Determining Step Sizes in Geometric Optimization Algorithms

Zhizhong Li[1,2], Deli Zhao[2], Zhouchen Lin[3,4*], Edward Y. Chang[5]
[1]School of Mathematics, Peking University, Beijing, China
[2]Advanced Algorithm Research Group, HTC, Beijing, China
[3]Key Laboratory of Machine Perception (MOE), School of EECS, Peking University, Beijing, China
[4]Cooperative Medianet Innovation Center, Shanghai, China
[5]HTC Research
lizz@pku.edu.cn, zhaodeli@gmail.com, zlin@pku.edu.cn, eyuchang@gmail.com

***Abstract*—Optimization on Riemannian manifolds is an intuitive generalization of the traditional optimization algorithms in Euclidean spaces. In these algorithms, minimizing along a search direction becomes minimizing along a search curve lying on a manifold. Computing such a curve to be subsequently searched upon is itself computational intensive. We propose a new minimization scheme aiming to find a better step size utilizing the first order information of the search curve. We prove that this scheme can provide further reduction for the cost function when the retraction and the vector transport are collinear. Then we adapt this scheme to propose a heuristic strategy for line search. In numerical experiments, we apply this heuristic strategy to one of the geometric algorithms for matrix completion and show its feasibility and the potential in accelerating computation.**

I. Introduction

Geometric optimization algorithms are generalizations of the traditional optimization methods in Euclidean spaces to Riemannian manifolds. Geometric concepts such as geodesic, Riemannian gradient, Riemannian Hessian and parallel transport are employed to replace the usual notions of straight line, gradient, Hessian and parallel translation. Thus, gradient descent method, conjugate gradient method, Newton's method and trust-region methods can be applied to manifolds without conceptual difficulties. Theories on optimizations over manifolds, especially on matrix manifolds, can be found in [1]–[6]. A Matlab toolbox for optimization on Riemannian manifolds that implements several general solvers and provides some familiar manifolds has also been developed [7].

One advantage of optimizing on manifolds is that it naturally converts a constrained optimization problems to an unconstrained one, given that the constraint conditions possess some good properties. Since these constraints are used to define manifold structures, they become invisible to solvers. The fixed-rank constraint for matrices is one of the examples that embrace manifold structures. Many geometric algorithms have been proposed to solve problems such as the low-rank matrix completion problem [8]–[11] and the regressions under fixed-rank constraints [12], [13].

Suppose that M is a Riemannian manifold and f is a continuously differentiable cost function defined on M. Our work focuses on determining the step size s_* along a descent search direction $\eta_0 \in T_{x_0}M$ at an iterate point $x_0 \in M$, where $T_{x_0}M$ is the tangent space of M at point x_0. This is a common subtask among the various optimization algorithms mentioned above and we refer to this step size selection task by "line search" in this paper. Mathematically, given the search curve $\phi(t)$ along direction η_0 which satisfies $\phi(0) = x_0$ and $\dot{\phi}(0) = \eta_0$, we want to choose a step size s_* such that the point $\phi(s_*)$ gives a sufficient reduction of the cost function compared to the value $f(x_0)$ at point x_0.

The basic line search strategy is setting a fixed initial step size s_0 and backtracking to meet the Armijo condition [14, Algorithm 3.1]. Another strategy to get the initial step size is by interpolation as described in [14, Equation (3.60)], and then backtracking. An adaptive step size strategy can also be used [15, Equation (17)]. The key idea for this adaptive strategy is to increase, keep, or decrease the step size based on the number of backtrackings in the last iteration.

If the manifold M is a submanifold and the cost function f can be extended to the ambient space $\overline{M}$, then a better initial guess for the step size could be made by taking the tangent line $\gamma(t) := x_0 + t\eta_0$ passing through x_0 as the first order approximation for the search curve ϕ. This is based on the assumption that the cost function f restricted on the straight line γ is much easier to compute and minimize. Algorithms in [8], [9], together with many other examples implementing this line search strategy have shown its effectiveness.

We further extend this idea and propose a minimization scheme (Algorithm 1), which aims to provide further reduction of the cost function for the line search. We prove our scheme's validity under the assumption that the retraction and the vector transport being collinear. For general cases, we present a modified scheme (Algorithm 2), which is shown to be a good heuristic for the step sizes in our numerical experiments.

The rest of the paper is organized as follows. Section II elaborates the minimization scheme and Section III describes the one-step-further heuristic based on that scheme. We show an application of the heuristic strategy to matrix completion in Section IV, and Section V concludes the paper.

II. Minimizing along a direction

Developing practical geometric optimization algorithms relies on some non-standard differential geometric concepts such as retraction [6, Definition 4.1.1] and vector transport [6, Definition 8.1.1]. They are analogues of the exponential map

*Corresponding author.

and the parallel transport in Riemannian geometry, respectively, and have the advantage of lowering the computation cost.

Let $f : M \to \mathbb{R}$ be a continuously differentiable cost function defined on a manifold M that is a submanifold of the Euclidean space $\mathbb{R}^n$. A Riemannian metric g is assigned to M such that (M, g) becomes a Riemannian manifold. The metric g may be problem-tailored and need not coincide with the standard metric of the ambient Euclidean space. We denote the vector transport by $\mathcal{T}$ and the associated retraction by $\mathcal{R}$. The domain of the cost function is extended to a neighborhood of the submanifold M. Assume that this neighborhood is large enough for our discussion. By an abuse of notation, we denote this extended cost function by f.

Given a point $x_0 \in M$ and a search direction $\eta_0 \in T_{x_0}M$, the search curve ϕ is defined as

$$\phi(t) := \mathcal{R}_{x_0}(t\eta_0), \tag{1}$$

where $\mathcal{R}_{x_0}(t\eta_0)$ is the retraction of the vector $t\eta_0$ at point x_0. We want to determine a step size s_* such that the value $f(\phi(s_*))$ provides a sufficient reduction for the cost function. Since minimizing directly on the curve is generally difficult, one can use the assumption that M being a submanifold of a Euclidean space and f being extended to the ambient space to define the first order approximation of curve ϕ as

$$\gamma(t) := x_0 + t\eta_0, \tag{2}$$

which is the tangent line to ϕ at point x_0, and then minimize the cost function $f(\gamma(t))$ on the staight line γ instead of on the curve ϕ to get a candidate step size s_*. This method generally gives a good initial value for the step size.

We may want to continue this approximation process and get a better estimation of the step size. The initial idea is illustrated in Figure 1.

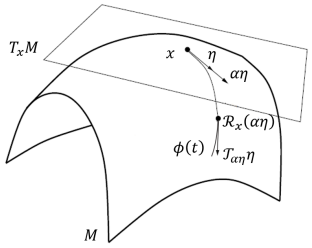


Fig. 1. Minimizing along the direction η on a manifold M. Suppose that the step size α is obtained by minimizing on the tangent line passing through $x \in M$ with direction $\eta \in T_xM$. Instead of using $x_1 := \mathcal{R}_x(\alpha\eta)$ as the next iterate point, the initial idea is to continue this minimizing process along this curve to get a better estimation of the step size based on point x_1. Vector $\mathcal{T}_{\alpha\eta}\eta$ is the direction η moved from x to x_1 by vector transport $\mathcal{T}$, which will serve as the minimizing direction for x_1.

Since definitions of retraction and vector transport focus on their local behaviors, we have to deal with several issues in order to make the above idea precise. The first one is that $\mathcal{T}_{\alpha\eta}\eta$ is not guaranteed to be a tangent vector of the curve $\phi(t) = \mathcal{R}_x(t\eta)$. This is fixed by introducing the notion of *collinearity*. A vector transport $\mathcal{T}$ is collinear with the associated retraction $\mathcal{R}$ given that vector $\mathcal{T}_{\alpha\eta}\eta$ is collinear with the tangent vector $\frac{\mathrm{d}}{\mathrm{d}t}\mathcal{R}_x(t\eta)|_{t=\alpha}$ at point $\mathcal{R}_x(\alpha\eta)$, where $x \in M$ is a point on M, $\eta \in T_xM$ is a tangent vector at point x, and $\alpha \in \mathbb{R}$ is a scalar. In Figure 1, if $\mathcal{T}$ is collinear with $\mathcal{R}$, then $\mathcal{T}_{\alpha\eta}\eta$ is tangent to the curve $\phi(t) = \mathcal{R}_x(t\eta)$ at point $\mathcal{R}_x(\alpha\eta)$. Parallel transport is collinear with exponential map on a Riemannian manifold. Another class of examples of collinearity can be found in [6, Section 8.1.2].

The second issue concerns with the non-degeneracy of $\mathcal{R}$ and $\mathcal{T}$ in a large range. Consider the following conditions:

1) Retraction $\mathcal{R}$ is non-degenerate along radial directions for all points $x \in M$. *i.e.*,
$$\left\| \frac{\mathrm{d}}{\mathrm{d}t}\Big|_{t=\alpha} \mathcal{R}_x(t\eta) \right\| \neq 0, \tag{3}$$
for all scalars $\alpha \in \mathbb{R}$ and all directions $\eta \in T_xM$.
2) Vector transport $\mathcal{T}$ is also non-degenerate along radial directions for all points $x \in M$ in the sense that
$$\|\mathcal{T}_{\alpha\eta}\eta\| \neq 0, \tag{4}$$
where $\eta \in T_xM$ is a direction at x and α is a scalar.

With these conditions, we propose a general minimization scheme in Algorithm 1.

Algorithm 1 Minimization scheme of Line Search

Input: Point $x_0 \in M$, descent direction $\eta_0 \in T_{x_0}M$ with $\|\eta_0\| = 1$, vector transport $\mathcal{T}$ that is collinear with retraction $\mathcal{R}$, and Equations (3) and (4) are satisfied.
Output: a sequence of step sizes $\{s_i\}_{i\geq 0}$.
1: Initialize $s_0 = 0$ and set $i = 0$.
2: Minimize f on the tangent line $\gamma_i(t) := x_i + t\eta_i$ of curve ϕ at point x_i to get a step size t_{i+1}, where $\eta_i \in T_{x_i}M$.
3: Stop if $t_{i+1} = 0$, when x_i is a minimizing point on γ_i.
4: Backtracking. Find the minimum integer j such that $f(\mathcal{R}_{x_0}((s_i + t_{i+1}/2^j)\eta_0)) < f(x_i)$.
5: Output $s_{i+1} = s_i + t_{i+1}/2^j$.
6: Set $x_{i+1} = \mathcal{R}_{x_0}(s_{i+1}\eta_0)$.
7: Let $\eta_{i+1} = \mathcal{T}_{s_{i+1}\eta_0}\eta_0$ and normalize it to be unit length.
8: Increase i by 1 and goto step 2.

Note that in Step 2 of the algorithm, an acceptable step size t_{i+1} should be at the decreasing side of $f(\gamma_i(t))$, *i.e.*, $t_{i+1} > 0$ if $\frac{\mathrm{d}}{\mathrm{d}t}f(\gamma_i(t))|_{t=0} > 0$ and vice versa. Also note that all the new iterate points are obtained by retractions at point x_0. This ensures that these points fall on the search curve starting at x_0. The next Proposition proves that Algorithm 1 works as it claims.

Proposition 1. *Under the specified conditions, Algorithm 1 either stop at a stationary point of f along the curve $\phi(t) := \mathcal{R}_{x_0}(t\eta_0)$, or generate a sequence of step size $\{s_i\}_{i\geq 0}$ such that the sequence of function values $\{f(x_i)\}_{i\geq 0}$ is a decreasing sequence, where $x_i := \phi(s_i) = \mathcal{R}_{x_0}(s_i\eta_0)$.*

Proof: The current iterate point is x_i and the current step size is s_i in the i-th iteration of Algorithm 1. Suppose $t_{i+1} \neq 0$ in Step 3. From the definition of the next iterate point x_{i+1} at Step 6, we can see that if the backtracking step (Step 4) succeeds, then the expected inequality $f(x_{i+1}) < f(x_i)$ holds. To ensure that Step 4 is always feasible, we need to show that t_{i+1} lies on the decreasing side of function $f(\phi_i(t))$, where $\phi_i(t)$ is defined as

$$\phi_i(t) := \phi(s_i + t) = \mathcal{R}_{x_0}((s_i + t)\eta_0). \tag{5}$$

This means that $t_{i+1} > 0$ if $\frac{\mathrm{d}}{\mathrm{d}t} f(\phi_i(t))|_{t=0} > 0$ and vice versa. Concerning the remark before this Proposition, it is equivalent to show that the signs of $\frac{\mathrm{d}}{\mathrm{d}t} f(\phi_i(t))|_{t=0}$ and $\frac{\mathrm{d}}{\mathrm{d}t} f(\gamma_i(t))|_{t=0}$ are the same.

It is trivial to verify that $\phi_i(0) = \gamma_i(0) = x_i$. For the given s_i, the tangent vector of $\phi_i(t)$ and $\gamma_i(t)$ at $t = 0$ are

$$\dot{\phi}_i(0) = \frac{\mathrm{d}}{\mathrm{d}t} \mathcal{R}_{x_0}((s_i + t)\eta_0)|_{t=0} = \frac{\mathrm{d}}{\mathrm{d}t} \mathcal{R}_{x_0}(t\eta_0)|_{t=s_i}, \quad (6)$$

and

$$\dot{\gamma}_i(0) = \eta_i = \mathcal{T}_{s_i\eta_0}\eta_0, \quad (7)$$

respectively. Since $\mathcal{T}$ is collinear with $\mathcal{R}$, we know that $\dot{\phi}_i(0)$ is collinear with $\dot{\gamma}_i(0)$. According to Equations (3) and (4), these two vectors have positive lengths, so there exists a nonzero scalar $k \neq 0$ such that

$$\dot{\phi}_i(0) = k\dot{\gamma}_i(0). \quad (8)$$

Thus the following equation holds,

$$\frac{\mathrm{d}}{\mathrm{d}t} f(\phi_i(t))|_{t=0} = k \frac{\mathrm{d}}{\mathrm{d}t} f(\gamma_i(t))|_{t=0}. \quad (9)$$

If we continuously vary s_i, the scalar k in Equation (8) will also change continuously. The facts that k never equals zero and $k = 1$ when $s_i = 0$ tell us that k will keep positive no matter how s_i changes. Equation (9) together with $k > 0$ guarantee that the signs of $\frac{\mathrm{d}}{\mathrm{d}t} f(\phi_i(t))|_{t=0}$ and $\frac{\mathrm{d}}{\mathrm{d}t} f(\gamma_i(t))|_{t=0}$ are the same.

If $t_{i+1} = 0$ in Step 3, we can see that x_i being a minimizing point of f on γ_i implies that x_i is a stationary point of f on ϕ_i from Equation (9). ■

From the proof we can see that the collinearity condition and the non-degeneracy conditions together ensure the long range behaviors of the retraction and the vector transport, otherwise, the algorithm may stop at a non-stationary point. We illustrate Algorithm 1 in Figure 2. The manifold is the unit circle S^1 and the cost function

$$f(x) := \|x - y_0\|^2 \quad (10)$$

is the squared distance from x to a given point y_0 on the plane. So the minimum point of f restricted on a subset of the plane is the point nearest to point y_0. Retraction is defined as $\mathcal{R}_x(\eta) := (x + \eta)/\|x + \eta\|$, where $\eta \in T_x S^1$ is a tangent vector at x. In the beginning, a point $x_0 \in S^1$ and a direction $\eta_0 \in T_{x_0} S^1$ is given.

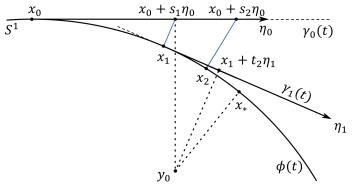


Fig. 2. The first two iterations of Algorithm 1 on S^1. First minimize on the tangent line γ_0 and get the first step size $t_1 = s_1$. Retract $\mathcal{R}_{x_0}(s_1\eta_0)$ to point x_1. Then minimize on the tangent line γ_1 to get a step size t_2. So we have $s_2 = t_1 + t_2$ and the next point $x_2 = \mathcal{R}_{x_0}(s_2\eta_0)$. We can see that x_2 has a lower cost value than x_1 and x_2 is also closer to the global minimum point x_* on $\phi(t) = \mathcal{R}_{x_0}(t\eta_0)$.

III. One-Step-Further Heuristic

The minimization scheme described in Algorithm 1 provides a way of refining step sizes. However the requirement is sometimes too strong in practice. Vector transport may not be collinear with the retraction in real applications, such as the example shown in Figure 3.

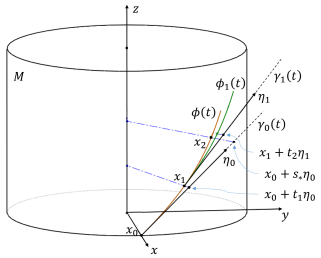


Fig. 3. Example of vector transport that is not collinear with retraction. Consider a retraction defined for a cylinder M in the three-dimensional Euclidean space as $\mathcal{R}_x(\eta) = ((a_1 + b_1)/c, (a_2 + b_2)/c, a_3 + b_3)$, where $x = (a_1, a_2, a_3)$, $\eta = (b_1, b_2, b_3)$, and $c = ((a_1 + b_1)^2 + (a_2 + b_2)^2)^{1/2}$. Vector transport $\mathcal{T}$ is defined by parallel translating the vector in Euclidean space and then projecting it to the tangent space of the target point. In this example, η_1 is the vector transport of η_0 from point x_0 to x_1. From the discrepancy of trajectories $\phi(t) = \mathcal{R}_{x_0}(t\eta_0)$ and $\phi_1(t) = \mathcal{R}_{x_1}(t\eta_1)$, we can see that η_1 is not tangent to ϕ. So the vector transport $\mathcal{T}$ is not collinear with the retraction $\mathcal{R}$. This figure also provides a running example of Algorithm 2.

In addition, we also have to trade off between accuracy and computation overhead. So based on our Algorithm 1, we propose the following one-step-further heuristic strategy in Algorithm 2 to keep things in balance.

Algorithm 2 One-Step-Further Heuristic

Input: Point $x_0 \in M$, direction $\eta_0 \in T_{x_0} M$, $\|\eta_0\| = 1$.
Output: step sizes s_*.

1: Minimize f on the tangent line $\gamma_0(t) := x_0 + t\eta_0$ of curve ϕ at point x_0 to get a step size t_1.
2: Set $x_1 = \mathcal{R}_{x_0}(t_1\eta_0)$,
3: Let $\eta_1 = \mathcal{T}_{t_1\eta_0}\eta_0$ and normalize it to be unit length.
4: Minimize f on the tangent line $\gamma_1(t) := x_1 + t\eta_1$ of curve ϕ at point x_1 to get a step size t_2.
5: Set $s_* = t_1 + t_2$.
6: Perform the Armijo backtracking. Modify s_* if necessary.

Note that t_2 serves as a modification of t_1. To prevent possible crashes, such as turning the step size s_* to a negative value, we restrict t_2 in a reasonable range when implementing Algorithm 2. For example, set $t_2 \in [b_0 t_1, b_1 t_1]$ where $-1 < b_0 \leq 0$ and $b_1 \geq 0$; otherwise, we do not trust t_2. The last step ensures that the output step size satisfy the Armijo condition.

When the iterate point comes near to the optimal solution, both the gradient and the step size become small. In this case, the manifold locally resembles a flat Euclidean manifold and the tangent line is good enough to represent the search curve. We can check whether very small t_2 ($|t_2| < \epsilon$ for some threshold ϵ) appears more than K times in a row to detect whether the algorithm enters a 'flat' area so it can stop computing t_2 to save time. For example, in Case 1 of our numerical experiments (Figure 4(b)), the algorithm enters a flat area at iteration 35 of the total 46 iterations.

Compared to the traditional methods, Algorithm 2 computes one more step, thus we expect a better estimation of

step size. This is shown in the numerical experiments section in which we apply it to a geometric algorithm called R3MC.

IV. Application to Matrix Completion

R3MC [8] is one of the state-of-the-art geometric algorithms for the low-rank matrix completion problem. The manifold in consideration is the set of rank-r matrices $\mathbb{R}_r^{n\times m}$ of size $n \times m$ for a fixed rank r. Let $X^* \in \mathbb{R}^{n\times m}$ be the partially known matrix, then the cost function f for the completion task can be written as

$$f(X) := \frac{1}{|\Omega|}\|\mathcal{P}_\Omega(X) - \mathcal{P}_\Omega(X^*)\|_F^2, \tag{11}$$

where $\Omega := \{(i,j)|X^*_{i,j} \text{ is given}\}$ is the index set for the known entries of X^*, $|\Omega|$ is the cardinality of Ω, $\|A\|_F$ is the Frobenius norm of a matrix A, and $\mathcal{P}_\Omega$ is the orthogonal sampling operator.

The fixed-rank manifold $\mathbb{R}_r^{n\times m}$ is homeomorphic to

$$\mathcal{M} := (\mathrm{St}(r,n) \times \mathrm{GL}(r) \times \mathrm{St}(r,m))/(\mathcal{O}(r) \times \mathcal{O}(r)), \tag{12}$$

where $\mathrm{St}(r,n)$ represents a Stiefel manifold [6, Section 3.3.2], $\mathrm{GL}(r)$ is a general linear group and $\mathcal{O}(r)$ is an orthogonal group. $\mathcal{M}$ is a quotient manifold formed by group action [8, Equation (2)] with the canonical projection map

$$\begin{aligned} \pi : \quad \overline{M} \quad &\to \quad \mathcal{M} \\ (U,R,V) &\mapsto URV^T, \end{aligned} \tag{13}$$

where $\overline{\mathcal{M}} := \mathrm{St}(r,n) \times \mathrm{GL}(r) \times \mathrm{St}(r,m)$ is the total space and $(U,R,V) \in \overline{\mathcal{M}}$. By Equation (13), the cost function can be lifted to $\overline{\mathcal{M}}$ as

$$\bar{f}(U,R,V) = \frac{1}{|\Omega|}\|\mathcal{P}_\Omega(URV^T) - \mathcal{P}_\Omega(X^*)\|_F^2. \tag{14}$$

Conceptually, R3MC performs a conjugate gradient method on the abstract manifold $\mathcal{M}$. However, actual computations are done in the total space $\overline{\mathcal{M}}$ utilizing various lifted objects from $\mathcal{M}$. For example, A point $x \in \mathcal{M}$ can be represented by an element $(U,R,V) \in \pi^{-1}(x) \subset \overline{\mathcal{M}}$. Similarly, retraction and vector transport in $\mathcal{M}$ can be computed through the retraction and vector transport defined on the total space. The facts that $\overline{\mathcal{M}}$ being a submanifold of the ambient vector space $\mathcal{E} := \mathbb{R}^{n\times r} \times \mathbb{R}^{r\times r} \times \mathbb{R}^{m\times r}$ and the lifted cost function $\bar{f}$ can naturally be extended to the ambient space $\mathcal{E}$ permit the application of Algorithm 2 to the line search subproblem.

Suppose that $\gamma(t) = \bar{x} + t\bar{\eta}$ is a line in the ambient space $\mathcal{E}$ passing through a point $\bar{x} = (U,R,V) \in \overline{\mathcal{M}}$ towards direction $\bar{\eta} = (\bar{\eta}_U, \bar{\eta}_R, \bar{\eta}_V) \in T_{\bar{x}}\overline{\mathcal{M}}$. Then the lifted cost function $\bar{f}$ restricted on γ is a degree-6 polynomial in variable t which has a degree 2 polynomial approximation

$$\begin{aligned} \tilde{f}(t) = \frac{1}{|\Omega|}\|\mathcal{P}_\Omega(URV^T - X^* + \\ t(\bar{\eta}_U RV^T + U\bar{\eta}_R V^T + UR\bar{\eta}_V^T))\|_F^2. \end{aligned} \tag{15}$$

R3MC uses the solution of $s_* = \arg\min_t \tilde{f}(t)$ as the initial guess for the step size. In the following numerical experiments, we replace this line search method by the one-step-further strategy described in Algorithm 2.

We conducted experiments using Matlab on a 2.90 GHz Intel Core i7 laptop with 8G RAM. The ground truth matrix X^* was generated randomly with exponential decaying singular values, and CN denotes its condition number. Details of the problem generation process can be found in [8, Section V]. In the experiments, elements of the known entries Ω were chosen uniformly randomly. The number of known entries was represented by the over-sampling factor (OS), which is defined as $|\Omega|/(nr + mr + r^2)$ for a rank r matrix. Algorithms stop when the cost function reduces below 10^{-10} or when they reach the maximum iteration number 500.

Three different line search methods were compared.

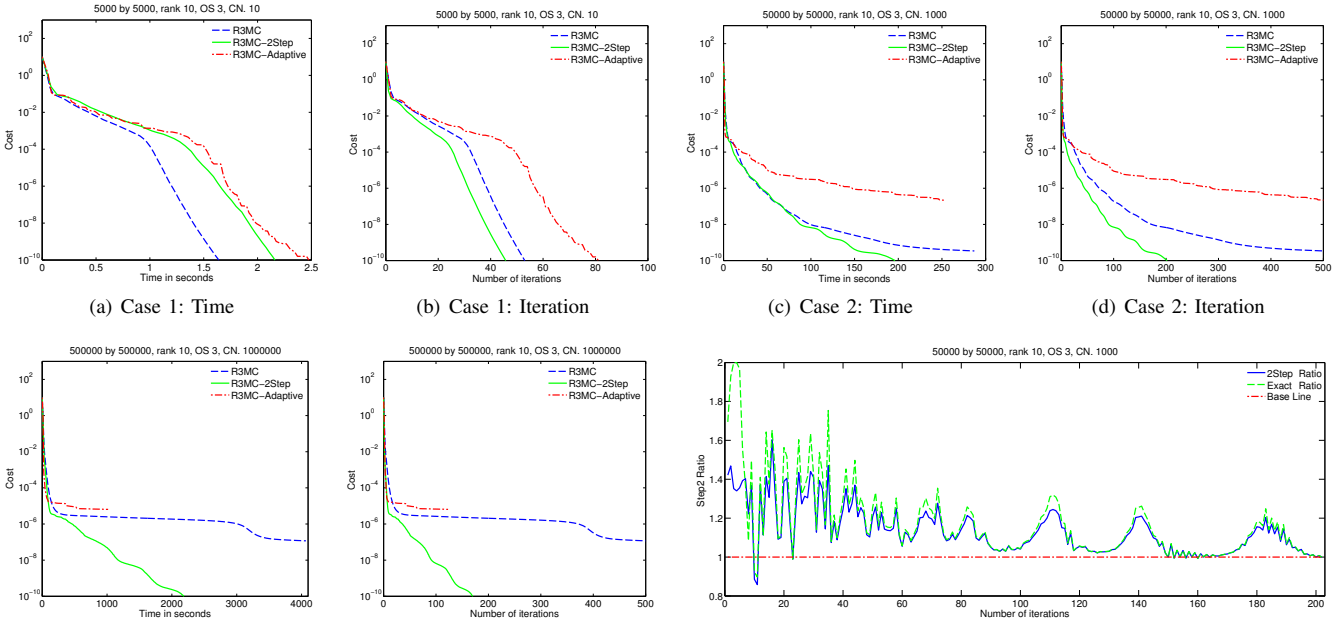
- R3MC. The original line search method [8, Section IV] in R3MC which computes the step size by minimizing the degree 2 approximation (Equation (15)) on the tangent line that passing through the iterate point.

- R3MC-2Step. The implementation of our one-step-further heuristic method (Algorithm 2). We also utilized the approximation polynomial for the cost function. Parameters described in Section III were set to $b_0 = -0.2$, $b_1 = 1$, $\epsilon = 0.005$ and $K = 5$.

- R3MC-Adaptive. The adaptive step size procedure in [15, Section 5.1] which updates the step size according to the number of Armijo backtracking in the last step.

In Figure 4, three example cases with increasing sizes and condition numbers are demonstrated. In these cases, we fixed the ranks to be 10 and set OS to be 3. When measured by total iteration numbers, Figures 4(b), 4(d) and 4(f) imply that our R3MC-2Step performs best among these methods. Since R3MC-2Step has to carry out an extra estimation for the second step size t_2, computation time required for each iterate is a little more than that of R3MC. Figure 4(a) illustrates the effect of this overhead. However, as the complexity of the problem increases, the advantage of better convergence per iteration compensates the cost due to extra computations and we can see from Figures 4(c) and 4(e) that R3MC-2Step outperforms R3MC and R3MC-Adaptive. This makes it suitable for large scale and ill-conditioned scenarios.

We now dig into details of R3MC-2Step in these examples. First, we compare the value of $\mathcal{R}_{x_0}(t_1\eta_0)$ and $\mathcal{R}_{x_0}((t_1+t_2)\eta_0)$ (using notations from Algorithm 1) to see whether the second step t_2 successfully decreases the cost function against the first step t_1. The success rate (SR) is listed in Table I and it shows that the second step seldom fails. Then, how good is this extra step? We compute the exact solution $\hat{s}$ in the range $[(1+b_0)t_1, (1+b_1)t_1]$ using Matlab's *fminbnd* function and compare it with the estimation of Algorithm 2, (t_1+t_2). Figure 4(g) illustrates this comparison for Case 2. Note that all the lengths are divided by t_1. So the values of (t_1+t_2) and $\hat{s}$ are represented by relative ratios (2-Step Ratio and Exact Ratio, respectively). They match very well in Figure 4(g) and there a trend that the closer to the solution, the better is the estimation. The mean absolute error (MAE) is listed in Table I. The average iterations (MI) for the *fminbnd* function is also listed in the Table which shows the efficiency of the heuristic strategy. Three other items (MB1, MB2, and MB3) in Table I are the average number of backtrackings per iterate for R3MC, R3MC-2Step and R3MC-Adaptive, respectively.



(a) Case 1: Time (b) Case 1: Iteration (c) Case 2: Time (d) Case 2: Iteration

(e) Case 3: Time (f) Case 3: Iteration (g) Case 2: Heuristic Step v.s. Exact Step

Fig. 4. Comparing three line search methods, R3MC, R3MC-2Step and R3MC-Adaptive, in three cases. The sizes are 5000×5000, 50000×50000, 500000×500000 and the condition numbers (CN) are 10, 1000 and 1000000, respectively.

	SR	MAE	MI	MB1	MB2	MB3
Case 1	100%	0.0191	11.41	0	0	1.012
Case 2	99.5%	0.0402	14.08	0	0	1.010
Case 3	100%	0.0609	15.75	0	0	1.155

TABLE I. STATISTICS OF THE THREE LINE SEARCH METHODS

V. CONCLUSION

The search curve in geometric optimization algorithms generally lacks closed-form expressions. This restricts the development of effective line search strategies that exploit higher order information of the search curve. To circumvent this difficulty, we use the concept of collinearity and the assistance of linear approximations of the curve to propose a minimization scheme which produces a series of refining step sizes. We prove its validity theoretically. To put it into practical use, we propose a one-step-further heuristic strategy, which is a modification of the minimization scheme. This heuristic can be used to replace the line search part of existing algorithms. We apply it to one of the state-of-the-art geometric algorithms for matrix completion R3MC and show its potential in accelerating the performance of existing algorithms.

ACKNOWLEDGMENT

Z. Lin is supported by 973 Program of China (grant no. 2015CB352502), NSF China (grant nos. 61272341 and 61231002), and Microsoft Research Asia Collaborative Research Program.

REFERENCES

[1] D. Gabay, "Minimizing a differentiable function over a differential manifold," *Journal of Optimization Theory and Applications*, vol. 37, no. 2, pp. 177–219, 1982.

[2] S. T. Smith, "Optimization techniques on Riemannian manifolds," in *Hamiltonian and gradient flows, algorithms and control*, ser. Fields Inst. Commun., 1994, vol. 3, pp. 113–136.

[3] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. MATRIX ANAL. APPL*, vol. 20, no. 2, pp. 303–353, 1998.

[4] W. Ring and B. Wirth, "Optimization methods on Riemannian manifolds and their application to shape space," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 596–627, 2012.

[5] P.-A. Absil, C. Baker, and K. Gallivan, "Trust-region methods on Riemannian manifolds," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 303–330, 2007.

[6] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009.

[7] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," *Journal of Machine Learning Research*, vol. 15, pp. 1455–1459, 2014.

[8] B. Mishra and R. Sepulchre, "R3MC: A Riemannian three-factor algorithm for low-rank matrix completion," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 2014.

[9] B. Vandereycken, "Low-rank matrix completion by Riemannian optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, p. 1214, 2013.

[10] N. Boumal and P.-A. Absil, "RTRMC: A Riemannian trust-region method for low-rank matrix completion," in *Advances in Neural Information Processing Systems 24 (NIPS)*, 2011, pp. 406–414.

[11] T. Ngo and Y. Saad, "Scaled gradients on Grassmann manifolds for matrix completion," in *Advances in Neural Information Processing Systems 24 (NIPS)*, 2012, pp. 1412–1420.

[12] G. Meyer, S. Bonnabel, and R. Sepulchre, "Linear regression under fixed-rank constraints: A Riemannian approach," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 545–552.

[13] ——, "Regression on fixed-rank positive semidefinite matrices: a Riemannian approach," *JMLR*, vol. 12, pp. 593–625, 2011.

[14] J. Nocedal and S. J. Wright, *Numerical optimization*, 2nd ed. New York: Springer, 2006.

[15] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre, "Fixed-rank matrix factorizations and Riemannian low-rank optimization," *Computational Statistics*, vol. 29, no. 3-4, pp. 591–621, 2014.