Advances in Applied Mathematics 应用数学进展, 2020, 9(8), 1146-1158
Published Online August 2020 in Hans. http://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2020.98134

Hans汉斯

The Improvement and Empirical Study of Nonlinear Programming Algorithm Based on Newton Method and Quasi-Newton Method

Weiwei Fu[1,2], Liwei Zhang[1*], Yu Dong[1]

[1]School of Mathematical Sciences, Dalian University of Technology, Dalian Liaoning
[2]College of Science, Liaoning Technical University, Fuxin Liaoning
Email: [*]lwzhang@dlut.edu.cn, fuweiwei0601@126.com

Received: Jul. 15th, 2020; accepted: Jul. 28th, 2020; published: Aug. 4th, 2020

Abstract

Nonlinear programming has always been a hot topic in the research of optimization theory. In this paper, three commonly used methods for solving unconstrained nonlinear programming (gradient method, Newton method and quasi-Newton method) are compared and analyzed. Two new algorithms are given by improving the original method, and the convergence of the improved algorithm is illustrated. On this basis, this paper makes an empirical study on the location of distribution center based on competition.

Keywords

Gradient, Convergence, Location Problem

基于牛顿法及拟牛顿法的非线性规划算法改进及实证研究

付巍巍[1,2]，张立卫[1*]，董　玉[1]

[1]大连理工大学数学科学学院，辽宁 大连
[2]辽宁工程技术大学理学院，辽宁 阜新
Email: [*]lwzhang@dlut.edu.cn, fuweiwei0601@126.com

[*]通讯作者。

文章引用: 付巍巍, 张立卫, 董玉. 基于牛顿法及拟牛顿法的非线性规划算法改进及实证研究[J]. 应用数学进展, 2020, 9(8): 1146-1158. DOI: 10.12677/aam.2020.98134

收稿日期：2020年7月15日；录用日期：2020年7月28日；发布日期：2020年8月4日

摘　要

非线性规划一直是优化理论研究的热点问题。本文将求解无约束非线性规划的三种常用方法(梯度法，牛顿法和拟牛顿法)进行对比分析，改进原有方法从而给出两种新算法，并说明改进算法的收敛性。在此基础上对基于竞争的配送中心选址问题进行了实证研究。

关键词

梯度，收敛，选址问题

1. 引言

非线性规划是最优化问题中的重要一类。一般的最优化问题可以表达成一个二元组 (D,f)，D 为任一集合，是问题的可行区域；f 为目标函数，是由集合 D 到一维欧式空间 R^1 的一个映射。问题是求出 D 中的一个元素 X^*，使对 $\forall X^* \in D$，有 $f(X^*) \le f(X)$，此时 X^* 称为全局最优解。

对于无约束非线性规划问题：

$$\min f(X)$$

其中 $X=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}$ 是 n 维空间中的向量点。

2. 梯度法

梯度法是 1847 年法国著名数学家 Cauchy 最早提出[1]。梯度法的基本原理：若目标函数 $f(X)$ 有一阶连续偏导数，在 X^* 取得极小点，通过在迭代点 $X^{(k)}$ 负梯度方向上的一维搜索，来确定使得 $f\left(X^{(k+1)}\right)=f\left(X^{(k)}+t_k d^{(k)}\right)$ 最小的 t_k (即最佳步长)，得到逐渐趋近于极小点 X^* 的序列 $\left\{X^{(k)}\right\}$，这种梯度法就是所谓的最速下降法。

梯度法方法简单，每迭代一次的工作量较小，所需要的存储量也小，即使从一个不好的初始点出发，也能保证算法的收敛性。而缺点是在极小点附近收敛的很慢。由于梯度是函数的局部性质，从局部看，在一点附近下降或上升的快，但从总体看可能走许多弯路。事实上，若 $X^{(k+1)}$ 是沿方向 $d^{(k)}=-\nabla f\left(X^{(k)}\right)$ 用一维搜索求得的，即 t_k 为 $\varphi(t)=f\left(X^{(k)}+td^{(k)}\right)$ 的极小点，那么 $\varphi'(t_k)=0$，即

$$\nabla f\left(X^{(k+1)}\right)^{\mathrm{T}}\cdot\nabla f\left(X^{(k)}\right)=0$$

说明相邻的两个迭代方向正交，迭代路线呈锯齿状，尤其在最优解附近，锯齿现象尤为严重，从而影响迭代速度，如图 1。另外，梯度法产生的序列是线性收敛的，而且收敛性质与最优解处的 Hessian 阵的特征值关系很大，且关于小的扰动是不稳定的，任何小的扰动都可能破坏算法的收敛性。

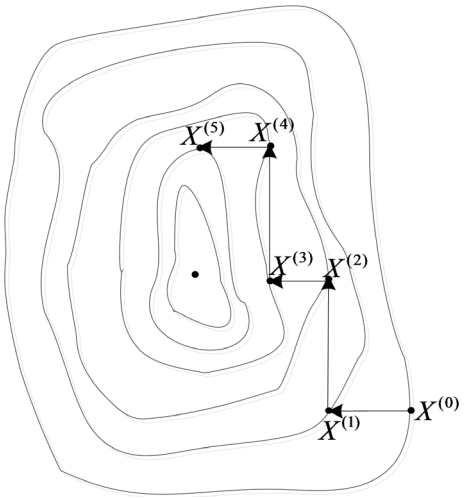

Figure 1. The iteration roadmap of Gradient method
图 1. 梯度法迭代路线图

例 1. 用梯度法求

$$\min f(X) = -2x_1x_2 - 2x_2 + x_1^2 + 2x_2^2$$

初始点 $X^{(0)} = (0,0)^{\mathrm{T}}$， $\varepsilon = 0.1$。

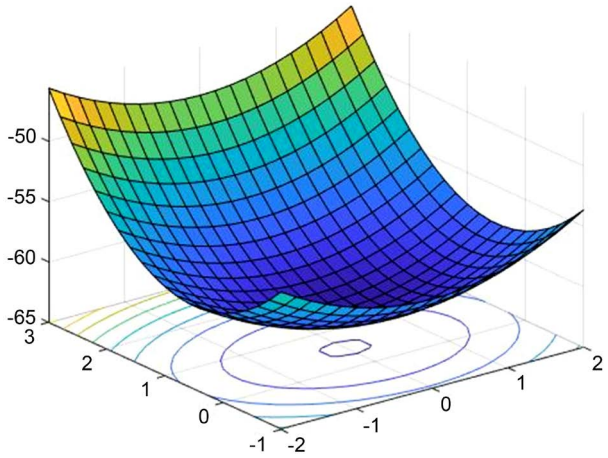

Figure 2. Function image of the objective function in example 1
图 2. 例 1 中目标函数的函数图像

解：图 2 为目标函数的函数图像，从图上可以看出二次函数的精确最优解 $X^* = (1,1)^{\mathrm{T}}$，目标函数的梯度为 $\nabla f(X) = -(2x_2 - 2x_1, 2x_1 + 2 - 4x_2)^{\mathrm{T}}$，所以 $\nabla f\left(X^{(0)}\right) = (0,-2)^{\mathrm{T}}$， $\left\|\nabla f\left(X^{(0)}\right)\right\| = 2 > 0.1$。

第一次迭代：

从 $X^{(0)} = (0,0)^{\mathrm{T}}$ 出发，沿方向 $d^{(0)} = -\nabla f\left(X^{(0)}\right) = \begin{pmatrix} 0 \\ 2 \end{pmatrix}$ 进行一维搜索， $t_0 = \frac{1}{4}$，在直线上的极大点

$$X^{(1)} = X^{(0)} + t_0 d^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} + \frac{1}{4}\begin{pmatrix} 0 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ \frac{1}{2} \end{pmatrix}$$

如此重复进行迭代，会如图所示，得到点列$\left(\frac{1}{2},\frac{3}{4}\right),\left(\frac{3}{4},\frac{3}{4}\right),\left(\frac{3}{4},\frac{7}{8}\right),\cdots$逐渐趋近于问题的最优解$X^*=\begin{pmatrix}1\\1\end{pmatrix}$，可以验证$\nabla f\left(X^*\right)=0$。具体迭代情况如图 3 及表 1 所示。

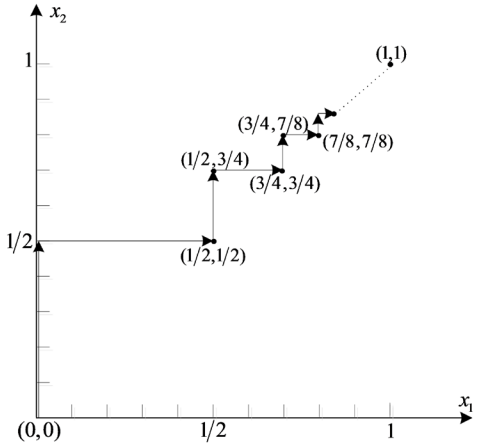

Figure 3. The iteration point and iteration roadmap of example 1 were solved by gradient method
图 3. 梯度法求解例 1 时迭代点及迭代路线图

Table 1. Initial value points and iteration table of example 1 were solved by gradient method
表 1. 梯度法求解例 1 的初值点和迭代次数表

初值	最优解	最优值	迭代次数	精度
(0, 0)	(0.9469, 0.9718)	−0.9986	39	$\varepsilon_1=0.1$
(100, 100)	(0.9201, 0.9506)	−0.9966	305	$\varepsilon_1=0.1$

3. 牛顿法

牛顿法最初由艾萨克·牛顿于 1736 年在 Method of Fluxions 中公开提出[2]。若非线性目标函数$f(X)$具有二阶连续偏导数，又设$X^{(k)}$为其极值点的某一近似，在这一点做$f(X)$的二阶泰勒展开，即：

$$f(X)\approx g(X)=f\left(X^{(k)}\right)+\nabla f\left(X^{(k)}\right)^{\mathrm{T}}\left(X-X^{(k)}\right)+\frac{1}{2}\left(X-X^{(k)}\right)^{\mathrm{T}}\nabla^2 f\left(X^{(k)}\right)\left(X-X^{(k)}\right)$$

其中$\nabla^2 f\left(X^{(k)}\right)$为$f(X)$在$X^{(k)}$处的 Hessian 矩阵。

为求$g(X)$的平稳点，令$\nabla g(X)=0$，即

$$\nabla f\left(X^{(k)}\right)+\nabla^2 f\left(X^{(k)}\right)\left(X-X^{(k)}\right)=0$$

设矩阵$\nabla^2 f\left(X^{(k)}\right)$可逆，则得到牛顿法的算法迭代公式：

$$X^{(k+1)}=X^{(k)}-\left[\nabla^2 f\left(X^{(k)}\right)\right]^{-1}\nabla f\left(X^{(k)}\right)$$

如果$f(X)$是二次函数，则其 Hessian 矩阵$\nabla^2 f(X)$为常数阵。在这种情况下，从任意一点出发，只要一步即可求出$f(X)$的极值点(假设 Hessian 矩阵正定)。

如果$f(X)$不是二次函数，二阶泰勒展式仅是一个近似表达式。此时，求得的极值点只是$f(X)$的近似极值点。在这种情况下，选取搜索方向：

$$d^{(k)}=-\left[\nabla^2 f\left(X^{(k)}\right)\right]^{-1}\nabla f\left(X^{(k)}\right)$$

$$X^{(k+1)}=X^{(k)}+d^{(k)}$$

按照这种方式求函数 $f(X)$ 极小点的方法称为**牛顿法**，称方向 $d^{(k)}=-\left[\nabla^2 f\left(X^{(k)}\right)\right]^{-1}\nabla f\left(X^{(k)}\right)$ 为牛顿方向。牛顿法收敛的速度很快，当 $f(X)$ 的二阶导数及 Hessian 矩阵的逆矩阵便于计算时，这一方法非常有效。

例 2 用牛顿法求解**例 1** 问题

$$\min f(X)=-2x_1x_2-2x_2+x_1^2+2x_2^2$$

初始点 $X^{(0)}=(0,0)^{\mathrm{T}}$，$\varepsilon=0.1$。

解：目标函数 $f(X)$ 的梯度为 $\nabla f(X)=\left(-2x_2+2x_1,-2x_1-2+4x_2\right)^{\mathrm{T}}$。

Hessian 阵 $H(X)=\nabla^2 f(X)=\begin{pmatrix}2&-2\\-2&4\end{pmatrix}$ 为正定矩阵(即目标函数为二次凸函数)。

初始点 $X^{(0)}=(0,0)^{\mathrm{T}}$，于是有 $\nabla f\left(X^{(0)}\right)=(0,-2)^{\mathrm{T}}$，此时 $\left\|\nabla f\left(X^{(0)}\right)\right\|=2>0.1$，开始第一次迭代

$$H\left(X^{(0)}\right)=\nabla^2 f\left(X^{(0)}\right)=\begin{pmatrix}2&-2\\-2&4\end{pmatrix},\quad H\left(X^{(0)}\right)^{-1}=\begin{pmatrix}1&\frac{1}{2}\\\frac{1}{2}&\frac{1}{2}\end{pmatrix}$$

根据牛顿法的迭代公式有：

$$X^{(1)}=X^{(0)}-H\left(X^{(0)}\right)^{-1}\nabla f\left(X^{(0)}\right)=\begin{pmatrix}0\\0\end{pmatrix}-\begin{pmatrix}1&\frac{1}{2}\\\frac{1}{2}&\frac{1}{2}\end{pmatrix}\begin{pmatrix}0\\-2\end{pmatrix}=\begin{pmatrix}1\\1\end{pmatrix}$$

于是 $\nabla f\left(X^{(1)}\right)=\begin{pmatrix}0\\0\end{pmatrix}$，所以 $X^*=X^{(1)}=\begin{pmatrix}1\\1\end{pmatrix}$ 为问题精确的极小点，极小值为 $f\left(X^{(1)}\right)=-1$。

显然，与例 1 对比可以知道，牛顿法的收敛速度要远远大于梯度法。

牛顿法每一步的迭代步长都是 $t_k=1$，这样虽然算法简单，但是会在一定的程度上影响收敛速度，所以每次迭代时，我们依然可以采用一维搜索的方法计算出最优步长，即找到搜索方向上的最优迭代点，该种方法被称为阻尼牛顿法。

选择非线性规划常用的测试函数[3]：$\min f=100\left(x_2-x_1^2\right)^2+\left(1-x_1\right)^2$，该函数图像如图 4 所示，此问题解析解为(1, 1)，最优值为 0，设误差为 0.1 进行迭代求解，结果见表 2。

由下面的分析可以知道，梯度法对于初值的依赖性不强(初值点从 $(0,0)\to(10,10)\to(20,20)\to(100,100)$，在满足精度要求的前提下，迭代次数从 $156\to18374\to57561\to619505$ 次)，越接近极值点收敛速度越慢，而恰好牛顿法对于初值的依赖性很强(初值点从 $(0,0)\to(10,10)\to(20,20)\to(40,40)$，迭代次数从 $3\to12598\to88854\to167330$ 次，初值点在(0, 0)时，只需迭代 3 次即可满足精度要求，而初值点在(10, 10)时，却需要迭代 12598 次才能满足精度要求)，在极值点附近收敛速度很快。再对比牛顿法和阻尼牛顿法，显然可以看出阻尼牛顿法由于每一次迭代都在搜索方向上寻求了最优的迭代点，所以他的收敛速度相比牛顿法会有明显的提升，但牛顿法固有的对初值依赖性强的问题并没有得到解决，所以我们想到结合两种算法，首先使用梯度法进行迭代，当

迭代点进入极值点的某个充分小邻域时，再使用阻尼牛顿法进行搜索，从而提升问题的收敛速度，本文将此种改进算法称作 **GN 算法**。

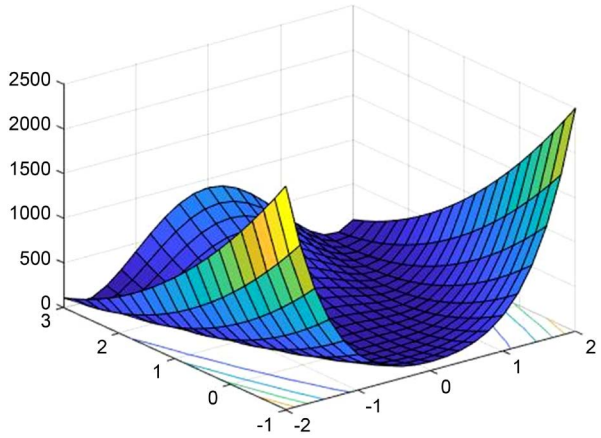

Figure 4. Function image of the test function
图 4. 测试函数的函数图像

Table 2. Initial value points and iteration table of the test function were solved by the three methods
表 2. 三种方法求解测试函数的初值点和迭代次数表

算法	初始值	最优解	最优值	迭代次数	精度
梯度法	(0, 0)	(0.9027, 0.8145)	0.0095	156	$\varepsilon_1 = 0.1$
	(10, 10)	(1.1200, 1.2549)	0.0144	18,374	$\varepsilon_1 = 0.1$
		(2.7811, 7.7360)	3.1725	5000	$\varepsilon_1 = 0.1$
	(20, 20)	(1.1186, 1.2518)	0.0141	57,561	$\varepsilon_1 = 0.1$
		(2.4586, 6.0482)	2.1287	50,000	$\varepsilon_1 = 0.1$
	(100, 100)	(1.0036, 1.0073)	1.3127e−05	619,505	$\varepsilon_1 = 0.1$
牛顿法	(0, 0)	(1.000, 1.000)	4,9304e−30	3	$\varepsilon_1 = 0.1$
	(10, 10)	(1.0498, 1.1020)	0.0025	12,598	$\varepsilon_1 = 0.1$
	(20, 20)	(1.0498, 1.1020)	0.0025	88,854	$\varepsilon_1 = 0.1$
	(40, 40)	(1.0498, 1.1020)	0.0025	167,330	$\varepsilon_1 = 0.1$
阻尼牛顿法	(0, 0)	(0.9501, 0.9027)	0.0025	317	$\varepsilon_1 = 0.1$
	(10, 10)	(7,4362, 55.2978)	41.4253	5000	$\varepsilon_1 = 0.1$
		(1.0498, 1.1020)	0.0025	12,597	$\varepsilon_1 = 0.1$
	(20, 20)	(1.0498, 1.1020)	0.0025	44,426	$\varepsilon_1 = 0.1$

GN 算法的算法步骤：

(1) 给出初始点 $X^{(0)} \in E^n$，精度 $\varepsilon_1 > 0, \varepsilon_2 > 0, \varepsilon_1 < \varepsilon_2$，令 $k = 0$；

(2) 计算 $\nabla f\left(X^{(k)}\right)$；

(3) 若 $\left\|\nabla f\left(X^{(k)}\right)\right\|<\varepsilon_1$，则迭代结束，取 $\bar{X}=X^{(k)}$，否则转(4)；

(4) 若 $\left\|\nabla f\left(X^{(k)}\right)\right\|\geq\varepsilon_1$。

(I) 若 $\left\|\nabla f\left(X^{(k)}\right)\right\|\geq\varepsilon_2$，用一维搜索求 $\varphi(t)=f\left(X^{(k)}+td^{(k)}\right)$ 的一个极小点 t_k，使 $f\left(X^{(k)}+t_kd^{(k)}\right)<f\left(X^{(k)}\right)$，其中 $d^{(k)}=-\nabla f\left(X^{(k)}\right)$。

令 $X^{(k+1)}=X^{(k)}+t_kd^{(k)},k=k+1$，转(2)。

(II) 若 $\left\|\nabla f\left(X^{(k)}\right)\right\|<\varepsilon_2$，用一维搜索求 $\varphi(t)=f\left(X^{(k)}+td^{(k)}\right)$ 的一个极小点 t_k，使 $f\left(X^{(k)}+t_kd^{(k)}\right)<f\left(X^{(k)}\right)$，其中 $d^{(k)}=-\left[\nabla^2f\left(X^{(k)}\right)\right]^{-1}\cdot\nabla f\left(X^{(k)}\right)$。

令 $X^{(k+1)}=X^{(k)}+t_kd^{(k)},k=k+1$，转(2)。

Table 3. Initial value points and iteration table of the test function were solved by the GN methods
表 3. GN 算法求解测试函数的初值点及迭代次数表

算法	初始值	最优解	最优值	迭代次数	精度
GN 算法	(0, 0)	(0.9905, 0.9810)	9.1200e−05	500	$\varepsilon_1=0.1,\varepsilon_2=0.5$
	(10, 10)	(1.0047, 1.0093)	2.1719e−05	16,000	$\varepsilon_1=0.1,\varepsilon_2=0.5$
		(1.0000, 1.0000)	9.3138e−10	17,000	$\varepsilon_1=0.1,\varepsilon_2=0.5$
	(100, 100)	(1.0000, 1.0000)	3.9211e−14	18,000	$\varepsilon_1=0.1,\varepsilon_2=0.5$

由表 3 可以看出，当初值点位于(100, 100)时，G-N 算法经 18,000 次迭代，即可得到非常精确(误差仅为 3.9211e−14)的最优解，而此时无论是牛顿法还是阻尼牛顿法甚至都是不能得到近似解的，因为牛顿法每一次迭代都严格要求迭代点处的 Hessian 矩阵可逆，而满足这点并不容易。我们改进后的算法只在极值点的充分小的邻域内使用阻尼牛顿法，既发挥了牛顿法迭代速度快的优点，又最大限度的避免了初值对其影响大的缺点，在局部最优解的充分小的邻域内依然可以保证至少二阶的收敛速度，是一种有效的改进算法。

4. 拟牛顿法

牛顿法，它的突出优点是收敛很快，但是运用牛顿法需要计算二阶偏导数，而目标函数的 Hessian 矩阵可能非正定。由于实际问题的目标函数往往相当复杂，计算二阶导数矩阵及其逆矩阵相当困难或根本不可能，因此，确定牛顿方向存在很大的障碍。为避免计算二阶导数矩阵 $\nabla^2f\left(X^{(k)}\right)$ 及其逆矩阵 $\left[\nabla^2f\left(X^{(k)}\right)\right]^{-1}$，提出了拟牛顿法。

拟牛顿法的基本思想：使用不含二阶导数的矩阵 H_k 近似替代牛顿法中的 Hessian 阵的逆矩阵 $\left[\nabla^2f\left(X^{(k)}\right)\right]^{-1}$，构造 H_k 应遵循如下三条原则：

(1) 每一步迭代均能按已有的信息确定下一个搜索方向；

(2) 每一步迭代均能使目标函数有所改进；

(3) 近似矩阵 H_k 最终应收敛于极值点处的 Hessian 矩阵的逆矩阵 $\left[\nabla^2f\left(X^{(k)}\right)\right]^{-1}$。

由于构造近似矩阵的方法不同，因而出现不同的拟牛顿法。拟牛顿法不是一种算法，而是一族公认的比较有效的算法。

前面已经给出了牛顿法的迭代方向，即牛顿方向为：

$$d^{(k)}=-\left[\nabla^2 f\left(X^{(k)}\right)\right]^{-1}\nabla f\left(X^{(k)}\right)$$

构造$\left[\nabla^2 f\left(X^{(k)}\right)\right]^{-1}$的近似矩阵$H_k$，得$d^{(k)}=-H_k\nabla f\left(X^{(k)}\right)$，称之为点$X^{(k)}$处的**拟牛顿方向**。

设在第k次迭代后，得到点$X^{(k+1)}$，将目标函数$f(X)$在$X^{(k+1)}$点处展开成Taylor级数，并取二阶近似，记$p^{(k)}=X^{(k+1)}-X^{(k)}$，$q^{(k)}=\nabla f\left(X^{(k+1)}\right)-\nabla f\left(X^{(k)}\right)$。

为了用不包含二阶导数的矩阵H_{k+1}取代牛顿法中的Hessian矩阵的逆矩阵$\left[\nabla^2 f\left(X^{(k)}\right)\right]^{-1}$，有理由令$H_{k+1}$满足：

$$p^{(k)}=H_{k+1}\cdot q^{(k)}$$

这个条件称为**拟牛顿条件**。

显然，若想使拟牛顿方向$d^{(k)}=-H_k\nabla f\left(X^{(k)}\right)$较好的近似于牛顿方向，下面基本条件应当得到满足：

(1) $d^{(k)}=-H_k\nabla f\left(X^{(k)}\right)$是目标函数$f(X)$在点$X^{(k)}$的下降方向；

(2) H_{k+1}满足$p^{(k)}=H_{k+1}\cdot q^{(k)}$；

(3) H_{k+1}与之间应具有某种简单的迭代关系

$$H_{k+1}=H_k+E_k$$

其中E_k是一个容易计算的矩阵，称之为校正矩阵。即$p^{(k)}=\left(H_k+E_k\right)\cdot q^{(k)}$。

(4) 若有$\nabla f\left(X^{(k)}\right)^{\mathrm{T}}\left[-H_k\nabla f\left(X^{(k)}\right)\right]>0$，即$\nabla f\left(X^{(k)}\right)^{\mathrm{T}}H_k\nabla f\left(X^{(k)}\right)<0$成立，则拟牛顿方向$d^{(k)}=-H_k\nabla f\left(X^{(k)}\right)$必是下降方向。

构造校正矩阵E_k的方法有很多，每一种特殊的构造方法对应一种具体的算法，因而拟牛顿法是一族算法，本文选择有代表性的变尺度算法，其主要的优点是：既避免了计算二阶导数矩阵及其求逆过程，又比梯度法收敛的速度快，特别是对高维问题具有显著的优越性。变尺度法最早由*Davidon*于1959年提出，后经*Fletcher*和*Powell*二人改进，因此变尺度法也被称为**DFP**法[2]。

变尺度法中，校正矩阵E_k的构造方式为：

$$E_k=\alpha_k u^{(k)}u^{(k)\mathrm{T}}+\beta_k v^{(k)}v^{(k)\mathrm{T}}$$

其中α_k,β_k是待定的数，$u^{(k)},v^{(k)}$是待定的向量。

将上式代入拟牛顿条件$p^{(k)}=\left(H_k+E_k\right)\cdot q^{(k)}$中，可以得到：

$$\alpha_k=\frac{1}{u^{(k)\mathrm{T}}q^{(k)}}=\frac{1}{p^{(k)\mathrm{T}}q^{(k)}}$$

$$\beta_k=\frac{-1}{v^{(k)\mathrm{T}}q^{(k)}}=\frac{-1}{q^{(k)\mathrm{T}}H_k q^{(k)}}$$

于是

$$E_k=\frac{p^{(k)}p^{(k)\mathrm{T}}}{p^{(k)\mathrm{T}}q^{(k)}}-\frac{H_k q^{(k)}q^{(k)\mathrm{T}}H_k}{q^{(k)\mathrm{T}}H_k q^{(k)}}$$

下面通过例**3**来展示变尺度法的计算过程。

例3 用**变尺度法(DFP)**求例1

$$\min f(X)=-2x_1x_2-2x_2+x_1^2+2x_2^2$$

初始点$X^{(1)}=(0,0)^{\mathrm{T}}$，初始矩阵$H_1=\begin{pmatrix}1&0\\0&1\end{pmatrix}$

解：由题可知 $f(X)=-2x_1x_2-2x_2+x_1^2+2x_2^2$

$$\nabla f(X)=\begin{pmatrix}2x_1-2x_2\\-2x_1-2+4x_2\end{pmatrix}$$

在点 $X^{(1)}=(0,0)^{\mathrm{T}}$ 处的梯度 $g_1=\begin{pmatrix}0\\-2\end{pmatrix}$，$\|g_1\|=2\neq 0$。

第一次迭代：

令搜索方向

$$d^{(1)}=-H_1g_1=-\begin{pmatrix}1&0\\0&1\end{pmatrix}\begin{pmatrix}0\\-2\end{pmatrix}=\begin{pmatrix}0\\2\end{pmatrix}$$

从 $X^{(1)}=(0,0)^{\mathrm{T}}$ 出发，沿方向 $d^{(1)}$ 进行一维搜索：

$$\min_{t\geq 0} f\left(X^{(1)}+td^{(1)}\right)$$

求得 $t_1=\frac{1}{4}$，令

$$X^{(2)}=X^{(1)}+t_1d^{(1)}=\begin{pmatrix}0\\0\end{pmatrix}+\frac{1}{4}\begin{pmatrix}0\\2\end{pmatrix}=\begin{pmatrix}0\\\frac{1}{2}\end{pmatrix}$$

此时 $g_2=\nabla f\left(X^{(2)}\right)=\begin{pmatrix}-1\\0\end{pmatrix}$，$\|g_2\|=1\neq 0$

第二次迭代：

令 $p^{(1)}=X^{(2)}-X^{(1)}=t_1d^{(1)}=\begin{pmatrix}0\\\frac{1}{2}\end{pmatrix}$

$$q^{(1)}=g_2-g_1=\begin{pmatrix}-1\\0\end{pmatrix}-\begin{pmatrix}0\\-2\end{pmatrix}=\begin{pmatrix}-1\\2\end{pmatrix}$$

于是

$$H_2=H_1+E_1=H_1+\frac{p^{(1)}p^{(1)\mathrm{T}}}{p^{(1)\mathrm{T}}q^{(1)}}-\frac{H_1q^{(1)}q^{(1)\mathrm{T}}H_1}{q^{(1)\mathrm{T}}H_1q^{(1)}}=\frac{1}{20}\begin{pmatrix}16&8\\8&9\end{pmatrix}$$

令

$$d^{(2)}=-H_2g_2=-\frac{1}{20}\begin{pmatrix}16&8\\8&9\end{pmatrix}\begin{pmatrix}-1\\0\end{pmatrix}=\frac{1}{5}\begin{pmatrix}4\\2\end{pmatrix}$$

从 $X^{(2)}=\left(0,\frac{1}{2}\right)^{\mathrm{T}}$ 出发，沿方向 $d^{(2)}$ 进行一维搜索：

$$\min_{t\geq 0} f\left(X^{(2)}+td^{(2)}\right)$$

求得 $t_2=\frac{5}{4}$，令

$$X^{(3)}=X^{(2)}+t_2d^{(2)}=\begin{pmatrix}1\\1\end{pmatrix}$$

此时 $g_3 = \nabla f\left(X^{(3)}\right) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$。

所以 $X^* = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ 为此问题的最优解。

在例 3 的讨论中，第一个尺度矩阵 H_1 为单位矩阵(对称正定阵)，以后的尺度矩阵 H_k 按算法步骤逐步形成。可以证明，如此构造的尺度矩阵均为对称正定阵，进而确保搜索方向为下降方向。

DFP 方法由于不需要在迭代点处计算 Hessin 矩阵的逆，所以在极值点附近的收敛速度比牛顿法更快，效率更高。可由表 4 列出的数值实验可以看出，当初始点距离极值点很远时，受迭代误差的影响，DFP 方法很难收敛到极值点。因此，为了提升收敛速度和求解的准确性我们同样可以在迭代点距离极值点很近时再使用拟牛顿方法求解问题。本文将此种改进方法称作 **GNN 算法**。

Table 4. Initial value points and iteration table of the test function were solved by the DFP methods
表 4. DFP 算法求解测试函数的初值点及迭代次数表

算法	初始值	最优解	最优值	迭代次数
DFP 拟牛顿算法	(0, 0)	(0.9459, 0.8946)	0.0029	13
	(4, 4)	(0.9756, 0.9543)	0.0012	2000
	(10, 10)	(1.0388, 1.0791)	0.0015	30,195
	(40, 40)	1.0e+14 * (−0.0000, 5.9352)	5.9352e+14	50,000~500,000

由表 4 和表 5，明显可以看出 GNN 算法的精确度和收敛速度对比 DFP 方法均有提升，是一种行之有效的优化算法。

Table 5. Initial value points and iteration table of the test function were solved by the GNN methods
表 5. GNN 算法求解测试函数的初值点及迭代次数表

算法	初始值	最优解	最优值	迭代次数	精度
GNN 算法	(0, 0)	(1.0000, 1.0000)	2.4159e−30	50	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$
	(4, 4)	(1.9602, 3.8455)	0.9230	1000	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$
		(1, 1)	0	2000	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$
	(10, 10)	(1.8947, 3.5927)	0.8013	14,000	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$
		(1, 1)	0	14,500	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$
	(20, 20)	(2.4586, 6.0482)	2.1287	50,000	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$
		(1, 1)	0	57,000	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.5$

5. 物流配送中心选址问题

假设在某区域内有 8 个需求点，需求点位置坐标如表 6 所示，选址区域 D 内已有 2 个配送中心，配送中心坐标分别为$\left(x_j, y_j\right)$，(4, 4)，(6, 5)，现欲在该区域内选择一个配送中心，问该如何选择配送中心的位置才能使总收益达到最大?

Table 6. Information table of the demand points
表 6. 需求点信息表

需求点	位置(a_i,b_i)	需求量v_i	需求点	位置(a_i,b_i)	需求量v_i
1	(4, 6)	70	5	(3, 3)	50
2	(2, 8)	40	6	(5, 1)	40
3	(7, 7)	50	7	(7, 3)	80
4	(8, 9)	60	8	(9, 2)	60

假设新建配送中心的位置在(x,y)处，根据 Huff 引力模型[4]，在选址区域 D 内，已存在多个配送中心，新建一个配送中心，采用配送中心利润最大原则进行，则问题模型为求新建配送中心利润的最大值[5]。

$\max\limits_{(x,y)\in D} F(x,y)=\omega\sum\limits_{i=1}^{n}p_i(x,y)\cdot v_i=\omega\sum\limits_{i=1}^{n}\dfrac{1}{1+\sum\limits_{j=1}^{m}\mathrm{e}^{-\lambda(d_{ij}-d_i)}}\cdot v_i$，其中 $p_i(x,y)$表示新建配送中心(x,y)分担需求点 i 的需求量的比例，$(a_i,b_i),i=1,2,\cdots,8$表示 8 个需求点的位置坐标，v_i表示需求点 i 的需求量，ω表示配送单位需求量的利润，λ为距离调节系数，d_i表示新建配送中心到需求点 i 的距离，d_{ij}表示已有配送中心 j 到需求点 i 的距离，即

$$d_{ij}=\sqrt{(a_i-x_j)^2+(b_i-y_j)^2};$$

$$d_i=\sqrt{(x-a_i)^2+(y-b_i)^2}$$

取$\omega=1,\lambda=0.5$，于是该问题可表示为如下非线性规划[6]：

$$\max_{(x,y)\in D} F(x,y)=\sum_{i=1}^{8}\frac{1}{1+\sum\limits_{j=1}^{2}\mathrm{e}^{-\frac{1}{2}(d_{ij}-d_i)}}v_i,$$

该问题目标函数如图 5 所示，若利用 Nelder-Mead simplex direct search 方法可求出该问题的最优解为$x^*=(6.9201,3.1403)$，在该处建立配送中心可获得利润的最大值为：157.6933。

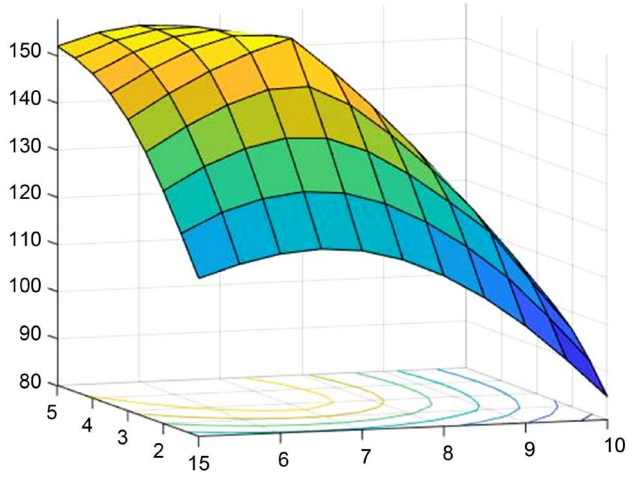

Figure 5. Function image of the objective function for the logistics distribution center problem
图 5. 物流配送中心问题目标函数图像

接下来，我们分别使用梯度法、牛顿法、拟牛顿方法以及改进的两种算法来求解此问题。

Table 7. Comparative analysis table of the five methods
表 7. 五种算法对比分析表

方法	初值点	最优解	最优值	迭代次数	精度
梯度法	(1, 1)	(6.9011, 3.1754)	157.6921	20	$\varepsilon = 0.1$
	(7, 4)	(6.9473, 3.0935)	157.6910	17	$\varepsilon = 0.1$
	(18, 18)	(6.8988, 3.1761)	157.6920	25	$\varepsilon = 0.1$
阻尼牛顿法	(1, 1)	(−9.3054, −5.9231)	0.1573	5	$\varepsilon = 0.1$
	(5, 4)	(6.9216, 3.1381)	157.6933	5	$\varepsilon = 0.1$
	(7, 4)	(6.9205, 3.1383)	157.6933	4	$\varepsilon = 0.1$
	(18, 18)	(20.9756, 20.7512)	0.0812	2	$\varepsilon = 0.1$
DFP 拟牛顿法	(1, 1)	(6.9201, 3.1403)	157.6933	11	$\varepsilon = 0.1$
	(7, 4)	(6.9201, 3.1403)	157.6933	11	$\varepsilon = 0.1$
	(20, 20)	(6.9201, 3.1403)	151.8361	190	$\varepsilon = 0.1$
GN 算法	(1, 1)	(6.9202, 3.1402)	157.6933	10	$\varepsilon_1 = 0.1, \varepsilon_2 = 1$
	(5, 4)	(6.9206, 3.1399)	157.6933	5	$\varepsilon_1 = 0.1, \varepsilon_2 = 1$
	(7, 4)	(6.9303, 3.1271)	157.6929	5	$\varepsilon_1 = 0.1, \varepsilon_2 = 1$
	(18, 18)	(6.9214, 3.1393)	157.6933	25	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.2$
GNN 算法	(1, 1)	(6.7779, 3.3778)	157.6579	10	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.2$
	(5, 4)	(6.8070, 3.3347)	157.6579	5	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.2$
	(7, 4)	(6.9608, 3.0714)	157.6883	5	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.2$
	(18, 18)	(6.8112, 3.3338)	157.6574	20	$\varepsilon_1 = 0.1, \varepsilon_2 = 0.2$

由表 7 的对比分析可以看出，五种算法中效率最差的是梯度法，一般而言梯度法只具有线性收敛速度，而阻尼牛顿法和 DFP 算法在极值点的充分小邻域内具有至少二阶的收敛速度。但是对于本问题而言，在极值点附近阻尼牛顿法的效率明显高于 DFP 算法，是因为该问题的运算量不大，各迭代点处 Hessian 矩阵的逆存在，阻尼牛顿法在有限步内迭代简单可逐渐收敛到极值点，而 DFP 算法在此邻域内因为要叠加误差，所以收敛的速度反而没有阻尼牛顿法快，但是当迭代点到达(18, 18)时，受 Hessian 矩阵的逆的存在性影响，阻尼牛顿法已经不能收敛到极值点，但此问题显然借助 DFP 算法可以解决。而毫无疑问改进的两种算法无论从迭代速度还是初值点的选择上都增加了问题的实用性，对初值点的选择没有那么强，同时迭代速度对比 DFP 算法有明显提升。

6. 结论

本文结合梯度法和阻尼牛顿法提出了 GN 算法，结合梯度法和变尺度法提出了 GNN 算法，从理论上解释算法的收敛性，用测试函数进行了收敛速度的验证。同时将几种算法应用于有竞争机制的物流中心选址问题，验证算法的实用性，同时给出算法的对比分析。

参考文献

[1] 陈宝林. 最优化理论与算法[M]. 北京: 清华大学出版社, 2005.

[2] 甘应爱. 运筹学: 本科版[M]. 北京: 清华大学出版社, 2005.

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

[4] Huff, D.I. (1964) Defining and Estimating a Trading Area. *Journal of Marking*, **28**, 34-38. https://doi.org/10.1177/002224296402800307

[5] Drezner, T. (1994) Locating a Single New Facility among Existing Unequally Facilities. *Journal of Regional Science*, **34**, 237-252. https://doi.org/10.1111/j.1467-9787.1994.tb00865.x

[6] 王瑞, 胡洁琼. 基于竞争的配送中心选址研究[J]. 物流科技, 2013(7): 54-55+60.