术的效率和鲁棒性、拓展优化理论在新领域新场景下的应用等方面。

参考文献

[1] 袁亚湘,孙文瑜.最优化理论与方法[M].北京：科学出版社,1998.

[2] 李东风编.统计计算[M].高等教育出版社,2017.

[3] 马昌凤编著.最优化方法及其 Matlab 程序设计[M].科学出版社,2010.

[4] 邢文训，谢金星. 现代优化计算方法. 清华大学出版社 1999

[5] Kim, Jong-Hwan & Myung, Hyun. (1997). Evolutionary programming techniques for constrained optimization problems. Evolutionary Computation, IEEE Transactions on. 1. 129 - 140.

[6] 李智勇,黄滔,陈少淼,李仁发.约束优化进化算法综述[J].软件学报,2017,28(06):1529-1546

[7] 胡一波. 解决约束优化问题的两种新的进化算法[D].西安电子科技大学,2006.

[8] 黎小圣. 进化优化算法的约束处理方法与应用[D].天津大学,2014.

[9] 罗志军. 序列二次规划算法的研究[D]. 桂林电子科技大学,2008. DOI:10.7666/d.D562176.
[10] 王勇, 蔡自兴, 周育人, 肖赤心. 约束优化进化算法. 软件学报, 2009, 20(1): 11-29.
[11] 庞丽萍/肖现涛.最优化方法[M].大连理工大学出版社,2021.