

基于梯度投影的广义滤子填充函数方法

张慧雯¹, 王薇¹, 李民¹, 徐以汎²

(1. 华东理工大学数学系, 上海 200237)

(2. 复旦大学管理学院, 上海 200433)

摘要: 本文研究了约束非凸全局优化问题. 利用滤子技术和填充函数的架构, 提出了一个基于梯度投影的广义滤子填充函数算法, 获得了较好的理论性质和数值效果. 文章修改了填充函数的定义以及滤子技术的适用范围, 推广了局部优化技术, 使之成为约束全局问题的有效求解方法之一.

关键词: 非凸全局优化; 约束函数; 填充函数; 三维滤子

MR(2010) 主题分类号: 90C30; 65K05

中图分类号: O221.2

文献标识码: A

文章编号: 0255-7797(2019)01-0029-13

1 引言与假设

本文讨论如下带线性约束的全局优化问题,

$$\begin{aligned} \min f(x), \\ \text{s.t. } Ax \leq b, \end{aligned} \quad (1.1)$$

其中 $f(x) : R^n \rightarrow R$ 为非凸函数, A 为 $m \times n$ 阶矩阵, $b = (b_1, b_2, b_3, \dots, b_m)^T \in R^m$. 对约束问题 (1.1), 称所有满足约束的点为可行点, 由所有可行点组成的集合 $X = \{x \in R^n : Ax \leq b\}$ 为可行域. 一般说来, 问题 (1.1) 有多个极小值, 用传统方法求解只能得到局部最优, 而且对于全局最优解的判定条件至今都缺少实用的研究成果, 这都使得问题求解仍然面临着许多困难.

填充函数法是求解多极值最优化问题的常用算法之一, 其主要思想是: 在求得问题的一个局部极小点后, 构造填充函数. 通过极小化该填充函数, 寻找比当前局部极小点更优的点, 进而得到更优的极小值^[1-5]. 与其他方法相比, 该算法思想简单, 而且仅用到经典算法, 所以易于实现且效率较高, 同时也可以推广到其他非线性问题以及离散数学规划的求解中^[6].

滤子技术最早由 Fletcher 和 Leyffer 提出, 他们详细讨论了滤子作为代替罚函数的工具在局部优化算法中的一些应用^[7,8]. 之后滤子技术在局部优化问题的求解中被认为是一种更有效的方法, 因其良好的数值效果, 许多学者继续进行了一系列的相关研究^[9-11].

梯度投影算法自从被 Rosen^[12] 提出后就引起了广泛的注意和系统的研究^[13,14], 由于该方法简单、实际应用的数值效果好, 在一些更有效的近代算法中也继续沿用了它的基本思想^[15,16]. 本文为了优化填充函数算法, 将滤子技术和改进的梯度投影方法融入到全局优化算法中, 提出了基于梯度投影的广义滤子填充函数算法求解问题 (1.1).

为方便起见, 下面做一些记号说明.

*收稿日期: 2017-05-17 接收日期: 2017-10-18

基金项目: 国家自然科学基金 (71372113).

作者简介: 张慧雯 (1993-), 女, 江苏南通, 硕士, 主要研究方向: 数学规划.

$J = \{1, 2, \dots, m\}$ 为指标集, A 的第 j 行为 a_j^T , 记 $c_j(x) = a_j^T x - b_j, j \in J$.

记约束违反度函数 $h(x) = \max(0, c_j(x), j \in J)$, $A(x) = (a_j, j \in J_0(x))$, 其中 $J_0(x) = \{j | c_j(x) \geq 0, j \in J\}$, 并且记 $\bar{J}_0(x) = J \setminus J_0(x)$.

$L(P)$ 表示问题 (1.1) 的局部最优解集合, $G(P)$ 表示问题 (1.1) 的全局最优解集合.

x^* 是 $f(x)$ 的一个稳定点, $S_1(x^*) = \{x | f(x) \geq f(x^*), x \in X \setminus \{x^*\}\}$ 称为高水平集, $S_2(x^*) = \{x | f(x) < f(x^*), x \in X\}$ 称为低水平集.

$\text{int}X$ 表示可行域 X 的内点集合, ∂X 表示可行域 X 的边界点集合.

考虑问题 (1.1), 我们首先提出以下假设:

[A1] $f(x)$ 只有有限多个极小值, 即存在直径充分大的闭箱 Ω 能包含所有极小值.

[A2] $\nabla f(x)$ 在 Ω 上连续.

[A3] $\forall x \in R^n, \{a_j, j \in J_0(x)\}$ 为线性无关向量组.

由 A1 和 A2 可知, 原问题的全局解必在有界闭区域 $X \cap \Omega$ 上, 因此可以认为 X 是有界闭集, 且算法中的 x 总是取自 Ω .

2 广义填充函数

定义 2.1 设 x^* 是问题 (1.1) 当前的局部极小值, 称 $T(x, x^*)$ 是 $f(x)$ 在 x^* 处的广义填充函数, 如果 $T(x, x^*)$ 满足

(i) $T(x, x^*)$ 在高水平集 $S_1(x^*)$ 上没有稳定点;

(ii) 如果 x^* 不是问题 (1.1) 的一个全局极小值点, 即 $x^* \notin G(P)$, 那么存在点 $x_1^* \in S_2(x^*)$, 使得 x_1^* 是 $T(x, x^*)$ 的极小值点.

下面给出求解问题 (1.1) 的广义填充函数.

设 x^* 是问题 (1.1) 的一个局部极小点, 定义单参数广义填充函数

$$T(x, x^*, r) = \frac{1 - e^{-\frac{1}{r^2}[f(x) - f(x^*) + r]}}{1 + \|x - x^*\|}, \quad (2.1)$$

其中

$$0 < r < \min\{|f(x_1^*) - f(x_2^*)| : f(x_1^*) \neq f(x_2^*); x_1^*, x_2^* \in L(P)\}. \quad (2.2)$$

根据 [A2], 显然 $T(x, x^*, r)$ 在可行域上是连续可微的. 下面来讨论 $T(x, x^*, r)$ 的性质.

定理 2.1 对充分小的 $r > 0$, 有如下结论成立

(i) 函数 $T(x, x^*, r)$ 在集合 $S_1(x^*) \setminus \partial X$ 上没有稳定点;

(ii) 当 $x - x^*$ 与 $\{a_j, j \in J_0(x)\}$ 线性无关时, 函数 $T(x, x^*, r)$ 在集合 $S_1(x^*) \cap \partial X$ 上没有稳定点.

证 (i) $\forall x \in S_1(x^*) \setminus \partial X$, 有

$$\begin{aligned} & \frac{(x - x^*)^T}{\|x - x^*\|} \nabla T(x, x^*, r) \\ = & \frac{(x - x^*)^T}{\|x - x^*\|} \frac{-e^{-\frac{1}{r^2}[f(x) - f(x^*) + r]} \left[-\frac{1}{r^2} \nabla f(x) \right] (1 + \|x - x^*\|) - \{1 - e^{-\frac{1}{r^2}[f(x) - f(x^*) + r]}\} \frac{x - x^*}{\|x - x^*\|}}{(1 + \|x - x^*\|)^2} \\ = & \frac{e^{-\frac{1}{r^2}[f(x) - f(x^*) + r]}}{r^2(1 + \|x - x^*\|)} \left\{ \frac{(x - x^*)^T}{\|x - x^*\|} \nabla f(x) - r^2 e^{\frac{1}{r^2}[f(x) - f(x^*) + r]} T(x, x^*, r) \right\}. \end{aligned}$$

根据 A2, $\left| \frac{(x-x^*)^T}{\|x-x^*\|} \nabla f(x) \right| \leq \|\nabla f(x)\|$ 在可行域内是有界的. 而由 $x \in S_1(x^*) \setminus \partial X$ 可知 $f(x) \geq f(x^*)$, 则 $T(x, x^*, r) > 0$. 所以当 $r > 0$ 充分小时, 有

$$\frac{(x-x^*)^T}{\|x-x^*\|} \nabla T(x, x^*, r) < 0, \quad (2.3)$$

即函数 $T(x, x^*, r)$ 在集合 $S_1(x^*) \setminus \partial X$ 上没有稳定点.

(ii) 假设 $\exists x_L \in S_1(x^*) \cap \partial X$ 是 $T(x, x^*, r)$ 的稳定点, 则有 $\Lambda(x_L) = (\lambda_1, \lambda_2, \dots, \lambda_m)^T$, 使得

$$\nabla T(x_L, x^*, r) = \sum_{j=1}^m \lambda_j a_j \quad (2.4)$$

成立, 其中 $\lambda_j \geq 0, \lambda_j c_j(x_L) = 0, j \in J$. 而由 (i) 的证明可知

$$\nabla T(x_L, x^*, r) = \frac{e^{-\frac{1}{r^2}[f(x_L)-f(x^*)+r]}}{r^2(1+\|x_L-x^*\|)} \nabla f(x_L) - \frac{T(x_L, x^*, r)}{1+\|x_L-x^*\|} \frac{x_L-x^*}{\|x_L-x^*\|} \quad (2.5)$$

且当 $r > 0$ 充分小时, 显然上式右端第一项趋于 0. 则由 (2.4), (2.5) 式知

$$x_L - x^* = - \sum_{j=1}^m \frac{\lambda_j \|x_L - x^*\| (1 + \|x_L - x^*\|)}{T(x_L, x^*, r)} a_j,$$

这与已知条件矛盾, 故函数 $T(x, x^*, r)$ 在集合 $S_1(x^*) \cap X$ 上没有稳定点.

由 (2.3) 式, 显然有

推论 2.1 $x - x^*$ 是函数 $T(x, x^*, r)$ 在 $S_1(x^*) \setminus \partial X$ 处的一个严格下降方向.

定理 2.2 如果 $x^* \in L(P)$, 但是 $x^* \notin G(P)$, 那么 $T(x, x^*, r)$ 在 $S_2(x^*)$ 上至少有一个极小值点.

证 由于 $G(P)$ 非空和 (2.2) 式的定义, 则必有 $x_G \in G(P)$, 使得 $T(x_G, x^*, r) < 0$. 而 $T(x, x^*, r)$ 在 X 上连续, 所以必存在函数的最小值点 $\bar{x} \in X$, 使得

$$T(\bar{x}, x^*, r) \leq T(x_G, x^*, r) < 0.$$

也就是说 $1 - e^{-\frac{1}{r^2}[f(x)-f(x^*)+r]} < 0$, 即 $f(\bar{x}) < f(x^*)$, $\bar{x} \in S_2(x^*)$.

推论 2.2 如果 $x \in G(P)$, 则对充分小的 $r > 0$, 函数 $T(x, x^*, r)$ 在可行域内部 $\text{int}X$ 没有稳定点.

由定理 2.1 和定理 2.2 可知, $T(x, x^*, r)$ 是一个广义填充函数.

3 滤子和梯度投影

滤子最初定义为由两个相互竞争的函数 $\phi(x)$ 和 $\psi(x)$ 组成的数对集合, 记为 $(\phi(x), \psi(x))$. 为了之后讨论方便, 本小节将给定的一阶连续可微函数 $Q(x)$ 作为目标函数, 即考虑如下问题

$$\begin{aligned} \min & Q(x), \\ \text{s.t.} & Ax \leq b. \end{aligned} \quad (3.1)$$

取 $\phi(x)$ 为 $Q(x)$, $\psi(x)$ 为 $h(x)$, 下面给出“支配”的相关概念.

定义 3.1 点 x^k 支配点 x^l 当且仅当 $Q(x^k) \leq Q(x^l)$ 且 $h(x^k) \leq h(x^l)$.

定义 3.2 滤子是由一系列互不支配的数对组成, 即若点 x^k 和 x^l 都在滤子中, 那么 $Q(x_k) \leq Q(x_l)$ 和 $h(x_k) \leq h(x_l)$ 两者必有之一不成立.

此外, 为了保证算法的下降性, 在本文中如果说点 x_{k+1} “被滤子接受” 当且仅当存在滤子中的点 x_l , 使得下面两个不等式

$$Q(x_{k+1}) < Q(x_l) - \beta h(x_l), \quad (3.2)$$

$$h(x_{k+1}) < (1 - \eta)h(x_l) \quad (3.3)$$

之一成立, 其中 $\beta, \eta \in (0, 1)$.

从以上概念可以看出, 滤子能够作为一个评判标准来决定对当前试探步 (沿下降方向寻找合适步长时假设的迭代点) 是否接受. 但是, 以 (3.2) 和 (3.3) 式作为滤子集在接受标准可能导致算法收敛到一个可行的非稳定点. 因此, 本文在下降搜索时又另外采用了其他准则.

对当前迭代点 x_k , 记

$$\begin{aligned} U(x_k) &= (u_j(x_k), j \in J_0(x))^\top = -(A(x_k)^\top A(x_k))^{-1} A(x_k)^\top \nabla Q(x_k), \\ P(x_k) &= I - A(x_k)(A(x_k)^\top A(x_k))^{-1} A(x_k)^\top, \end{aligned}$$

其中 I 为 n 阶单位矩阵, 称 $P(x_k)$ 为 x_k 处的投影矩阵, 并记 x_k 处的搜索方向为

$$d_k = \begin{cases} -P(x_k)\nabla Q(x_k), & \text{当 } x_k \in X, \\ -P(x_k)\nabla Q(x_k) + \rho_k B(x_k)^\top \omega, & \text{当 } x_k \notin X, \end{cases} \quad (3.4)$$

其中

$$\begin{aligned} \rho_k &= \frac{\nabla Q(x_k)^\top P(x_k)\nabla Q(x_k) + h(x_k)}{2|U(x_k)^\top \omega| + 1}, \\ B(x_k) &= (A(x_k)^\top A(x_k))^{-1} A(x_k)^\top, \quad \omega = (\omega_j, j \in J_0(x_k))^\top, \quad \omega_j = -1. \end{aligned}$$

对于当前点的试探步长 α_k , 本文要求一个足够下降量标准 (转换条件)

$$m_k(\alpha_k) < 0, \quad [-m_k(\alpha_k)]^{s_1} (\alpha_k)^{1-s_1} > \delta_1 [h(x_k)]^{s_2} \quad (3.5)$$

成立. 固定参数为 $\delta_1 > 0$, $s_2 > 1$, $s_1 > 2s_2$, 其中 $m_k(\alpha_k) := \alpha_k \nabla Q(x_k)^\top d_k$.

若转换条件 (3.5) 成立, 则可以不再受滤子接受准则约束, 只要求目标函数的 Armijo 条件

$$Q(x_k(\alpha_k)) \leq Q(x_k) + \delta_2 m_k(\alpha_k) \quad (3.6)$$

成立. 其中 $\delta_2 \in (0, \frac{1}{2})$ 是一个固定的常数. 不难看出, 若对某一试探步长 α_k , 转换条件 (3.5) 成立, 而 Armijo 条件 (3.6) 不成立, 那么对于再小一点的步长, 转换条件也将有可能不再成立.

有时算法在搜索过程中目标函数逐步下降但迭代点却离可行域越来越远, 此时就需要进入可行性恢复阶段. 下面简述下可行性恢复的具体过程.

设点 $x \in \Omega$ 但 $x \notin X$, 给定一个固定的参数 $\varepsilon > 0$, 不妨设有 $c_{i_j}(x) > \varepsilon, j = 1, 2, \dots, k, c_{i_j}(x) \leq \varepsilon, j = k+1, \dots, m$, 即存在 k 个约束不在可接受的范围内, 需要进行可行性恢复, 则求解问题

$$\begin{aligned} \min \quad & h(x), \\ \text{s.t.} \quad & c_{i_j}(x) \leq \varepsilon, j = k+1, \dots, m. \end{aligned} \quad (3.7)$$

对于进入可行性恢复阶段的判定, 本文采用如下准则.

如果当前迭代步 α_k 足够大, 转换条件 (3.5) 对于某个 $\alpha \leq \alpha_k$ 是成立的, 那么仍有可能存在某个小一点的步长能被 Armijo 条件 (3.6) 接受, 此时就不用进入可行性恢复阶段. 故由 (3.5) 式可知, 若 $\nabla Q(x_k)^T d_k < 0$, 只要 $\alpha_k > \frac{\delta_1 [h(x_k)]^{s_2}}{[-\nabla Q(x_k)^T d_k]^{s_1}}$, 就不进入可行性恢复阶段.

然而, 当转换条件 (3.5) 不成立时, 考虑如下两个线性近似

$$\begin{aligned} Q(x_k + \alpha_k d_k) &= Q(x_k) + \alpha_k \nabla Q(x_k)^T d_k + o(\|\alpha_k d_k\|), \\ h(x_k + \alpha_k d_k) &= \max\{0, c_j(x_k) + \alpha_k a_j^T d_k + o(\|\alpha_k d_k\|), j \in \overline{J_0(x)}\}. \end{aligned}$$

当 $a_j^T d_k < 0$ 时, 若 $\alpha_k \leq -\frac{\eta c_j(x_k)}{a_j^T d_k}$, 则无法满足约束违反度函数的足够下降量. 类似地,

当 $\nabla Q(x_k)^T d_k < 0$ 时, 若 $\alpha_k \leq -\frac{\beta Q(x_k)}{\nabla Q(x_k)^T d_k}$, 目标函数的足够下降量也不再满足. 即当 $a_j^T d_k < 0$ 时, 对目标函数的下降存在一个最小试探步长

$$\alpha_k^{\min} := \theta \cdot \begin{cases} \min \left\{ \frac{\delta_1 [h(x_k)]^{s_2}}{[-\nabla Q(x_k)^T d_k]^{s_1}}, -\frac{\beta Q(x_k)}{\nabla Q(x_k)^T d_k}, -\frac{\eta c_j(x_k)}{a_j^T d_k} \right\}, & \text{若 } \nabla Q(x_k)^T d_k < 0, \\ -\frac{\eta c_j(x_k)}{a_j^T d_k}, & \text{否则.} \end{cases} \quad (3.8)$$

当步长 $\alpha_k < \alpha_k^{\min}$ 时, 算法进入可行性恢复阶段, 其中 $\theta \in (0, 1]$ 为一常数, 用来补充线性化时被省略的高阶项, 避免不必要的可行性恢复.

当 $a_j^T d_k = 0$ 时, 最小试探步长即为

$$\alpha_k^{\min} := \theta \cdot \min \left\{ \frac{\delta_1 [h(x_k)]^{s_2}}{[-\nabla Q(x_k)^T d_k]^{s_1}}, -\frac{\beta Q(x_k)}{\nabla Q(x_k)^T d_k} \right\}. \quad (3.9)$$

4 基于梯度投影的广义滤子填充函数算法及其性质

本文研究的是带线性约束的全局优化问题, 同时从目标函数、填充函数和约束违反度函数三个角度考虑, 所以将一般滤子推广到三维, 构成滤子 $(f(x), T(x, x^*), h(x))$. 即

(i) 点 x_k 支配点 x_l 当且仅当 $f(x_k) \leq f(x_l)$, $T(x_k, x^*) \leq T(x_l, x^*)$ 和 $h(x_k) \leq h(x_l)$ 同时成立.

(ii) 若点 x_k 和点 x_l 都在滤子中, 那么在 $f(x_k) \leq f(x_l)$, $T(x_k, x^*) \leq T(x_l, x^*)$ 和 $h(x_k) \leq h(x_l)$ 中至少有一个不等式不成立.

在本文中, 用 F_k 表示点 $\{x_l | l \leq k\}$ 的集合, 也就是说 $(f(x_l), T(x_l, x^*), h(x_l))$ 是当前滤子中的元素, F_k 为当前滤子. 另外, $|F|$ 表示滤子 F 所包含的元素个数.

此外, 如果说点 x_{k+1} “被滤子 F_k 接受” 当且仅当存在点 $x_l \in F_k$, 使得下面三个不等式

$$f(x_{k+1}) < f(x_l) - \beta_1 h(x_l), \quad (4.1)$$

$$T(x_{k+1}, x^*) < T(x_l, x^*) - \beta_2 h(x_l), \quad (4.2)$$

$$h(x_{k+1}) < (1 - \eta)h(x_l) \quad (4.3)$$

之一成立, 其中 $\beta_1, \beta_2, \eta \in (0, 1)$.

定义 4.1 将数组 $(f(x), T(x, x^*), h(x))$ 加入到当前滤子中, 并把被 $(f(x), T(x, x^*), h(x))$ 支配的所有数组移出滤子的过程称为更新滤子. 即

$$\begin{aligned} F_{k+1} &= F_k \cup \{x_{k+1}\} \setminus \{x_l \in F_k \mid f(x_{k+1}) \\ &\leq f(x_l), T(x_{k+1}, x^*) \leq T(x_l, x^*), h(x_{k+1}) \leq h(x_l)\}. \end{aligned} \quad (4.4)$$

根据三维滤子的定义可知, 如果初始滤子集非空, 那么在任意点处的滤子集非空, 即 $|F| \geq 1$. 在提出广义滤子填充函数算法之前, 我们先给出试探步长 α_k 的迭代算法.

回溯线搜索算法

步骤 0 令 $\alpha_k = 1$.

步骤 1 若 $\alpha_k < \alpha_k^{\min}$ 时, 转步骤 5; 否则, 计算下一代 $x_k(\alpha_k) = x_k + \alpha_k d_k$.

步骤 2 若滤子中存在数组能支配 $x_k(\alpha_k)$, 则拒绝试探步, 转步骤 4; 否则, 进入步骤 3.

步骤 3 检验足够下降量

3.1 情况 I 若 $x_k \in X$ 时, (3.5) 和 (3.6) 式同时成立, 并且 $x_k(\alpha_k) \in X$, 则 α_k 为所求步长; 否则, 转步骤 4.

3.2 情况 II 若 $x_k \notin X$ 且 (3.5) 式成立时, (3.6) 式成立, 则 α_k 为所求步长; 否则, 转步骤 4.

3.3 情况 III 若 $x_k \notin X$ 且 (3.5) 式不成立时, 试探点能被滤子接受, 则 α_k 为所求步长; 否则, 转步骤 4.

步骤 4 选择新的试探步长 $\alpha_k = \frac{1}{2}\alpha_k$, 转步骤 2.

步骤 5 可行性恢复: 求解问题 (3.7), 并更新滤子集.

本算法的目的是对当前迭代点 x_k 和方向 d_k , 寻找合适的下降步长. 当 x_k 不需要进行可行性恢复且下一步迭代 $x_k(\alpha_k)$ 不被滤子支配时, 试探步长 α_k 还需要满足足够下降量或者滤子接受准则. 算法中对足够下降量的检验作了详细的情况分析, 确保当前点无论是否在可行域内一定存在可被接受的步长.

下面给出主算法.

算法

步骤 0 任取 $x_0 \in R^n$, 给定精度参数 $\varepsilon > 0$, 邻域半径 $\delta > 0$, r_0, r, G, N 为变量 x 的维数. 令 $k := 0$, 若当前点没有填充函数值, 则令其分量 $T = Z$, Z 充分大. 初始化滤子集 $F_0 = (f(x_0), Z, h(x_0))$.

步骤 1 取 $Q(x)$ 为 $f(x)$, 若有 $d_k = -P(x_k)\nabla f(x_k) = 0$, 当 $U(x_k) \geq 0$ 和 $h(x_k) = 0$ 同时成立, 令 $x^* = x_k$, 转步骤 5; 当存在 $u_j(x_k) < 0, j \in J_0(x)$ 时, 转步骤 4; 否则, 进入步骤 2.

步骤 2 回溯线搜索, 若因可行性恢复跳出, 转步骤 1; 否则, 进入步骤 3.

步骤 3 令 $x_{k+1} = x_k(\alpha_k), k \leftarrow k + 1$, 更新滤子, 转步骤 1.

- 步骤 4 令 $J_0(x_k) \setminus \{j\}$, 重新构造 $A(x_k)$, 转步骤 1.
 步骤 5 在 x^* 处构造填充函数 $T(x, x^*, r)$, 令 $S = 1$.
 步骤 6 若 $S > 2N$, 则令 $r = \frac{r}{10}$, 转步骤 12; 否则, 令 $k := 0$, 取 $x_k \in O(x^*, \delta) \setminus \{x^*\}$.
 步骤 7 取 $Q(x)$ 为 $T(x)$, 若 $d_k = 0$, 令 $F = \{x^*\}$, $S = S + 1$, 转步骤 6; 否则进入步骤 8.
 步骤 8 回溯线搜索, 若因可行性恢复跳出, 转步骤 11; 否则, 进入步骤 9.
 步骤 9 令 $x_{k+1} = x_k(\alpha_k)$, $k \leftarrow k + 1$, 更新滤子.
 步骤 10 若 $|F| = 1$, 则令 $k := 0$, 转步骤 1; 否则, 进入步骤 7.
 步骤 11 若 $|F| > G$, 则令 $F = \{x^*\}$, $S = S + 1$, 转步骤 6; 否则, 转步骤 7.
 步骤 12 若 $r < r_0$, 则停止, x^* 为全局最优; 否则, 进入步骤 5.

下面讨论该算法的相关性质.

引理 4.1 矩阵 $P(x)$ 是半正定矩阵, 且满足 $P(x)^2 = P(x)$.

定理 4.1 (i) 当 $x_k \in X$ 且非 KKT 点时, 由算法得到的 d_k 满足

$$\nabla Q(x_k)^T d_k < 0, \quad a_j^T d_k = 0, j \in J_0(x_k);$$

(ii) 当 $x_k \notin X$ 时, 由算法得到的 d_k 满足 $a_j^T d_k < 0, j \in J_0(x_k)$.

证 (i) 显然, 当 $x_k \in X$ 且非 KKT 点时, 有

$$\nabla Q(x_k)^T d_k = -\nabla Q(x_k)^T P(x) \nabla Q(x) = -\|P(x) \nabla Q(x)\|^2 < 0.$$

另外

$$A(x_k)^T d_k = -A(x_k)^T (I - A(x)(A(x)^T A(x))^{-1} A(x)^T) \nabla Q(x_k) = 0,$$

即 $a_j^T d_k = 0, j \in J_0(x_k)$, 结论得证.

(ii) 当 $x_k \notin X$ 时,

$$A(x_k)^T d_k = -A(x_k)^T P(x_k) \nabla Q(x_k) + \rho_k A(x_k)^T B(x_k)^T \omega = \rho_k \omega < 0,$$

即 $a_j^T d_k < 0, j \in J_0(x_k)$, 结论得证.

定理 4.2 回溯线搜索有限终止.

证 (i) 若 $x_k \in X$ 时, $\alpha_k^{\min} = 0$. 由定理 4.1 知在非 KKT 点处, $\nabla Q(x_k)^T d_k < 0$, 则必有 $m_k(\alpha_k) < 0$, 并且 $[-m_k(\alpha_k)]^{s_1} (\alpha_k)^{1-s_1} > 0 = \delta_1 [h(x_k)]^{s_2}$, 即 (3.5) 式成立.

另一方面, 当 α_k 足够小时, 有

$$\begin{aligned} Q(x_k(\alpha_k)) - Q(x_k) &= Q(x_k) + \alpha_k \nabla Q(x_k)^T d_k + o(\alpha_k d_k) - Q(x_k) \\ &= \alpha_k \nabla Q(x_k)^T d_k + o(\alpha_k d_k) \leq \delta_2 \alpha_k \nabla Q(x_k)^T d_k, \end{aligned}$$

即 (3.6) 式成立. 因此若 $x_k \in X$ 时, 算法必能找到合适的 α_k 被接受.

(ii) 若 $x_k \notin X$ 时, $\alpha_k^{\min} > 0$, 此时要么在步骤 3 产生一个可被接受的新迭代, 要么进入可行性恢复阶段.

综上, 算法必能找到合适的 α_k , 回溯线搜索总会有限终止.

为了证明算法性质, 本文给出如下假设

[A4] 存在 $M_m > 0$, 使得 $|m_k(\alpha)| \leq M_m \alpha$.

定理 4.3 若假设都成立, 则有

$$\lim_{k \rightarrow \infty} h(x_k) = 0. \quad (4.5)$$

证 (i) 若滤子集仅在有限步扩大, 假设 K , 当迭代 $k > K$ 时, 滤子集不再扩充. 则由算法可知, 对所有 $k \geq K$, (3.5) 和 (3.6) 式都成立, 则

$$\delta_1 [h(x_k)]^{s_2} < [-m_k(\alpha_k)]^{s_1} \alpha_k^{1-s_1} \leq M_m^{s_1} \alpha_k.$$

故 $\alpha_k > \frac{\delta_1 [h(x_k)]^{s_2}}{M_m^{s_1}}$. 由 $1 - \frac{1}{s_1} > 0$, 有

$$Q(x_{k+1}) - Q(x_k) \leq \delta_2 m_k(\alpha_k) < -\delta_2 \delta_1^{\frac{1}{s_1}} (\alpha_k)^{1-\frac{1}{s_1}} [h(x_k)]^{\frac{s_2}{s_1}} < -\delta_1 \delta_2 M_m^{1-s_1} [h(x_k)]^{s_2}.$$

故对 $i = 1, 2, \dots$,

$$Q(x_{K+i}) = Q(x_K) + \sum_{k=K}^{K+i-1} [Q(x_{k+1}) - Q(x_k)] < Q(x_K) - \delta_1 \delta_2 M_m^{1-s_1} \sum_{k=K}^{K+i-1} [h(x_k)]^{s_2}.$$

由 $Q(x_{K+i})$ 下有界可知, (4.5) 式成立.

(ii) 若滤子集一直扩大, 下降搜索时所得序列 $\{x_{K_i}\}$, 令 $\{x_{k_i}\}$ 是其子序列, 滤子集在 $k_i \in \{K_i\}$ 扩大, 对所有 i , 不妨设存在常数 $C_1 \in R$ 和 $C_2 > 0$, 使得 $Q(x_{k_i}) \geq C_1$, $h(x_{k_i}) \leq C_2$. 由定理 4.1 和可行性恢复可知, 必有 $\lim_{i \rightarrow \infty} h(x_{k_i}) = 0$ 成立, 则 (4.5) 式成立.

下面验证滤子的相关性质.

定理 4.4 设 $x_k \in S_1(x^*)$, 当前滤子集为 F_k . 若 $r > 0$ 充分小, 则一定存在 $\alpha > 0$, 使得 $x_{k+1} = x_k + \alpha d$ 被滤子 F_k 接受.

证 由定理 4.1 可知, 当 $x_k \in X$ 且不是问题 (3.1) 的 KKT 点时, 由算法得到的 d_k 满足

$$\nabla Q(x_k)^T d_k < 0; \quad a_j^T d_k = 0, \quad j \in J_0(x),$$

所以当 $x_k \in S_1(x^*)$ 时, 算法中的迭代方向一定是 $f(x_k)$ 或 $T(x_k, x^*, r)$ 的下降方向. 根据算法的下降性以及滤子的定义, 必有 $x_l \in F_k$, 使得 $f(x_{k+1}) < f(x_l)$ 或 $T(x_{k+1}, x^*) < T(x_l, x^*)$ 成立, 即 x_{k+1} 能被滤子 F_k 接受.

下面讨论滤子集 F_k 和低水平集 $S_2(x^*)$ 的关系.

定理 4.5 设 x_k 处的滤子集为 $F_k = \{x_l \mid 1 \leq l \leq k\}$. 如果点 x_{k+1} 被 F_k 接受, 并且支配所有 x_l ($\forall x_l \in F_k$), 则 $x_{k+1} \in S_2(x^*)$.

证 若 $x_{k+1} \notin X$, 则对 $x_l \in F_k \cap X$, x_{k+1} 必然无法支配, 故 $x_{k+1} \in X$. 若存在 $x_l \in F_k \cap S_2(x^*)$, 因为 x_{k+1} 支配 x_l , 结论显然成立. 否则, 对所有 $x_l \in F_k \cap (S_1(x^*) \cup \{x^*\})$, 根据滤子的定义, 点 x^* 肯定在 F_k 内. 而 x_{k+1} 被 F_k 接受且支配 x^* , 由不等式 (4.1)–(4.3) 可知 $f(x_{k+1}) < f(x^*)$, 即 $x_{k+1} \in S_2(x^*)$.

定理 4.6 设 x_k 处的滤子集为 $F_k = \{x_l \mid 1 \leq l \leq k\} \subseteq S_1(x^*)$, 且 $S_2(x^*) \neq \emptyset$. 则必存在点 $x_N \in S_2(x^*)$ 使得 $\forall x_l \in F_k$, x_N 支配 x_l .

证 首先, $\forall x_N \in S_2(x^*)$, 有 $f(x_N) < f(x_l)$ ($\forall x_l \in F_k \cap S_1(x^*)$).

又由于 (2.2) 式和 $S_2(x^*) \neq \emptyset$, 一定存在 $x_N \in S_2(x^*)$, 使得 $e^{-\frac{1}{r^2}[f(x_N)-f(x^*)+r]} > 1$, 即 $T(x_N, x^*, r) < 0$. 因为 $x_l \in F_k \cap S_1(x^*)$, 显然 $T(x_l, x^*, r) > 0$, 则有 $T(x_N, x^*, r) < T(x_l, x^*, r), \forall x_l \in F_k \cap S_1(x^*)$.

另外, 根据 $S_1(x^*)$ 和 $S_2(x^*)$ 的定义, x_l 和 x_N 均在可行域内, 则 $h(x_l) = h(x_N) = 0$, 所以 x_N 支配 x_l .

根据定理 4.6, 如果滤子集里只有一个元素, 且该元素支配 x^* , 则算法进入到 x^* 的低水平集里.

定理 4.7 设 x_k 处的滤子集为 $F_k = \{x_l \mid 1 \leq l \leq k\} \subseteq S_2(x^*)$, 进一步搜索得 $x^{**} \in L(P)$, 则 x^{**} 必能支配 F_k 中所有的点.

证 由 $f(x^{**}) < f(x_l) (\forall x_l \in F_k \cap S_2(x^*))$ 得

$$e^{-\frac{1}{r^2}[f(x^{**})-f(x^*)+r]} > e^{-\frac{1}{r^2}[f(x_l)-f(x^*)+r]}, \quad (4.6)$$

则当 r 充分小时, $T(x^{**}, x^*, r) < T(x_l, x^*, r)$ 必能成立, 并且 $h(x^{**}) = h(x_l) = 0$, 所以 x^{**} 必支配 x_l .

5 数值结果

本节主要验证基于梯度投影的广义滤子填充函数算法的有效性. 这些算例都是在 Matlab2014b 运行环境下运行, 处理器为 Intel(R) Core(TM) i5-2410M CPU @ 2.30GHZ 2.30GHZ, 系统类型为 64 位操作系统, 基于 x64 的处理器. 算法的参数设置如下: $s_1 = 2.5$, $s_2 = 1.2$, $\delta_1 = \delta_2 = \beta_1 = \beta_2 = \eta = 10^{-6}$, $r_0 = 1$, $r = 10^{-3}$, $G = 500$, $\delta = 10^{-3}$.

算法的数值结果在表 1-4 中列出, 其中各个符号定义如下.

k : 第 k 次求解得局部极小点; x_k^0 : 进行第 k 次局部极小点迭代的初始点; x_k^* : 第 k 个局部极小点; $f(x_k^*)$: 第 k 个局部极小值; F_k : 在第 k 次达到局部极小点时的滤子.

算例 5.1

$$\begin{aligned} \min \quad & f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2), \\ \text{s.t.} \quad & x_1 + x_2 \leq -2, \quad x_1 - 5x_2 \leq 3.5, \quad -3 \leq x_i \leq 2, \quad i = 1, 2. \end{aligned}$$

取在可行域外的初始点 $x_1^0 = (0.25397, 1.82675)^T$, 得到全局极小点

$$x^\infty = (-1.38766, -0.69384)^T,$$

全局极小值为 $f(x^\infty) = 0.42196$, 总运行时间为 8.513 秒.

算例 5.2

$$\begin{aligned} \min \quad & f(x) = \left(\sum_{i=1}^5 i \cos[(i+1)x_1 + i] \right) \left(\sum_{i=1}^5 i \cos[(i+1)x_2 + i] \right) \\ & + [(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2], \\ \text{s.t.} \quad & 10x_1 + 5x_2 \leq -10, \quad 5x_1 - 10x_2 \leq -10, \quad -10 \leq x_i \leq 10, \quad i = 1, 2. \end{aligned}$$

取在可行域外的初始点 $x_1^0 = (-2.37284, 4.58849)^T$, 得到全局极小点

$$x^\infty = (-7.70562, -0.80032)^T,$$

表 1: 算例 5.1

k	x_k^0	x_k^*	$f(x_k^*)$	F_k
1	$\begin{pmatrix} 0.25397 \\ 1.82675 \end{pmatrix}$	$\begin{pmatrix} -1.41238 \\ -0.98248 \end{pmatrix}$	1.60708	$\begin{pmatrix} -1.41238 \\ -0.98248 \end{pmatrix}$
2	$\begin{pmatrix} -1.39267 \\ -0.97853 \end{pmatrix}$	$\begin{pmatrix} -1.39687 \\ -0.97937 \end{pmatrix}$	1.56761	$\begin{pmatrix} -1.39687 \\ -0.97937 \end{pmatrix}$
3	$\begin{pmatrix} -1.39601 \\ -0.69780 \end{pmatrix}$	$\begin{pmatrix} -1.38766 \\ -0.69384 \end{pmatrix}$	0.42196	$\begin{pmatrix} -1.38766 \\ -0.69384 \end{pmatrix}$

表 2: 算例 5.2

k	x_k^0	x_k^*	$f(x_k^*)$	F_k
1	$\begin{pmatrix} -2.37284 \\ 4.58849 \end{pmatrix}$	$\begin{pmatrix} -6.47590 \\ -0.80032 \end{pmatrix}$	-98.05298	$\begin{pmatrix} -6.47590 \\ -0.80032 \end{pmatrix}$
2	$\begin{pmatrix} -7.71452 \\ -0.80032 \end{pmatrix}$	$\begin{pmatrix} -7.70562 \\ -0.80032 \end{pmatrix}$	-147.26943	$\begin{pmatrix} -7.70562 \\ -0.80032 \end{pmatrix}$

全局极小值为 $f(x^\infty) = -147.26943$, 总运行时间为 4.566 秒.

算例 5.3

$$\begin{aligned} \min \quad & f(x) = -25(x_1 - 2)^2 - (x_2 - 2)^2 - (x_3 - 1)^2 - (x_4 - 4)^2 - (x_5 - 1)^2 - (x_6 - 4)^2, \\ \text{s.t.} \quad & x_3 + x_4 \geq 4, \quad x_5 + x_6 \geq 4, \quad x_1 - 3x_2 \leq 2, \quad -x_1 + x_2 \leq 2, \quad x_1 + x_2 \leq 6, \quad x_1 + x_2 \geq 2, \\ & 0 \leq x_1 \leq 6, \quad 0 \leq x_2 \leq 8, \quad 1 \leq x_3 \leq 5, \quad 0 \leq x_4 \leq 6, \quad 1 \leq x_5 \leq 5, \quad 0 \leq x_6 \leq 10. \end{aligned}$$

取在可行域外的初始点 $x_1^0 = (3.28329, 0.83175, 0.89576, 1.54505, 5.04430, 1.52569)^\top$, 得到全

表 3: 算例 5.3

k	x_k^0	x_k^*	$f(x_k^*)$	F_k
1	$\begin{pmatrix} 3.28329 \\ 0.83175 \\ 0.89576 \\ 1.54505 \\ 5.04430 \\ 1.52569 \end{pmatrix}$	$\begin{pmatrix} 5.000000 \\ 1.00000 \\ 5.00000 \\ 0.00000 \\ 5.00000 \\ 0.00000 \end{pmatrix}$	-290	$\begin{pmatrix} 5.000000 \\ 1.00000 \\ 5.00000 \\ 0.00000 \\ 5.00000 \\ 0.00000 \end{pmatrix}$
2	$\begin{pmatrix} 5.000000 \\ 1.00000 \\ 5.00000 \\ 0.00000 \\ 5.00000 \\ 8.00186 \end{pmatrix}$	$\begin{pmatrix} 5.000000 \\ 1.00000 \\ 5.00000 \\ 0.00000 \\ 5.00000 \\ 10.00000 \end{pmatrix}$	-310	$\begin{pmatrix} 5.000000 \\ 1.00000 \\ 5.00000 \\ 0.00000 \\ 5.00000 \\ 0.00000 \end{pmatrix}$

局极小点 $x^\infty = (5, 1, 5, 0, 5, 10)^\top$, 全局极小值为 $f(x^\infty) = -310$, 总运行时间为 15.939 秒.

表 4: 算例 5.4

k	x_k^0	x_k^*	$f(x_k^*)$	F_k
1	$\begin{pmatrix} 0.9058, 0.1270, 0.9134, \\ 0.6324, 0.8147, 0.0975, \\ 0.2785, 0.5469, 0.9575, \\ 0.9649, 0.1576, 0.9706, \\ 0.9572, 0.4854, 0.8003, \\ 0.1419, 0.4218, 0.9157, \\ 0.7922, 0.9595 \end{pmatrix}^\top$	$\begin{pmatrix} 0.7238, 0.3449, 0.1552, \\ 0.3448, 0.2163, 0.2837, \\ 0.32481, 0.1752, 0.60404, \\ 0.5005, -0.0005, 0.5005, \\ 0.3287, 0.3483, 0.4141, \\ 0.0859, 0.4141, 0.62843, \\ 0.5663, 0.4099 \end{pmatrix}^\top$	3.21485	$\begin{pmatrix} 0.7238, 0.3449, 0.1552, \\ 0.3448, 0.2163, 0.2837, \\ 0.32481, 0.1752, 0.60404, \\ 0.5005, -0.0005, 0.5005, \\ 0.3287, 0.3483, 0.4141, \\ 0.0859, 0.4141, 0.62843, \\ 0.5663, 0.4099 \end{pmatrix}^\top$
12	$\begin{pmatrix} 0.4640, 0.0360, 0.4707, \\ 0.0302, 0.4698, 0.0305, \\ 0.4695, 0.0305, 0.4695, \\ 0.5426, -0.0426, 0.5426, \\ -0.0271, 0.5271, 0.2646, \\ 0.2354, 0.2646, 0.6577, \\ 0.3251, 0.1749 \end{pmatrix}^\top$	$\begin{pmatrix} 0.4640, 0.0360, 0.4707, \\ 0.0302, 0.4698, 0.0305, \\ 0.4695, 0.0305, 0.4695, \\ 0.5426, -0.0426, 0.5426, \\ -0.0271, 0.5271, 0.2646, \\ 0.2354, 0.2646, 0.6577, \\ 0.3251, 0.1749 \end{pmatrix}^\top$	2.45084	$\begin{pmatrix} 0.4640, 0.0360, 0.4707, \\ 0.0302, 0.4698, 0.0305, \\ 0.4695, 0.0305, 0.4695, \\ 0.5426, -0.0426, 0.5426, \\ -0.0271, 0.5271, 0.2646, \\ 0.2354, 0.2646, 0.6577, \\ 0.3251, 0.1749 \end{pmatrix}^\top$
24	$\begin{pmatrix} 0.4634, 0.0366, 0.4775, \\ 0.0227, 0.4773, 0.0232, \\ 0.4768, 0.0232, 0.4768, \\ 0.5377, -0.0377, 0.5377, \\ -0.0221, 0.5221, 0.2691, \\ 0.2309, 0.2691, 0.6398, \\ 0.3312, 0.1688 \end{pmatrix}^\top$	$\begin{pmatrix} 0.4634, 0.0366, 0.4779, \\ 0.0221, 0.4779, 0.0228, \\ 0.4772, 0.0228, 0.4772, \\ 0.5374, -0.0374, 0.5374, \\ -0.0218, 0.5218, 0.2693, \\ 0.2307, 0.2693, 0.6389, \\ 0.3315, 0.1685 \end{pmatrix}^\top$	2.43902	$\begin{pmatrix} 0.4634, 0.0366, 0.4779, \\ 0.0221, 0.4779, 0.0228, \\ 0.4772, 0.0228, 0.4772, \\ 0.5374, -0.0374, 0.5374, \\ -0.0218, 0.5218, 0.2693, \\ 0.2307, 0.2693, 0.6389, \\ 0.3315, 0.1685 \end{pmatrix}^\top$
36	$\begin{pmatrix} 0.4289, 0.0711, 0.4296, \\ 0.0704, 0.4296, 0.0704, \\ 0.4296, 0.0704, 0.4296, \\ 0.0704, 0.4296, 0.0704, \\ 0.4297, 0.0703, 0.4297, \\ 0.0703, 0.4297, 0.0703, \\ 0.4297, 0.0703 \end{pmatrix}^\top$	$\begin{pmatrix} 0.4289, 0.0711, 0.4295, \\ 0.0705, 0.4295, 0.0705, \\ 0.4295, 0.0705, 0.4295, \\ 0.0705, 0.4295, 0.0705, \\ 0.4298, 0.0702, 0.4298, \\ 0.0702, 0.4298, 0.0702, \\ 0.4298, 0.0702 \end{pmatrix}^\top$	0.55291	$\begin{pmatrix} 0.4289, 0.0711, 0.4295, \\ 0.0705, 0.4295, 0.0705, \\ 0.4295, 0.0705, 0.4295, \\ 0.0705, 0.4295, 0.0705, \\ 0.4298, 0.0702, 0.4298, \\ 0.0702, 0.4298, 0.0702, \\ 0.4298, 0.0702 \end{pmatrix}^\top$
43	$\begin{pmatrix} 0.4291, 0.0709, 0.4292, \\ 0.0708, 0.4292, 0.0708, \\ 0.4292, 0.0708, 0.4292, \\ 0.0708, 0.4292, 0.0708, \\ 0.4292, 0.0708, 0.4292, \\ 0.0708, 0.4292, 0.0708, \\ 0.4292, 0.0708 \end{pmatrix}^\top$	$\begin{pmatrix} 0.4291, 0.0709, 0.4291, \\ 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709, 0.4291, \\ 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709, 0.4291, \\ 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709 \end{pmatrix}^\top$	0.55285	$\begin{pmatrix} 0.4291, 0.0709, 0.4291, \\ 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709, 0.4291, \\ 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709, 0.4291, \\ 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709 \end{pmatrix}^\top$

算例 5.4

$$\min f(x) = \sum_{i=1}^{20} \left[x_i^2 - \frac{1}{10} \cos(5\pi x_i) \right],$$

$$\text{s.t. } x_i + x_{i+1} \geq 0.5, i = 1, 2, \dots, 19, \quad -1 \leq x_i \leq 1, i = 1, 2, \dots, 20.$$

取在可行域外的初始点

$$x_1^0 = (0.9058, 0.1270, 0.9134, 0.6324, 0.8147, 0.0975, 0.2785, 0.5469, 0.9575, 0.9649, \\ 0.1576, 0.9706, 0.9572, 0.4854, 0.8003, 0.1419, 0.4218, 0.9157, 0.7922, 0.9595)^\top,$$

得到全局极小点

$$x^\infty = (0.4291, 0.0709, 0.4291, 0.0709, 0.4291, 0.0709, 0.4291, 0.0709, 0.4291, 0.0709, \\ 0.4291, 0.0709, 0.4291, 0.0709, 0.4291, 0.0709, 0.4291, 0.0709, 0.4291, 0.0709)^\top,$$

全局极小值为 $f(x^\infty) = 0.55285$, 总运行时间为 409.516 秒. 由于迭代得的局部极小点比较多, 在本文中只列出部分迭代结果.

以上的数值结果说明了基于梯度投影的广义滤子填充函数算法的有效性. 算例 5.1 与算例 5.2 的最优解在内部, 其中算例 5.1 首先搜索到边界上的 KKT 点, 再通过填充函数迭代到低水平集, 并最终找到了目标函数的稳定点. 算例 5.2 首先在可行域内找到稳定点, 再通过两阶段迭代找到最优解. 算例 5.3 的最优解在边界上, 其搜索情况与算例 5.1 类似. 算例 5.4 的情况较为复杂, 首先从可行域外搜索到了 KKT 点, 然后在多次迭代后达到最优解. 前三个算例由于规模较小, 耗时均较短. 而算例 5.4 的维数较高, 程序的总运行时间相对长了很多.

参 考 文 献

- [1] Ge R P. A filled function method for finding a global minimizer of a function of several variables [J]. *Math. Prog.*, 1990, 46(1): 191–204.
- [2] Zhang L S, Ng C K, Li D, Tian W W. A new filled function method for global optimization [J]. *J. Glob. Optim.*, 2004, 28(1): 17–43.
- [3] Yang Y J, Shang Y L. A new filled function method for unconstrained global optimization [J]. *Appl. Math. Comput.*, 2005, 173(1): 501–512.
- [4] Liang Y M, Zhang L S, Li M M, Han B S. A filled function method for global optimization [J]. *J. Comput. Appl. Math.*, 2006, 205(1): 16–31.
- [5] Wang W, Zhang X S, Li M. A filled function method dominated by filter for nonlinearly global optimization [J]. *J. Appl. Math.*, 2015, 8(3): 8–18.
- [6] Wu C Z, Kok L T, Volker R. A filled function method for optimal discrete-valued control problems [J]. *Glob. Optim.*, 2009, 44(2): 213–225.
- [7] Fletcher R, Leyffer S. Nonlinear programming without a penalty function [J]. *Math. Prog.*, 2002, 91(2): 239–269.
- [8] Fletcher R, Leyffer S, Toint P L. On the global convergence of a filter-SQP algorithm [J]. *Siam J. Optim.*, 2006, 13(1): 44–59.
- [9] Michael U, Stefan U, Luis N V. A globally convergent primal-dual interior-point filter method for nonlinear programming [J]. *Math. Prog.*, 2004, 100(2): 379–410.
- [10] Andreas W, Lorenz T B. Line search filter methods for nonlinear programming: local convergence [J]. *SIAM J. Optim.*, 2005, 16(2): 1–31.
- [11] 胡铨, 王薇. 求解带箱式约束全局优化问题的滤子填充函数方法 [J]. *运筹学学报*, 2016, 03: 57–67.
- [12] Rosen J B. The gradient projection method for nonlinear programming, Part I. linear constraints [J]. *SIAM*, 1960, 8(1): 181–217.
- [13] Zhang X S. On the convergence of Rosen’s gradient projection method: three-dimension case [J]. *Acta Math. Appl. Sin., Engl. Ser.*, 1987, 3(3): 280–288.
- [14] Du D. Remarks on the convergence of Rosen’s gradient projection method [J]. *Acta Math. Appl. Sin., Engl. Ser.*, 1987, 3(3): 270–279.
- [15] Wang W, Hua S L, Tang J J. A generalized gradient projection filter algorithm for inequality constrained optimization [J]. *J. Appl. Math.*, 2013, 2013(2): 4819–4828.
- [16] Gao J, Wang W. A generalized gradient projection filter method for arbitrary initial point [J]. *Oper. Res. Trans.*, 2013, 17(2): 124–130.

A GENERALIZED FILTER FILLED FUNCTION METHOD BASED ON GRADIENT PROJECTION

ZHANG Hui-wen¹, WANG Wei¹, LI Min¹, XU Yi-fan²

(1. *Department of Mathematics, East China University of Science and Technology,
Shanghai 200237, China*)

(2. *School of Management, Fudan University, Shanghai 200433, China*)

Abstract: In this paper, non-convex global optimization problems with constraints are studied. By using the structures of filter and filled function, a generalized filter filled function algorithm based on gradient projection is presented and the theoretical properties and numerical results are obtained. The algorithm modifies the definition of the filled function and the application scope of the filter technique, which extends the local optimization technique and makes it one of effective methods to solve the global optimization problems with constraints.

Keywords: non-convex global optimization; constraint function; filled function; three-dimensional filter

2010 MR Subject Classification: 90C30; 65K05