

Guide of Matlab

内容概要： 数学建模算法

创建时间： 2022/4/7 13:41

更新时间： 2022/4/17 15:27

作者： TwinkelStar

蛮力法

Brute Force

1、操作系统相关环境

1) 硬件环境：

➤ 电脑

2) 软件环境：

➤ Matlab2018a(程序设计软件)

3) 操作系统(2 选 1)：

➤ Windows10

2、蛮力法

蛮力法是一种简单直接地解决问题的方法(暴力求解)，常常直接基于问题的描述和所涉及的概念定义。注意，这里的“力”是指计算机的计算的运算能力。一般来说，蛮力策略常常是最容易应用的办法。虽然巧妙和高效的算法很少来自于蛮力法，但我们不应该忽略它作为一种重要的算法设计策略的地位。

第一，蛮力法可以解决广阔领域的各种问题。实际上，它可能是

唯一一种几乎什么问题都能解决的一般性方法。

第二，对于一些重要的问题（如排序、查找、字符串匹配等）来说，蛮力法可以产生一些合理的算法。

第三，如果要解决的问题实例不多，而且蛮力法可以用一种能够接受的速度对实例求解，那么设计一个更高效算法所花费的代价很可能是值得的。

第四，即使效率通常很低，仍可使用蛮力法解决一些小规模的问题实例。第五，蛮力法可以作为研究或教学目的的服务，如可以以之为准绳，来衡量同样问题的更高效算法（如计算最坏时间复杂度）。

1) 设计思想

蛮力法是指采用遍历（扫描）技术，即采用一定的策略将待求解问题的所有元素依次处理一次，从而找出问题的解。依次处理所有元素是蛮力法的关键，为了避免陷入重复试探，应保证处理过的元素不再被处理。

蛮力法本质是先有策略地穷举，然后验证。简单来说，就是列举问题所有可能的解，然后去看看是否满足题目要求，是一种逆向解题方式。

3、算法分析

1) 算法优点

- 逻辑简单清晰，根据逻辑编写的程序简单清晰。
- 对于一些需要高正确性的重要问题，它产生的算法一般而言复杂度虽高但合理的。
- 解决问题的领域广。适用于一些规模较小，时间复杂度要求不高的问题。
- 可以作为其他高校算法的衡量标准，即作为被比较的例子。

2) 算法缺点

算法的主要缺点就算缺少人的思考，算法设计简单但往往缺乏效率，在所有的解空间中通过搜索解。

4、例题

1) 在象棋算式里，不同的棋子代表不同的数，有以下算式，设计一个算法求这些棋子各代表哪些数字。

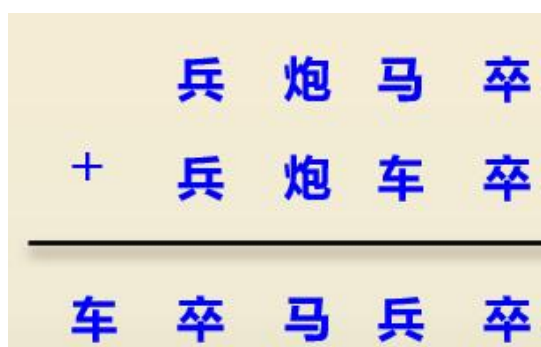


图 1.案例 1

fig1. Example1 one

直接采用穷举法的思想计算，对于五个棋子的取值分别进行枚举，然后判断取值是否成立，满足苏式要求，运行代码 `example1.m` 代码如图 2 所示：

```
for a=0:9
    for b=0:9
        for c=0:9
            for d=0:9
                for e=0:9
                    if a~=b && a~=c && a~=d && a~=e && b~=c && b~=d && b~=e && c~=d && c~=e && d~=e
                        m1 = a * 1000 + b * 100 + c * 10 + d;
                        m2 = a * 1000 + b * 100 + e * 10 + d;
                        s = e * 10000 + d * 1000 + c * 100 + a * 10 + d;
                        if s==m1+m2
                            disp(a)
                            disp(b)
                            disp(c)
                            disp(d)
                            disp(e)
                        end
                    end
                end
            end
        end
    end
end
```

图 2.案例 1 代码

fig2. Example one code

最后的运行结果为：

兵： 5； 炮： 2； 马： 4； 卒： 0； 车： 1。

验算之后等式成立。

2) 求解函数 $(x_1-5)^2+(x_2-7)^2$ 的最小值, x_1 、 x_2 的取值范围为 $[0,10]$ 。

从题目中分析不难得知, 这是函数是一条抛物线, 当 x_1 取 5, x_2 取 7 时, 函数取得最小值。使用蛮力法的思想, 生成两组样本数据, 范围在 $[0,10]$ 之间, 然后喂入函数最后寻找最小值对应的取值即可, 编写目标函数程序 f.m, 运行代码 example2.m, 代码如图 3 所示:

```
x1 = linspace(0,10,100);  
x2 = linspace(0,10,100);  
count=1  
for i=1:100  
    for j=1:100  
        y = f(x1(i),x2(j));  
        distr(count) = y;  
        count = count + 1;  
    end  
end  
plot(distr(1:100:10000), "*")  
title("目标函数曲线")
```

图 3.案例 2 代码

fig3. Example two code

最后的运行结果为:

最小值为: 0.0034690337720640393;

x_1 的取值为: 4.94949494949495;

x_2 的取值为: 6.96969696969697。

当 x_1 近似取 5, x_2 近似取 7 时, 函数有最极值, 最小值为 0

绘制函数图形, 如图 4 所示:

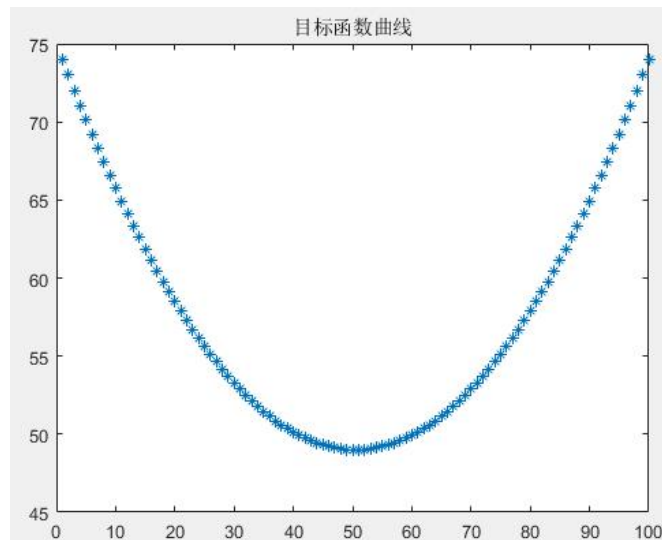


图 4. 案例 2 函数图形

fig4. Example two function img

3) 考虑一个线性规划的小规模案例，使得目标函数 $3x + 5y$ 取得极大值，尝试用蛮力法求解，约束条件为：

$$x + y \leq 4 \quad (1)$$

$$x + 3y \leq 6 \quad (2)$$

$$x \geq 0 \text{ 且 } y \geq 0 \quad (3)$$

运行程序 example3.m，代码如图 5 所示：

```
count=1
for i = 1:1000
    for j = 1:1000
        x=i/100;
        y=j/100;
        if x+y <= 4 && x + 3*y <=6
            dict{count}=3*x+5*y;
            count = count + 1;
        end
    end
end
plot(dict{1:100:count}, "-r*");
title("目标函数曲线")
```

图 5. 案例 3 代码

fig5. Example three code

绘制图形，如图 6 所示

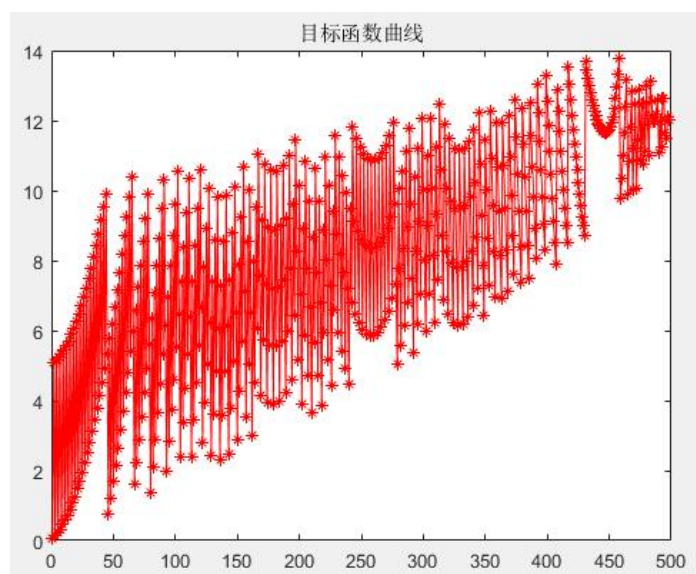


图 6.案例 3 函数图形

fig6. Example three function img

从图像以及结果得知，函数在第 45347 次迭代时产生的最大值，最大值为：14.0，x 的值为：3.0，y 的值为：1.0。

但蛮力法的这种暴力枚举的方法在时间选择上不可取，尽管理论上来说蛮力法可以穷举出所有的可能性，但对于一些规划问题和图论问题，需要在短时间内完成方案的选择，蛮力法的表现就不尽人意，并且，从图 6 中可以看出，蛮力法求解解空间并不问题，只是在解空间里面到处“试错”。

在已有的计算资源的情况下，对于一些基础的问题可以用蛮力法求解，但考虑到时间复杂度和代码的优化上来说，尽量设计出更好的程序，便于优化方案。