AnyProg

C++ 数学优化包

pangpang@hi-nginx.com

2019 年 11 月 29 日

目录

Chapter 1

起源

应该有一个 C++ 版的 LINGO。
它必须免费、高效而且易用。
它必须以 C++ 的方式解决 LINGO 需要面对的问题。
它的名字是：AnyProg[1]。

[1] Any Programming。

Chapter 2

功能

AnyProg 可用于求解以下模型定义的所有问题:

$$
\begin{aligned}
\min_{x\in\mathbf{R}^n} \quad & f(x) \\
\texttt{s.t.} \quad & g_i(x) \le 0, \qquad i = 0,\cdots,m_s-1 \\
& h_j(x) = 0, \qquad j = 0,\cdots,m_t-1 \\
& x_k \in [L_k, U_k], \qquad k = 0,\cdots,n-1
\end{aligned}
\tag{2.1}
$$

其中，f 是目标函数，g_i 和 h_j 分别是不等式约束函数 (m_s 个) 和等式约束函数 (m_t 个)，x 是一个 n 维实变量，其分量 x_k 位于闭区间 $[L_k, U_k]$。模型 (2.1) 不仅适合于连续优化问题，也适合于离散优化问题，还适合于混合优化问题。对 AnyProg 来说，更重要的概念是解的全局性，而不是变量的连续性。同时，目标函数和约束函数是否线性并不重要，AnyProg 并不区别看待它们。

AnyProg 能处理的问题的类型十分广泛：**LP**[1]、**QP**[2]、**NLP**[3]、**MILP**[4]、**MIQP**[5]、**MINLP**[6]。此外，它也能进行数据的非线性最小二乘拟合，以及非线性方程组的求解。

[1]线性优化
[2]二次优化
[3]非线性优化
[4]混合整数线性优化
[5]混合整数二次优化
[6]混合整数非线性优化

Chapter 3

技巧

3.1 线性优化

线性优化的模型定义最好采用矩阵方式，类似于 MATLAB。如例 (3.1)：

$$
\begin{aligned}
\min_{x \in R^2} \quad & f(x) = 8x_0 + x_1 \\
\texttt{s.t.} \quad & g_0(x) = \quad x_0 + 2x_1 \quad \geq -14 \\
& g_1(x) = \quad -4x_0 - x_1 \quad \leq -33 \\
& g_2(x) = \quad 2x_0 + x_0 \quad \leq 20
\end{aligned} \tag{3.1}
$$

对于不等于约束，如果采用的是 $\geq$ 方式，则两边乘以 -1 使之反转。如此，原问题可用矩阵表示如下：

$$
obj = \begin{pmatrix} 8 \\ 1 \end{pmatrix}, A = \begin{pmatrix} -1 & -2 \\ -4 & -1 \\ 2 & 1 \end{pmatrix}, b = \begin{pmatrix} 14 \\ -33 \\ 20 \end{pmatrix}.
$$

在 AnyProg 中，类 `real_block` 用来表示矩阵。因此，例 (3.1) 模型定义如下：

```
#include <anyprog/anyprog.hpp>

int main(int argc, char** argv)
{
    size_t dim = 2;
    anyprog::real_block obj(dim, 1), A(3, dim), b(3, 1);
    obj << 8, 1;
    A << -1, -2,
        -4, -1,
        2, 1;
    b << 14, -33, 20;

    anyprog::optimization::range_t range = { 0, 100 };

    anyprog::optimization opt(obj, range);
    opt.set_inequation_condition(A, b);
    auto ret = opt.solve();
```

```
    anyprog::optimization::print(opt.is_ok(), ret, obj);
    return 0;
}
```

编译、运行后可知，解是 $f(6.5,7)=59$。本例不包含等式约束。若有，则可效法 A、b，炮制 Aeq 和 beq，然后使用方法 `set_equation_condition(Aeq,beq)` 添加等式约束，继而求解。本例中的 `range` 是范围，表示 $\forall i, x_i \in [0,100]$。若变量的不同分量有不同的范围限制，可用容器 `std::vector` 包装范围类 `range_t` 列表，并按顺序定义各分量范围。若有较好初值，则可替换范围参数。使用范围参数意味着初值是随机的，有可能不能求解。此时可多次运行程序。随机初值引起的解的不确定性问题。解决该问题的最佳方法是用 `search` 方法取代 `solve` 方法；前者对应于全局解，后者对应于局部解。使用该方法时，对于局部解较少的模型，务必减少前两个参数[1]的数值。

3.2 二次优化

二次优化的模型定义如下：

$$
\begin{aligned}
\min_{x\in\mathbf{R}^n} \quad & f(x)=\frac{1}{2}x^T H x + c^T x \\
\text{s.t.} \quad & A\cdot x \le b \\
& Aeq\cdot x = beq \\
& x_k \in [L_k, U_k], \qquad\qquad k=0,\cdots,n-1
\end{aligned}
\tag{3.2}
$$

该模型用矩阵形式表示。在求解该类模型时，最好的方式就是顺其自然。比如下例：

$$
\begin{aligned}
\min_{x\in\mathbf{R}^2} \quad & f(x)=\frac{1}{2}x_0^2 + x_1^2 - x_0x_1 - 2x_0 - 6x_1 \\
\text{s.t.} \quad & g_0(x) = \quad x_0 + x_1 \qquad\qquad \le 2 \\
& g_1(x) = \quad -x_0 + 2x_1 \qquad \le 2 \\
& g_2(x) = \quad 2x_0 + x_1 \qquad\quad \le 3
\end{aligned}
\tag{3.3}
$$

其矩阵表示如下：

$$
H=\begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}, A=\begin{pmatrix} 1 & 1 \\ -1 & 2 \\ 2 & 1 \end{pmatrix}, b=\begin{pmatrix} 2 \\ 2 \\ 3 \end{pmatrix}, c=\begin{pmatrix} -2 \\ -6 \end{pmatrix}
$$

因此，它的模型定义为：

```
#include <anyprog/anyprog.hpp>

int main(int argc, char** argv)
{
    size_t dim = 2;
    anyprog::real_block H(dim, dim), c(dim, 1);
    H << 1, -1,
        -1, 2;
```

[1] `search` 方法的前两个参数的默认值分别是 100 和 30，这是专门应对变态测试模型的基本配置。对于局部解较少的普通模型，两个数值都可以大幅度降低；一般设置 10,3 即可。

```
    c << -2, -6;

    anyprog::optimization::function_t obj = [&](const anyprog::real_block& x) {
        anyprog::real_block ret = 0.5 * (x.transpose() * H * x) + c.transpose() * x;
        return ret(0, 0);
    };

    anyprog::real_block A(3, dim), b(3, 1);
    A << 1, 1,
        -1, 2,
        2, 1;
    b << 2, 2, 3;

    anyprog::optimization::range_t range = { 0, 100 };

    anyprog::optimization opt(obj, range, dim);
    opt.set_inequation_condition(A, b);
    auto ret = opt.solve();
    anyprog::optimization::print(opt.is_ok(), ret, obj);
    return 0;
}
```

编译、运行，其解为 $f(0.666667, 1.33333) = -8.22222$

3.3 非线性优化

3.4 混合整数优化

3.5 TSP 问题

3.6 Assignment 问题

3.7 数据拟合问题

3.8 方程组求解

Chapter 4

展望