J. Parallel Distrib. Comput. 93–94 (2016) 132–145

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Spline-based parallel nonlinear optimization of function sequences

Tal Ben-Nun[a,b,*], Amnon Barak[a], Uri Raviv[b]

[a] *Department of Computer Science, The Hebrew University of Jerusalem, Jerusalem 9190401, Israel*

[b] *Department of Chemistry, The Hebrew University of Jerusalem, Jerusalem 9190401, Israel*

HIGHLIGHTS

- Presents a scalable algorithm for optimizing sequences of functions in parallel.
- Develops an objective function for modeling nonlinear dynamical system optimization.
- Demonstrates three real-world applications and analyzes the results.
- Performance shows that the algorithm benefits from heterogeneous HPC clusters.
- Two distributed variants of the algorithm are proposed and evaluated.

ARTICLE INFO

Article history:
Received 6 January 2015
Received in revised form
22 September 2015
Accepted 21 April 2016
Available online 28 April 2016

Keywords:
Nonlinear optimization
Parallel optimization
Dynamical systems
Curve fitting

ABSTRACT

Nonlinear dynamical system optimization problems exist in many scientific fields, ranging from computer vision to quantitative finance. In these problems, the underlying optimized parameters exhibit a certain degree of continuity, which can be formulated as a discrete sequence of nonlinear functions. Traditionally, such problems are either solved by ad-hoc algorithms or via independent optimization of the underlying functions. The former solutions are difficult to define and develop, requiring expertise in the field, while the latter approach does not take advantage of the inherent sequential properties of the functions. This paper presents a parallel spline-based algorithm for nonlinear optimization of function sequences, with emphasis on dataset sequences that represent dynamically evolving systems. The presented algorithm provides results that are more coherent with fewer evaluations than independent optimization of the sequence functions. We elaborate on the heuristic approach, the motivation behind using splines to model dynamical systems, and the various tiers of concurrency built into the algorithm. Furthermore, we present two distributed variants of the algorithm and compare their convergence with the serial version. The performance of the algorithm is demonstrated on benchmarks and real-world problems in audio signal decomposition, small angle X-ray scattering analysis, and video tracking of arbitrary objects.

1. Introduction

Large-scale nonlinear optimization problems are recently gaining importance in the scientific community. One subset of these problems focuses on modeling dynamical systems. In this subset, the objective function contains an additional continuous dimension (e.g. time). Since real-world experiments consist of discrete observations, the problem can be formulated as a sequence of related functions to optimize. Examples of nonlinear dynamical system problems include analytical mechanics and physics simulations [23,25]; sensor-based (e.g. GPS) navigation [35]; nonlinear econometric models in financial time series [52]; and other time-evolving processes such as phase transitions in thermodynamic systems.

Current solutions to sequential problems are often area-specific [41,43,13,45], allowing little to no flexibility on the structure of the data (i.e., algorithms that operate on videos will usually not perform well on audio waves); do not leverage parallel and distributed systems; and are not always robust to various input conditions, such as noise. Furthermore, devising such solutions requires deep understanding of the research area, with which specifically-crafted heuristics are added to the algorithms. On the other hand, nonlinear modeling is simpler and only requires basic knowledge of the underlying process.

The importance of function sequence optimization stems from the fact that it estimates models more efficiently by using less parameters and assuming continuity. Moreover, the results of

* Corresponding author at: Department of Computer Science, The Hebrew University of Jerusalem, Jerusalem 9190401, Israel.
E-mail address: talbn@cs.huji.ac.il (T. Ben-Nun).

http://dx.doi.org/10.1016/j.jpdc.2016.04.011
0743-7315/© 2016 Elsevier Inc. All rights reserved.

optimized long sequences (e.g., a month of video footage) can be automatically subdivided to segments using the underlying parameters [24,47,4], a process which aids in finding anomalies and the critical points of the dynamical system.

Traditional optimization of each sequence function independently does not guarantee the coherency of the solution with respect to the sequence, nor produce well-informed parameter estimates based on the continuous properties of the system. This results in an unnecessarily large amount of function evaluations, which may be costly for some problems (e.g., when using numerical integration). More importantly, such algorithms are not scalable with respect to the length of the sequence, with which the memory consumption can grow linearly and sometimes quadratically, depending on the method.

In this paper, we design a novel algorithm that we call Discrete Sequence Optimization (DSO). This algorithm provides a generalized solution to constrained and unconstrained optimization of nonlinear function sequences. As we shall show, DSO converges to better solutions with less evaluations when the underlying functions represent a dynamical system.

Parallel and distributed nonlinear optimization algorithms generally provide better results than their sequential counterparts [44] by evaluating many estimates simultaneously to evade sub-optimal local convergence. The DSO algorithm is inherently parallel and can be distributed across many computing nodes, essentially capable of achieving maximal performance on HPC clusters with multi-GPU machines. It is designed with three tiers of concurrency: data parallelism, task parallelism, and distributed computing. For the distributed computing tier, two different variants of DSO are presented in this paper. These variants utilize distributed global optimization principles to achieve more accurate results than the sequential version.

We also show that the memory complexity of DSO is independent of the length of the sequence. This is especially important for solving large-scale optimization problems on massively parallel architectures, which typically contain a limited amount of RAM and no memory swapping capabilities.

To demonstrate the effectiveness of DSO, we present two benchmarks and three real-world applications for sequence optimization. These applications cover a wide range of research areas, optimizing audio signals for source instrument decomposition, analyzing complex fluid X-ray scattering datasets, and fitting image sequences to track arbitrary objects in videos.

The paper is organized as follows. Section 2 depicts the theoretical background for the algorithm, used throughout the paper. Section 3 presents DSO in detail, along with its distributed counterparts. Case studies that demonstrate the accuracy of DSO are presented in Section 4, and various performance parameters of the algorithm are evaluated in Section 5. Related work and conclusions are discussed in Sections 6 and 7 respectively.

2. Background

This section provides theoretical background for the algorithm presented in this paper.

2.1. Statement of the problem

The problem of unconstrained nonlinear dynamical system optimization can be formally defined as follows. Given an objective function $f : \mathbb{R}^{d+1} \to \mathbb{R}$, denoted as $f(\tau, x)$ where $x \in \mathbb{R}^d$ is some parameter vector, f is defined in the range $\tau \in [a, b]$ and smooth with respect to τ; let g be a scalarization function, which quantifies the values of f in the range $\tau \in [a, b]$ to a real value; find a continuously differentiable parameter function $\mathcal{X}^*(\tau) : \mathbb{R} \to \mathbb{R}^d$ that satisfies:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}:\mathbb{R}\to\mathbb{R}^d} g\left(f, \mathcal{X}, a, b\right).$$

A specific instance of this problem, where $g(f, \mathcal{X}, a, b) \equiv \int_a^b f(\tau, \mathcal{X}(\tau), \mathcal{X}'(\tau))d\tau$, solves the Euler–Lagrange equation [16], which is prominently used in analytical mechanics.

This paper focuses on the discrete version of this problem, finding the parameter matrix $X^* \in \mathbb{R}^{m\times d}$ that satisfies the equation:

$$X^* = \arg\min_{X\in\mathbb{R}^{m\times d}} g\begin{pmatrix} f_1\left(X_{1,*}\right) \\ \vdots \\ f_m\left(X_{m,*}\right)\end{pmatrix}, \tag{1}$$

where $f_t : \mathbb{R}^d \to \mathbb{R}$, $t \in [m]$ is a discrete sequence of m related functions, exhibiting sequential continuity (see below); X^* is sequentially continuous along its rows; and $X_{t,*}$ is equivalent to row t of X.

Since optimizing the f_t functions independently can produce incoherent results with respect to the sequence, it is necessary to find an objective function $g : \mathbb{R}^m \to \mathbb{R}$ that constrains the parameter values along the sequence in order to enforce continuity. To address this issue, we define a metric for sequential continuity as the discrete version of function smoothness, using finite differences instead of derivatives. A parameter sequence is continuous if the sum of its absolute second-order differences, $\|\ddot{X}_t\|$ (defined in Appendix A.1) for $1 < t < m$, is minimal. The scalarization function g is therefore given by:

$$g(\vec{f}, X, \lambda) \equiv \sum_{t=1}^{m} f_t\left(X_{t,*}\right) + \lambda \sum_{t=2}^{m-1}\sum_{i=1}^{d} |c_i \ddot{X}_{t,i}|, \tag{2}$$

where $c \in \mathbb{R}^d$ is a fixed non-negative vector that normalizes the dimensions of the parameters, and $\lambda \geq 0$ is a regularization parameter for the second term. In this paper, the two parts of Eq. (2) are referred to as the data and smoothness terms respectively. The second term functions as an ℓ_1-norm based regularization, applied via Lagrangian relaxation [27]. This technique follows the principle of Tikhonov regularization [50], adding the sequential continuity constraint to the minimized function directly.

2.2. Constrained sequence optimization

The respective constrained problem of discrete sequence optimization is defined by:

$$\begin{aligned} X^* = \arg\min_{X\in\mathbb{R}^{m\times d}} \quad & g(\vec{f}, X, \lambda), \\ \text{s.t.} \quad & X \in C_g \ \wedge \ \forall_{t\in[m]} : X_{t,*} \in C_\ell, \end{aligned}$$

where $C_g \subset \mathbb{R}^{m\times d}$ is the feasible subset of the result matrix, representing global constraints; and $C_\ell \subset \mathbb{R}^d$ is the feasible subset of individual results, representing local constraints. Global constraints enforce sequential coherency of the optimization results, whereas local constraints restrict the optimization of each function independently.

For example, linear box constraints, which are extensively used throughout the paper, can be defined by $C_g = \left\{x' \in \mathbb{R}^{dm} : A_g x' \leq b_g\right\}$ and $C_\ell = \left\{x \in \mathbb{R}^d : Ax \leq b\right\}$, where n, k are the number of global and local constraints respectively, $A_g \in \mathbb{R}^{n\times dm}$, $b_g \in \mathbb{R}^n$, x' is the vector representation of X (having $X_{i,j} = x'_{id+j}$), $A \in \mathbb{R}^{k\times d}$ and $b \in \mathbb{R}^k$.