MATLAB in Economics and Management

本科金融系 17020083 许楚楚

2020 年 6 月 28 日

摘要

本篇解决了老师布置的 5 道关于 Matlab 的题目，分别使用了栈，分枝定界法，最速下降法，Scaling 方法和 FIBONACCI search 的方法，每一道题主要围绕着解题思路、代码实操、结果分析三个方面进行阐述。注：本报告用 LaTeX 格式编写，源代码可以在压缩包中获得。

目录

1 Problem Set 1: 栈的使用

1.1 题目

八皇后问题是一个以国际象棋为背景的问题：如何能够在 8x8 的国际象棋棋盘上放置八个皇后，使得任何一个皇后都无法直接吃掉其他皇后。为了达到此目的，任两个皇后都不能处于同一条横行、纵行或斜线上。

要求：使用数据结构中的栈实现，最终结果采用矩阵形式输出。

1.2 解题思路

本题在求解过程中没有采用任何技巧，使用的是暴力枚举 (brute force)，利用递归的方法 (recursion) 实现深度优先的逐次枚举。为了从一定程度上降低复杂度 (complexity)，在遇到不合适的解的时候直接终止 (break) 后续棋子的摆放程序，跳至下一次尝试。

1.3 代码实操

首先定义了基本的变量，其中包含 3 个全局变量 (global variable) 依次为：`condense_board`、`solution`和`n`。`condense_board`是 1x8 的行向量，用于表示一整个棋盘，例如`condense_board`中第 5 个元素值为 8 则表示第 5 行的皇后放在第 8 列。`solution`是为了记录符合条件的八皇后问题解的变量，n 表示的是棋盘的大小 (在这里是 8x8)。主要的程序操作还是由`function search_at_depth(y1)`实现，其中输入参数 y1 是需要决定皇后摆放位置的行数。该函数实现的主要逻辑判断有：

1. 被检查行是否是第 1 至 8 行中的一行，如果是则为该行选择皇后位置–逐次在第`y1`行的第 1 至 8 列尝试皇后的位置，如果皇后的列数位置与之

前的皇后一样 (`x1 == x2`)，尝试失败；如果皇后的斜对角存在其他皇后 (`(y1+x1) == (y2+x2)` 和`(y1-x1) == (y2-x2)`)，尝试也失败。上述采割条件都通过后，才算合格，进入下一行的测试。这里通过递归的方法实现`search_at_depth(y1+1)`。

2. 如果输入行数大于 8，说明该方案已经通过前面 8 轮测试，为可行解，则输出该解；

3. 如果上述条件皆不符合，i.e. 输入行数非大于 1 的整数，说明输入有问题，报错。

1.4 结果分析

最终产生了 92 种可能的摆放情况，在此列举第 34 种[1]：

```
The 34th solution
    0     0     0     1     0     0     0     0
    0     1     0     0     0     0     0     0
    0     0     0     0     0     0     1     0
    0     0     0     0     1     0     0     0
    1     0     0     0     0     0     0     0
    0     0     0     0     0     0     0     1
    0     0     0     0     0     1     0     0
    0     0     1     0     0     0     0     0
```

2 Problem Set 2: 最大化目标函数的分枝定界法

2.1 题目

1. 初始求解整数规划的松弛问题：求解其松弛线性规划，若为整数解，极为整数规划的最优解。否则，初始下界设为 $-\infty$。

2. 建立分枝树：在任何一个（子）问题中，从不满足整数要求的变量中选出一个进行处理，通过加入一对互斥的约束将一个（子）问题分支为两个受到进一步约束的子问题，缩小搜索的区域。由此，子问题若不满足整数要求，则进一步向下分枝，形成一个分枝树。

3. 定界与剪枝：通过不断分枝和求各个子问题，分枝定界法将不断修正已经得到的最优整数解确定的下界。其中，求解子问题可能出现以下结果：

[1] The writer's lucky number

- 无可行解，无须继续分枝
- 得到一个整数解，无需继续分枝，更新下界
- 得到一个非整数解，视目标函数值的情况决定剪枝还是继续分枝

4. 按照上述步骤迭代，每当下界被修改以后，应当检查所有还没有求解过的子问题并剪去那些目标函数值小于新的下界的子问题。

要求：使用编写的程序求解一个需要迭代两次以上的整数规划问题。

Example 2.16 用分枝定界法求解整数规划问题

$$\begin{aligned} & max \qquad x_1 + x_2 \\ \text{s.t.} \quad & \begin{cases} 4x_1 - 2x_2 \geq 1, \\ 4x_1 + x_2 \leq 11, \\ 2x_2 \geq 1, \\ x_1, x_2 \geq 0 \text{且取整数值.} \end{cases} \end{aligned}$$

2.2 程序设计思路

本题旨在利用解决线性规划的方法解决整数规划，运用的是分枝定界法(branch-and-bound method)。首先需要标准化上述整数规划问题为 Matlab 的标准型，除去整数约束，得到松弛线性规划，该松弛线性规划问题标准化结果如下：

$$\begin{aligned} & min \qquad -x_1 - x_2 \\ \text{s.t.} \quad & \begin{cases} -4x_1 + 2x_2 \leq -1, \\ 4x_1 + x_2 \leq 11, \\ 0x_1 - 2x_2 \leq -1, \\ x_1, x_2 \geq 0. \end{cases} \end{aligned}$$

在 Matlab 中初始化问题代码如下，得到的解并非是整数解，因此我们需要建立分枝树，通过分枝定界法排查最优整数解。

```
c = [-1;-1];
A = [-4 2; 4 1; 0 -2];
b = [-1; 11; -1;];
lb = [0; 0];
up = [Inf; Inf];
```

接下来就是代码实现的部分，主要的思路是，先求除去整数条件后的解松弛线性规划问题，并对解进行判定。如果无解，说明条件更为严格的整数规划问题也无解；如果返回整数解，即为整数规划问题的解；如果返回非整数解，进入分枝定界法的运行。分枝定界法的运行流程如下：输入主枝 (mainNode) 的参数求解线性规划问题，在进入分枝环节前需要做 3 次判定：1. 判定是否有解，无解则剪枝，`return 0;`；2. 目标函数值是否低于当前上界 (lowerBound)，如果不低于当前上界，则没有继续运行的意义，剪枝；3. 解是否为整数解，如果是，更新当前上界 (upperBound)，存储决策变量和目标函数值至 candidnate solution 中 (`solution`)，剪枝；如果解非整数解，则分枝。分枝部分需要做的就是通过改变上下界的取值收紧约束条件，利用 recursion 的原理在此调用函数 branchbound 得到 2 个分枝。

注：在传统分枝定界求解时，是对下界进行设定，而计算机的思维与人不同，需要将最大化问题转换为最小化问题，随之设定的应该是上界而非下界。

2.3 结果分析

通过运行主函数文件`main.m`中的 test case 2 可以得到如下解。

```
================================================

The optimal integer solution is:     2     3

The obj value is:    -5

================================================
```

该结果与`doubleConfirm.m` 文件中通过 Matlab 内置命令 intlinprog 得到的结果一致。如果需进一步确认可以使用`main.m`和`doubleConfirm`给出的 test case 1 进行测试或者自定函数。

3 Problem Set 3: 最速下降法

3.1 题目

考虑下面的问题

$$\min \qquad 5x^2 + 5y^2 - xy - 11x + 11y + 11 \tag{1}$$

找出满足一阶必要条件的解，证明该解为全局最优解。若用最速下降法求解该问题，收敛率是多少？选定 $x = y = 0$ 为初始点，最速下降法 (最多) 需要多少次迭代才能将函数值降至 10^{-11}?

3.2 找出满足一阶必要条件的解

第一小问：根据一阶必要条件的的推论 [2]，可知满足条件的解出现在梯度为 0 的时候，则利用该条件在 Matlab 中进行实操。在 Matlab 中需要先输入定义符号目标函数，利用 gradient 命令得到目标函数关于 x 和 y 的偏导数，再利用 solve 命令得到当偏导数为 0 时的解。找到的满足一阶必要条件的解为 $x = 1, y = -1$，此时目标函数的值为 0。

3.3 证明该解为全局最优解

第二小问：这里我们首先需要利用 Theorem 2.3 [3] “若目标函数的海赛矩阵是正定的，该函数就是凸函数” 来证明目标函数为定义在凸集上的凸函数。再根据 Theorem 2.4 [4] “凸函数的局部最优解就是全局最优解” 证明在第一小问中得到的解是全局最优解。通过利用 det 命令求顺序主子式，求得得顺序主子式解大于 0，说明目标函数为凸函数，那么第一问得到的局部最优解即全局最优解。

3.4 求解最速下降法的收敛率

首先利用 eig 命令求出海赛矩阵 $\mathbf{H}$ 的特征值，再用 max 和 min 命令求出特征值中最大和最小的特征值，得到 r(condition number) 等于最大特征值除以最小特征值，最后收敛率 (covergence ratio) 也由相应公式得到，为 0.0100。

3.5 最多迭代次数

证明. 根据 Theorem 2.8[5]，收敛的最差情况是：

$$E(x_{k+1}) = \left(\frac{A-a}{A+a}\right)^2 E(x_k)$$

再根据 $E(x_k)$ 的定义[6]、初始条件 $x_0 = (0, 0)$ 和收敛率 $= 0.01$，得到：

$$E(x_0) = 11$$

[2]Corollary 2.1: Let Ω be a subset of $\mathbb{R}^n$ and let $f \in C^1$ be a function on Ω. If x^* is a relative minimum point of f over Ω and if x^* is an interior point of Ω, then $\bigtriangledown f(x^*)d = 0$.

[3]Theorem 2.3: Let $f \in C^2$. Then f is convex over a convex set Ω containing an interior point if and only if the Hessian matrix F of f is positive semidefinite.

[4]Theorem 2.4: Let f be a convex function defined on the convex set Ω. Then the set Γ where f achieves its minimum is convex, and any relative minimum of f is a global minimum

[5]Theorem 2.8: For any $x_0 \in \mathbb{R}^n$, the steepest descent method converges to the unique minimum point x^* of f. Further more, there holds at every step k, $E(x_{k+1}) \leq (\frac{A-a}{A+a})^2 E(x_k)$

[6]$E(x) = \frac{1}{2}(x - x^*)^t H(x - x^*)$

$$E(x_1) = 0.01E(x_0) = 0.11$$

$$E(x_2) = 0.01E(x_1) = 0.0011$$

$$\cdots \qquad \cdots$$

设经过 n 次的迭代后 $E(x_k) \leq 10^{-11}$

$$11 * 0.01^n = 10^{-11}$$

$$n = log_{0.01}(10^{-11})$$

$$n = 6.0207$$

□

所以得出结论：最多只需要经过 7 次的迭代就可以将函数值降至 10^{-11}。虽然说理论上，最多迭代的次数不会超过 7 次，但是通过运行文件`gradient.m`中 question 4 的部分，得到 count = 1，即在实际操作中仅需进行一次迭代便可得出最优解。

4 Problem Set 4: Scaling 方法

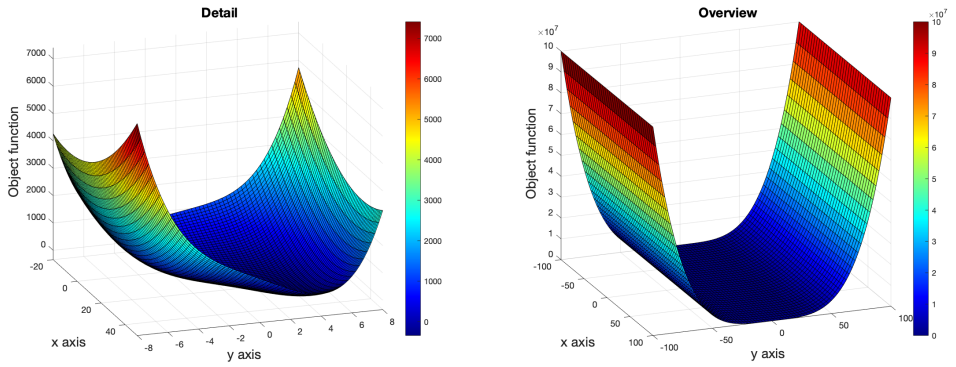
4.1 题目

已知目标函数

$$f(x, y) = x^2 + 5xy + y^4 - 25x - 8y \tag{2}$$

用最速下降法求最小值。现采用 scaling 方法，其原理是令 Hessian 矩阵主对角线上元素的值近似相等。令 $z = 7y$ 对上式做代换，重新用最速下降法求最小值，并比较两次迭代次数的结果。

4.2 可视化



该函数底部看似是对称平滑的曲面 (右图)，但实际上底部是凹凸不行、不对称的 (左图)。

4.3 程序设计思路

首先利用fminsearch函数计算出目标函数的最小值，得出当 $x = 20, y = 3$ 的时候，目标函数值最小，为-340。接下来是进行最速下降的操作：定义符号变量 x, y, z, a[7]，利用gradient命令计算出梯度表达式，开始迭代操作。这里利用while循环设定迭代结果与实际最小值之间的误差必须小于 0.000001 才算迭代得到最优解。在while循环内主要做 2 件事情：一、确定方向；二、确定最优步长。方向是梯度 $\bigtriangledown f(x_k)$ 取反方向，最优步长 α 是通过对 $f(x_k - \alpha \bigtriangledown f(x_k)^t)$ 求导、设定求导后函数为 0 得到，在 Matlab 中的具体实现是：`double(solve(diff(fa,a),'Real',true))`，因为可能会出现复数解，因此需要在solve中设定参数Real确保得到整数解，double是为了将分数解转换为小数存储 (如果以分式形式存储，运行会非常慢，这里是对于精确度和运行速度做了一个 tradeoff)。以此迭代循环，直至和最优值之间的误差小于 0.000001 结束循环。

上述是对原函数的操作，对于缩放 (Scaling) 后的目标函数，操作方法实际上是完全一致的，只是需要在个别函数名称上进行改变。

4.4 结果分析

通过运行 beforeScaling.m 文件，得到结果如下，第一个数字是 x 的取值、第二个数字是 y 的取值、第三个数字是最终目标函数的取值。为了得到误差小于 0.000001 的解，一共进行了 79 次尝试、78 次迭代。

```
The 79 time try
Solution:
   19.9990
    2.9999


Object value: -343.0000
Condition number:  61.3327
```

接下来让我们看一下在 Scaling 之后的程序运行效率。通过运行 afterScaling.m 文件，得到结果如下，不同于 Scaling 之前，第二个数字是 z 的取值，大约为 beforeScaling.m 运行得到 y 值的 3 倍，符合预期。由此可见，为了得到误差小于 0.000001 的解，仅仅只进行了 12 次尝试、11 次迭代，运行效率大大提高。

```
The 12 time try
Solution:
```

[7] a 为最优步长

```
   19.9998
   20.9995

Object value: -343.0000
Condition number:   2.0453
```

差距的原因是因为 scaling 前后的海赛矩阵不同，导致了对应特征矩阵的最大和最小的特征值不一样，得到的 r(condition number) 不一样，收敛率也不同。通过在源代码中加入计算 r 的代码，让计算机输出了 scaling 前后的 r，scaling 前的 r 大约在 61 左右，而 scaling 后的 r 降至到了 2 左右，根据 Theorem 2.8 [8]，$(\frac{A-a}{A+a})^2 = (\frac{r-1}{r+1})^2$, r 越小说明收敛的速度越快，由此可以说明缩放后收敛速度更快的原因。

5 Problem Set 5: FIBONACCI 搜索

5.1 题目

对于 FIBONACCI 搜索方法，令搜索区间的初始长度为 $d_1 = c_2 - c_1$

1. 证明 $d_k = \frac{F_{N-k+1}}{F_N} d_1$
2. 插入两个试验点后计算函数值，若第一次迭代的结果是保留左端点 c_1，求下一个插入点坐标的两种表达式，并验证它们的等价性。

5.2 d_k 与 d_1 关系证明

证明. 假设 d_i 是进行 i 次操作后不确定区间的宽度，N 是总的操作次数

根据前一项与后一项之间的比为 fibonacci 数列之比，得：

$$d_2 = \left(\frac{F_{N-1}}{F_N}\right) d_1$$

$$d_3 = \left(\frac{F_{N-2}}{F_{N-1}}\right) d_2$$

$$\cdots$$

$$d_{k-1} = \left(\frac{F_{N-k+2}}{F_{N-k+3}}\right) d_{k-2}$$

$$d_k = \left(\frac{F_{N-k+1}}{F_{N-k+2}}\right) d_{k-1}$$

[8] Theorem 2.8: For any $x_0 \in \mathbb{R}^n$, the steepest descent method converges to the unique minimum point x^* of f. Further more, there holds at every step k, $E(x_{k+1}) \leq (\frac{A-a}{A+a})^2 E(x_k)$

根据迭代法消元得：

$$d_k = \left(\frac{F_{N-k+1}}{\cancel{F_{N-k+2}}}\right) \cdot \left(\frac{\cancel{F_{N-k+2}}}{\cancel{F_{N-k+3}}}\right) \cdot \cdots\cdots \left(\frac{\cancel{F_{N-2}}}{\cancel{F_{N-1}}}\right) \cdot \left(\frac{\cancel{F_{N-1}}}{F_N}\right) d_1$$

$$d_k = \frac{F_{N-k+1}}{F_N} d_1$$

□

5.3 插入表达式等价性证明

方法一是通过更新后上下界 b_k 和 a_k 与中间插入点 t_1 和 t_2 的关系而得。在当前题目给定条件下，第一次迭代结果是保留左点点 c_1，那么新的下界是原来的 a_0、新的上界是原来的 t_2、新的的 t_2 是原来的 t_1，那么我们现在需要求解的便是新的 t_1 值。假定新的 t_1 为 t_1'，根据 t_1' 距离上界的距离为区间长 $d2$ 乘以当前 fibnacci 序列比例，得到表达式 1:

$$t_1' = b_1 - \left(\frac{F_{k-2}}{F_{k-1}}\right) d_2 \tag{3}$$

方法二是根据画图得到了表达式 2:

$$t_1' = a_0 + d_2 - d_3 \tag{4}$$

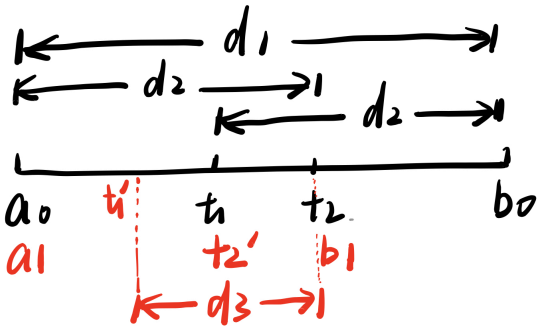


图 1: 表达式 2 图例

证明. 通过等式变换关系，表达式 1 可以通过下述转变得到：

$$t_1' = b_1 - \left(\frac{F_{k-2}}{F_{k-1}}\right) d_2$$

$$t_1' = b_1 - d_3$$

通过等式变换关系，表达式 2 可以通过下述转变得到：

$$t_1' = a_0 + d_2 - d_3$$
$$t_1' = t_2 - d_3$$
$$t_1' = b_1 - d_3$$

两者等价性得证。 □