A First-Order Variable Metric Method Invariant to Strictly Increasing Function Value Transformations

Dirk V. Arnold and Yuhan Fu

Faculty of Computer Science, Dalhousie University

Abstract

Incorporating multiple elements of evolution strategies, a first-order variable metric method for unconstrained optimization that is invariant to strictly increasing function value transformations is proposed. The algorithm's performance is evaluated relative to that of a quasi-Newton algorithm using a large set of test problems.

1 Introduction

Unconstrained numerical optimization considers the problem of minimizing an objective function $f : \mathbb{R}^n \to \mathbb{R}$. A fundamental distinction in algorithms for unconstrained optimization is between methods that assume differentiability of f and that use gradient vectors in the determination of the sequences of iterates that they generate, and those that do not. The former are known as first-order algorithms, the latter are zeroth-order or derivative-free.

Quasi-Newton methods are among the most important first-order methods, and through their integration in other algorithms they form a cornerstone of many methods for constrained optimization as well. They derive curvature estimates (i.e., second-order information), from differences between the gradient vectors of successive iterates. By preconditioning gradient vectors with curvature information (i.e., employing a variable metric), performance significantly superior to steepest descent can be achieved on many problems. BFGS (named after C. G. Broyden, R. Fletcher, D. Goldfarb, and D. Shanno) is often considered the most commonly used quasi-Newton method, and implementations are available in GSL, R, SciPy, MATLAB, and Mathematica.

Evolution strategies are stochastic zeroth-order optimization algorithms that neither assume the availability of first-order information nor do they try to approximate it. Sequences of iterates are generated by randomly sampling points, and the information gained through the evaluation of those points is used to update the parameters of the sampling distributions. A primary concern in the development of evolution strategies is the preservation of desirable invariance properties (Hansen et al., 2011). For example, evolution strategies, which base their decisions on comparisons of objective function values without using those values in arithmetic expressions, are invariant with respect to strictly increasing transformations of the objective function. They behave identically on all members of classes of problems that are related though such a transformation and are thus more robust than algorithms that excel on some members of a class but exhibit inferior performance on others.

A milestone in the development of evolution strategies was the addition of covariance matrix adaptation (Hansen and Ostermeier, 2001; Hansen et al., 2003), which enables the algorithms to effectively cope with poorly conditioned problems. Covariance matrix adaptation evolution strategies learn a variable metric by increasing variances of the sampling distribution in those directions where good points are located. We argue that covariance matrix adaptation is thus a form of reinforcement learning. The evolution strategy (the agent) samples points (the action) which are then evaluated using the objective

function (the environment). The result is a reward for sampling directions that lead to good points, adjusting the state of the agent in such a way that the likelihood of sampling even longer steps in those directions increases in subsequent iterations.

In this paper we present a first-order algorithm for unconstrained optimization that preconditions the gradient vector with information obtained through reinforcement learning. In Section 2 we provide brief discussions of quasi-Newton methods and of evolution strategies in as far as relevant in the present context. Section 3 introduces RLVM, a reinforcement learning based approach to variable metric learning in first-order settings that incorporates multiple design choices and ideas underlying evolution strategies. In contrast to quasi-Newton methods, RLVM is invariant to strictly increasing transformations of the objective function. Section 4 presents an experimental comparison of RLVM with BFGS. We find that the quasi-Newton approach is generally advantageous on quadratic problems, where the quadratic models used in those methods are a perfect fit, but that in non-quadratic settings significant advantages of the reinforcement learned based approach may be observed. Section 5 concludes with a brief discussion.

2 Background

Section 2.1 presents a brief overview of quasi-Newton methods that will allow us to contrast our approach. Section 2.2 summarizes important concepts and ideas from evolution strategies that we will utilize in a first-order setting in Section 3.

2.1 Quasi-Newton Methods

Quasi-Newton methods are iterative first-order optimization algorithms that use a quadratic approximation to the objective function in their update of a variable metric. In iteration k, the state of a quasi-Newton method consists of iterate $\mathbf{x}_k \in \mathbb{R}^n$ and $n \times n$ matrix $\mathbf{B}_k$ that serves as an approximation to the Hessian matrix at point $\mathbf{x}_k$. The minimizer of second-order Taylor approximation

$$f_2(\mathbf{x}) = f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^{\mathrm{T}}(\mathbf{x} - \mathbf{x}_k) + \tfrac{1}{2}(\mathbf{x} - \mathbf{x}_k)^{\mathrm{T}}\mathbf{B}_k(\mathbf{x} - \mathbf{x}_k)$$

to the objective function at iterate $\mathbf{x}_k$ lies in direction

$$\mathbf{p}_k = -\mathbf{B}_k^{-1}\nabla f(\mathbf{x}_k) \tag{1}$$

from $\mathbf{x}_k$, and the next iterate $\mathbf{x}_{k+1}$ is determined by a line search in that direction. The approximation to the Hessian matrix is then updated to satisfy secant equation

$$\mathbf{B}_{k+1}(\mathbf{x}_{k+1} - \mathbf{x}_k) = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)\,. \tag{2}$$

That is, changes in the gradient vector from one iteration to the next are exploited to infer information regarding the Hessian matrix at the current point. If $n > 1$, then Eq. (2) is under-determined and $\mathbf{B}_{k+1}$ is obtained by performing a low-rank update of $\mathbf{B}_k$ that typically ensures that some other conditions hold (e.g., symmetry).

The first quasi-Newton method was proposed by Davidon (1959) and is commonly referred to as DFP (after W. C. Davidon, R. Fletcher, and M. J. D. Powell). Among the most commonly used quasi-Newton methods in use today is BFGS, which performs update

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k))(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k))^{\mathrm{T}}}{(\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k))^{\mathrm{T}}(\mathbf{x}_{k+1} - \mathbf{x}_k)} - \frac{\mathbf{B}_k(\mathbf{x}_{k+1} - \mathbf{x}_k)(\mathbf{x}_{k+1} - \mathbf{x}_k)^{\mathrm{T}}\mathbf{B}_k}{(\mathbf{x}_{k+1} - \mathbf{x}_k)^{\mathrm{T}}\mathbf{B}_k(\mathbf{x}_{k+1} - \mathbf{x}_k)}\,. \tag{3}$$

That update is of rank two and can be shown to maintain positive definiteness of the approximation to the Hessian matrix if the steps satisfy some conditions (Nocedal and Wright, 2006). When objective function f is convex quadratic, then matrices $\mathbf{B}_k$ converge to the Hessian matrix of the problem.

Clearly, storage requirements of the method are quadratic in the dimension n of the problem. Computational requirements at first sight appear cubic as Eq. (1) involves matrix inversion. However, as updates to the approximation of the Hessian matrix are of low rank, the Sherman/Morrison formula can be employed to directly update the inverse of that matrix, resulting in quadratic overall costs.

2.2 Evolution Strategies

Originally devised by Rechenberg (1973) and Schwefel (1975), evolution strategies are stochastic zeroth-order optimization methods that in each iteration generate multiple points by sampling from an unbiased distribution centred at the current iterate. They then form a linear combination of the steps leading to those points, with coefficients determined by the ranks of their corresponding objective function values, to form the next iterate. As steps between successive iterates are determined by ranks, they are not impacted by rank-preserving transformations of function values.

Recognizing that the performance of evolution strategies on poorly conditioned problems can be improved significantly by adapting the shape of the sampling distribution, Hansen and Ostermeier (2001) and Hansen et al. (2003) proposed covariance matrix adaptation evolution strategies (CMA-ES). Comprehensive descriptions of the algorithm are provided by Hansen et al. (2015) and Hansen (2023). CMA-ES sample from a normal distribution with covariance matrix $\sigma^2\mathbf{C}$, where $\mathbf{C}$ is a symmetric, positive definite $n \times n$ matrix and $\sigma > 0$ is a further parameter that controls the overall scale. Central to CMA-ES is an update of the matrix controlling the sampling distribution according to

$$\mathbf{C}_{k+1} = (1-c)\mathbf{C}_k + c\,\mathbf{C}_k^{1/2}\mathbf{Z}_k\mathbf{C}_k^{1/2}\,, \tag{4}$$

where subscripts denote iteration number, $(\cdot)^{1/2}$ is the matrix square root, and $c > 0$ determines how rapidly the information represented in matrix $\mathbf{C}$ is replaced. Omitting iteration numbers for clarity, symmetric $n \times n$ matrix $\mathbf{Z}$ is computed as

$$\mathbf{Z} = \sum_{i=1}^{\lambda} w_i \mathbf{z}_{i;\lambda}\mathbf{z}_{i;\lambda}^{\mathrm{T}}\,, \tag{5}$$

where vectors $\mathbf{z}_j$, $j = 1, 2, \ldots, \lambda$, are standard normally distributed and $i;\lambda$ is the index of that vector $\mathbf{z}_j$ with the ith best value of $f(\mathbf{x} + \sigma\mathbf{C}^{1/2}\mathbf{z}_j)$. Weights w_i decrease monotonically with increasing i and thus ensure that larger weights are associated with those directions in which relatively better points are sampled. Jastrebski and Arnold (2006) proposed the use of negative weights to actively decrease variances in unfavourable directions, but care must be taken to avoid the loss of positive definiteness of matrix $\mathbf{C}$ in that case. It can be observed empirically under some conditions that if the objective function is convex quadratic, then $\mathbf{C}$ approximately converges to a matrix proportional to the inverse Hessian matrix of the problem. Due to its use of rank based weights, the algorithm is invariant to strictly increasing transformations of function values.

Matrix $\mathbf{A} = \mathbf{C}^{1/2}$ can be regarded as implementing a linear transformation that is applied to step vectors $\mathbf{z}_j$, $j = 1, 2, \ldots, \lambda$. As $\mathbf{A}$ is symmetric and positive definite, it represents a scaling operation with the eigenvalues of $\mathbf{A}$ forming the positive scale factors applied to the axes that form the matrix's eigenvectors. The result of using the matrix update in Eq. (4) is that scales are increased in those directions in which relatively good points are sampled (and, if using negative weights, decreased in those where points with relatively poorer objective function values are found). Covariance matrix adaptation can thus be considered a form of reinforcement learning, where good decisions result in a "reward" that impacts future actions. Sampling directions that lead to good points results in an adjustment of the state of the algorithm in such a way that the likelihood of sampling further steps in those directions increases.

A further significant contribution to the development of evolution strategies was the proposal by Krause and Glasmachers (2015) to replace the additive matrix update in Eq. (4) with a multiplicative

update according to

$$\mathbf{C}_{k+1} = \mathbf{C}_k^{1/2} \exp(c\mathbf{Z}_k)\mathbf{C}_k^{1/2} \,. \tag{6}$$

For small values of c the additive and multiplicative updates are nearly identical. A primary benefit of the multiplicative update is that negative weights can be used in Eq. (5) without having to fear a loss of positive definiteness of $\mathbf{C}$.

Finally, Glasmachers and Krause (2020) proposed using curvature estimates to learn covariance matrices for evolution strategies in a zeroth-order setting. In a sense, the approach pursued in the following is the reverse in that we strive to utilize information gained through reinforcement learning in a first-order setting.

3 Reinforcement Learning Based Variable Metric Method

This section introduces RLVM, a reinforcement learning based variable metric method for first-order optimization that is invariant to strictly increasing transformations of the objective function. The algorithm maintains a matrix $\mathbf{B}$ that stores axis scales akin to the covariance matrix in CMA-ES. In contrast to zeroth-order CMA-ES, which use the square root of that matrix to transform sums of random vectors weighted according to the ranks of the function values of their corresponding points, RLVM uses it to transform the negative of the normalized gradient vector $\mathbf{g} = \nabla f(\mathbf{x})/\|\nabla f(\mathbf{x})\|$ at the current iterate. A strictly increasing transformation of function values changes the magnitude of the gradient vector, but not its direction. As the algorithm only uses normalized gradient vectors, it is invariant to such transformations.

In order to learn a variable metric, directions in which axis scales should be increased or decreased need to be identified. RLVM, in relative terms, increases the scale of the axis in the direction formed by the sum of two successive steps, and, akin to BFGS (compare Eq. (1)), it decreases the scale of the axis in the direction formed by their difference. In the spirit of reinforcement learning, a longer scale factor associated with the direction of the sum of two successive steps will increase the length of future steps in those directions; a smaller scale factor associated with their difference will reduce them. Similar to the use of the matrix update in Eq. (6), this is accomplished by updating matrix $\mathbf{B}$ multiplicatively according to

$$\mathbf{B}_{k+1} = \zeta_k \mathbf{B}_k^{1/2} \exp\left(c\mathbf{Z}_k\right) \mathbf{B}_k^{1/2} \,, \tag{7}$$

where

$$\begin{aligned} \mathbf{Z}_k &= \tfrac{1}{2}\left((\mathbf{g}_0 + \mathbf{g}_1)(\mathbf{g}_0 + \mathbf{g}_1)^{\mathrm{T}} - (\mathbf{g}_0 - \mathbf{g}_1)(\mathbf{g}_0 - \mathbf{g}_1)^{\mathrm{T}}\right) \\ &= \mathbf{g}_0\mathbf{g}_1^{\mathrm{T}} + \mathbf{g}_1\mathbf{g}_0^{\mathrm{T}} \,, \end{aligned} \tag{8}$$

and where $\mathbf{g}_0 = \nabla f(\mathbf{x}_{k-1})/\|\nabla f(\mathbf{x}_{k-1})\|$ and $\mathbf{g}_1 = \nabla f(\mathbf{x}_k)/\|\nabla f(\mathbf{x}_k)\|$ are successive normalized gradient vectors. Parameter $c > 0$ governs the magnitude of the update. Irrespective of the choice of c, matrix $\mathbf{B}$ will remain positive definite. As the quasi-Newton update in Eq. (1), the update in Eq. (7) is of rank two.

Akin to the adaptation of parameter σ in evolution strategies, scalar $\zeta_k > 0$ in Eq. (7) allows RLVM to effect changes to the overall scale of the steps made by the algorithm. An approach to the adaptation of the scale parameter in evolution strategies proposed by Ostermeier et al. (1994) amounts to increasing the scale if successive steps point in similar directions, and to decrease it if they tend to be closer to antiparallel. Similarly, we choose to set

$$\zeta_k = \exp\left(d\left(\mathbf{g}_0^{\mathrm{T}}\mathbf{g}_1 - e\right)\right) \,, \tag{9}$$

where $d > 0$ and e are scalar constants. As a result, the overall scale of matrix $\mathbf{B}$ is increased if consecutive gradients form an angle with a cosine exceeding e; it is decreased if the cosine of that angle is less than e. Parameter $d > 0$ governs the magnitude of the updates. In the experiments described in Section 4,

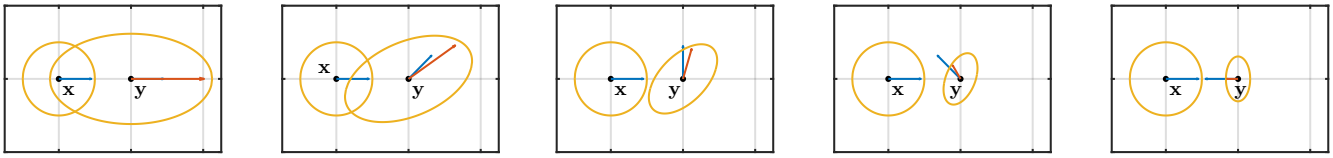

Figure 1: Illustration of the adaptation of the transformation matrix in a single iteration. Depicted are five examples with the angle between successive negative, normalized gradient vectors (shown in blue) increasing from left to right.

parameter settings $c = 0.6$, $d = 0.7$, and $e = 0.4$ are employed throughout and prove useful across all test problems and dimensions.

To illustrate the effect of adapting the transformation matrix according to Eq. (7), consider a step from point $\mathbf{x}$ to point $\mathbf{y}$ and assume that the initial transformation matrix is the identity matrix. Figure 1 depicts five different scenarios in dimension $n = 2$, each showing one step of the algorithm. Negatives of normalized gradient vectors $\mathbf{g}_0 = \nabla f(\mathbf{x})/\|\nabla f(\mathbf{x})\|$ and $\mathbf{g}_1 = \nabla f(\mathbf{y})/\|\nabla f(\mathbf{y})\|$ are shown in blue. Shown in yellow are a circle centred at point $\mathbf{x}$ and an ellipse centred at point $\mathbf{y}$ that illustrates the result of transforming the circle using the linear transformation $\mathbf{B}^{1/2}$, where $\mathbf{B}$ has been computed based on $\mathbf{g}_0$ and $\mathbf{g}_1$ using Eq. (7). The red arrows are negative normalized gradient vectors transformed by the updated matrix and thus show the directions that the algorithm will subsequently proceed in from $\mathbf{y}$. In the first of the scenarios shown, $\mathbf{g}_0$ and $\mathbf{g}_1$ are identical, resulting in a transformation matrix that has the effect of increasing scales, primarily in the direction of the (identical) normalized gradients. The eigenvalues of the updated transformation matrix are 1.234 and 2.248. The red arrow depicting $-\mathbf{B}^{1/2}\mathbf{g}_1$ points in the same direction as $\mathbf{g}_1$ but is markedly longer, resulting in a longer step in the next iteration. In the second scenario the angle formed by $\mathbf{g}_0$ and $\mathbf{g}_1$ is acute. The change in $\mathbf{B}$ has the effect of elongating the length of the step taken in the next iteration, and to skew in the direction common to both normalized gradient vectors. In the third scenario $\mathbf{g}_0$ and $\mathbf{g}_1$ are orthogonal, and the matrix update shrinks the length of the next step taken as the angle between $\mathbf{g}_0$ and $\mathbf{g}_1$ is now larger than the neutral angle determined by e. The eigenvalues of the transformation matrix are 0.644 and 1.174. In the fourth scenario the angle formed by $\mathbf{g}_0$ and $\mathbf{g}_1$ is obtuse, resulting in a shrinking of the step primarily in the direction of the difference between the two normalized gradient vectors and in a more severe decrease of the length of the following step overall. The final scenario depicts a case where $\mathbf{g}_0$ and $\mathbf{g}_1$ are antiparallel, resulting in eigenvalues of the updated transformation matrix of 0.336 and 0.613, and in a significantly smaller step in the following iteration.

Figure 2 presents pseudo-code for RLVM. The algorithm takes as input a starting point $\mathbf{x}$, in the absence of further information initializes matrix $\mathbf{B}$ to the $n \times n$ identity matrix, and determines the objective function value and normalized gradient vector of the starting point. Lines 3 through 13 form the main loop of the algorithm. Line 4 determines transformation matrix $\mathbf{A}$ as the square root of $\mathbf{B}$. Line 5 takes a step from $\mathbf{x}$ to new point $\mathbf{y}$ by applying the transformation matrix to the negative normalized gradient vector at $\mathbf{x}$. Line 6 uses the objective function to obtain the function value and normalized gradient vector at $\mathbf{y}$. Line 7 then performs matrix adaptation as proposed in Eq. (7), and Line 8 adapts the overall scale according to Eq. (8). Line 9 implements basic safeguards designed to ensure proper operation of the algorithm despite the limited accuracy inherent in floating point representations of real numbers. The update in Line 7 may result in matrix $\mathbf{B}$ not being perfectly symmetric, and as Hansen (2023) we address this by mirroring the upper triangular part of the matrix and using it to replace the lower triangular part. More significantly, cumulative updates of $\mathbf{B}$ over multiple iterations can result in the condition number of the matrix becoming so large that taking its square root in Line 4 of the algorithm fails. To avoid this, we monitor the condition number of $\mathbf{B}$ and add small terms $\delta\mathbf{I}_n$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix and δ is chosen in direct proportion to the smallest eigenvalue of $\mathbf{B}$, to the matrix until the condition number drops below a threshold of 10^{14}. Lines 10 through 12 replace $\mathbf{x}$ with $\mathbf{y}$

Input:

- $\mathbf{x} \in \mathbb{R}^n$

1: Let $\mathbf{B} \leftarrow \mathbf{I}_n$.
2: Let $f_0 \leftarrow f(\mathbf{x})$ and $\mathbf{g}_0 \leftarrow \nabla f(\mathbf{x}) / \|\nabla f(\mathbf{x})\|$.
3: **while** stopping criteria are not satisfied **do**
4: Let $\mathbf{A} \leftarrow \mathbf{B}^{1/2}$.
5: Let $\mathbf{y} \leftarrow \mathbf{x} - \mathbf{A}\mathbf{g}_0$.
6: Let $f_1 \leftarrow f(\mathbf{y})$ and $\mathbf{g}_1 \leftarrow \nabla f(\mathbf{y}) / \|\nabla f(\mathbf{y})\|$.
7: Let $\mathbf{B} \leftarrow \mathbf{A} \exp\left(c(\mathbf{g}_0\mathbf{g}_1^{\mathrm{T}} + \mathbf{g}_1\mathbf{g}_0^{\mathrm{T}})\right) \mathbf{A}$.
8: Let $\mathbf{B} \leftarrow \exp\left(d(\mathbf{g}_0^{\mathrm{T}}\mathbf{g}_1 - e)\right) \mathbf{B}$.
9: Ensure symmetry and potentially regularize.
10: **if** $f_1 < f_0$ **then**
11: Let $\mathbf{x} \leftarrow \mathbf{y}$, $f_0 \leftarrow f_1$ and $\mathbf{g}_0 \leftarrow \mathbf{g}_1$.
12: **end if**
13: **end while**

Figure 2: Reinforcement learning based variable metric method.

if the objective function value of the latter is superior to that of the former. MATLAB code implementing RLVM as well as for running the experiments described in Section 4 is included with this submission.

Regarding algorithm internal costs, the update of matrix $\mathbf{B}$ in Lines 7 and 8 and can be accomplished in a number of operations quadratic in the dimension of the problem in a manner similar to the multiplicative update of the covariance matrix described by Krause and Glasmachers (2015). However, the benefit of this acceleration is limited as the computation of the matrix square root in Line 4 requires cubic time regardless. Amortized algorithm internal costs can be reduced to quadratic in the dimension of the problem by performing the computation of the matrix square root infrequently as described by Hansen (2023), but a systematic evaluation of the consequences remains to be performed.

4 Experiments

We experimentally evaluate RLVM using two sets of test problems. Section 4.1 employs three scalable families of test functions commonly used to compare black-box optimization algorithms. Section 4.2 aggregates results from a much larger collection of test functions. Comparisons are with BFGS as implemented in the MATLAB *Optimization Toolbox*. That toolbox also includes an implementation of the DFP method, but in the documentation indicates that it does not work very well and is available for educational purposes. We have confirmed that observation and interpret it partly as evidence for the difficulty of creating a successful implementation of a quasi-Newton algorithm.

4.1 Initial Tests

This section considers three parameterized families of test problems selected to provide interpretable insights into the relative capabilities of BFGS and RLVM. The problems are unimodal and separable. However, neither algorithm exploits separability, and in all cases the coordinate system could be rotated without impacting the results. Optimal function values for all problems are zero. All runs are initialized to start at points sampled from multivariate normal distributions centred at the optimal solutions and with diagonal covariance matrix and component-wise standard deviations of 10^3. Both algorithms use the same sets of starting points. Each evaluation of the objective function returns both the function value

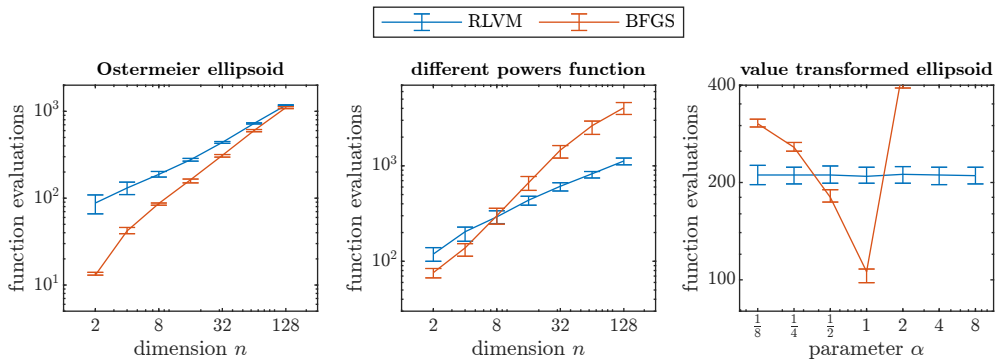

Figure 3: Numbers of points evaluated before reaching termination accuracy for three families of test problems.

and the gradient vector. Computational cost is quantified as the number of function evaluations that an algorithm requires to achieve a solution of a prescribed accuracy.

The left-hand subplot in Fig. 3 shows the numbers function evaluations required by RLVM and BFGS to optimize the Ostermeier ellipsoid defined as

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} a_i x_i^2 \,, \tag{10}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ and $a_i = 10^{6(i-1)/(n-1)}$ for $i = 1, 2, \ldots, n$. The problem is convex quadratic with a condition number of 10^6 of its Hessian matrix and is included in the BBOB 2009 test set (Hansen et al., 2009). Runs of either algorithm terminate once a point with a function value no larger than 10^{-6} has been found. Lines in the plot connect median numbers of function evaluations, and error bars show the 10th and 90th percentiles across 101 runs. Numbers of function evaluations to reach termination accuracy are lower for BFGS than for RLVM. This is expected as the problem is quadratic and thus the ideal use case for the quasi-Newton algorithm, which iteratively approximates the inverse Hessian matrix. It is interesting to see though that while the gap between the numbers of function evaluations required by the two algorithms is close to a factor of eight for $n = 2$, it gradually narrows with increasing dimension. For $n = 128$, the median number of function evaluations required by RLVM is less than ten percent larger than that required by BFGS.

The middle subplot in Fig. 3 shows the numbers function evaluations required by RLVM and BFGS to optimize the different powers function defined as

$$f_2(\mathbf{x}) = \sqrt{\sum_{i=1}^{n} |x_i|^{2+4(i-1)/(n-1)}} \,. \tag{11}$$

The function is included in the BBOB 2009 test set, and runs are terminated when a point with a function value no larger than 10^{-6} has been reached. While for small dimensions BFGS requires fewer function evaluations than RLVM, that relation is inverted for $n > 8$ and the performance advantage of RLVM ultimately widens to a factor of almost four at $n = 128$. On this non-quadratic problem the Hessian matrix varies throughout the optimization domain, and the quasi-Newton method loses its advantage compared to the reinforcement learning based approach.

Finally, the right-hand subplot in Fig. 3 shows results for function value transformed Ostermeier ellipsoid defined as

$$f_3(\mathbf{x}; \alpha) = (f_1(\mathbf{x}))^{\alpha} \,. \tag{12}$$

For $\alpha > 0$ the transformation of values composed with the Ostermeier ellipsoid defined in Eq. (10) is a strictly increasing function. The same test problem has been used by Auger et al. (2009) in order to

illustrate the importance of invariance in zeroth-order search. In contrast to that reference, we consider the test function in a first-order setting where gradient vectors are immediately available rather than having to be approximated through finite differencing. The values shown in the plot are for dimension $n = 10$. Either algorithm terminates successfully once a point with an objective function value no larger than $10^{-6\alpha}$ has been found. As reflected in the plot, RLVM, which is invariant to the transformation of function values, requires numbers of function evaluations that are independent of α. In contrast, numbers of function evaluations required by BFGS, which are markedly lower than those required by RLVM in an interval of α values that includes 1 (i.e., the quadratic case), increase notably for both large and small values of α. For values of $\alpha \geq 4$ BFGS fails to attain solutions of the required accuracy.

4.2 Andrei Test Problems

While the previous section has shown that RLVM can outperform BFGS on problems where transformations are applied to function values or to individual variables in quadratic functions, we strive to compare the algorithms on a problem set with more diverse characteristics. One such set has been gathered by Andrei (2008). The set contains 101 unconstrained test problems, 18 of which are quadratic, 61 are unimodal without being quadratic, and 22 are multimodal. All of the problems are smooth, thus allowing the computation of gradients. All of them have prescribed starting points, and, importantly, all are parameterized by the dimension.

We conduct experiments on the Andrei test problems, varying the experimental parameters along two independent axes. First, we run experiments for multiple dimensions of the optimization domains. The minimum useful dimension for most of the problems is $n = 2$, but fifteen of the problems require that n is a multiple of 3, and three require that n is a multiple of 4. Beyond the minimum useful dimension we also run experiments for $n = 12$ and for $n = 72$. RLVM becomes less useful for dimensions significantly in excess of those values as its algorithm internal costs increase cubically with n. And second, rather than using the starting points identified by Andrei (2008), we sample starting points randomly and run experiments for multiple choices of their distribution. All but twelve of the problems have a single global optimizer. Denoting that optimizer as $\mathbf{x}^*$ and the starting point proposed by Andrei as $\mathbf{x}_0$, we generate starting points by sampling from normal distributions with mean $\mathbf{x}^*$ and with covariance matrix $(\gamma\|\mathbf{x}_0 - \mathbf{x}^*\|)^2\mathbf{I}_n/sqrt(n)$, where $\gamma \in \{1, 10, 100\}$. For $\gamma = 1$, the expected squared distance from the optimizer is the same as for the starting point identified by Andrei. We refer to the three different choices for parameter γ as near starts, medium starts, and distant starts, respectively. For those twelve problems that do not possess a finite global optimizer we sample starting points from normal distributions with mean $\mathbf{x}_0$ and with covariance matrices $\gamma^2\mathbf{I}_n/sqrt(n)$.

To compare algorithm performance, we use empirical cumulative running time distributions as described by Hansen et al. (2021). For those problems that have finite global optima, 21 function value targets geometrically uniformly spaced between $f(\mathbf{x}^*) + 10^{-6}$ and $f(\mathbf{x}^*) + 0.1(f(\mathbf{x}_0) - f(\mathbf{x}^*))$, where $\mathbf{x}_0$ are the starting points proposed by Andrei, are used. Eleven of the 101 problems do not possess a finite global optimum, and sequences of points with function values converging to $-\infty$ exist. For those problems we use 21 objective function value targets geometrically uniformly spaced between -10^2 and -10^6. For each combination of problem, dimension, and value of γ (which parameterizes the distribution of starting points), 21 runs are conducted for each algorithm. Runs are terminated when either the strictest function value target has been reached or the L^∞-norm of the gradient vector is less than 10^{-8}. The latter criterion is satisfied at the merely local optimizers of the multimodal functions.

Figure 4 depicts empirical cumulative running time distributions for the minimum useful dimensions of the problems (i.e., $n \in \{2, 3, 4\}$). The columns from left to right show distributions for all problems, for those problems that are quadratic, problems that are unimodal without being quadratic, and multimodal problems. Rows correspond to different distributions of starting points, with the expected distance from the optimizer increasing from top to bottom. In each plot, the horizontal axis represents the number of function evaluations, the vertical axis the fraction of all function value targets reached. Beside RLVM

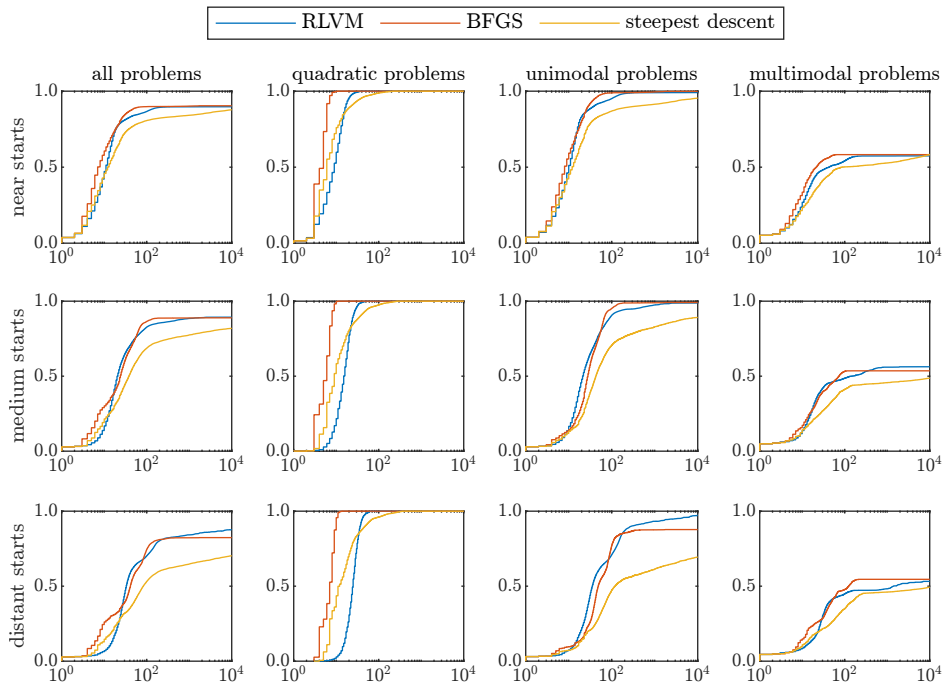

Figure 4: Empirical cumulative running time distributions for minimal dimension. Horizontal axes represent numbers of function evaluations, vertical axes the fraction of function value targets reached.

and BFGS, a steepest descent algorithm that does not employ a variable metric and that is available in the MATLAB *Optimization Toolbox* is also included. It can be seen that across all of the problems, the distribution plots for near starts are dominated by BFGS. For small function evaluation budgets, steepest descent is slightly more successful than RLVM, though the latter algorithm dominates the former for larger numbers of function evaluations. The strict dominance of BFGS over RLVM disappears with increasing distance of the starting points from the optimizer, and RLVM attains the greatest number of function value targets for distant starts after 10^4 function evaluations.

These observations become more interpretable by considering separate distributions for quadratic problems, unimodal problems that are not quadratic, and multimodal problems. On the quadratic problems, all three algorithms eventually attain all function value targets. In line with the observations made in Section 4.1 on the Ostermeier ellipsoid, BFGS requires significantly fewer function evaluations than RLVM. Over 80 percent of the targets are reached by the steepest descent algorithm, which incorporates a line search, before they are reached by the reinforcement learning based approach. The advantage of BFGS over RLVM increases with increasing distance of the starting points from the optimizer. This is intuitive as the quasi-Newton algorithm converges rapidly after having generated a good approximation of the inverse Hessian matrix.

On those problem that are unimodal without being quadratic, the distributions depicted in Fig. 4 show much less of an advantage, if any, for BFGS. Despite their unimodality, none of the algorithms reach all of the target function values within 10^4 function evaluations. For distant starts, the fraction of all unimodal problems that are eventually solved is significantly higher for RLVM than for BFGS. Of the 84 unimodal instances (across all three starting point scenarios) that RLVM has not solved after 10^4 iterations, 63 are on the extended Hiebert function, which is characterized by a very narrow and curved valley leading to the optimizer and where BFGS also fails in over half of the runs.

Finally, distributions for the multimodal problems in Fig. 4 suggest that while the quasi-Newton method often is the first to reach function value targets, the number of runs that eventually converge to the global optimizer is similar for BFGS and RLVM. Neither algorithm appears to have significantly superior global optimization capabilities.

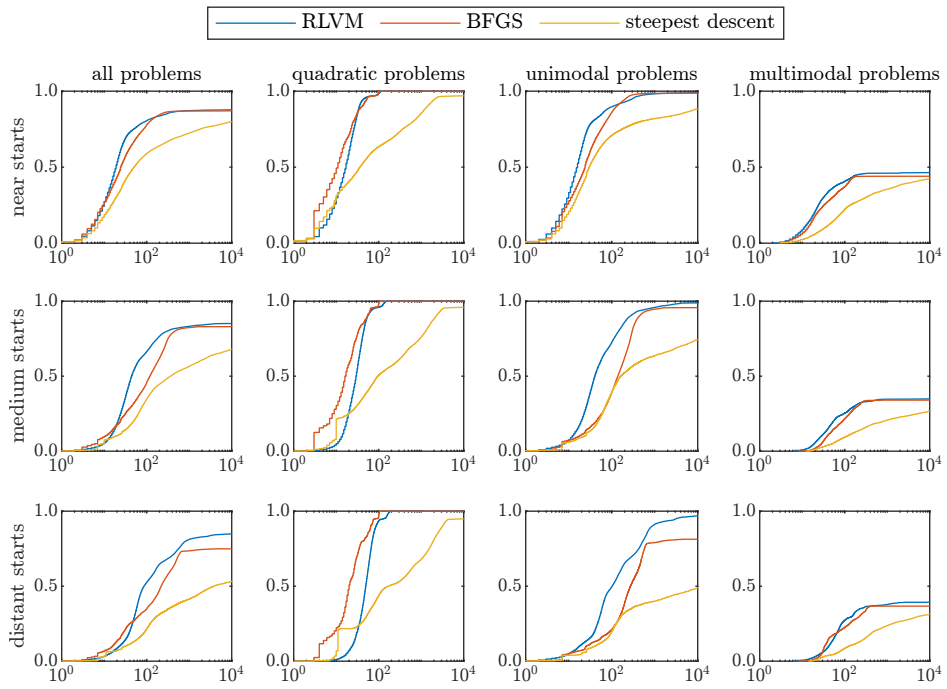

Figure 5: Empirical cumulative running time distributions for dimension $n = 12$. Horizontal axes represent numbers of function evaluations, vertical axes the fraction of function value targets reached.

Empirical cumulative running time distributions for dimension $n = 12$ are shown in Fig. 5. Across all problems, RLVM dominates the distributions for numbers of function evaluations in the tens for near starts and for numbers of function evaluations in the high tens or larger for medium and distant starts. As for the minimum useful dimensions, the distributions for quadratic problems are mostly dominated by BFGS. However, in line with observations made for the Ostermeier ellipsoid in Section 4.1, the performance advantage that the quasi-Newton method enjoys over the reinforcement learning based approach now is markedly more narrow in the region where the more stringent function value targets are reached, and the steepest descent approach falls far behind. The plots for unimodal but not quadratic problems are now in most places clearly dominated by RLVM, especially in the case of distant starts, where BFGS fails to reach a significant fraction of targets within 10^4 function evaluations. Of the 102 problem instances that RLVM fails to solve within that budget, 63 are again on the extended Hiebert function, and BFGS fails to solve all but one of those as well. The running time distributions for multimodal problems reveal that compared to the minimum useful dimensions of the problems a larger fraction of runs now fail to converge to the global optimizer. A majority of the runs that do not converge to the global optimizer terminate as a result of near vanishing gradients and presumably converge to merely local optimizers. Differences between BFGS and RLVM again appear minor.

Empirical cumulative running time distributions for dimension $n = 72$ are shown in Fig. 6 and continue the trend. While BFGS continues to reach the easiest targets on the quadratic problems significantly earlier than RLVM does, the reinforcement learning based algorithm dominates the quasi-Newton algorithm for a range of function evaluation budgets where the harder targets are attained. The single quadratic problem that both BFGS and RLVM are slowest to solve and that is responsible for the kinks near the top of the distribution curves is the *harkerp2* function originally due to Harker and Pang (1990). At the optimizer, the Hessian matrix of that problem for $n = 72$ has a condition number in excess of $7 \cdot 10^7$, and RLVM applies regularization in every run. On unimodal but not quadratic problems the distribution curves of RLVM clearly dominate those of BFGS for medium and distant starts, and they do so in a large range of function evaluation budgets for near starts. Any differences on multimodal problems are far less pronounced.

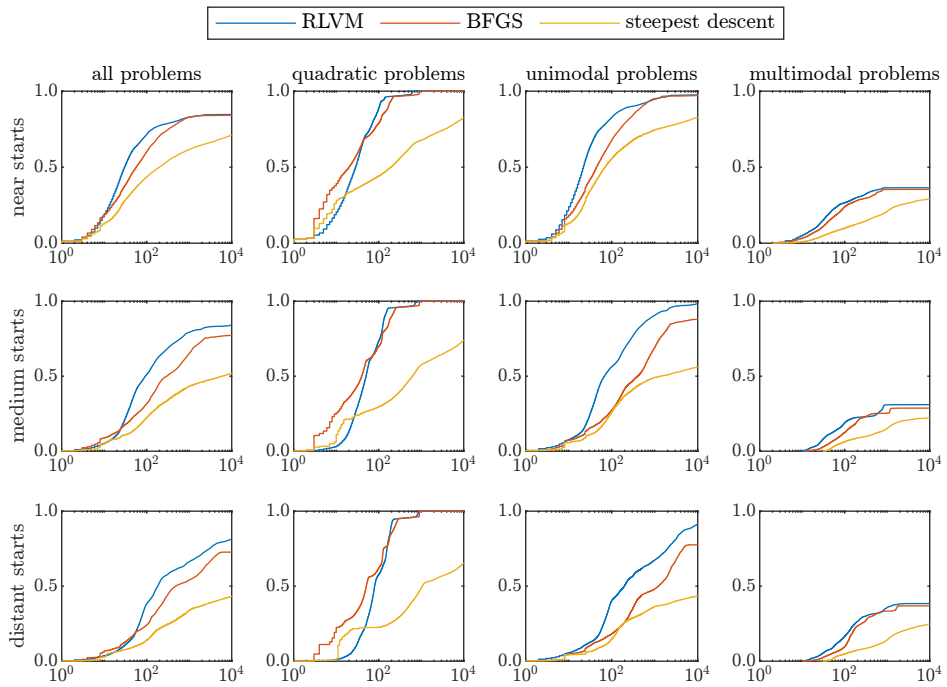

Figure 6: Empirical cumulative running time distributions for dimension $n = 72$. Horizontal axes represent numbers of function evaluations, vertical axes the fraction of function value targets reached.

5 Conclusion

We have proposed RLVM, a first-order optimization algorithm derived from design elements of evolution strategies. Specifically, the use of directions irrespective of the magnitude of gradient vectors, the adaptation of a global scale factor, the reinforcement learning based update of a variable metric, and the use of multiplicative matrix updates have all been lent from the zeroth-order strategies and been used in a first-order setting. The behaviour of the resulting algorithm is invariant to strictly increasing transformations of objective function values and thus possesses a level of robustness that quasi-Newton methods do not. Experiments on a set of 101 test problems suggest that despite its simplicity, RLVM often outperforms BFGS on problems that are not quadratic, and that for those problems where BFGS is superior for small dimensions, that advantage may decrease or even be reversed in higher-dimensional settings.

A significant limitation of RLVM are its algorithm internal costs, which in each iteration are cubic in the problem dimension. Even if, akin to the use of deferred eigen decompositions in CMA-ES (Hansen, 2023), computing the matrix square root needed to obtain the transformation matrix only after a number of iterations that grows linearly with the dimension, truly large-scale applications and being able to compete with limited-memory BFGS (Liu and Nocedal, 1989) require strategy variants with a significantly smaller memory footprint. Evolution strategies with memory requirements linear in the dimension have been proposed by Akimoto and Hansen (2016), Loshchilov (2017), and Akimoto and Hansen (2020), and it remains to adapt their design elements to a first-order setting.

ACKNOWLEDGEMENTS

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

Y. Akimoto and N. Hansen. 2016. Projection-based restricted covariance matrix adaptation for high dimension. In *GECCO '16: Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM Press, New York, NY, 197–204. https://doi.org/10.1145/2908812.2908863

Y. Akimoto and N. Hansen. 2020. Diagonal acceleration for covariance matrix adaptation evolution strategies. *Evolutionary Computation* 28, 3 (2020), 405–435. https://doi.org/10.1162/evco_a_00260

N. Andrei. 2008. An unconstrained optimization test functions collection. *Advanced Modeling and Optimization* 10, 1 (2008), 147–161.

A. Auger, N. Hansen, J. M. Perez Zerpa, R. Ros, and M. Schoenauer. 2009. Experimental comparisons of derivative free optimization algorithms. In *Experimental Algorithms: 8th International Symposium, SEA 2009*, J. Vahrenhold (Ed.). Springer Verlag, Dortmund, Germany, 3–15. https://doi.org/10.1007/978-3-642-02011-7_3

W. C. Davidon. 1959. *Variable metric method for minimization*. Technical Report ANL-5990. Argonne National Lab. https://doi.org/10.2172/4252678

T. Glasmachers and O. Krause. 2020. The Hessian estimation evolution strategy. In *Parallel Problem Solving from Nature — PPSN XVI*, T. Bäck et al. (Eds.). Springer Verlag, Berlin, Heidelberg, 597–609. https://doi.org/10.1007/978-3-030-58112-1_41

N. Hansen. 2023. The CMA evolution stratagy: A tutorial. arxiv:1604.00772. https://doi.org/10.48550/arXiv.1604.00772

N. Hansen, D. V. Arnold, and A. Auger. 2015. Evolution strategies. In *Springer Handbook of Computational Intelligence*, J. Kacprzyk and W. Pedrycz (Eds.). Springer Verlag, Berlin, Heidelberg, 871–898. https://doi.org/10.1007/978-3-662-43505-2_44

N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D Brockhoff. 2021. COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software* 36, 1 (2021), 114–144. https://doi.org/10.1080/10556788.2020.1808977

N. Hansen, S. Finck, R. Ros, and A. Auger. 2009. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. Technical Report RR-6869. INRIA. Available at http://hal.inria.fr/inria-00362633/en/.

N. Hansen, S. D. Müller, and P. Koumoutsakos. 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary Computation* 11, 1 (2003), 1–18. https://doi.org/10.1162/106365603321828970

N. Hansen and A. Ostermeier. 2001. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation* 9, 2 (2001), 159–195. https://doi.org/10.1162/106365601750190398

N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. 2011. Impacts of invariance in search: When CMA-ES and PSO face ill-conditioned and non-separable problems. *Applied Soft Computing* 11, 8 (2011), 5755–5769. https://doi.org/10.1016/j.asoc.2011.03.001

P. T. Harker and J. S. Pang. 1990. A damped Newton method for the linear complementarity problem. In *Computational Solution of Nonlinear Systems of Equations*, E. Allgower (Ed.). Lectures in Applied Mathematics, Vol. 26. American Mathematical Society, Providence, RI, 265–284.

G. A. Jastrebski and D. V. Arnold. 2006. Improving evolution strategies through active covariance matrix adaptation. In *IEEE World Congress on Computational Intelligence — WCCI 2006*. IEEE Press, New York, NY, 9719–9726. https://doi.org/10.1109/CEC.2006.1688662

O. Krause and T. Glasmachers. 2015. A CMA-ES with multiplicative covariance matrix updates. In *GECCO '15: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM Press, New York, NY, 281–288. https://doi.org/10.1145/2739480.2754781

D. C. Liu and J. Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical Programming* 45, 3 (1989), 503–528. https://doi.org/10.1007/BF01589116

I. Loshchilov. 2017. LM-CMA: an alternative to L-BFGS for large scale black-box optimization. *Evolutionary Computation* 25, 1 (2017), 142–171. https://doi.org/10.1162/EVCO_a_00168

J. Nocedal and S. Wright. 2006. *Numerical Optimization* (2nd ed.). Springer Science+Business Media, LLC, New York, NY. https://doi.org/10.1007/978-0-387-40065-5

A. Ostermeier, A. Gawelczyk, and N. Hansen. 1994. Step-size adaptation based on non-local use of selection information. In *Parallel Problem Solving from Nature — PPSN III*, Y. Davidor et al. (Eds.). Springer Verlag, Berlin, Heidelberg, 189–198. https://doi.org/10.1007/3-540-58484-6_263

I. Rechenberg. 1973. *Evolutionsstrategie — Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart-Bad Cannstatt.

H.-P. Schwefel. 1975. *Evolutionsstrategie und numerische Optimierung*. Ph. D. Dissertation. Technische Universität Berlin.