基于演化策略的矩阵流形黑盒优化方法

何笑雨[1)] 周育人[1)] 陈泽丰[2)]

[1)](中山大学数据科学与计算机学院 广州 510006)

[2)](南洋理工大学计算机科学与工程学院 新加坡 639798 新加坡)

摘 要 该文使用演化算法求解一类定义在黎曼流形之上的矩阵优化问题. 传统演化算法大多针对欧式空间中的优化问题而设计，因此难以直接用于矩阵流形优化. 该文根据矩阵流形的几何结构，将原始的流形约束问题转换为一系列切空间中的无约束问题，并通过将经典的协方差矩阵适应技术从欧式空间扩展到黎曼流形，提出了用于求解矩阵流形黑盒优化问题的流形搜索方向适应演化策略. 所提出的算法在矩阵流形的每个切空间中使用多元高斯分布引导搜索，并不断更新概率分布从而提高产生优质解的似然性. 该文设计了在切空间中仅使用单位阵和少量搜索方向构造大规模高斯分布的概率建模方法，并提出了在矩阵流形的不同切空间中更新和移动概率模型的方法. 所提出的算法不依赖于全局坐标系、不需要额外的编码方案、独立于特定坐标基的选择，所有演化算子均可表示为矩阵线性运算,因而能够达到较高的运行效率. 该文在三类经典的矩阵流形优化问题上进行了仿真实验. 数值结果表明，在黑盒优化场景中，所提出算法的性能显著优于或相当于主流的流形优化器.

关键词 演化策略；矩阵流形；搜索方向适应；黑盒优化；协方差矩阵适应

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2020.01604

An Evolution Strategy for Black-box Optimization on Matrix Manifold

HE Xiao-Yu[1)] ZHOU Yu-Ren[1)] CHEN Ze-Feng[2)]

[1)] (*School of Data and Computer Science, Sun Yat-sen University, Guangzhou* 510006)

[2)] (*School of Computer Science and Engineering, Nanyang Technological University, Singapore* 639798 *Singapore*)

Abstract This work concerns the black-box approach towards a class of matrix optimization problems defined on Riemannian manifolds. Due to their non-Euclidean nature, these problems are intractable for traditional evolutionary algorithms. In this work, we transform, according to the manifold structure, the originally manifold-constrained problem into a sequence of unconstrained ones in the tangent spaces, and propose a manifold search direction adaptation evolution strategy by extending the covariance matrix adaptation mechanism from Euclidean space to matrix manifolds. The proposed algorithm maintains, in each tangent space, a multi-variate Gaussian distribution to guide the search and iteratively adapts it in order to increase the likelihood of reproducing high-quality solutions. This work provides the method of building the high-dimensional probability model with an identity matrix and only a few tangent search directions. The method of moving probability models between different tangent spaces is also discussed. The proposed algorithm is computationally efficient as all its genetic operators are performed using linear matrix transformations, independent of global coordinates, additional encoding schemes, or specifications of orthogonal bases. We verify the performance of the proposed algorithm on three matrix manifold optimization problems and the results show that it is superior to or competitive with other state-of-the-artsin black-box settings.

收稿日期：2020-01-01；在线发布日期：2020-02-07. 本课题得到国家自然科学基金（61773410, 61673403）和中国博士后科学基金(2019M663234)资助. **何笑雨**，博士，主要研究领域为演化算法，hxyokokok@foxmail.com。**周育人**（通信作者），博士，教授，主要研究领域为演化算法，zhouyuren@mail.sysu.edu.cn。**陈泽丰**，博士，主要研究领域为演化算法，

Keywords　evolution strategy; matrix manifold; search direction adaptation; black-box optimization; covariance matrix adaptation

1　引　言

矩阵流形优化又称黎曼优化，主要关注于定义在黎曼流形上的矩阵优化问题. 这类问题可定义为

$$\min_{\boldsymbol{X}\in\mathcal{M}} f(\boldsymbol{X}),$$

其中，$\boldsymbol{X}$ 为矩阵形式的决策变量，f 为目标函数，$\mathcal{M}$ 为黎曼流形. 通常，$\mathcal{M}$ 不是欧式空间甚至不是线性空间，但是在任意一点 $\boldsymbol{X}$ 处都可以局部线性化为一个切空间，记为 $T_{\boldsymbol{X}}\mathcal{M}$. 在这一切空间中，可以定义关于 $\boldsymbol{X}$ 的内积运算 $\langle\cdot,\cdot\rangle_{\boldsymbol{X}}$，因此，$T_{\boldsymbol{X}}\mathcal{M}$ 为欧式空间. 如上定义的问题广泛地存在于数据挖掘[1−3]、数值代数[4,5]、信号处理[6,7]、电子设计[8]等领域. 例如，在使用谱聚类[9]进行无监督学习任务时，需要找到一组高维数据在低维子空间中的嵌入表示，该任务可归结为寻找使得 $tr(\boldsymbol{U}^T\boldsymbol{L}\boldsymbol{U})$ 最大化的正交矩阵 $\boldsymbol{U}$ 的优化问题，其中 $\boldsymbol{L}$ 为数据点相似图的拉普拉斯矩阵，$tr(\cdot)$ 表示矩阵的迹. 由于决策变量 $\boldsymbol{U}$ 需要满足正交约束，因此其可行域不是线性空间，而是欧式空间中的子流形.

虽然矩阵流形优化问题在现实中有着广泛的应用，但传统数值优化方法在求解这类问题时，通常面临着两方面困难：一是在优化过程中保持流形约束非常困难. 传统的流形约束处理方法，例如使用罚函数、将流形参数化等，依赖于特定参数的选择或问题本身的结构；二是搜索区域的非凸性使得问题具有许多局部最优解，在大多数情形下传统算法无法保证得到全局最优解.

早在 18 世纪关于微分流形的研究中，研究者就已经观察到矩阵流形具有的特殊几何结构可以用于构造流形优化算法. 然而直到 2000 年以来，由于这类问题在学术研究和工程实践中的广泛存在，相关课题才在数值优化领域被系统性的提出. 目前，流形优化算法主要基于两类框架：

（1）基于测地线的框架. Edelman 等人[10]最早在数值优化的范畴内研究了矩阵流形的几何结构，提出了矩阵流形优化的通用框架. 其基本思想是沿着测地线（Geodesic）方向[11]迭代下降. 测地线是直线在曲面中的扩展，其计算过程通常借助于微分流形的相关工具和技术. 对于某些常用流形，测地线有解析式，因此所需计算量较少. 此外，也可以使用伪测地线[12]代替测地线，进一步提升算法运行效率而不影响算法性能. 然而，对于那些测地线不存在解析式的流形，测地线依然需要通过微分方程求解，因而运行效率低下，只能用于小规模问题.

（2）基于回撤的框架. Manton[13]在微分几何的范畴之外重新设计了流形优化方法. 其算法迭代地求解形如 $f(\pi(\boldsymbol{X}))$ 的目标函数，其中 $\pi:T_{\boldsymbol{X}}\mathcal{M}\to\mathcal{M}$ 为测地线的近似算子，因而将流形约束问题转换为了切空间中的无约束问题. 近几年来，学者们提出了针对不同流形的不同近似算子[14−17]，并将它们统称为回撤（Retraction）. 一个典型的回撤算子通常包含了在切空间中移动和投影回流形的两个基本步骤，其计算过程不依赖测地线，因而效率较高. 此外，常见流形的回撤算子大多可以通过线性代数计算，因而适用面比基于测地线的方法更广.

目前常用的流形优化算法大多在上述框架中使用最速下降、共轭梯度、牛顿法等经典方法[5,18−21]产生解. 文献[14,22,23]对这些矩阵流形上的经典算法进行了细致地综述. 由于矩阵流形局部同胚于欧式空间，因此这类算法大多能保证局部收敛性. 同时，这些算法也存在一定缺陷：一方面，由于流形的非凸性，这些算法难以保证全局收敛性；另一方面，越来越多的现实问题被建模成非光滑或不可导的问题，因而经典方法往往难以直接应用. 这些缺陷的存在对矩阵流形问题的黑盒优化方法提出了迫切需求.

演化算法（Evolutionary Algorithm, EA）[24]是黑盒优化的经典方法. EA 不依赖于问题本身特性，具有较强的全局搜索能力，因此可以期望 EA 能在矩阵流形优化中有较好的表现. 然而，现阶段的演化算法大多针对欧式空间中的优化问题而设计，当用于矩阵流形优化时，将导致以下两个方面问题：

（1）大多数演化算法运行于向量空间而非矩阵空间，从而造成了演化流形优化的第一个核心困难：如何对矩阵进行有效编码. 这一困难的本质在于矩阵流形往往不能占据整个矩阵线性空间，因而，算法在表示矩阵或某个搜索方向时所需的自由度小于矩阵大小. 显然，若直接把矩阵编码为向量，则会引入冗余变量，造成计算资源的浪费；将矩阵参数化虽然可以避免这一问题，但往往依赖于问题本身特性，且计算量较大.

（2）目前常用的演化算子大多依赖于决策空间的欧氏几何结构，引入了演化流形优化的第二个核

心困难：如何设计演化算子. 例如，差分演化（Differential Evolution, DE）[25]中的变异算子、演化策略（Evolution Strategy, ES）[26]中的重组算子在本质上都是线性变换，因而需要决策空间为线性空间. 再比如，DE 中的交叉算子以及遗传算法（Genetic Algorithm, GA）[27]中几乎所有的交叉和变异算子都隐含地使用到了坐标的概念，因此这些算子必须在欧式空间中才能运行. 此外，协方差矩阵适应演化策略（Covariance Matrix Adaptation Evolution Strategy, CMA-ES）[28]中协方差矩阵的更新以及粒子群优化（Particle Swarm Optimization, PSO）[29]中粒子速度的更新，不仅需要欧式空间，还要求该欧式空间在算法的整个运行过程中具有一个全局坐标系从而保证更新的有效性.

目前，演化算法领域的研究者已对上述两个问题进行了初步研究. 文献[30]在 PSO 的速度更新机制中引入了黎曼梯度，提升算法在求解低秩张量近似问题时的性能. 文献[6,31]则对标准 PSO 进行了简单的扩展. 在所提出的算法中，粒子的速度位于该粒子所决定的切空间中，每个粒子沿着相应的测地线方向前进. 文献[31]将 CMA-ES 推广至三维球面空间，在球面上沿测地线方向移动候选解及相应的概率分布，求解核磁共振成像中的形状重建问题. 该文献还在球面空间上实现了 DE 用作对比算法，其使用对数映射将可行解映射至切空间，在切空间中使用 DE 算子产生候选解后用指数映射投影回球面. 文献[32]将文献[31]中的变异强度适应机制与标准 ES 相结合，并在球面空间中分析了算法收敛性能. 文献[33]提出了变异强度自适应的标准 ES，用于在三维欧式空间与三维球面空间组成的积流形上求解室内定位问题.

上述研究成果虽然在黎曼流形的黑盒优化任务上表现出了可观的应用前景，但仍然存在一些可改进的空间. 例如，文献[6,30,31]中的 PSO 改进算法在移动粒子后，需要将粒子投影回切空间以更新对应的速度向量. 这一过程需要计算测地线的逆映射，因而只在少数几类流形中可以实现，算法欠缺通用性. 文献[31]将 CMA-ES 中的协方差矩阵视为黎曼流形上的双线性映射，在移动概率分布时需要进行矩阵分解，因此并未在本质上解决概率分布需要额外编码的问题. 文献[31]中的改进 DE 算法还需要在切空间中进行显式的正交化过程，计算效率较低.

本文立足于解决现有演化算法在求解矩阵流形优化问题时存在的困境，在基于回撤的算法框架内，提出了流形搜索方向适应演化策略（Manifold Search Direction Adaptation Evolution Strategy, MSDA-ES）. 该算法有以下主要特性和创新之处：

（1）把经典的 CMA-ES 算法从欧式空间推广到黎曼流形之中，设计了在黎曼流形上维护和更新多元高斯分布的方法.

（2）提出了先在切空间产生候选矩阵，后映射至流形进行评估的采样方案，所有解可使用矩阵直接表示，而演化操作则通过线性代数完成，算法不再需要额外的编码策略.

（3）提出了使用少量切向量构造和更新协方差矩阵的方法，算法运行过程与特定的坐标系无关，不依赖于全局坐标系，也不需要显式地构造局部坐标系，运行效率较高.

本文的第二章首先介绍矩阵流形的基础知识；随后，在第三章给出 MSDA-ES 算法的具体实现；第四章在一组测试问题中检验所提算法的有效性；最后，在第五章总结全文并给出未来的研究方向.

2 研究基础

本节介绍矩阵流形优化问题的几个具体例子，同时，给出了流形几何结构相关的结论和方法，这些基础知识将在下一章中用于算法设计.

2.1 三类常见的矩阵流形优化问题

本研究考虑自变量为 $n\times p$ 矩阵的黎曼流形优化问题，所涉及的流形结构包括 Stiefel 流形、Grassmann 流形以及 Oblique 流形. 需要注意的是，本研究所提出的流形演化策略采用通用的流形优化框架，因此其适用范围不局限于这三类流形.

（1）Stiefel 流形. 首先关注以下定义在 $n\times p$ 正交矩阵集合上的函数 $f(\boldsymbol{X})$ 最小化问题：

$$\min_{\boldsymbol{X}\in R^{n\times p}} f(\boldsymbol{X})$$
$$\text{s.t. } \boldsymbol{X}^T\boldsymbol{X}=\boldsymbol{I}_p,$$

其中，$\boldsymbol{I}_p$ 表示 $p\times p$ 单位阵. 这一问题的可行域称为 Stiefel 流形[34]，记作 $St(n,p)$.

（2）Grassmann 流形. 本文要研究的第二个问题非常类似于上述第一个问题，其主要区别是，该问题的目标函数值表现出旋转不变性. 该问题可通过以下方式定义：

$$\min_{\boldsymbol{X}\in R^{n\times p}} f(\boldsymbol{X})=f(\boldsymbol{X}\boldsymbol{Q})$$
$$\text{s.t. } \boldsymbol{X}^T\boldsymbol{X}=\boldsymbol{I}_p,$$

其中，$\boldsymbol{Q}$ 为任意的 $p\times p$ 正交矩阵. 以上定义可知，目标函数 f 的值仅依赖于 $\boldsymbol{X}$ 的列空间，而和基的选取无关. 这一问题的可行域称为 Grassmann 流形，

记为 $Gr(n,p)$，该问题与子空间追踪等多种实际应用有紧密的联系[14].

（3）Oblique 流形. 本章关注的最后一个例子具有以下形式

$$\min_{\boldsymbol{X}\in R^{n\times p}} f(\boldsymbol{X})$$
$$\text{s.t. } diag(\boldsymbol{X}^T\boldsymbol{X}) = \boldsymbol{I}_p,$$

其中，符号 $diag(\boldsymbol{Y})$ 表示将方阵 $\boldsymbol{Y}$ 中所有非对角线元素设为 0 时所得的对角阵. 这一问题的可行域常常被称为 Oblique 流形，记作 $Ob(n,p)$. 许多二值整数规划问题经过松弛后都可以转化为此类问题[12].

2.2　矩阵流形的基础知识

本节简要介绍有关上述三类流形的常用符号和结论. 需要注意的是，本节中出现的所有图例仅考虑了 Stiefel 流形. 相对而言，Grassmann 流形和 Oblique 流形更加抽象，无法使用图片展示其几何结构. 但本节中给出的维度计算方法、切空间投影、回撤算子等内容对上述三类流形均有效. 至于 Grassmann 流形和 Oblique 流形的具体性质，读者可在文献[10,14]中找到.

2.2.1　切空间与法空间

对于流形 $\mathcal{M}$ 上的任意一点 $\boldsymbol{X}$，都存在一线性空间以 $\boldsymbol{X}$ 为原点、与 $\mathcal{M}$ 相切于 $\boldsymbol{X}$. 这一空间称为关于 $\boldsymbol{X}$ 的切空间，记为 $T_{\boldsymbol{X}}\mathcal{M}$. 本研究假设 $\mathcal{M}$ 为黎曼流形，因此切空间同时为欧式空间. 将切空间 $T_{\boldsymbol{X}}\mathcal{M}$ 的正交补称为法空间，记为 $N_{\boldsymbol{X}}\mathcal{M}$. 图 1 给出了 $St(3,1)$ 流形的切空间和法空间的例子. 该流形为嵌入在 3 维空间中的一个球面，其任意一点对应的切空间为球的 2 维切平面，而相应的法空间为经过这一点的法线.

2.2.2　流形的维度

在演化算法中，所涉及的诸多参数往往和可行域的维度有关. 因此，在进一步设计流形演化算法前，需要了解所求流形优化问题的维度. 一个流形的维度等于该流形所对应切空间的维度，亦即参数化切空间所需自由元素的个数. 对于上述三类流形，其维度计算方式列于表 1 中.

2.2.3　切空间投影与回撤运算

对于矩阵流形 $\mathcal{M}$ 上的点 $\boldsymbol{X}$，可以定义从矩阵空间到切空间的正交投影 $P_{\boldsymbol{X}}: R^{n\times p}\to T_{\boldsymbol{X}}\mathcal{M}$. 为了方便起见，在投影时将 $\boldsymbol{X}$ 视为矩阵空间 $R^{n\times m}$ 的原点. 对于上述三类矩阵流形，表 1 给出了相应的切空间投影计算方法.

同样，对于切空间中的任意一点，都可以定义一个将该点映射回流形的变换，称为回撤（Retraction），记为 $R_{\boldsymbol{X}}: T_{\boldsymbol{X}}\mathcal{M}\to\mathcal{M}$. 回撤算子可定义为满足如下两个条件的连续可微映射[15]：（1）对任意的 $\boldsymbol{X}\in\mathcal{M}$ 满足 $R_{\boldsymbol{X}}(\boldsymbol{0}) = \boldsymbol{X}$，其中 $\boldsymbol{0}$ 表示 $T_{\boldsymbol{X}}\mathcal{M}$ 的原点；（2）对任意的 $\boldsymbol{H}\in T_{\boldsymbol{X}}\mathcal{M}$ 满足 $\frac{\mathrm{d}}{\mathrm{d}t}R_{\boldsymbol{X}}(t\boldsymbol{H})\big|_{t=0} = \boldsymbol{H}$. 理论上，测地线也可视为一个回撤算子，但通常认为回撤算子是测试线函数的一个近似. 此外，对于同一类流形可以定义多种不同的回撤算子，本章使用文献[14,35]描述的定义方法，其具体的表达式在表 1 中给出.

图 2 给出了一个 2 维流形中切空间投影与回撤算子的示意图. 图中，带箭头直线表示切空间投影 $P_{\boldsymbol{X}}$，该投影将位于 $R^{n\times p}$ 中的点 $\boldsymbol{Y}$ 映射至 $T_{\boldsymbol{X}}\mathcal{M}$. 带箭头曲线表示回撤算子 $R_{\boldsymbol{X}}$，其将 $T_{\boldsymbol{X}}\mathcal{M}$ 中的点 $\boldsymbol{Z}$ 映射至 $\mathcal{M}$.

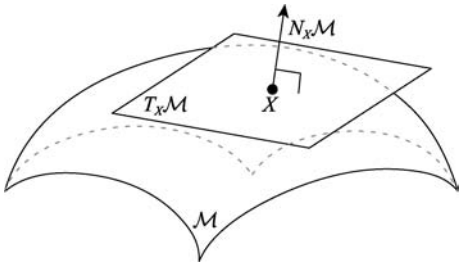


图 1　$St(3,1)$ 流形及相应的切空间和法空间

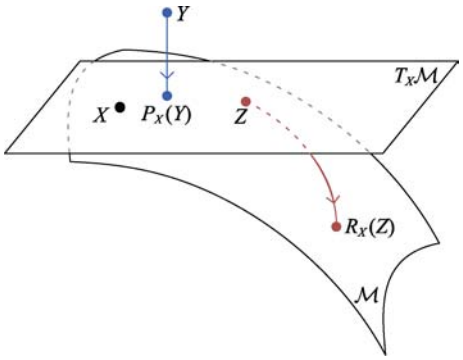


图 2　切空间投影与回撤算子

2.2.4　CMA-ES 及其局限性

CMA-ES 是求解连续型黑盒优化问题性能最好的演化算法之一. CMA-ES 维持了一个多元高斯分布引导搜索，并且通过学习优质解的变量间依赖不断更新协方差矩阵，从而表现出对决策空间旋转变换的性能不变性. 由于使用了演化路径技术对群体的成功变异方向进行累积，只需要较小的群体规模即可实现较快的协方差矩阵更新速度. CMA-ES 还采用了累积步长控制技术（Cumulative Step-size Adaptation，CSA），使得变异强度的适应机制独立

于特定坐标系. 此外，由于 CMA-ES 算法的环境选择过程仅依赖于目标函数值的比较结果，因此算法具有保序变换不变性. 这些优良特性使得 CMA-ES 在许多现实中有着广泛的应用[36–38].

表 1 三类矩阵流形的基本性质

$\mathcal{M}$	特性
$St(n,p)$	$dim\mathcal{M}=np-\frac{1}{2}(p+1)p$ $P_{\boldsymbol{X}}(\boldsymbol{Y})=\boldsymbol{Y}-\frac{1}{2}\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{Y}+\boldsymbol{Y}^T\boldsymbol{X})$ $R_{\boldsymbol{X}}(\boldsymbol{Z})=\boldsymbol{qr}(\boldsymbol{X}+\boldsymbol{Z})$
$Gr(n,p)$	$dim\mathcal{M}=(n-p)p$ $P_{\boldsymbol{X}}(\boldsymbol{Y})=\boldsymbol{Y}-\boldsymbol{X}\boldsymbol{X}^T\boldsymbol{Y}$ $R_{\boldsymbol{X}}(\boldsymbol{Z})=\boldsymbol{U}\boldsymbol{V}^T$，其中 $\boldsymbol{U}$ 和 $\boldsymbol{V}$ 分别为矩阵 $\boldsymbol{X}+\boldsymbol{Z}$ 的 SVD 分解中的左奇异矩阵和右奇异矩阵
$Ob(n,p)$	$dim\mathcal{M}=(n-1)p$ $P_{\boldsymbol{X}}(\boldsymbol{Y})=(\hat{\boldsymbol{Y}}_{:,1},\cdots,\hat{\boldsymbol{Y}}_{:,p})$，其 $\hat{\boldsymbol{Y}}_{:,i}=\boldsymbol{Y}_{:,i}-\boldsymbol{X}_{:,i}(\boldsymbol{X}_{:,i}^T\boldsymbol{Y}_{:,i})$ $R_{\boldsymbol{X}}(\boldsymbol{Z})=(\hat{\boldsymbol{Z}}_{:,1},\cdots,\hat{\boldsymbol{Z}}_{:,p})$，其中 $\hat{Z}_{:,i}=\frac{\boldsymbol{X}_{:,i}+\boldsymbol{Z}_{:,i}}{\lvert\boldsymbol{X}_{:,i}+\boldsymbol{Z}_{:,i}\rvert_2}$

注：（1）$\boldsymbol{qr}(\cdot)$ 表示 QR 分解中得到的正交矩阵；（2）表中使用的 QR 分解和 SVD 分解均为精简型（economy-sized）分解；（3）$|\cdot|_2$ 表示向量的 L_2 范数；（4）$\boldsymbol{M}_{:,i}$ 表示矩阵 $\boldsymbol{M}$ 中的第 i 列.

如前文所述，CMA-ES 存在一个显著的局限性，即其只能运行于欧式空间. 例如，算法维持的协方差矩阵仅在欧式空间中才有定义，而协方差矩阵的更新还依赖于全局坐标系. 对于流形优化问题，一方面，决策变量间的线性关系仅在切空间中的局部区域有定义，另一方面，可行域并不存在一个全局坐标系. 这些性质使得 CMA-ES 无法直接应用于矩阵流形优化. 文献[31]通过在流形上平行移动双线性映射实现了协方差矩阵的更新，通过对切向量的平行移动实现了演化路径的更新. 然而，该方法至少在两方面存在可改进的空间. 首先，双线性映射的平行移动需要对矩阵进行分解，隐含地引入了对流形参数化的过程，因此运行效率较低，无法应用于大规模环境；其次，将演化路径累积机制直接推广至黎曼流形后的几何意义不明确，事实上，文献[32]已经从理论上证明，在黎曼流形上直接使用 CSA 技术无法确保算法收敛.

3 流形搜索方向适应演化策略

3.1 基本思想

MSDA-ES 是 CMA-ES 在矩阵流形中的推广. 它使用标准的 (μ,λ)-ES 算法框架，并利用一个多元高斯分布引导群体搜索. 在每一次迭代中，算法在某一切空间中产生出 λ 个解，利用回撤算子将解映射至流形从而完成函数评估，最后，使用其中最好的 μ 个解更新搜索分布.

假设矩阵流形 $\mathcal{M}$ 的维度为 d，即 $d=dim\mathcal{M}$. MSDA-ES 在其第 g 代运行中维持了以下策略参数：

- $\boldsymbol{X}^{(g)}\in\mathcal{M}$：矩阵流形上的一个可行解；
- $\sigma^{(g)}\in R_+$：变异强度；
- $\boldsymbol{C}^{(g)}\in R^{d\times d}$：协方差矩阵.

上述协方差矩阵 $\boldsymbol{C}^{(g)}$ 用于描述切空间 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中决策变量间的线性依赖. 使用这些参数，构造如下的多元高斯分布：

$$(\sigma^{(g)})^2\mathcal{N}(\boldsymbol{0}_d,\boldsymbol{C}^{(g)}),$$

其中，$\boldsymbol{0}_d$ 表示 d 维 $\boldsymbol{0}$ 向量. 由于 $\boldsymbol{X}^{(g)}$ 是切空间 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 的原点，因此服从上述分布的随机变量自然地分布于 $\boldsymbol{X}^{(g)}$ 周围. 因此，虽然在上述分布中并没有指明均值，但 $\boldsymbol{X}^{(g)}$ 实际上起到了均值的作用.

在第 g 代中，MSDA-ES 的工作流程如下：

1. 从高斯分布 $(\sigma^{(g)})^2\mathcal{N}(\boldsymbol{0}_d,\boldsymbol{C}^{(g)})$ 中采样出 λ 个位于 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中的候选矩阵 $\boldsymbol{Y}_1,\cdots,\boldsymbol{Y}_\lambda$.

2. 对候选矩阵排序，使得 $f(R_X(\boldsymbol{Y}_{1:\lambda}))\leqslant\cdots\leqslant f(R_{\boldsymbol{X}}(\boldsymbol{Y}_{\lambda:\lambda}))$. 其中，$\boldsymbol{Y}_{i:\lambda}$ 表示排名第 i 的候选矩阵.

3. 对 μ 个最好的候选矩阵重组以得到新的矩阵 $\overline{\boldsymbol{Y}}$. 将 $\overline{\boldsymbol{Y}}$ 使用回撤算子映射至流形作为下一代群体均值 $\boldsymbol{X}^{(g+1)}$.

4. 使用 $\overline{Y}$ 更新搜索分布.

5. 将分布移动至新的切空间 $T_{\boldsymbol{X}^{(g+1)}}\mathcal{M}$，同时计算 $\sigma^{(g+1)}$ 和 $\boldsymbol{C}^{(g+1)}$.

上述步骤在以下几个方面区别于标准 CMA- ES：

- CMA-ES 在整个决策空间中维护概率分布，产生的候选解也分布于整个决策空间，而在 MSDA-ES 中，概率分布的维护以及候选解的采样均在切空间中完成. 其原因在于流形与切空间是局部同胚的，因此在法空间中进行搜索只会浪费计算资源.
- CMA-ES 产生候选解后可直接进行函数评估，而 MSDA-ES 采样得到的候选矩阵并非可行解，因此需要在函数评估前使用回撤算子映射回流形（见上述第二步）.
- CMA-ES 在一个固定的全局坐标系中维护其概率分布，而在 MSDA-ES 中，群体的演化过程伴随着切空间的变化，因此，其概率分布在不同代可能位于不同的切空间中. 对于均值 $\boldsymbol{X}^{(g+1)}$，需要使用回撤算子将其变换至矩阵流形以确保其可行性（见上述第三步）. 对于协

方差矩阵以及变异强度，还需要将其移动至新的切空间（见上述第五步）.

3.2　编码策略

对矩阵进行有效的编码是使用传统演化算法求解流形优化问题时面临的主要困难. 本节介绍MSDA-ES 中对解和概率分布的表示方法,可以有效克服上述困难.

3.2.1　协方差矩阵建模

标准 CMA-ES 使用一个规模为 $d\times d$ 的协方差矩阵存储所有自变量之间的两两线性关联，因此具有 $O(d^2)$ 的时间和空间复杂度. 由于在矩阵流形优化中，$d=O(np)$ 常常为一个较大的值，因此上述表示协方差矩阵的方法不适用于矩阵流形优化.

为了降低存储和更新协方差矩阵的复杂度，MSDA-ES 中的协方差矩阵具有以下形式：

$$\boldsymbol{C}^{(g)}=(1-c_{cov})\boldsymbol{I}_d+c_{cov}\sum_{i=1}^{m}\boldsymbol{q}_i^{(g)}(\boldsymbol{q}_i^{(g)})^T \tag{1}$$

其中，$\boldsymbol{q}_1^{(g)},\cdots,\boldsymbol{q}_m^{(g)}\in R^d$ 为 m 个向量，c_{cov} 为权系数. 在算法中，向量 $\boldsymbol{q}_i^{(g)}$ 描述了可能产生高质量解的方向，因此将其称为搜索方向.

公式(1)使用 $\sum_{i=1}^{m}\boldsymbol{q}_i^{(g)}(\boldsymbol{q}_i^{(g)})^T$ 估计协方差矩阵，当 $m\ll d$ 时,维护协方差矩阵所需复杂度可以显著降低，从而在性能和效率之间达到折衷. 同时，由于该估计量的秩不超过 m，为了保证协方差矩阵的正定性，将其与单位阵进行线性组合，并使用 c_{cov} 作为组合系数. 该线性组合使得公式(1)具有 Shrinkage 估计器[39]的形式,因此在保持原有估计量特征向量不变的情况下，改善了其正定性，有助于描述高维空间中的变量依赖关系. 需要注意的是，上述模型只用于说明解的采样原理，模型本身并不真正的出现在算法实现中.

3.2.2　采样方法

除了具有较低的算法复杂度，使用公式(1)对协方差矩阵建模还有另外一个优点，即采样过程与坐标系无关.

假设 $\{\boldsymbol{E}_i\in R^{n\times p}\mid i=1,\cdots,d\}$ 为 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 的一组基. 为了从该组基所确定的坐标系中产生出一个候选矩阵，可首先对 $\mathcal{N}(0,\boldsymbol{C}^{(g)})$ 进行概率采样：

$$\boldsymbol{y}=\sqrt{1-c_{cov}}\boldsymbol{z}_0+\sqrt{c_{cov}}\sum_{i=1}^{m}\boldsymbol{q}_i^{(g)}z_i.$$

该过程产生的向量中包含了候选矩阵的坐标值，其中，$\boldsymbol{z}_0\in R^d$ 为服从标准高斯分布 $\mathcal{N}(\boldsymbol{0},\boldsymbol{I}_d)$ 的随机向量，$z_i\in R$ 为服从标准高斯分布 $\mathcal{N}(0,1)$ 的随机数，$i\in\{1,\cdots,m\}$. 随后，该坐标向量所对应的候选矩阵可表示为

$$\begin{aligned}Y&=\sum_{j=1}^{d}\boldsymbol{E}_j\left(\sqrt{1-c_{cov}}\boldsymbol{z}_{0,j}+\sqrt{c_{cov}}\sum_{i=1}^{m}\boldsymbol{q}_{i,j}^{(g)}z_i\right)\\&=\sqrt{1-c_{cov}}\sum_{j=1}^{d}\boldsymbol{E}_j\boldsymbol{z}_{0,j}+\sqrt{c_{cov}}\sum_{i=1}^{m}\left(\sum_{j=1}^{d}\boldsymbol{E}_j\boldsymbol{q}_{i,j}^{(g)}\right)z_i,\end{aligned}$$

其中，$\boldsymbol{z}_{0,j}$ 和 $\boldsymbol{q}_{i,j}^{(g)}$ 分别表示 $\boldsymbol{z}_0$ 和 $\boldsymbol{q}_i^{(g)}$ 中的第 j 个元素.

令 $\boldsymbol{Q}_i^{(g)}=\sum_{j=1}^{d}\boldsymbol{E}_j\boldsymbol{q}_{i,j}^{(g)}\in R^{n\times p}$，则 $\boldsymbol{Q}_i^{(g)}$ 是一个定义在 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中、坐标为 $\boldsymbol{q}_i^{(g)}$ 的矩阵. 因此，在上述协方差矩阵的建模中，可以使用矩阵 $\boldsymbol{Q}_i^{(g)}$ 代替向量 $\boldsymbol{q}_i^{(g)}$ 从而在采样过程中避免使用坐标基. 由于 $\boldsymbol{Q}_i^{(g)}$ 和 $\boldsymbol{q}_i^{(g)}$ 发挥了相同的作用，本文的后续部分将该矩阵也同样称为搜索方向.

对于 $\sum_{j=1}^{d}\boldsymbol{E}_j\boldsymbol{z}_{0,j}$，从直观上看，其计算过程中不可避免地涉及到了坐标基. 然而，可以观察到其本质上为 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中服从标准高斯分布的随机变量. 由于标准高斯分布与特定坐标基无关，因此可以将其替换成同一空间中呈标准高斯分布的其他随机变量. 另一方面,记 $\boldsymbol{L}\in R^{n\times m}$ 为每一个元素均独立地服从 $\mathcal{N}(0,1)$ 的矩阵，可知 $\boldsymbol{L}$ 为矩阵空间 $R^{n\times m}$ 中服从标准高斯分布的随机变量；又由于 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 为 $R^{n\times m}$ 的子空间,而 $P_{\boldsymbol{X}^{(g)}}$ 为相应的正交投影,因此 $P_{\boldsymbol{X}^{(g)}}(\boldsymbol{L})$ 的分布为 $\boldsymbol{L}$ 的分布的边缘分布，亦即，$P_{\boldsymbol{X}^{(g)}}(\boldsymbol{L})$ 为 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中服从标准高斯分布的随机变量. 因此，可使用 $P_{\boldsymbol{X}^{(g)}}(\boldsymbol{L})$ 代替 $\sum_{j=1}^{d}\boldsymbol{E}_j\boldsymbol{z}_{0,j}$ 完成采样过程.

综上所述，当考虑变异强度时，MSDA-ES 使用以下公式产生一个候选矩阵

$$\boldsymbol{Y}=\sigma^{(g)}\left(\sqrt{1-c_{cov}}P_{\boldsymbol{X}^{(g)}}(\boldsymbol{L})+\sqrt{c_{cov}}\sum_{i=1}^{m}\boldsymbol{Q}_i^{(g)}z_i\right) \tag{2}$$

其中，$\boldsymbol{L}\in R^{n\times p}$ 为每个元素都独立地服从 $\mathcal{N}(0,1)$ 的矩阵，$\boldsymbol{Q}_i^{(g)}\in R^{n\times p}$ 为矩阵形式的搜索方向，$z_i\in R$ 为服从 $\mathcal{N}(0,1)$ 的随机数，$i\in\{1,\cdots,m\}$.

3.2.3　解的表示方法

从公式(2)可知，候选矩阵的产生过程与切空间中的坐标基无关，因此，每一个候选矩阵可以直接

表示成矩阵形式而无需额外的参数化过程. 需要注意的是，这一方法在表示一个矩阵时引入了冗余的参数，但由于在常见的流形优化问题中 p 通常为一个较小的数，因而该编码策略中消耗的额外存储空间几乎可以忽略. 另一方面，使用该方法时，矩阵的采样过程仅需要标准的线性代数运算，因此其计算效率较高.

3.3 协方差矩阵适应

标准 CMA-ES 通过协方差矩阵适应提高在整个欧式空间中重现成功变异方向的似然性，与此对应，MSDA-ES 中协方差矩阵适应的目标是在切空间中提高重现成功变异方向的似然性. 然而在 MSDA-ES 中，该过程伴随着切空间的改变，需要进行如下的特殊处理.

3.3.1 确定切空间

在第 g 代中，当产生了全部 λ 个候选矩阵后，使用回撤算子将它们映射至流形 $\mathcal{M}$. 对映射回流形的候选矩阵进行评估和排序，使得 $f(R_{X^{(g)}}(\boldsymbol{Y}_{1:\lambda}))\leqslant\cdots\leqslant f(R_{X^{(g)}}(\boldsymbol{Y}_{\lambda:\lambda}))$，其中，$\boldsymbol{Y}_{i:\lambda}$ 表示排名第 i 的候选矩阵.

此后，对排名最好的 μ 个候选矩阵进行线性组合，得到新的分布均值：

$$\bar{\boldsymbol{Y}}=\sum_{i=1}^{\mu}\omega_i\boldsymbol{Y}_{i:\lambda},$$

其中，$\omega_1,\cdots,\omega_\mu$ 为组合系数且满足 $\sum_{i=1}^{\mu}\omega_i=1$，$\omega_1\geqslant\cdots\geqslant\omega_\mu>0$. 上述操作与标准 CMA-ES 更新群体均值的过程一致. 需要注意的是，由于所有候选矩阵都位于 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中，而根据定义 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 为欧式空间，因此，此处进行的线性组合是合法运算.

最后，使用回撤算子将 $\bar{\boldsymbol{Y}}$ 从切空间 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 映射至流形 $\mathcal{M}$，得到以下新的矩阵：

$$\boldsymbol{X}^{(g+1)}=R_{\boldsymbol{X}^{(g)}}(\bar{\boldsymbol{Y}}).$$

上述得到的新矩阵 $\boldsymbol{X}^{(g+1)}$ 将在下一代中作为概率分布的均值，其相应的切空间则在下一代中作为搜索空间. 换言之，上述过程完成了一次切空间的变换. 切空间的变换在 MSDA-ES 中至关重要，如果算法在搜索过程中保持切空间固定不变，则新产生的候选矩阵将逐渐远离初始切空间的原点，由于流形与其切空间的同胚关系仅仅是定义在局部区域的，因此切空间中的候选矩阵与流形上的可行解之间的对应关系将越来越弱. 相反，使用以上方法在每一次迭代中确定一个新的切空间作为当前代的搜索区域，使得每一代中产生的候选矩阵都分布在原点的周围，能够有效避免上述问题.

3.3.2 计算成功变异方向

演化策略中，成功变异方向是指群体均值的移动方向. MSDA-ES 通过学习成功变异方向的决策变量间依赖关系以不断更新其概率分布. 该方向由下式给出

$$\boldsymbol{Z}=\sqrt{\mu_{eff}}\,\frac{\bar{\boldsymbol{Y}}}{\sigma^{(g)}},$$

其中，$\mu_{eff}=1/\sum_{j=1}^{\mu}\omega_j^2$. 可以验证，在不存在选择压力时，$\boldsymbol{Z}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{C}^{(g)})$. 这表明 $\boldsymbol{Z}$ 的方差为协方差矩阵的一个无偏估计，因而在后续步骤中，将使用 $\boldsymbol{Z}$ 更新搜索方向.

3.3.3 更新搜索方向

搜索方向适应机制的核心思想是利用成功变异方向 Z 分别构造出 m 个协方差矩阵的无偏估计，分别用于更新 m 个搜索方向.

（1）更新 $\boldsymbol{Q}_1^{(g)}$：令 $\boldsymbol{Z}_1=\boldsymbol{Z}$，则 $\boldsymbol{Z}_1$ 的方差为 $\boldsymbol{C}^{(g)}$ 的一个无偏估计，因而第一个搜索方向按如下方式更新：

$$\boldsymbol{Q}_1'=(1-c_c)\boldsymbol{Q}_1^{(g)}+\sqrt{c_c(2-c_c)}\boldsymbol{Z}_1,$$

其中，$c_c\in(0,1)$ 为搜索方向的更新率. 当不存在选择压力时，若 $\boldsymbol{Q}_1^{(g)}\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{C}^{(g)})$，则 $\boldsymbol{Q}_1'\sim\mathcal{N}(\boldsymbol{0},\boldsymbol{C}^{(g)})$，亦即上式为无偏更新. 需要注意的是，上式与标准 CMA-ES 中演化路径的更新过程具有相同形式，因此 MSDA-ES 中所使用的第一个搜索方向即为演化路径. 此外，由于 $\boldsymbol{Z}_1=\boldsymbol{Z}$，因而其方差为（关于成功变异方向的）极大似然估计，即，以上更新得到的协方差矩阵增大了重现成功变异方向的似然性.

（2）更新 $\boldsymbol{Q}_2'^{(g)},\cdots,\boldsymbol{Q}_m'^{(g)}$：同样使用成功变异方向对余下的 $m-1$ 个搜索方向进行更新. 然而，若使用与上述完全相同的更新方式，则这些搜索方向将变得越来越接近，导致概率分布的退化. 为了保持搜索方向之间的独立性，为余下的每个搜索方向 $\boldsymbol{Q}_j'^{(g)}$ 构造一个新的成功变异方向 $\boldsymbol{Z}_j$；在完成相应的更新后，从 $\boldsymbol{Z}_j$ 中减去其在已更新搜索方向（记为 $\boldsymbol{Q}_j'$）上的投影，用以构造 $\boldsymbol{Z}_{j+1}$ 从而更新下一个搜索方向. 换言之，算法使得每个 $\boldsymbol{Z}_{j+1}$ 都正交于 $\boldsymbol{Q}_j'$，因此，所得的 $\boldsymbol{Q}_{j+1}'$ 也将趋于与 $\boldsymbol{Q}_j'$ 保持独立，增强了在高维空间中估计协方差矩阵的鲁棒性.

具体而言，在更新第 j 个搜索方向后（$j\in$

$\{1,\cdots,m-1\}$），计算 Z_j 与其在 $\boldsymbol{Q}'_j$ 上的投影间的残差：

$$\boldsymbol{Z}_j - t_j \cdot \boldsymbol{Q}'_j$$

其中，$t_j = \langle \boldsymbol{Z}_j, \boldsymbol{Q}'_j \rangle_{\boldsymbol{X}^{(g)}} / \langle \boldsymbol{Q}'_j, \boldsymbol{Q}'_j \rangle_{\boldsymbol{X}^{(g)}}$ 为投影的相对长度，$\langle \cdot,\cdot \rangle_{\boldsymbol{X}^{(g)}}$ 为流形在 $\boldsymbol{X}^{(g)}$ 的切空间中导出的内积运算.

上述投影残差已经与 $\boldsymbol{Q}'_j$ 正交，但由于其概率分布在运算中发生了改变，因此无法直接用于更新下一个搜索方向. 为了避免这一问题、提升协方差矩阵更新的无偏性，对投影残差进行如下缩放：

$$\boldsymbol{Z}_{j+1} = \frac{1}{\sqrt{1+t_j^2}}(\boldsymbol{Z}_j - t_j \cdot \boldsymbol{Q}'_j).$$

可以验证，缩放系数 $\frac{1}{\sqrt{1+t_j^2}}$ 使得在满足下列三个条件时 $\boldsymbol{Z}_{j+1} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(g)})$：（1）不存在选择压力，（2）$\boldsymbol{Z}_j \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(g)})$，（3）$\boldsymbol{Q}'_j \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}^{(g)})$. 即，$\boldsymbol{Z}_{j+1}$ 的方差同样是 $\boldsymbol{C}^{(g)}$ 的无偏估计. 利用这一性质，第 $j+1$ 个搜索方向可按如下方式更新：

$$\boldsymbol{Q}'_{j+1} = (1-c_c)\boldsymbol{Q}^{(g)}_{j+1} + \sqrt{c_c(2-c_c)}\boldsymbol{Z}_{j+1}.$$

可见，在更新 m 个搜索方向时，算法在保证搜索方向无偏更新的同时减少了搜索方向间的依赖关系.

3.3.4　移动协方差矩阵

在上述过程中，更新后的搜索方向 $\boldsymbol{Q}'_i$ 位于当前代的切空间 $T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中，为了能够使协方差矩阵用于下一代搜索过程，需要将搜索方向移动至由 $\boldsymbol{X}^{(g+1)}$ 所决定的新的切空间. 为此，可对其使用如下的正交投影：

$$\boldsymbol{Q}_i^{(g+1)} = P_{\boldsymbol{X}^{(g+1)}}(\boldsymbol{Q}'_i).$$

上述所得的 $\boldsymbol{Q}_i^{(g+1)}$ 矩阵可用于在新的切空间中重构协方差矩阵.

图 3 给出了上述整个协方差矩阵适应过程的示意图. 图中，$T_{\boldsymbol{X}^{(g)}}\mathcal{M}$ 中的细的黑色箭头表示当前代搜索方向 $\boldsymbol{Q}_i^{(g)}$，使用当前代的成功变异方向对其进行更新后，产生了一个新的矩阵 $\boldsymbol{Q}'_i$（加粗箭头标注）. 在矩阵空间 $R^{n\times p}$ 中将 $\boldsymbol{Q}'_i$ 的起点平行移动（使用虚线箭头表示）到 $\boldsymbol{X}^{(g+1)}$，将其投影至 $T_{\boldsymbol{X}^{(g+1)}}\mathcal{M}$ 从而得到新的搜索方向 $\boldsymbol{Q}_i^{(g+1)}$. $\boldsymbol{Q}_i^{(g+1)}$ 将用于在下一代中重构协方差矩阵，图中可见，新的协方差矩阵的等密度轮廓线在更新后向着 $\boldsymbol{Q}_i^{(g+1)}$ 方向进一步延伸，因此提高了重现成功变异方向的似然性.

3.4　算法实现

本节给出 MSDA-ES 的具体实现细节. 需要注意的是，虽然上述章节已经完整地描述了协方差矩阵适应过程，但还需要额外的机制对变异强度 $\sigma^{(g)}$ 进行适应. 考虑到 MSDA-ES 使用了非标准的低秩模型对协方差矩阵进行建模，本节选择文献[40]中提出的基于群体的成功率准则（Population Success Rule, PSP）完成变异强度适应.

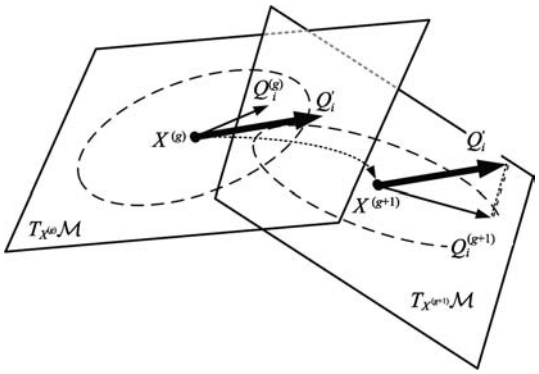


图 3　协方差矩阵适应示意图

3.4.1　伪代码

算法 I 给出了 MSDA-ES 算法的具体实现. 在算法开始前，将矩阵 $\boldsymbol{X}^{(g)}$ 初始化至流形 $\mathcal{M}$ 的任意位置，将搜索方向 $\boldsymbol{Q}_i^{(g)}$ 初始化为 0 矩阵，将用于 PSR 的统计量 $s^{(g)}$ 初始化为 0，将变异强度 $\sigma^{(g)}$ 设置为 1（第 2-3 行）.

在第 g 代，算法首先利用公式(2)产生 λ 个候选矩阵，并将这些矩阵映射至 $\mathcal{M}$ 后按目标函数值排序（第 5-7 行）. 排序结果中较好的 μ 个候选矩阵通过线性组合得到新的群体均值和成功变异方向（第 8-10 行）.

随后，按照第 3.3 节中描述的方法进行协方差矩阵适应（第 11-16 行）. 具体而言，首先使用成功变异方向 $\boldsymbol{Z}_j$ 对搜索方向 $\boldsymbol{Q}_j^{(g)}$ 进行更新，再计算 $\boldsymbol{Z}_j$ 在更新后的搜索方向（即 $\boldsymbol{Q}'_j$）上的相对投影长度 t_j，最后对 $\boldsymbol{Z}$ 在 $\boldsymbol{Q}'_j$ 上的投影残差进行缩放，作为新的成功变异方向 $\boldsymbol{Z}_{j+1}$ 用于更新下一个搜索方向.

最后，在算法的第 17-19 行，使用文献[40]中介绍的 PSR 对变异强度进行适应. 该方法首先将当前代与上一代群体混合，对混合群体按目标函数值从小到大排序，分别计算两代群体在混合群体中的秩和. 将秩和变化值归一化后与预设值 z^* 进行比较并进行累积，得到 $s^{(g+1)}$，其中 c_σ 为累积率. 上述步骤不断重复，直至满足终止条件.

算法 1.　流形搜索方向适应演化策略 MSDA-ES

输入：$\mathcal{M}$：矩阵流形；$f: R^{n\times p} \to R$：目标函数；$R_X: T_X\mathcal{M} \to \mathcal{M}$：回撤算子；$P_X: R^{n\times p} \to T_X\mathcal{M}$：切空间投影.

输出：已找到的最优解.

1:　$g \leftarrow 0$

2:　$\boldsymbol{X}^{(g)} \leftarrow \mathcal{M}$ 中随机分布的矩阵，$\sigma^{(g)} \leftarrow 1$

3: $s^{(g)} \leftarrow 0$， $\boldsymbol{Q}_i^{(g)} \leftarrow \boldsymbol{0}_{n\times p}$， $i \in \{1,\cdots,m\}$

4: WHILE(未满足终止条件){

5: $\boldsymbol{Y}_1,\cdots,\boldsymbol{Y}_\lambda \leftarrow$ 公式(2)采样

6: $f^{(g)} = \left\{ f\left(R_{\boldsymbol{X}^{(g)}}\left(\boldsymbol{Y}_{1:\lambda}\right)\right),\cdots, f\left(R_{\boldsymbol{X}^{(g)}}\left(\boldsymbol{Y}_{\lambda:\lambda}\right)\right)\right\}$

7: 将 $F^{(g)}$ 按升序排列

8: $\bar{\boldsymbol{Y}} \leftarrow \sum_{i=1}^{\mu} \omega_i \boldsymbol{Y}_{i:\lambda}$

9: $\boldsymbol{X}^{(g+1)} \leftarrow R_{\boldsymbol{X}^{(g)}}\left(\bar{\boldsymbol{Y}}\right)$

10: $\boldsymbol{Z}_1 = \sqrt{\mu_{eff}}\bar{\boldsymbol{Y}} / \sigma^{(g)}$

11: FOR ($j \leftarrow 1:m$){

12: $\boldsymbol{Q}_j' = (1-c_c)\boldsymbol{Q}_j^{(g)} + \sqrt{c_c(2-c_c)}\boldsymbol{Z}_j$

13: $t_j = \left\langle \boldsymbol{Z}_j, \boldsymbol{Q}_j' \right\rangle_{X^{(g)}} / \left\langle \boldsymbol{Q}_j', \boldsymbol{Q}_j' \right\rangle_{X^{(g)}}$

14: $\boldsymbol{Z}_{j+1} = \frac{1}{\sqrt{1+t_j^2}}\left(\boldsymbol{Z}_j - t_j \cdot \boldsymbol{Q}_j'\right)$

15: $\boldsymbol{Q}_j^{(g+1)} = P_{\boldsymbol{X}^{(g+1)}}\left(\boldsymbol{Q}_j'\right)$

16: }

17: $r^{(g)}, r^{(g-1)} \leftarrow F^{(g)}, F^{(g-1)}$ 在 $F^{(g)} \cup F^{(g-1)}$ 中的秩和

18: $s^{(g+1)} \leftarrow (1-c_\sigma)s^{(g)} + c_\sigma\left(\frac{r^{(g-1)} - r^{(g)}}{\lambda^2} - z^*\right)$

19: $\sigma^{(g+1)} \leftarrow \sigma^{(g)}\exp(s^{(g+1)})$

20: $g \leftarrow g+1$

21: }

3.4.2 参数设置

λ、μ、ω_i 等参数来源于标准 CMA-ES，因此使用文献[41]中的方法对其设置：$\lambda = 4+3\ln(d), \mu = 0.5\lambda, \omega_i = \frac{\ln(\mu+1)-\ln(i)}{\mu\ln(\mu+1) - \sum_{j=1}^{\mu}\ln(j)}$．$c_{cov}$ 为协方差矩阵的更新率，由于文献[42]指出，在信息几何优化框架中，协方差矩阵以 $O(1/\sqrt{d})$ 的速度收敛至目标函数的 Hessian 矩阵的逆，因此将其设置为 $c_{cov} = 0.4/\sqrt{d}$．c_c 为搜索方向的学习率，其合理的值应与成功变异方向的自由度成反比，因而将其设置为 $c_c = 0.25/\sqrt{d}$．c_σ 和 z^* 为变异强度适应机制 PSR 中引入的参数，因此按文献[40]中介绍的原则进行设置：$c_\sigma = 0.3, z^* = 0.25$．$m$ 为使用的搜索方向的个数．通常，m 越大，算法能够捕获的有效搜索方向越多，因而性能越好；然而，使用过多的搜索方向也将增加算法复杂性．综合以上考虑，将其设置为 $m = 10$．需要注意的是，上述部分参数虽然来源于其他算法，但其原始文献中的设置是为欧式空间中的优化而设计的，因此，当把这些参数应用于 MSDA-ES 时，这些参数中涉及到的问题维度应该设置为流形维度（即 $d = dim\mathcal{M}$）而非自变量矩阵的大小（即 mn）．

3.4.3 时间复杂度

算法的时间复杂度由算法执行的回撤算子和切空间投影的次数决定．在算法的每一代中，产生每一个候选矩阵和更新每一个搜索方向各需执行一次切空间投影．评估每一个候选矩阵和产生新的群体均值各需执行一次回撤算子．因此，在使用第 3.4.2 节中推荐的参数时，算法平摊至每一次函数评估的时间复杂度，在 Stiefel 流形和 Grassmann 流形上为 $O(np^2)$，在 Oblique 流形上为 $O(np)$．

3.4.4 不变性分析

不变性是设计演化策略时需要重点考量的性质之一．不变性是指对目标函数或决策空间施加某些变换后，算法的运行行为或最终结果保持不变．当演化策略具有不变性时，其在某一个具体问题上展现出的性能可以推广至一大类问题，表明该算法具有普适性．

本文所提出的 MSDA-ES 与标准 CMA-ES 类似，具有目标函数保序变换不变性．具体而言，若 g 表示任意严格单调递增函数，则 MSDA-ES 在求解 $\min_{\boldsymbol{X}\in\mathcal{M}} f(\boldsymbol{X})$ 和 $\min_{\boldsymbol{X}\in\mathcal{M}} g(f(\boldsymbol{X}))$ 时将表现出完全相同的性能．其原因在于 MSDA-ES 并未直接利用目标函数值，而是利用了不同解之间的目标函数值的大小关系．由于函数 g 是严格单调递增的，在群体中不改变目标函数值的序关系，因此算法性能与 g 无关．

另一方面，当决策空间具有某些不变性时，MSDA-ES 能够维持这种不变性．以 Grassmann 流形上的优化问题为例，该流形具有旋转不变性，因此当对某一优化问题的决策变量施加旋转变换后，虽然问题的形式发生了变换，但 MSDA-ES 依然将其视为同样的优化问题，展现出一致的算法性能．以下命题描述了 MSDA-ES 在求解 Grassmann 流形上的优化问题时所具有的决策空间旋转变换不变性．

命题 1. 令 A_1 和 A_2 分别为算法 1 求解优化问题 $\min_{\boldsymbol{X}\in Gr(n,p)} f(\boldsymbol{X})$ 和 $\min_{\boldsymbol{X}\in Gr(n,p)} \hat{f}(\boldsymbol{X}) = f(\boldsymbol{XU})$ 时的两个算法实例，其中 $\boldsymbol{U} \in R^{p\times p}$ 为正交矩阵．令 $S^{(g)} = (\boldsymbol{X}^{(g)}, \boldsymbol{Q}_1^{(g)},\ldots,\boldsymbol{Q}_m^{(g)}, \sigma^{(g)}, Z^{(g)})$ 和 $\hat{S}^{(g)} = (\hat{\boldsymbol{X}}^{(g)}, \hat{\boldsymbol{Q}}_1^{(g)},\ldots,\hat{\boldsymbol{Q}}_m^{(g)}, \hat{\sigma}^{(g)}, \hat{Z}^{(g)})$ 分别表示 A_1 和 A_2 运行在第 g 代时的状态．令 $G(S^{(g)}) = (\boldsymbol{X}^{(g)}\boldsymbol{V}, \boldsymbol{Q}_1^{(g)}\boldsymbol{V},\ldots,\boldsymbol{Q}_m^{(g)}\boldsymbol{V}, \sigma^{(g)}, Z^{(g)})$ 表示状态变换函数，其中 $\boldsymbol{V} \in R^{p\times p}$ 为任意正交矩阵．假设 A_1 在第 g 代使用随机变量 $\boldsymbol{L}^{(g)}, z_i^{(g)} (i = 1,..,m)$ 产生候选矩阵，则当 $\hat{S}^{(0)} = G(S^{(0)})$ 且 A_2 在第 g 代使用随机变量 $\hat{\boldsymbol{L}}^{(g)} = \boldsymbol{L}^{(g)}, \hat{z}_i^{(g)} = z_i^{(g)} (i = 1,..,m)$ 产生候选矩阵时，有

$$\hat{S}^{(g)} = G(S^{(g)})$$

对任意 $g = 1,2,\ldots$ 成立.

命题 1 的证明见补充材料. 命题 1 表明，当对 Grassmann 流形优化问题的决策变量施加某些旋转变换时，只要算法的初始条件进行相应的变换（例如，$\hat{\boldsymbol{X}}^{(g)}, \hat{\boldsymbol{Q}}_i^{(g)}$ 分别与 $\hat{\boldsymbol{X}}^{(g)}, \hat{\boldsymbol{Q}}_i^{(g)}$ 具有相同列空间，而 $\hat{\sigma}^{(g)}, \hat{Z}^{(g)}$ 分别与 $\sigma^{(g)}, Z^{(g)}$ 保持一致），则算法的后续行为将保持一致. 需要指出的是，虽然命题 1 需要假设 $\hat{\boldsymbol{L}}^{(g)} = \boldsymbol{L}^{(g)}, \hat{z}_i^{(g)} = z_i^{(g)} (i = 1,..,m)$，但由于这些随机变量独立地采样自标准正态分布，因此即使在不满足上述假设时，$\hat{S}^{(g)}$ 与 $G(S^{(g)})$ 的概率分布也是相一致的. 此外，命题 1 描述的不变性的本质，在于旋转变化在 Grassmann 流形上为等价关系，因此命题 1 可直接推广至任意流形与给定等价关系导出的商流形，即，对任意商流形，只要算法初始状态是等价的，则对决策变量施加的任意等价变换不影响算法运行行为.

4　仿真实验

4.1　测试问题

4.1.1　基于割线的数据降维

基于割线的降维（Secant-based Dimension Reduction, SDR）用于在一个高维数据集中找到一个线性子空间，使得相应的正交投影尽可能地成为单射函数[43,44].

令 $\boldsymbol{X} \in R^{n\times n}$ 为样本容量为 N 的数据矩阵. $\boldsymbol{X}$ 的单位割线集合为 $\Sigma = \{\dfrac{\boldsymbol{X}_{:,i} - \boldsymbol{X}_{:,j}}{|\boldsymbol{X}_{:,i} - \boldsymbol{X}_{:,j}|_2} | \ j = 1,\cdots,N\}$. 令 $\mathcal{U}$ 表示所求的子空间，那么当 Σ 与 $\mathcal{U}$ 的正交补不存在交集时，$\mathcal{U}$ 对应的正交投影为单射. 因此，该问题的优化任务是在 Grassmann 流形上找到一个正交矩阵 $\boldsymbol{U}$，该矩阵张成 $\mathcal{U}$ 且最小化以下函数：

$$\min_{\boldsymbol{U}\in Gr(n,p)} f(\boldsymbol{U}) = -\min_{\boldsymbol{s}\in\Sigma} |\boldsymbol{U}\boldsymbol{U}^T\boldsymbol{s}|_2 .$$

本实验中，数据矩阵 $\boldsymbol{X}$ 通过 $\boldsymbol{X} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Z}$ 产生. 其中，$\boldsymbol{Q} \in R^{n\times n}$ 为一随机正交矩阵，$\boldsymbol{\Lambda} \in R^{n\times n}$ 为对角阵且对角线元素 $\boldsymbol{\Lambda}_{i,i} = \beta^{1-i}$，$\boldsymbol{Z} \in R^{n\times n}$ 为随机矩阵且每个元素独立地服从 $\mathcal{N}(0,1)$. 参数 β 控制数据矩阵的条件数，在实验中固定为 1.05. 从集合 {50,100, 150,200,250,300} 中选择 n，从 $\{3,5,7,9,11\}$ 中选择 p，共计得到 30 组不同的测试用例.

4.1.2　Thomson 问题

Thomson 问题为著名的 3 维装箱问题，其在数学、化学、物理等领域有着广泛的应用[45]. 这一问题需要在一个球面上安放 p 个电荷，使得总静电能最小. 该问题的定义如下：

$$\min_{\boldsymbol{X}\in Ob(n,p)} f(\boldsymbol{X}) = \sum_{i=1}^{p} |\boldsymbol{X}_{:,i}|_2^{-1},$$

其中，n 固定为 3. Thomson 问题的目标函数虽然可微，但存在着大量局部最优解. 目前，仅在 p 取几个特殊值时该问题存在有效解法，而在更一般的情况下则难以求解. 在实验中，设置 $p = 25i, i \in \{1,2,\cdots,20\}$，因此，共产生 20 组测试用例.

4.1.3　二次最小化问题

二次最小化问题（Quadratic Minimization, QM）是科学计算和数据挖掘中的常见问题[17]，其试图找到一个 $n\times p$ 正交矩阵从而最小化如下函数：

$$\min_{\boldsymbol{X}\in St(n,p)} f(\boldsymbol{X}) = \frac{1}{2} tr(\boldsymbol{X}^T\boldsymbol{A}\boldsymbol{X}) + tr(\boldsymbol{G}^T\boldsymbol{X}),$$

其中，$A \in R^{n\times n}$ 为对称矩阵，$G \in R^{n\times p}$ 为任意矩阵，$tr(\cdot)$ 表示矩阵的迹. 线性特征值问题为该问题在 $G = 0$ 时的特例. 该问题的目标函数不具有旋转不变性，因而其可行域为 Stiefel 流形而非 Grassmann 流形.

在该问题中，使用 $\boldsymbol{A} = \boldsymbol{P}\boldsymbol{\Lambda}\boldsymbol{P}^T$ 随机产生矩阵 $\boldsymbol{A}$，其中 $\boldsymbol{P} \in R^{n\times n}$ 为随机正交矩阵，$\boldsymbol{\Lambda} \in R^{n\times n}$ 为对角阵且 $\boldsymbol{\Lambda}_{i,i} = \beta^{1-i}$. 矩阵 $\boldsymbol{G}$ 由 $\boldsymbol{G} = \boldsymbol{Q}\boldsymbol{D}$ 得到，其中 $\boldsymbol{Q} \in R^{n\times p}$ 为随机矩阵，其每一个元素独立地采样自 [0,1] 上的均匀分布，$\boldsymbol{D} \in R^{p\times p}$ 为对角阵且 $\boldsymbol{D}_{i,i} = \zeta^{i-1}/|\boldsymbol{Q}_{:,i}|_2$. 使用 $\beta = \zeta = 1.05$，且从 $n \in \{50,100,150,200,250,300\}$ 以及 $p \in \{3,5,7,9,11\}$ 中产生出 30 组不同的测试用例.

4.2　实验设置

4.2.1　用于比较的算法

本研究选择最速下降法（Steepest Descent Method, SD）、PSO、DE 以及 CMA-ES 作为对比算法. 其中，SD 和 PSO 为开源工具箱 Manopt[35]中的内置求解器. DE 和 CMA-ES 根据文献[31]中的方法实现，并进行了一定的修改以增强性能，读者可在附录中找到其伪代码. 这些算法的设置如下：

（1）SD：该方法在基于回撤的流形优化框架中使用经典的梯度下降法[14]. 算法在每个切空间中使用 Armijo 准则[46]进行线搜索. 由于本文内容针对黑盒优化，因此算法中所需的梯度通过有限微分法估计得到. 具体而言，算法首先在矩阵空间 $R^{n\times p}$ 中使用前向微分估计目标函数的梯度，之后使用切空间投影将梯度映射至切空间. 由于本文所考虑的三类流形均为嵌入子流形，因而所得的投影

即为黎曼梯度. 在上述过程中，估计一次梯度消耗 $np+1$次函数评估. 该算法的所有参数按 Manopt 工具箱默认设置.

（2）PSO：该方法最早来源于文献[6]，其在标准 PSO 中引入测地线从而在 3 维旋转群中进行启发式搜索. Manopt 工具箱对其进行了扩展，使其在测地线可逆时作为一般性的流形优化算法. 在本实验中，由于 Stiefel 流形上的测地线不可逆，因而仅在 Grassmann 流形和 Oblique 流形上测试该算法. 该算法的所有参数按 Manopt 工具箱默认设置.

（3）DE：按照文献[31]使用指数映射及其逆映射将传统 DE 算法推广至矩阵流形. 与文献[31]采用 DE/rand/1/bin 策略不同，本文采用了 DE/rand/1/either-or 策略，即，对每个解按概率 P_F 执行变异操作或按概率 $1-P_F$ 执行算术交叉操作. 该策略使所有演化算子均为线性变换，不再需要计算每个解的坐标值以进行传统的二进制交叉操作. 依据文献[47]，设置 $P_F=0.4$，差分向量缩放因子 $F=0.7$，算术交叉系数 $k=0.85$. 与 PSO 类似，该算法依赖指数映射的逆，因此不用于 Stiefel 流形上的测试问题.

（4）CMA-ES：根据文献[31]，使用平行移动实现矩阵流形上演化路径的更新. 然而，考虑到文献[31]只涉及小规模问题，而本文实验使用的测试问题维度较高，为了避免平行移动双线性映射中的耗时操作，引入了文献[48]中提出的有限内存矩阵适应技术. 该技术是协方差矩阵适应机制在大规模环境中的变体，使用少量几个演化路径重建协方差矩阵，因此与 MSDA-ES 有相同的算法复杂度，保证了实验的公平性. 为了提升算法在大规模环境中的收敛速度，算法参数按文献[48]设置.

4.2.2 终止条件

本实验在函数评估次数超过 *maxFEs* 时终止所有算法. 对于 SDR 问题，$maxFEs=50np$，对于其他问题，$maxFEs=100np$. 此外，由于不同算法使用了不同的框架，因而为每个算法单独设置以下的终止条件：对 SD，算法在梯度的 2 范数小于 10^{-6} 或 Armijo 步长小于 10^{-8} 时终止算法；对 PSO 和 DE，最后 100 代中最优目标函数值的改进值小于 10^{-8} 时终止算法；对于 MSDA-ES 和 CMA-ES，变异强度小于 10^{-6} 时终止算法.

4.2.3 性能指标与统计方法

对于每一个算法，记录其所得的目标函数值作为性能指标，目标函数值越小表明算法性能越好. 所有算法均独立重复 20 次，使用中值和四分位距衡量结果的平均水平和浮动范围. 在每一个测试用例上，使用置信水平为 0.05 的 Wilcoxon 秩和检验比较 MSDA-ES 和相应算法之间的统计差异. 使用多重检验比较所有算法在所有测试用例上的综合性能. 该检验首先根据 Friedman 测试计算每个算法的平均排名，并使用 Bonferroni 后置过程进行校正，校正后的置信水平作为最终检验结果.

除了直接比较数值结果，还使用数据剖面图（Data profiles）对所有算法的综合性能做出可视化的对比. 在数据剖面图中，横坐标（τ）表示某个算法所分配的计算资源，纵坐标（$\rho(\tau)$）表示在给定 τ 值时算法能够求解的问题的百分比. 在绘制剖面图时，对于一个算法，若其在一个测试用例 t 上满足 $f_{\mu_t}\leqslant f_{min}+10^{-3}\left(f_{max}-f_{min}\right)$，则称该算法可在 μ_t 次函数评估内求解 t，其中 f_{μ_t} 为该算法在 μ_t 次函数评估内所获得的最优目标函数值，f_{max} 为所有对比算法初始解中的最大目标函数值，f_{min} 为 μ_t 次函数评估内所有算法得到的最优函数值. 数据剖面（曲线）可定义为

$$\rho(\tau)=\frac{1}{|\mathcal{TS}|}\left|\left\{t\in\mathcal{TS},\frac{\mu_t}{s_t+1}\leqslant\tau\right\}\right|,$$

其中，$|\cdot|$ 表示集合的元素个数，$\mathcal{TS}$ 表示测试用例的集合，s_t 表示测试用例 t 的大小. s_t 设置为矩阵大小，即 $s_t=np$，因此 $\frac{\mu_t}{s_t+1}$ 可解释为一次梯度估计所需的评估次数. 这一缩放因子使得数据剖面独立于问题规模. 在数据剖面图中，可以在给定 τ 值时通过数据剖面曲线的位置衡量算法优劣. 同时，可以计算数据剖面的曲线下面积（Area under the Curve，$AUC=\int\rho(\tau)\mathrm{d}\tau$）衡量算法的综合性能，算法的 *AUC* 值越大表明算法的综合性能越好.

4.3 数值结果

4.3.1 SDR 问题上的最终结果

表 2 给出了 MSDA-ES、SD、PSO、DE 以及 CMA-ES 在基于割线的数据降维问题上的最终实验结果. 显然，MSDA-ES 在所有问题上均显著优于其他对比算法. SDR 问题的难度主要来源于其不可微性，因此 SD 使用的有限微分法难以估计出精确的下降方向. 此外，该问题的目标函数非凸，因而确定性算法在该问题上难以有效收敛至全局最优解. 由于这两方面原因，SD 的性能表现最差. PSO、DE 和 CMA-ES 由于采用了演化算法框架，不依赖于问题的可微性或凸性，因而在该问题上性能优于 SD. 然而，上述三个演化算法依然在所有测试用例上均弱

表 2　在 SDR 问题上 MSDA-ES 等算法最终结果的中值和四分位距(每个测试用例上的最优值和次优值分别用深色背景和浅色背景标注)

n	p	MSDA-ES	SD	PSO	DE	CMA-ES
50	3	−3.02E-1(3.2E-2)	−8.60E-2(2.7E-2)+	−1.21E-1(2.3E-2)+	−1.28E-1(1.7E-2)+	−2.18E-1(3.9E-2)+
50	5	−5.52E-1(2.4E-2)	−1.82E-1(4.0E-2)+	−2.22E-1(2.1E-2)+	−2.37E-1(1.3E-1)+	−4.34E-1(2.9E-2)+
50	7	−7.11E-1(3.6E-2)	−2.51E-1(4.4E-2)+	−2.93E-1(2.3E-2)+	−3.16E-1(1.6E-1)+	−5.56E-1(5.0E-2)+
50	9	−8.05E-1(3.5E-2)	−3.25E-1(4.7E-2)+	−3.71E-1(2.2E-2)+	−3.83E-1(2.1E-1)+	−6.94E-1(4.2E-2)+
50	11	−8.69E-1(1.1E-2)	−3.66E-1(8.0E-2)+	−4.18E-1(2.3E-2)+	−4.26E-1(2.1E-1)+	−7.62E-1(1.7E-2)+
100	3	−3.19E-1(5.3E-2)	−5.81E-2(1.8E-2)+	−8.48E-2(1.7E-2)+	−9.40E-2(1.9E-2)+	−2.16E-1(2.0E-2)+
100	5	−5.63E-1(3.8E-2)	−1.31E-1(3.2E-2)+	−1.61E-1(1.4E-2)+	−2.04E-1(1.1E-1)+	−4.40E-1(3.6E-2)+
100	7	−7.15E-1(2.6E-2)	−1.95E-1(5.8E-2)+	−2.09E-1(1.8E-2)+	−2.48E-1(1.3E-1)+	−5.88E-1(3.0E-2)+
100	9	−8.05E-1(2.0E-2)	−2.33E-1(4.7E-2)+	−2.55E-1(2.1E-2)+	−2.94E-1(2.2E-2)+	−6.86E-1(2.6E-2)+
100	11	−8.75E-1(1.3E-2)	−2.69E-1(4.5E-2)+	−2.92E-1(1.6E-2)+	−3.35E-1(2.1E-1)+	−7.68E-1(3.5E-2)+
150	3	−3.08E-1(3.1E-2)	−5.66E-2(1.7E-2)+	−7.22E-2(4.6E-3)+	−7.97E-2(1.2E-2)+	−2.17E-1(4.2E-2)+
150	5	−5.56E-1(2.6E-2)	−1.20E-1(2.8E-2)+	−1.28E-1(9.7E-3)+	−1.62E-1(8.1E-2)+	−4.39E-1(4.2E-2)+
150	7	−7.11E-1(2.4E-2)	−1.56E-1(3.8E-2)+	−1.72E-1(1.6E-2)+	−2.00E-1(1.2E-1)+	−5.99E-1(3.9E-2)+
150	9	−8.03E-1(1.9E-2)	−2.04E-1(3.7E-2)+	−2.09E-1(2.0E-2)+	−2.83E-1(1.3E-1)+	−6.82E-1(2.5E-2)+
150	11	−8.71E-1(1.3E-2)	−2.26E-1(2.9E-2)+	−2.39E-1(2.2E-2)+	−3.02E-1(1.6E-1)+	−7.47E-1(1.6E-2)+
200	3	−3.20E-1(2.0E-2)	−5.38E-2(1.9E-2)+	−5.90E-2(1.5E-2)+	−7.18E-2(5.1E-2)+	−2.22E-1(2.8E-2)+
200	5	−5.47E-1(3.2E-2)	−1.00E-1(2.6E-2)+	−1.12E-1(1.9E-2)+	−1.46E-1(8.2E-2)+	−4.47E-1(4.1E-2)+
200	7	−7.16E-1(1.7E-2)	−1.42E-1(3.4E-2)+	−1.51E-1(1.5E-2)+	−1.91E-1(1.1E-1)+	−6.11E-1(1.7E-2)+
200	9	−8.04E-1(2.4E-2)	−1.78E-1(4.1E-2)+	−1.84E-1(2.0E-2)+	−2.03E-1(3.6E-2)+	−6.97E-1(3.3E-2)+
200	11	−8.66E-1(2.0E-2)	−2.06E-1(3.3E-2)+	−2.07E-1(8.0E-3)+	−3.03E-1(1.6E-1)+	−7.54E-1(3.3E-2)+
250	3	−3.31E-1(1.9E-2)	−4.89E-2(1.6E-2)+	−5.47E-2(7.7E-3)+	−6.83E-2(5.4E-2)+	−2.21E-1(2.3E-2)+
250	5	−5.67E-1(1.9E-2)	−8.41E-2(3.9E-2)+	−1.02E-1(1.5E-2)+	−1.20E-1(8.2E-2)+	−4.62E-1(3.3E-2)+
250	7	−7.10E-1(3.2E-2)	−1.31E-1(3.4E-2)+	−1.33E-1(1.4E-2)+	−1.64E-1(9.9E-2)+	−6.09E-1(2.8E-2)+
250	9	−8.03E-1(1.3E-2)	−1.62E-1(1.8E-2)+	−1.63E-1(1.3E-2)+	−1.98E-1(1.1E-1)+	−7.04E-1(4.4E-2)+
250	11	−8.67E-1(1.1E-2)	−1.90E-1(3.0E-2)+	−1.85E-1(1.3E-2)+	−2.09E-1(3.2E-2)+	−7.54E-1(2.3E-2)+
300	3	−3.27E-1(3.2E-2)	−4.63E-2(1.6E-2)+	−5.32E-2(7.1E-3)+	−6.85E-2(3.8E-2)+	−2.30E-1(2.7E-2)+
300	5	−5.64E-1(3.0E-2)	−7.95E-2(1.7E-2)+	−9.33E-2(9.8E-3)+	−1.43E-1(6.9E-2)+	−4.51E-1(2.8E-2)+
300	7	−7.22E-1(2.3E-2)	−1.17E-1(2.4E-2)+	−1.21E-1(1.4E-2)+	−2.24E-1(9.5E-2)+	−6.07E-1(3.0E-2)+
300	9	−8.07E-1(1.9E-2)	−1.47E-1(2.2E-2)+	−1.49E-1(1.8E-2)+	−1.74E-1(6.1E-2)+	−6.97E-1(3.6E-2)+
300	11	−8.66E-1(1.1E-2)	−1.77E-1(1.7E-2)+	−1.70E-1(1.5E-2)+	−2.02E-1(8.7E-2)+	−7.61E-1(2.2E-2)+
		+ \ - \ =	30 \ 0 \ 0	30 \ 0 \ 0	30 \ 0 \ 0	30 \ 0 \ 0

注：“+”表示 MSDA-ES 显著优于（基于显著性水平 0.05 的 Wilcoxon 秩和检验）对比算法；“-”表示相反，即对比算法显著优于 MSDA-ES；若不存在显著差异，则用“=”表示. 这些符号在本文其他表格中具有相同含义.

于 MSDA-ES. PSO 和 DE 性能较弱的原因可能在于其无法有效求解大规模问题[49,50]. 事实上，目前绝大多数 PSO 和 DE 的相关研究都仅针对于 100 维以下的问题，而本文中问题的维度普遍在 150 至 2000 之间. 目前使用 PSO 和 DE 求解大规模问题的研究大多依赖于对全局坐标系的分解[51−53]，因此无法应用于流形优化之中. 相比之下，由于 MSDA-ES 和 CMA-ES 引入了专门设计的大规模协方差矩阵建模

技术，因此在求解流形优化问题中有明显的优势. MSDA-ES 相比于 CMA-ES 的性能优势主要来源于其搜索方向适应机制. 正如文献[31,32]指出的，传统的演化路径累积技术在黎曼流形中的几何意义不明确，因而间接影响了协方差矩阵适应和变异强度适应机制，造成算法性能的降低. 由此可见，MSDA-ES 中专门设计的流形搜索方向适应机制对于求解矩阵流形上的优化问题是必要的.

4.3.2 Thomson 问题上的最终结果

由于 Thomson 问题包含了大量局部最优解，因此可以预期，使用演化策略框架的 MSDA-ES 能够在该问题上表现出良好的性能. 表 3 给出的实验结果很大程度上验证了该观点. MSDA-ES 在所有测试用例上均取得最优结果，且在几乎所有测试用例上显著优于所有对比算法. 对于 SD，由于目标函数具有可微性，因而其收敛速度较快，能够在所有测试用例中得到仅次于 MSDA-ES 的结果. PSO 和 DE 虽然采用了演化算法框架，在避免陷入局部最优解方面存在一定优势，但由于收敛能力不足，因此最终结果相对较差. 此外，我们发现 MSDA-ES 在大多数问题上对应的 IQR 值较小，表明其在每一次独立运行中都能够产生出类似的解，说明了 MSDA-ES 采用的演化策略框架能够在保持收敛性能的前提下避免算法陷入局部最优解. CMA-ES 的表现类似于 MSDA-ES，但 IQR 值相对更大. 其原因如前所述，可能是由于在黎曼流形上使用了针对欧式空间设计的演化路径累积技术. 事实上，使用切空间投影在矩阵流形上移动演化路径时，不可避免地会缩短演化路径长度，导致变异强度过度缩小，造成算法提前收敛. MSDA-ES 中的变异强度适应机制不依赖于演化路径或搜索方向，仅和群体的目标函数值有关，因此未发生类似的问题.

表 3 在 Thomson 问题上 MSDA-ES 等算法最终结果的中值和四分位距(每个测试用例上的最优值和次优值分别用深色背景和浅色背景标注)

n	p	MSDA-ES	SD	PSO	DE	CMA-ES
3	25	2.44E+2(8.0E-4)	2.44E+2(1.0E-3)=	2.46E+2(1.1E+0)+	2.57E+2(2.2E+0)+	2.44E+2(2.9E-2)+
3	50	1.06E+3(8.7E-2)	1.06E+3(3.9E-2)+	1.08E+3(4.3E+0)+	1.11E+3(7.7E+0)+	1.06E+3(1.1E-1)+
3	75	2.45E+3(9.4E-2)	2.45E+3(1.3E-1)=	2.52E+3(9.3E+0)+	2.59E+3(1.4E+1)+	2.45E+3(1.8E-1)+
3	100	4.45E+3(2.5E-1)	4.45E+3(2.7E-1)=	4.57E+3(1.4E+1)+	4.68E+3(2.0E+1)+	4.45E+3(2.2E-1)+
3	125	7.04E+3(1.6E-1)	7.04E+3(2.4E-1)+	7.24E+3(3.1E+1)+	7.38E+3(2.5E+1)+	7.04E+3(2.8E-1)+
3	150	1.02E+4(2.5E-1)	1.02E+4(4.0E-1)+	1.05E+4(2.7E+1)+	1.07E+4(4.9E+1)+	1.02E+4(3.2E-1)+
3	175	1.40E+4(2.4E-1)	1.40E+4(4.7E-1)+	1.45E+4(5.9E+1)+	1.47E+4(3.9E+1)+	1.40E+4(4.7E-1)+
3	200	1.84E+4(3.7E-1)	1.84E+4(4.5E-1)+	1.90E+4(6.3E+1)+	1.93E+4(4.0E+1)+	1.84E+4(4.7E-1)+
3	225	2.35E+4(4.0E-1)	2.35E+4(7.5E-1)+	2.42E+4(1.1E+2)+	2.45E+4(8.7E+1)+	2.35E+4(6.2E-1)+
3	250	2.91E+4(4.0E-1)	2.91E+4(4.8E-1)+	2.99E+4(8.6E+1)+	3.03E+4(5.2E+1)+	2.91E+4(4.9E-1)+
3	275	3.53E+4(3.3E-1)	3.53E+4(6.7E-1)+	3.63E+4(1.5E+2)+	3.67E+4(1.0E+2)+	3.53E+4(7.5E-1)+
3	300	4.21E+4(4.2E-1)	4.21E+4(9.4E-1)+	4.33E+4(1.3E+2)+	4.38E+4(7.0E+1)+	4.21E+4(3.8E-1)+
3	325	4.96E+4(5.1E-1)	4.96E+4(4.2E-1)+	5.09E+4(1.6E+2)+	5.15E+4(8.5E+1)+	4.96E+4(5.9E-1)+
3	350	5.76E+4(6.6E-1)	5.76E+4(9.7E-1)+	5.92E+4(1.5E+2)+	5.99E+4(9.5E+1)+	5.76E+4(1.1E+0)+
3	375	6.63E+4(6.2E-1)	6.63E+4(1.1E+0)+	6.80E+4(1.7E+2)+	6.88E+4(1.1E+2)+	6.63E+4(1.1E+0)+
3	400	7.56E+4(5.3E-1)	7.56E+4(1.2E+0)+	7.76E+4(1.6E+2)+	7.84E+4(9.6E+1)+	7.56E+4(8.1E-1)+
3	425	8.55E+4(1.1E+0)	8.55E+4(1.8E+0)+	8.78E+4(2.5E+2)+	8.85E+4(1.0E+2)+	8.55E+4(7.8E-1)+
3	450	9.60E+4(1.1E+0)	9.60E+4(1.4E+0)+	9.85E+4(2.0E+2)+	9.94E+4(9.2E+1)+	9.60E+4(1.3E+0)+
3	475	1.07E+5(7.1E-1)	1.07E+5(1.7E+0)+	1.10E+5(2.4E+2)+	1.11E+5(1.4E+2)+	1.07E+5(1.1E+0)+
3	500	1.19E+5(9.4E-1)	1.19E+5(1.5E+0)+	1.22E+5(2.1E+2)+	1.23E+5(1.5E+2)+	1.19E+5(1.2E+0)+
		+ \ - \ =	17 \ 0 \ 3	20 \ 0 \ 0	20 \ 0 \ 0	20 \ 0 \ 0

4.3.3 QM 问题上的最终结果

QM 问题为定义在 Stiefel 流形上的矩阵优化问题，由于在该流形中测地线不可逆，因此本组实验不对 PSO 和 DE 进行测试. 相比于以上两类问题，QM 问题中的目标函数是光滑的，因此使用梯度的算法能够有效求解该问题，同时，在满足某些假设时能够达到全局最优解[17]. 因此，该组实验的主要目的并非是比较算法的优劣，而是验证 MSDA-ES 作为通用

流形优化器的可能性. 表 4 给出了实验结果，可以观察到，MSDA-ES 和 CMA-ES 性能类似，在 2 个测试用例上取得了显著优势，而在另外 28 个测试用例上不存在显著差异. 相比于 SD，MSDA-ES 在 30 个问题中的 18 个上获得了更好的最终结果，但同样在 28 个测试用例上不存在显著的统计差异. 因此，可以从该组实验中得出如下结论：在相对简单的问题上，MSDA-ES 的性能不弱于具有理论支撑的算法.

表 4　在 QM 问题上 MSDA-ES 等算法最终结果的中值和四分位距(每个测试用例上的最优值和次优值分别用深色背景和浅色背景标注)

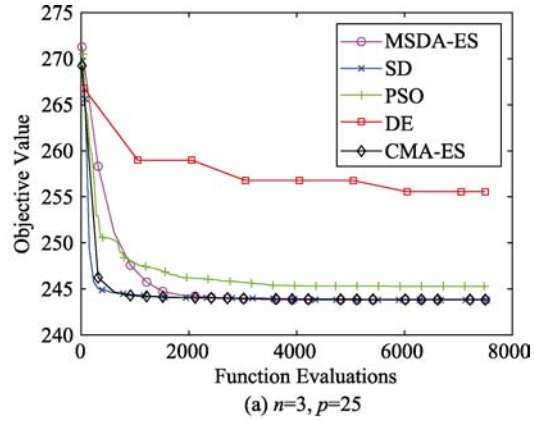
n	p	MSDA-ES	SD	CMA-ES
50	3	−1.98E+0(5.9E-2)	−1.98E+0(6.7E-2)=	−1.99E+0(7.0E-2)=
50	5	−3.36E+0(1.7E-1)	−3.32E+0(7.9E-2)=	−3.37E+0(9.3E-2)=
50	7	−4.78E+0(1.2E-1)	−4.75E+0(1.2E-1)=	−4.83E+0(1.4E-1)=
50	9	−6.42E+0(1.8E-1)	−6.41E+0(1.7E-1)=	−6.37E+0(1.3E-1)=
50	11	−8.12E+0(1.9E-1)	−8.01E+0(3.0E-1)=	−8.13E+0(1.1E-1)=
100	3	−2.16E+0(4.8E-2)	−2.14E+0(5.9E-2)=	−2.17E+0(3.4E-2)=
100	5	−3.69E+0(6.0E-2)	−3.69E+0(6.7E-2)=	−3.67E+0(6.5E-2)=
100	7	−5.31E+0(8.1E-2)	−5.34E+0(6.7E-2)=	−5.30E+0(1.1E-1)=
100	9	−7.07E+0(1.2E-1)	−7.06E+0(8.3E-2)=	−7.03E+0(1.8E-1)=
100	11	−8.93E+0(1.8E-1)	−8.98E+0(1.1E-1)=	−8.96E+0(1.3E-1)=
150	3	−2.21E+0(3.5E-2)	−2.23E+0(3.0E-2)-	−2.22E+0(4.1E-2)=
150	5	−3.80E+0(4.4E-2)	−3.82E+0(5.4E-2)=	−3.80E+0(3.9E-2)=
150	7	−5.48E+0(1.1E-1)	−5.48E+0(8.5E-2)=	−5.47E+0(6.3E-2)=
150	9	−7.36E+0(1.5E-1)	−7.32E+0(6.0E-2)=	−7.32E+0(7.2E-2)=
150	11	−9.27E+0(1.0E-1)	−9.25E+0(1.3E-1)=	−9.26E+0(6.2E-2)=
200	3	−2.25E+0(3.5E-2)	−2.25E+0(3.2E-2)=	−2.24E+0(2.7E-2)=
200	5	−3.85E+0(5.1E-2)	−3.86E+0(4.8E-2)=	−3.86E+0(5.2E-2)=
200	7	−5.58E+0(4.3E-2)	−5.56E+0(4.5E-2)+	−5.57E+0(5.8E-2)=
200	9	−7.43E+0(5.9E-2)	−7.41E+0(1.0E-1)=	−7.38E+0(7.8E-2)+
200	11	−9.40E+0(1.1E-1)	−9.41E+0(1.1E-1)=	−9.40E+0(1.1E-1)=
250	3	−2.26E+0(1.7E-2)	−2.27E+0(3.3E-2)=	−2.27E+0(3.2E-2)=
250	5	−3.89E+0(4.9E-2)	−3.89E+0(4.9E-2)=	−3.88E+0(4.4E-2)=
250	7	−5.61E+0(4.8E-2)	−5.61E+0(3.7E-2)=	−5.61E+0(4.1E-2)=
250	9	−7.49E+0(4.6E-2)	−7.48E+0(3.6E-2)=	−7.46E+0(5.4E-2)+
250	11	−9.51E+0(8.1E-2)	−9.49E+0(4.7E-2)=	−9.49E+0(8.7E-2)=
300	3	−2.28E+0(2.3E-2)	−2.28E+0(2.5E-2)=	−2.27E+0(3.0E-2)=
300	5	−3.90E+0(3.4E-2)	−3.91E+0(3.3E-2)=	−3.89E+0(3.6E-2)=
300	7	−5.65E+0(3.8E-2)	−5.63E+0(5.4E-2)=	−5.65E+0(5.4E-2)=
300	9	−7.52E+0(9.0E-2)	−7.52E+0(7.0E-2)=	−7.52E+0(7.0E-2)=
300	11	−9.55E+0(7.9E-2)	−9.54E+0(6.1E-2)=	−9.55E+0(8.1E-2)=
		+ \ - \ =	1 \ 1 \ 28	2 \ 0 \ 28

4.3.4　收敛行为研究

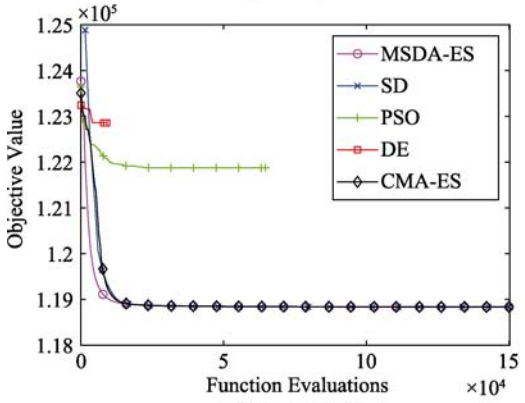
上节中的实验仅反映了各个算法在终止后的最终结果，其实验结果可能受到终止条件的影响. 因此，本节将使用收敛曲线图描述算法在特定测试用例上的目标函数值随评估次数的变化趋势. 对于每一个测试问题，分别选择规模最大和最小的两个测试用例. 用于绘图的数据来源于 20 次独立运行中结果最接近于中值的那一次.

图 4、图 5、图 6 给出了各算法的收敛曲线，有如下观察结果：

（1）MSDA-ES 的收敛速度很快，在大多数测试问题上，能够在约 20%~30%的总评估次数内接近最终解. 在图 4(b)和图 5(b)所示的高维问题中，其收敛速度甚至超过了 SD. CMA-ES 具有类似的收敛行为，而它们的收敛速度都比 PSO 和 DE 更快. 这一观察结果与欧氏空间优化的相关研究基本一致[54].

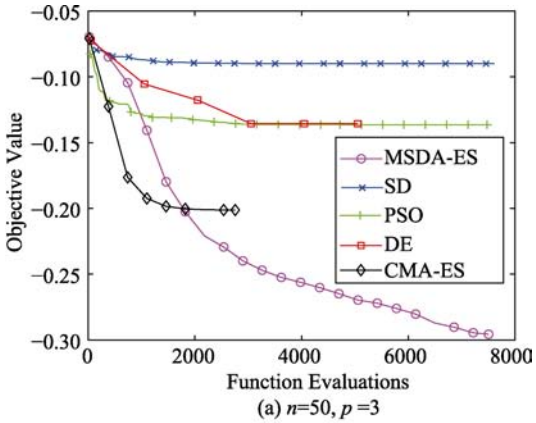


(a) n=3, p=25

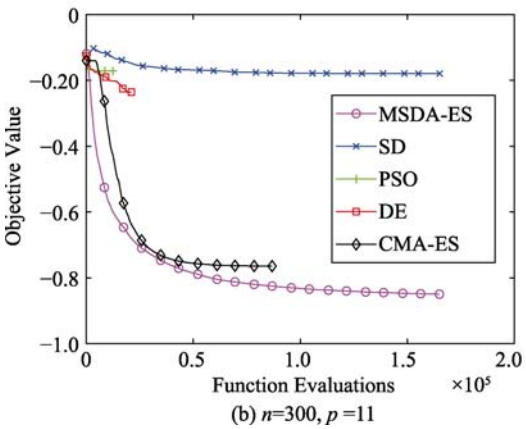


(b) n=3, p=500

图 4 Thomson 问题上的收敛曲线



(a) n=50, p =3



(b) n=300, p =11

图 5 SDR 问题上的收敛曲线

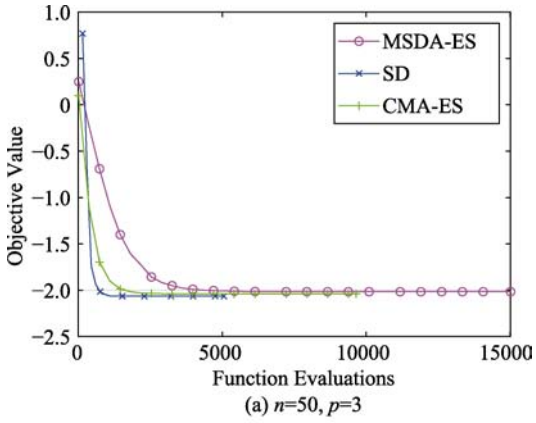
（2）MSDA-ES 在所有测试用例中具有相似的收敛行为. 与之相反，SD 对目标函数的可微性高度敏感，其在不可微的 SDR 问题上几乎在初始化后立刻就陷入了停滞状态，而在其他问题上的收敛速度则相对较快. DE 和 PSO 的收敛行为受到问题类型和问题规模的影响，例如，由图 5 可知，PSO 和 DE 在问题规模变大时更易提前收敛.

（3）在相对较简单的低维 QM 问题上，MSDA-ES 和 CMA-ES 的收敛速度较 SD 更慢，但 SD 的速度优势在问题维度升高后则不再显著. 其可能的原因在于 SD 中梯度估计时所产生的误差在运行中不断累积，导致 SD 在接近最优解时难以达到较高的精度.

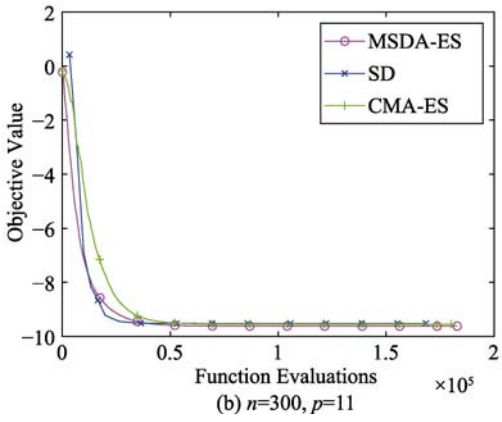
4.3.5 综合性能

上述实验仅反映了算法在特定测试用例上的性能，本节使用数据剖面图和多重检验测试所提出的算法在所有测试用例上的综合性能.

（1）数据剖面图. 使用数据剖面图（见图 7）展示各个算法在给定问题能够成功求解的测试用例的比例. 由于 PSO 和 DE 无法应用于 QM 问题，而在 SDR 和 Thomson 问题上也无法成功求解任何一个测试用例，因此它们并未绘制在数据剖面图中.



(a) n=50, p=3



(b) n=300, p=11

图 6 QM 问题上的收敛曲线

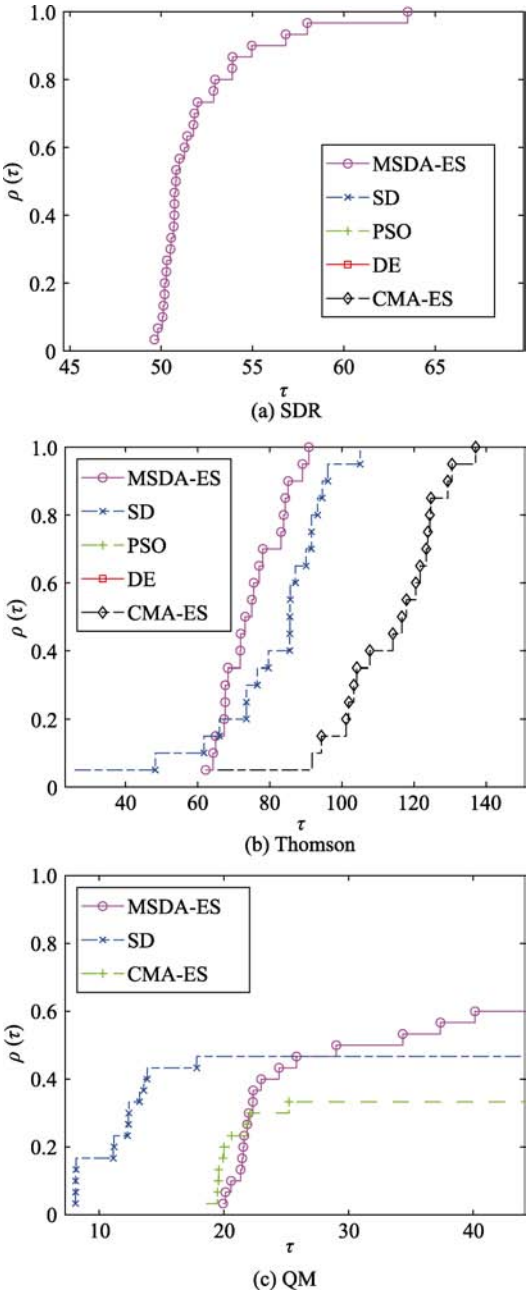


图 7　数据剖面图

SDR 问题（见图 7(a)）具有不可微的目标函数，因此在三个测试问题中求解难度最大. 除 MSDA-ES 以外的所有对比算法（包括同样使用演化策略框架的 CMA-ES）均无法求解任何一个测试用例. 因此，仅有 MSDA-ES 的数据剖面曲线反映在该图中.

对于图 7(b)所示的 Thomson 问题，MSDA-ES 和 SD 的数据剖面曲线相互重叠，说明它们在运行的不同阶段有各自的优势. 例如，当 $50<\tau<60$ 时五个算法中仅有 SD 能够求解出约 10%的测试用例，而当$\tau>60$ 时，MSDA-ES 的数据剖面曲线总体上位于 SD 的数据剖面曲线的左上方，表明当允许使用更多的计算资源时，MSDA-ES 能够求解出更多的问题. 其原因可能是，尽管 Thomson 问题具有可微的目标函数，但其并非光滑，因此 MSDA-ES 比 SD 具有性能优势. CMA-ES 的整个数据剖面曲线都位于 MSDA-ES 的右侧，虽然其最终可以求解所有问题，但达到相同比例时大约比 MSDA-ES 多使用一倍的计算资源，表明其收敛速度较慢.

在图 7(c)所示的 QM 问题上可以观察到，SD 仅需要使用极少的计算资源即可求解相当数量的问题，而 MSDA-ES 在获得充分计算资源时往往可以求解更多的问题. 在 τ=20 时，CMA-ES 比 MSDA-ES 能够求解稍多的问题，而在此之后所能求解的问题反而最少. 考虑到 QM 问题相对简单，MSDA-ES 在运行初期收敛速度较慢的原因可能在于更新协方差矩阵需要较多的计算资源，因此，提升协方差矩阵适应机制的效率可作为未来的一个研究方向.

（2）多重检验. 本节使用 Friedman 测试和 Bonferroni 后置过程对所有算法在给定测试问题的所有测试用例上进行多重检验. 相比于前文中直接统计 Wilcoxon 秩和检验在每个测试用例上的结果，多重检验可降低族内误差，体现算法的整体性能. 表 5 给出了在三个不同测试问题上的多重检验结果. 根据 Friedman 测试，MSDA-ES 在三个不同的测试问题上均排名第一. 根据 Bonferroni 后置过程，当置信水平为 0.05 时，MSDA-ES 在 SDR 问题上显著优于 SD、PSO 以及 DE，在 Thomson 问题上显著优于 PSO、DE 和 CMA-ES. 若将置信水平设置为 0.1，则 MSDA-ES 在 SDR 问题上还能够显著优于 CMA-ES.

4.3.6　问题规模对算法性能的影响

本节研究 MSDA-ES 关于流形维度的可扩展性. 对于每个测试问题，将所有测试用例的流形维度从小到大排序，等分为小规模、中规模和大规模三组. 在每一分组中，使用 4.3.5 节中使用的多重检验对所有算法的性能进行分析，所得结果呈现在表 6 中.

从表 6 可知，在 SDR 和 Thomson 问题上，MSDA-ES 在所有分组中均取得了最好的平均排名，其他四个对比算法的平均排名结果在三个不同分组中保持一致，且与表 5 中的结果基本相符. 这说明在上述两个问题上，所有算法对问题规模都不敏感. 产生这一结果的原因是由于上述两个问题的困难来源于目标函数本身的非光滑性而非决策空间的复杂性. 因此，算法性能主要由算法处理复杂目标函数的能力决定，受到流形维度的影响较小.

相比于 SDR 和 Thomson 问题，QM 问题的目标函数非常简单，且本质上等价于凸优化问题. 在该问题上，算法性能受到目标函数性质的影响较

表 5 不同测试问题上的多重检验结果(根据 **Friedman** 检验结果，排名第一的算法使用粗体标注)

	SDR		Thomson		QM	
	Rank	p-Value	Rank	p-Value	Rank	p-Value
MSDA-ES	**10.52**		**14.37**		**29.63**	
SD	81.93	0.00	29.16	1.00	30.72	1.00
PSO	74.22	0.00	70.50	0.00	N/A	N/A
DE	55.35	0.00	90.50	0.00	N/A	N/A
CMA-ES	30.48	0.08	47.97	0.00	31.15	1.00

注:“Rank”表示在给定测试问题上,根据 Friedman 检验得到的平均排名. “p-Value“表示使用 Bonferroni 后置过程进行矫正后,MSDA-ES 与对比算法在所有测试用例上性能差异的显著性水平. 由于 PSO 和 DE 无法应用于 QM 问题，相关数据使用 N/A 标准. 这些符号在本文其他表格中具有相同含义.

表 6 不同规模的测试问题上的多重检验结果(根据 **Friedman** 检验结果，排名第一的算法使用粗体标注)

		小规模		中规模		大规模	
		Rank	p-Value	Rank	p-Value	Rank	p-Value
SDR	MSDA-ES	**10.55**		**10.53**		**10.50**	
	SD	85.05	0.00	81.60	0.00	79.15	0.00
	PSO	68.87	0.00	75.88	0.00	77.91	0.00
	DE	57.57	0.00	54.03	0.01	54.44	0.01
	CMA-ES	30.48	1.00	30.48	1.00	30.50	1.00
Thomson	MSDA-ES	**18.22**		**13.01**		**11.46**	
	SD	26.36	1.00	29.98	1.00	31.47	1.00
	PSO	70.50	0.01	70.50	0.00	70.50	0.00
	DE	90.50	0.00	90.50	0.00	90.50	0.00
	CMA-ES	46.91	0.64	48.51	0.22	48.58	0.27
QM	MSDA-ES	30.80		**29.25**		**28.85**	
	SD	30.46	1.00	30.98	1.00	30.72	1.00
	CMA-ES	**30.25**	1.00	31.27	1.00	31.94	1.00

小，而受到流形维度的影响相对较大，因此，在该测试问题上呈现出了与在 SDR 和 Thomson 问题上完全不同的实验结果. 如表 6 所示，MSDA-ES 在小规模分组中排名最差,但在中规模和大规模分组中排名最好，可见，MSDA-ES 算法性能受到流形维度的影响,且当维度较高时具有一定的性能优势. MSDA- ES的这一性能优势主要来源于其使用的流形搜索方向适应机制. 该机制仅需使用少量的几个切向量即可描述切空间中的线性依赖关系,因此可在高维流形中捕获有效的搜索方向,实现较快的搜索.

5 结 论

本文将经典的 CMA-ES 算法从欧式空间推广至黎曼流形之中，提出了一个用于求解矩阵流形优化问题的搜索方向适应演化策略 MSDA-ES. MSDA-ES 使用回撤算子处理流形约束,将原始的流形约束问题转换成为一系列切空间中的无约束问题. 在每个切空间中,MSDA-ES 使用一个单位阵和若干个搜索方向构造协方差矩阵，并使用相应的多元高斯分布引导搜索. 本文在涉及三种矩阵流形的经典问题中测试了 MSDA-ES 的性能，实验结果表明，MSDA-ES 比已有算法在黑盒优化任务中具有显著优势. 下一阶段,我们计划将 MSDA-ES 推广至幂流形、积流形等更为复杂的拓扑结构. 同时，求解多目标矩阵流形优化也是未来的研究任务之一.

参 考 文 献

[1] Vandereycken B. Low-rank matrix completion by Riemannian optimization. SIAM Journal on Optimization, 2013, 23(2): 1214-1236

[2] Cambier L, Absil P. Robust low-rank matrix completion by Riemannian optimization. SIAM Journal on Scientific Computing, 2016, 38(5): S440-S460

[3] Keshavan R H, Montanari A, Oh S. Matrix completion from a few entries. IEEE Transactions on Information Theory, 2010, 56(6): 2980-2998

[4] Bradbury W W, Fletcher R. New iterative methods for solution of the eigenproblem. Numerische Mathematik, 1966, 9(3): 259- 267
[5] Zhao Z, Bai Z-J, Jin X-Q. A Riemannian newton algorithm for nonlinear eigenvalue problems. SIAM Journal on Matrix Analysis and Applications, 2015, 36(2): 752-774
[6] Borckmans P B, Absil P. Oriented bounding box computation using particle swarm optimization//Proceedings of the 18th European Symposium on Artificial Neural Networks. Bruges, Belgium, 2010
[7] Rudin L I, Osher S, Fatemi E. Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 1992, 60(1–4): 259-268
[8] Yang C, Meza J C, Wang L-W. A constrained optimization algorithm for total energy minimization in electronic structure calculations. Journal of Computational Physics, 2006, 217(2): 709-721
[9] von Luxburg U. A tutorial on spectral clustering. Statistics and Computing, 2007, 17(4): 395-416
[10] Edelman A, Arias T A, Smith S T. The geometry of algorithms with orthogonality constraints. SIAM Journal on Matrix Analysis and Applications, 1998, 20(2): 303-353
[11] Spivak M. A Comprehensive Introduction to Differential Geometry. The 3rd edition. Houston, USA: Publish or Perish, 1999
[12] Wen Z, Yin W. A feasible method for optimization with orthogonality constraints. Mathematical Programming, 2013, 142(1-2): 397-434
[13] Manton J H. Optimization algorithms exploiting unitary constraints. IEEE Transactions on Signal Processing, 2002, 50(3): 635-650
[14] Absil P, Mahony R, Sepulchre R. Optimization algorithms on matrix manifolds. Princeton, USA: Princeton University Press, 2008
[15] Absil P, Malick J. Projection-like retractions on matrix manifolds. SIAM Journal on Optimization, 2012, 22(1): 135-158
[16] Jiang B, Dai Y-H. A framework of constraint preserving update schemes for optimization on Stiefel manifold. Mathematical Programming, 2015, 153(2): 535-575
[17] Gao B, Liu X, Chen X, et al. A new first-order algorithmic framework for optimization problems with orthogonality constraints. SIAM Journal on Optimization, 2018, 28(1): 302-332
[18] Ring W, Wirth B. Optimization methods on Riemannian manifolds and their application to shape space. SIAM Journal on Optimization, 2012, 22(2): 596-627
[19] Huang W, Absil P, Gallivan K A. A Riemannian symmetric rank-one trust-region method. Mathematical Programming, 2015, 150(2): 179-216
[20] Hosseini S, Huang W, Yousefpour R. Line search algorithms for locally Lipschitz functions on Riemannian manifolds. SIAM Journal on Optimization, 2018, 28(1): 596-619
[21] Huang W, Absil P, Gallivan K. A Riemannian BFGS method without differentiated retraction for nonconvex optimization problems. SIAM Journal on Optimization, 2018, 28(1): 470-495
[22] Absil P, Mahony R, Sepulchre R. Optimization on manifolds: methods and applications//Proceedings of the 14th Belgian- French-German Conference on Optimization (BFG09). Leuven, Belgium, 2009: 125-144
[23] Huang W. Optimization algorithms on Riemannian manifolds with applications [Ph. D. Thesis]. Florida State University, Tallahassee, USA: 2013
[24] Boussaïd I, Lepagnot J, Siarry P. A survey on optimization metaheuristics. Information Sciences, 2013, 237: 82-117
[25] Storn R, Price K. Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 1997, 11(4): 341-359
[26] Beyer H-G, Schwefel H-P. Evolution strategies–A comprehensive introduction. Natural computing, 2002, 1(1): 3-52
[27] Goldberg D E. Genetic algorithms in search, optimization, and machine learning. Reading, Mass: Addison-Wesley Pub. Co, 1989
[28] Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation, 2001, 9(2): 159-195
[29] Banks A, Vincent J, Anyakoha C. A review of particle swarm optimization. Part I: background and development. Natural Computing, 2007, 6(4): 467-484
[30] Borckmans P B, Ishteva M, Absil P-A. A modified particle swarm optimization algorithm for the best low multilinear rank approximation of higher-order tensors//Proceedings of the 7th International Conference on Swarm Intelligence. Brussels, Belgium, 2010: 13-23
[31] Colutto S, Fruhauf F, Fuchs M, et al. The CMA-ES on Riemannian manifolds to reconstruct shapes in 3-D voxel images. IEEE Transactions on Evolutionary Computation, 2010, 14(2): 227-245
[32] Arnold D V. On the use of evolution strategies for optimization on spherical manifolds. Parallel Problem Solving from Nature – PPSN XIII. Ljubljana, Slovenia, 2014: 882-891
[33] Arnold D V, Lu A. An evolutionary algorithm for depth image based camera pose estimation in indoor environments// Proceedings of the 2016 Congress on Evolutionary Computation. Vancouver, BC, Canada, 2016: 3871-3878
[34] Stiefel E. Richtungsfelder und fernparallelismus in n-dimensionalen mannigfaltigkeiten. ETH Zurich, 1935
[35] Boumal N, Mishra B, Absil P A, et al. Manopt, a matlab toolbox for optimization on manifolds. Journal of Machine Learning Research, 2013, 15(1): 1455-1459
[36] Hansen N, Niederberger A S P, Guzzella L, et al. A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion. IEEE Transactions on Evolutionary Computation, 2009, 13(1): 180-197
[37] Beyer H-G, Finck S. On the design of constraint covariance matrix self-adaptation evolution strategies including a cardinality constraint. IEEE Transactions on Evolutionary Computation, 2012, 16(4): 578-596
[38] Suttorp T, Hansen N, Igel C. Efficient covariance matrix update for variable metric evolution strategies. Machine Learning, 2009, 75(2): 167-197
[39] Schäfer J, Strimmer K. A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics. Statistical Applications in Genetics and Molecular Biology, 2005, 4(1)
[40] Loshchilov I. LM-CMA: An alternative to L-BFGS for large-scale black box optimization. Evolutionary Computation, 2017, 25(1): 143-171
[41] Nikolaus Hansen, Sibylle D. Müller, Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary Computation, 2003, 11(1): 1-18

[42] Beyer H-G. Convergence analysis of evolutionary algorithms that are based on the paradigm of information geometry. Evolutionary Computation, 2014, 22(4): 679-709
[43] Broomhead D S, Kirby M J. Dimensionality reduction using secant-based projection methods: the induced dynamics in projected systems. Nonlinear Dynamics, 2005, 41(1–3): 47-67
[44] Absil P, Hosseini S. A collection of nonsmooth Riemannian optimization problems. Nonsmooth Optimization and Its Applications. Cham: Springer International Publishing, 2019: 1-15
[45] Thomson J J. On the structure of the atom: an investigation of the stability and periods of oscillation of a number of corpuscles arranged at equal intervals around the circumference of a circle; with application of the results to the theory of atomic structure. Philosophical Magazine, 1904, 7(39): 237-265
[46] Polak E. Optimization: algorithms and consistent approximations. 1997 edition. New York: Springer, 1997
[47] Das S, Suganthan P N. Differential evolution: a survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 4-31
[48] Loshchilov I, Glasmachers T, Beyer H G. Large scale black-box optimization by limited-memory matrix adaptation. IEEE Transactions on Evolutionary Computation, 2019, 23(2): 353-358
[49] van den Bergh F, Engelbrecht A P. A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation, 2004, 8(3): 225-239
[50] Sepesy Maučec M, Brest J. A review of the recent use of Differential Evolution for Large-Scale Global Optimization: An analysis of selected algorithms on the CEC 2013 LSGO benchmark suite. Swarm and Evolutionary Computation, 2019, 50: 100428
[51] Vesterstrom J, Thomsen R. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems//Proceedings of the 2004 Congress on Evolutionary Computation. Portland, USA, 2004: 1980-1987
[52] Li X, Yao X. Tackling high dimensional nonseparable optimization problems by cooperatively coevolving particle swarms// Proceedings of the 2009 Congress on Evolutionary Computation. Trondheim, Norway, 2009: 1546-1553
[53] Li X, Yao X. Cooperatively coevolving particle swarms for large scale optimization. IEEE Transactions on Evolutionary Computation, 2012, 16(2): 210-224
[54] Molina D, LaTorre A, Herrera F. An insight into bio-inspired and evolutionary algorithms for global optimization: review, analysis, and lessons learnt over a decade of competitions. Cognitive Computation, 2018, 10(4): 517-544

附录 1. 命题 1 的证明

假设 $\hat{S}^{(g)}=G(S^{(g)})$，使用归纳法证明 $\hat{S}^{(g+1)}=G(S^{(g+1)})$.

(1)首先，证明 $\hat{\boldsymbol{X}}^{(g+1)}=\boldsymbol{X}^{(g+1)}\boldsymbol{V}$.

根据假设 $\hat{\boldsymbol{L}}^{(g)}=\boldsymbol{L}^{(g)}\boldsymbol{V},\hat{z}_i^{(\mathrm{g})}=z_i^{(\mathrm{g})}$，有

$$\begin{aligned}\hat{\boldsymbol{Y}}_i^{(\mathrm{g})}&=\hat{\sigma}^{(\mathrm{g})}(\sqrt{1-c_{cov}}P_{\hat{\boldsymbol{X}}}^{(\mathrm{g})}(\hat{\boldsymbol{L}}^{(\mathrm{g})})+\sqrt{c_{cov}}\sum_{i=1}^{m}\boldsymbol{Q}_i^{(\mathrm{g})}\hat{z}_i^{(\mathrm{g})})\\&=\sigma^{(\mathrm{g})}(\sqrt{1-c_{cov}}P_{\boldsymbol{X}}^{(g)}(\boldsymbol{L}^{(\mathrm{g})}\boldsymbol{V})+\sqrt{c_{cov}}\sum_{i=1}^{m}\boldsymbol{Q}_i^{(\mathrm{g})}\boldsymbol{V}z_i^{(\mathrm{g})})\\&=\sigma^{(\mathrm{g})}(\sqrt{1-c_{cov}}P_{\boldsymbol{X}}^{(g)}(\boldsymbol{L}^{(\mathrm{g})})+\sqrt{c_{cov}}\sum_{i=1}^{m}\boldsymbol{Q}_i^{(\mathrm{g})}z_i^{(\mathrm{g})})\boldsymbol{V}\\&=\boldsymbol{Y}_i^{(\mathrm{g})}\boldsymbol{V}.\end{aligned}$$

其中，上式第二行利用了 Grassmann 流形上切空间投影的性质 $P_{\boldsymbol{X}^{(\mathrm{g})}}=P_{\boldsymbol{X}^{(\mathrm{g})}\boldsymbol{V}}$.

根据回撤算子的定义，可知

$R_{\hat{\boldsymbol{X}}^{(\mathrm{g})}}(\hat{\boldsymbol{Y}}_i^{(g)})=R_{\boldsymbol{X}^{(\mathrm{g})}\boldsymbol{V}}(\boldsymbol{Y}_i^{(\mathrm{g})}\boldsymbol{V})=\ R_{\boldsymbol{X}^{(\mathrm{g})}}(\boldsymbol{Y}_i^{(\mathrm{g})})\boldsymbol{V}$，进而有

$$f(R_{\hat{\boldsymbol{X}}^{(\mathrm{g})}}(\hat{\boldsymbol{Y}}_i^{(\mathrm{g})}))=f(R_{\boldsymbol{X}^{(\mathrm{g})}}(\boldsymbol{Y}_i^{(\mathrm{g})})\boldsymbol{V})=f(R_{\boldsymbol{X}^{(\mathrm{g})}}(\boldsymbol{Y}_i^{(\mathrm{g})})).$$

因此，A_1 和 A_2 对应的候选矩阵具有相同的列空间，被映射至流形上的同一位置，具有相同的目标函数值.

由于 A_1 和 A_2 的群体具有相同的目标值和排序结果，因此在重组时，有

$$\hat{\bar{\boldsymbol{Y}}}=\sum_{i=1}^{\mu}\omega_i\hat{\boldsymbol{Y}}_{i:\lambda}^{(\mathrm{g})}=\sum_{i=1}^{\mu}\omega_i\boldsymbol{Y}_{i:\lambda}^{(\mathrm{g})}\boldsymbol{V}=\bar{\boldsymbol{Y}}\boldsymbol{V},$$

$$\hat{\boldsymbol{X}}^{(\mathrm{g}+1)}=R_{\hat{\boldsymbol{X}}^{(\mathrm{g})}}(\hat{\bar{\boldsymbol{Y}}})=R_{X^{(\mathrm{g})}V}(\bar{\boldsymbol{Y}}\boldsymbol{V})=\boldsymbol{X}^{(\mathrm{g}+1)}\boldsymbol{V}.$$

(2)其次，证明 $\hat{\boldsymbol{Q}}_i^{(g+1)}=\boldsymbol{Q}_i^{(g+1)}\boldsymbol{V}\left(i=1,\ldots,m\right)$.

在更新第一个搜索方向时，

$$\hat{\boldsymbol{Z}}_1=\sqrt{\mu_{eff}}\,\frac{\hat{\bar{\boldsymbol{Y}}}}{\hat{\sigma}^{(\mathrm{g})}}=\sqrt{\mu_{eff}}\,\frac{\bar{\boldsymbol{Y}}\boldsymbol{V}}{\sigma^{(\mathrm{g})}}=\boldsymbol{Z}_1\boldsymbol{V}$$，可知 A_1 和 A_2 中成功变异方向具有相同列空间，因此后续更新得到的搜索方向同样具有相同列空间. 具体而言，对 A_2 中的第一个搜索方向更新可得

$$\begin{aligned}\hat{\boldsymbol{Q}}_1'&=(1-c_c)\hat{\boldsymbol{Q}}_1^{(\mathrm{g})}+\sqrt{(2-c_c)c_c}\,\hat{\boldsymbol{Z}}_1\\&=(1-c_c)\boldsymbol{Q}_1^{(\mathrm{g})}\boldsymbol{V}+\sqrt{(2-c_c)c_c}\,\boldsymbol{Z}_1\boldsymbol{V}=\boldsymbol{Q}_1'\boldsymbol{V}.\end{aligned}$$

进而有

$$\begin{aligned}\hat{t}_1&=\frac{\langle\hat{\boldsymbol{Z}}_1,\hat{\boldsymbol{Q}}_1'\rangle_{\hat{\boldsymbol{X}}^{(g)}}}{\langle\hat{\boldsymbol{Q}}_1',\hat{\boldsymbol{Q}}_1'\rangle_{\hat{\boldsymbol{X}}^{(g)}}}=\frac{\langle\boldsymbol{Z}_1\boldsymbol{V},\boldsymbol{Q}_1\boldsymbol{V}\rangle_{\hat{\boldsymbol{X}}^{(g)}}}{\langle\hat{\boldsymbol{Q}}_1'\boldsymbol{V},\boldsymbol{Q}_1V\rangle_{\hat{\boldsymbol{X}}^{(g)}}}=\frac{\langle\boldsymbol{Z}_1,\boldsymbol{Q}_1'\rangle_{\hat{\boldsymbol{X}}^{(g)}}}{\langle\boldsymbol{Q}_1',\boldsymbol{Q}_1'\rangle_{\hat{\boldsymbol{X}}^{(g)}}}\\&=\frac{\langle\boldsymbol{Z}_1,\boldsymbol{Q}_1'\rangle_{\boldsymbol{X}^{(g)}}}{\langle\boldsymbol{Q}_1',\boldsymbol{Q}_1'\rangle_{\boldsymbol{X}^{(g)}}}=t_1.\end{aligned}$$

即，成功变异方向在更新后的第一个搜索方向上的相对投影长度相同. 同时，由

$$\hat{\boldsymbol{Z}}_2=\frac{1}{\sqrt{1+\hat{t}_1^2}}(\hat{\boldsymbol{Z}}_1-\hat{t}_1\hat{\boldsymbol{Q}}_1^{[g+1]})=\boldsymbol{Z}_2V$$

用于更新第二个搜索方向的成功变异方向同样具有相同的列空间，同理可得，

$$\hat{\boldsymbol{Q}}_i'=\boldsymbol{Q}_i'\boldsymbol{V},i=1,\ldots,m.$$

因此，对 $i=1,\ldots,m$，有

$$\hat{\boldsymbol{Q}}_i^{(g+1)}=P_{\hat{\boldsymbol{X}}^{(g+1)}}(\hat{\boldsymbol{Q}}_i')=P_{\boldsymbol{X}^{(g+1)}V}(\boldsymbol{Q}_i'\boldsymbol{V})=P_{\boldsymbol{X}^{(g+1)}}(\boldsymbol{Q}_i')\boldsymbol{V}=\boldsymbol{Q}_i^{(g+1)}\boldsymbol{V}.$$

(3)最后，由于 A_1 和 A_2 中的群体具有相同的目标函数值，因此连续两代群体目标函数值的秩和差值相同. 由于

变异强度适应机制仅依赖于该秩和差值，因此有

$$\hat{s}^{(g+1)} = s^{(g+1)}, \hat{\sigma}^{(g+1)} = \sigma^{(g+1)}.$$

证毕.

附录 2. 用于矩阵流形优化的 DE 算法

算法 2. DE

输入：$\mathcal{M}$：矩阵流形；$f: R^{n\times p} \to R$：目标函数；$Exp_{\boldsymbol{X}}: T_{\boldsymbol{X}}\mathcal{M} \to \mathcal{M}$：指数映射；$Log_{\boldsymbol{X}}: \mathcal{M} \to T_{\boldsymbol{X}}\mathcal{M}$：对数映射；$N_p$：群体规模；$F$：差分向量缩放因子；$k$：算术交叉系数；$P_F$：执行变异的概率.

输出：已找到的最优解.

FOR ($i \leftarrow 1: N_p$)　$\boldsymbol{X}_i \leftarrow \mathcal{M}$ 中随机分布的矩阵
WHILE(未满足终止条件){
FOR ($i \leftarrow 1: N_p$) {
　$j_1, j_2, j_3 \leftarrow \{1,2,\ldots,N_p\}$ 中不同的三个随机数
　IF($rand < P_F$)
　$\boldsymbol{U} = \mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_1}) + F(\mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_2}) - \mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_3}))$
　ELSE
　$\boldsymbol{U} = \mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_1}) + k(\mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_2}) + \mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_3}) - 2\mathrm{Log}_{\boldsymbol{X}_i}(\boldsymbol{X}_{j_1}))$
　IF($f(Exp_{\boldsymbol{X}_i}(\boldsymbol{U})) < f(\boldsymbol{X}_i)$)　$\boldsymbol{X}_i \leftarrow Exp_{\boldsymbol{X}_i}(\boldsymbol{U})$
　}
}

rand 指[0,1]上均匀分布的随机数.

附录 3. 用于矩阵流形优化的CMA-ES算法

算法 3. CMA-ES

输入：$\mathcal{M}$：矩阵流形；$f: R^{n\times p} \to R$：目标函数；$R_{\boldsymbol{X}}: T_{\boldsymbol{X}}\mathcal{M} \to \mathcal{M}$：回撤算子；$P_{\boldsymbol{X}}: R^{n\times p} \to T_{\boldsymbol{X}}\mathcal{M}$：切空间投影.

输出：已找到的最优解.

$g \leftarrow 0$
$\boldsymbol{X}^{(g)} \leftarrow \mathcal{M}$ 中随机分布的矩阵，$\sigma^{(g)} \leftarrow 1$
$p_\sigma^{(g)} \leftarrow \boldsymbol{0}_{n\times p}$，$\boldsymbol{Q}_i^{(g)} \leftarrow \boldsymbol{0}_{n\times p}$，$i \in \{1,\cdots,m\}$
WHILE(未满足终止条件){
　FOR ($i \leftarrow 1:\lambda$) {
　　$\boldsymbol{Z}_i \leftarrow P_{X^{(g)}}(\boldsymbol{L})$
　　$\boldsymbol{Y}_i \leftarrow \boldsymbol{Z}_i$
　FOR ($j \leftarrow 1: m$)
　　$\boldsymbol{Y}_i \leftarrow (1-c_{d,j})\boldsymbol{Y}_i + c_{d,j}\boldsymbol{Q}_j^{(g)}(\boldsymbol{Q}_j^{(g)T}\boldsymbol{Y}_i)$
　}
　$F^{(g)} = \left\{ f(R_{\boldsymbol{X}^{(g)}}(\sigma^{(g)}\boldsymbol{Y}_1)), \cdots, f(R_{\boldsymbol{X}^{(g)}}(\sigma^{(g)}\boldsymbol{Y}_\lambda)) \right\}$
　将 $F^{(g)}$ 按升序排列
　$z \leftarrow \sqrt{\mu_{eff}} \sum_{i=1}^{\mu} \omega_i \boldsymbol{Z}_{i:\lambda}$
　$X^{(g+1)} \leftarrow R_{\boldsymbol{X}^{(g)}}\left(\sum_{i=1}^{\mu} \omega_i \sigma^{(g)} \boldsymbol{Y}_{i:\lambda} \right)$
　$\boldsymbol{p}_\sigma^{(g+1)} = (1-c_\sigma)\boldsymbol{p}_\sigma^{(g)} + \sqrt{c_\sigma(2-c_\sigma)}\boldsymbol{z}$
　FOR ($j \leftarrow 1:$ m)
　　$\boldsymbol{Q}_j^{(g+1)} = P_{\boldsymbol{X}^{(g+1)}}((1-c_{c,j})\boldsymbol{Q}_j^{(g)} + \sqrt{c_{c,j}(2-c_{c,j})}\boldsymbol{z})$
　$\sigma^{(g+1)} \leftarrow \sigma^{(g)} exp\left(\frac{c_\sigma}{2}\left(\frac{|\boldsymbol{p}_\sigma^{(g+1)}|^2}{dim\mathcal{M}} - 1 \right) \right)$
}

HE Xiao-Yu, Ph.D., His research interests focus on evolutionary computation.

ZHOU Yu-Ren, Ph.D., professor. His research interests focus on evolutionary computation.

CHEN Ze-Feng, Ph.D., His research interests focus on evolutionary computation.

Background

This work focuses on the manifold constrained matrix optimization problems which abound in data mining, numerical linear algebra, signal processing, and electronic design. Most existing approaches are white-box, thereby the efficiency and applicability strongly dependent on the problem structure and the inherent mathematic properties. The algorithm presented here, to the best of our knowledge, is the first retraction-based evolutionary manifold optimizer. This algorithm is a general-purpose solver that can be applied to most kinds of manifolds whenever the retraction operator and tangent space projection are well defined. This work gives some inspirations on black-box approaches to complicated matrix manifold problems which may not be handled easily with traditional algorithms.

This work was supported by National Natural Science Foundation of China (61773410, 61673403) and China Postdoctoral Science Foundation (2019M663234).