OpReg-Boost: Learning to Accelerate Online Algorithms with Operator Regression

Anonymous Authors[1]

Abstract

We present OpReg-Boost, a novel acceleration scheme to boost the convergence properties and lessen the asymptotical error of online algorithms for time-varying (weakly) convex optimization problems. OpReg-Boost is built to learn the closest algorithm to a given online algorithm that has all the algorithmic properties one needs for fast convergence, and it is based on the concept of operator regression. We show how to build OpReg-Boost by using a Peaceman-Rachford solver, and further boost [... interpolation-MAP..]. Simulation results showcase the [better/increase/..] properties of OpReg-Boost w.r.t. the more classical [...], and its close relative convex-regression-boost which is also novel but significantly less performing.

1. Introduction

In recent years, we have witnessed the growing interest in time-varying optimization problems, where the cost function, constraints, or both are parametrized over data streams, and therefore are changing over time (**?**). Notable applications for the machine learning community are subspace tracking for video streaming and online sparse subspace clustering, see (**???**) and references therein.

Solving time-varying optimization problems with online algorithms amounts to find and track continuously changing optimizers. If we let the optimizers trajectory be defined as $\boldsymbol{x}^*(t)$, a notable results for online algorithms in this context is that if the *path-length* $\sum_{k=1}^{T} \|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\|$, for subsequent time instances t_{k+1}, t_k, is not sub-linear in T, then online algorithms will have an asymptotical error $O(1)$ (**??**) (or $O(T)$ in terms of pertinent notions of regret)[1].

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]Jad..

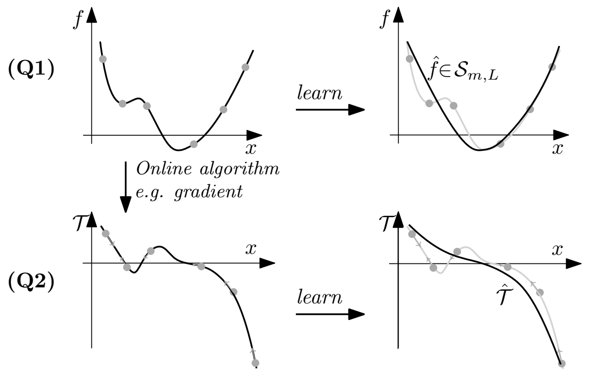

Figure 1. ...

While errors can be made small, for instance with the use of prediction-correction algorithms (**??**), the existence of an asymptotical error remains a distinctive feature of online algorithms for time-varying optimization.

An eye-opening intuition is then to use the existence of said error to one's advantage, by introducing regularizations in the problem formulation that boost the algorithms convergence. In time-varying scenarios, where convergence rate is crucial, often time then one obtains smaller asymptotical error w.r.t. the original problem if one uses regularizations than if they do not. To reiterate, surprisingly, often *there is no trade-off between accuracy and speed* and regularized problems offer better speed and asymptotical errors w.r.t. the original problem, even though we are modifying the problem formulation (**??**).

By building on this field-defining fact, once can ask what is the best regularization for a time-varying problem at hand? In this paper, we explore this question in two slightly different angles. First, we ask ourselves:

(Q1) *what is the closest problem to a given time-varying problem that has all the functional properties we need for fast convergence?*

This gives rise to convex-regression-based boosting.

To fix the ideas, let us consider Figure.. where we have depicted a .. (as a bonus, since we define ..)

Convex regression is however not the only way to learn good problems, and not the best one (as we are going to explore in the simulation section). A better way is operator regression, which stems from the second question we ask:

(Q2) *what is the closest algorithm to a given online algorithm for solving a time-varying problem that has all the algorithmic properties we need for fast convergence?*

Explain OpReg..

In this paper, we offer the following contributions,

1. .

1.1. Literature review

Time-varying optimization and online algorithms are widely treated in (**???**), while a more machine learning approach is offered in (**?**) and subsequent work. The notion that regularizations help convergence possibly without introducing extra asymptotical error is presented in the papers (**???**). While we refer to (**??**) for seminal papers close in spirit in the static domain.

Learning the optimize and regularize is a growing research topic, and we cite (**??**), as related work, even though not on the problem we are interested in solving here.

Convex regression is treated extensively in (**???**), while recently being generalized to smooth strongly convex functions (**?**) based on A. Taylor's work (**?**).

Operator regression is a recent and at the same old topic. We are going to build on the recent work (**?**) and the 19?? paper (**?**).

The acceleration schemes that we compare with are ..

1.2. Notation

Notation is wherever possible standard. We use the concept of μ-weakly convex function, to indicate a function $f : \mathbb{R}^n \to \mathbb{R}$ that becomes convex by adding the term $\frac{\mu}{2}\|\boldsymbol{x} - \boldsymbol{x}_0\|_2^2$, $\mu > 0$ (see (**???**)). The set of convex functions on $\mathbb{R}^n$ that are L-smooth (i.e., have L-Lipschitz continuous gradient) and m-strongly convex is indicated with $\mathcal{S}_{m,L}(\mathbb{R}^n)$.

An operator $\mathcal{T} : \mathbb{R}^n \to \mathbb{R}^n$ is said to be non-expansive iff $\|T\boldsymbol{x} - T\boldsymbol{y}\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, whereas it is ζ-contractive, $\zeta \in (0,1)$, iff $\|T\boldsymbol{x} - T\boldsymbol{y}\| \leq \zeta\|\boldsymbol{x} - \boldsymbol{y}\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$.

2. Problem Formulation

We are interested in solving the following time-varying optimization problem,

$$\boldsymbol{x}^*(t) \in \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}; t) + g(\boldsymbol{x}; t) \tag{1}$$

where $f : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ is closed, proper, and μ-weakly convex, whereas $g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}$ is closed, convex and proper function, optionally with $g \equiv 0$. Upon sampling the problem at discrete time instances t_k, we then obtain the sequence of time-invariant problems,

$$\boldsymbol{x}^*(t_k) \in \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}; t_k) + g(\boldsymbol{x}; t_k), \qquad k \in \mathbb{N}. \tag{2}$$

Our overarching goal is to set up an online algorithm $\mathcal{A}$ that generate a sequence of approximate optimizers $\{\boldsymbol{x}_k\}_{k\in\mathbb{N}}$, such that the asymptotical tracking error

$$\limsup_{k\to\infty} \|\boldsymbol{x}_k - \boldsymbol{x}^*(t_k)\|, \tag{3}$$

is as small as possible. Our main blanket assumption is that optimizers[2] at subsequent time instances cannot be arbitrarily far apart, that is,

$$\|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\| \leq \varDelta < +\infty, \qquad k \in \mathbb{N}, \tag{4}$$

which in turn guarantees that the path-length grows linearly in time,

$$\sum_{k\in\mathbb{N}} \|\boldsymbol{x}^*(t_{k+1}) - \boldsymbol{x}^*(t_k)\| \leq \varDelta k. \tag{5}$$

The first important and known result here is that if function f is in $\mathcal{S}_{m,L}(\mathbb{R}^n)$ uniformly in k, then an online algorithm A (here a forward-backward algorithm) can obtain linear convergence to the asymptotical error bound $\varDelta/(1-\gamma)$, with $\gamma = ..$; much weaker results hold for the general (weakly) convex case (**?**).

The second important result is that a Tikhonov regularization ....

With this in place, our two main research questions are

- **Q1.** Can we project the weakly convex function f onto $\mathcal{S}_{m,L}(\mathbb{R}^n)$
- **Q1.** Can we project the A onto ??

3. Q1. Convex Regression

We briefly introduce here the concept of convex regression, while we leave the main technical details to (**???**). Suppose one has collected ℓ noisy measurements of a convex function

[2]..

$\varphi(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$ (say y_i) at points $\boldsymbol{x}_i \in \mathbb{R}^n$, $i \in I_\ell$. Then convex regression is a least-squares approach to estimate the generating function based on the measurements. Formally, letting $\varphi \in \mathcal{S}_{m,L}(\mathbb{R}^n)$, then one would like to solve the infinite-dimensional problem,

$$\hat{\varphi}_\ell \in \underset{\psi \in \mathcal{S}_{m,L}(\mathbb{R}^n)}{\arg\min} \left\{ \sum_{i \in I_\ell} (y_i - \psi(\boldsymbol{x}_i))^2 \right\}. \tag{6}$$

The problem can be then equivalently decomposed into an estimation on the data points, to find the true function values and gradients at the data point ($\boldsymbol{f} = [\varphi_i]_{i \in I_\ell}$, $\boldsymbol{\delta} = [\nabla \varphi_i]_{i \in I_\ell}$) which turns out to be a convex quadratically constrained quadratic program,

$$\underset{\boldsymbol{f} \in \mathbb{R}^\ell,\, \boldsymbol{\delta} \in \mathbb{R}^{n\ell}}{\text{minimize}} \quad \sum_{i \in I_\ell} (y_i - \varphi_i)^2 \tag{7a}$$

$$\text{subject to :} \quad \varphi_i - \varphi_j - \boldsymbol{\delta}_j^\top (\boldsymbol{x}_i - \boldsymbol{x}_j) \geq \frac{1}{2(1 - m/L)} \left(\frac{1}{L} \|\boldsymbol{\delta}_i - \boldsymbol{\delta}_j\|_2^2 + m \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 - 2\frac{m}{L} (\boldsymbol{\delta}_j - \boldsymbol{\delta}_i)^\top (\boldsymbol{x}_j - \boldsymbol{x}_i) \right), \quad \forall i, j \in I_n \tag{7b}$$

And an interpolation scheme, that extends the point estimate over the whole space maintaining the functional properties,

$$\hat{\varphi}_\ell(\boldsymbol{x}) = \text{conv}(p_i(\boldsymbol{x})) + \frac{m}{2} \|\boldsymbol{x}\|_2^2 \in \mathcal{S}_{m,L}(\mathbb{R}^n), \tag{8}$$

where

$$p_i(\boldsymbol{x}) := \frac{L-m}{2} \|\boldsymbol{x} - \boldsymbol{x}_i\|_2^2 + (\boldsymbol{\delta}_i^* - m\boldsymbol{x}_i)^\top \boldsymbol{x} + - \boldsymbol{\delta}_i^{*,\top} \boldsymbol{x}_i + f_i^* + m/2 \|\boldsymbol{x}_i\|_2^2, \tag{9}$$

and where $\text{conv}(\cdot)$ indicates the convex hull.

Then, to solve Q1, one could consider a number of evaluations of the function $f(\boldsymbol{x}; t_k)$ as noisy measurements of an underlying convex function $\varphi_k \in$ .. and using the least-squares approach above to *project* function $f(\cdot; t_k)$ onto the space of "good" functions ..

4. Q2. Operator Regression

A slightly different approach to convex regression is to look at the algorithm directly, instead of the function. First of all, we remind the reader that any algorithm can be seen as an operator that maps the current approximate optimizer $\boldsymbol{x}_k$ into a new approximate optimizer $\boldsymbol{x}_{k+1}$. For example, a gradient algorithm of the form,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k; t_k) \equiv \underbrace{(I - \alpha \nabla_x f_k)}_{=:\mathcal{T}_k}(\boldsymbol{x}_k) \tag{10}$$

where $\mathcal{T}_k : \mathbb{R}^n \to \mathbb{R}^n$ is the gradient algorithm operator. Interpreting algorithms as operators (possibly averaged, monotone, etc.) has been extremely fruitful for characterizing their convergence properties (**??**).

The counterpart of a smooth strongly convex function φ_k in convex regression, is an operator that is contracting, i.e., .. In fact, while smooth strongly convex functions give rise to ...-contracting gradient algorithm operators, the space of contracting operators is larger (i.e., a function f that .. could give rise to ..).

The key idea now is to interpret .. as measurement of a true .. operator that we want to estimate.

First of all, we define the observations

$$\boldsymbol{y}_i = \mathcal{T}_f \boldsymbol{x}_i + \boldsymbol{e}_i, \quad i \in [D]$$

of the operator we want to regularize. Now, using Fact 2.2 in (Ryu et al., 2020) we know that an operator $\mathcal{T}$ is ζ-contractive interpolable if and only if it satisfies

$$\|\mathcal{T}\boldsymbol{x}_i - \mathcal{T}\boldsymbol{x}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \quad \forall i, j \in [D],\ i \neq j.$$

Therefore, we can define the following constrained regression problem:

$$\hat{\boldsymbol{t}}_i = \underset{\boldsymbol{t}_i \in \mathbb{R}^n}{\arg\min} \frac{1}{2} \sum_{i \in [D]} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|^2 \\ \text{s.t. } \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \ \forall i, j \in [D], i \neq j \tag{11}$$

where again the cost function is a least square terms on the observations and the constraints enforce contractiveness.

4.1. Auto-tuning

4.2. PRS-based solver

5. OpReg-Boost

..

To iron this..

5.1. Problem formulation

Consider the following *convex, time-varying* optimization problem

$$\boldsymbol{x}^*(t) \in \arg\min f(\boldsymbol{x}; t) + g(\boldsymbol{x}; t) \tag{12}$$

where $f : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R}$ and $g : \mathbb{R}^n \times \mathbb{R}_+ \to \mathbb{R} \cup \{+\infty\}$ are closed, convex and proper functions, optionally with $g \equiv 0$. In particular, we are interested in solving the following sequence of static problems derived from sampling (12) at the times $\{t_k\}_{k \in \mathbb{N}}$, $t_{k+1} - t_k = T_\text{s}$:

$$\boldsymbol{x}^*(t) \in \arg\min f(\boldsymbol{x}; t_k) + g(\boldsymbol{x}; t_k). \tag{13}$$

Problem (13) is convex but not strongly so, which means that the performance of online algorithms applied to it is worse than the performance attainable for strongly convex problems. As an example, consider the case $g \equiv 0$, and apply an online gradient descent: if the problem is convex the (static) regret is $O(\sqrt{T})$, with T the number of sampling times, while if the problem is strongly convex the regret is $O(\log(T))$, section 3.1 in (Hazan, 2016). Instead of citing Hazan, we should derive a result in an appendix, for example for the fixed point residual

As we can see from the results just mentioned, in general in online optimization it is not possible to achieve zero regret, due to the dynamic nature of the problem. However, for strongly convex problems smaller regrets can be achieved.

The goal then is to design learning techniques that allow to achieve strongly convex-like performance while tracking a solution trajectory of the problem (13) with good accuracy.

5.2. Q2. Operator regression

First of all, we define the observations

$$\boldsymbol{y}_i = \mathcal{T}_f \boldsymbol{x}_i + \boldsymbol{e}_i, \quad i \in [D]$$

of the operator we want to regularize. Now, using Fact 2.2 in (Ryu et al., 2020) we know that an operator $\mathcal{T}$ is ζ-contractive interpolable if and only if it satisfies

$$\|\mathcal{T}\boldsymbol{x}_i - \mathcal{T}\boldsymbol{x}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2, \quad \forall i,j \in [D],\ i \neq j.$$

Therefore, we can define the following constrained regression problem:

$$\begin{aligned} \hat{\boldsymbol{t}}_i = \operatorname*{arg\,min}_{\boldsymbol{t}_i \in \mathbb{R}^n} & \frac{1}{2} \sum_{i \in [D]} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|^2 \\ \text{s.t. } & \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \ \forall i,j \in [D], i \neq j \end{aligned} \tag{14}$$

where again the cost function is a least square terms on the observations and the constraints enforce contractiveness.

Auto-tuning contraction We can modify (14) by including $w = \zeta^2$ as an unknown of the regression problem, which becomes

$$\begin{aligned} \hat{\boldsymbol{t}}_i = \operatorname*{arg\,min}_{\boldsymbol{t}_i \in \mathbb{R}^n} & \frac{1}{2} \sum_{i \in [D]} \|\boldsymbol{t}_i - \boldsymbol{y}_i\|^2 + \frac{c}{2} w^2 \\ \text{s.t. } & \|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 - w \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \leq 0 \ \forall i,j \in [D], i \neq j \end{aligned} \tag{15}$$

this way the contraction constant does not need to be specified, and is *auto-tuned* by the regression.

PRS-based solver .... brief description of basic characteristics, reference to the appendix ....

6. OpReg-Boost

Let now $\mathcal{T}_k : \mathbb{R}^n \to \mathbb{R}^n$ be a solver for the convex problem observed at time t_k, and assume that the operator has a term $\mathcal{T}_{f,k}$ depending exclusively on f (*e.g.* we have $\mathcal{T}_k = \mathcal{T}'_k \circ \mathcal{T}_{f,k}$). Since the sampled problem is convex, the solver $\mathcal{T}_k$ is non-expansive (actually, averaged, to guarantee convergence).

The idea then is to try and learn an approximation of $\mathcal{T}_k$ that is *contractive* rather than non-expansive.

Again, w.l.o.g. we are only interested in learning the term $\mathcal{T}_{f,k}$ that depends on f. The algorithm can be described as follows: at each time t_k do:

1. sample a new problem, that is, observe $f(\boldsymbol{x}; t_k)$ and $g(\boldsymbol{x}; t_k)$;
2. approximate the term $\mathcal{T}_{f,k}$ of the solver with a contractive one, and
3. apply the resulting solver. For example if $\mathcal{T}_k = \mathcal{T}'_k \circ \mathcal{T}_{f,k}$, then we apply $\mathcal{T}_k = \mathcal{T}'_k \circ \hat{\mathcal{T}}_{f,k}$ where $\hat{\mathcal{T}}_{f,k}$ is the learned operator.

The learning step is performed using the (novel) constrained operator regression OpReg described in the following.

In particular we have the following algorithm make this pseudo code For $\ell \in 0, 1, \dots, M$ (for some $M \in \mathbb{N}$):

- let $\boldsymbol{x}_k^\ell$ be the current approximate solution, we choose $D-1$ points around it, *e.g.*

  $$\boldsymbol{x}_i = \boldsymbol{x}_k^\ell + \boldsymbol{p}_i,\ i = 2, \dots, D,\ \boldsymbol{p}_i \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I}_n)$$

  and set $\boldsymbol{x}_1 = \boldsymbol{x}_k^\ell$;
- we compute the data $\boldsymbol{t}_i = \mathcal{T}_{f,k}\boldsymbol{x}_i$, $i = 1, \dots, D$, for example $\boldsymbol{t}_i = \boldsymbol{x}_i - \alpha \nabla f(\boldsymbol{x}_i; t_k)$;
- we solve the OpReg, and use the approximate operator at $\boldsymbol{x}_k^\ell$, that is $\boldsymbol{t}_1$, in the chosen solver $\mathcal{T}_k$:

  $$\boldsymbol{x}_k^{\ell+1} = \operatorname{prox}_{\alpha g}(\boldsymbol{t}_1)$$

  where α is a step-size.

6.1. Interpolated version

7. Numerical Results

7.1. Simulations set-up

We consider the following time-varying problem:

$$\boldsymbol{x}^*(t_k) = \operatorname*{arg\,min}_{\boldsymbol{x} \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}(t_k)\|^2 + w \|\boldsymbol{x}\|_1 \tag{16}$$

with $n = 10$, $\boldsymbol{A}$ matrix with maximum and minimum (non-zero) eigenvalues $\sqrt{L} = 10^8$, $\sqrt{\mu} = 1$, and with rank 5; $\boldsymbol{y}(t_k)$ has sinusoidal components with 3 zero components. Due to $\boldsymbol{A}$ being rank deficient, the cost f is convex but not strongly so.

7.2. Results

References

Hazan, E. Introduction to Online Convex Optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

Ryu, E. K., Taylor, A. B., Bergeling, C., and Giselsson, P. Operator Splitting Performance Estimation: Tight Contraction Factors and Optimal Parameter Selection. *SIAM Journal on Optimization*, 30(3):2251–2271, 2020.

Simonetto, A. Smooth Strongly Convex Regression. In *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 2130–2134, Amsterdam, January 2021. IEEE.

A. PRS-Based QCQP solver

In this section we present a solver for OpReg which can be efficiently parallelized, inspired by the approach in (Simonetto, 2021).

The idea is as follows: each pair of data points $i, j \in [D]$, $i \neq j$, gives rise to one constraint, for a total of $D(D-1)/2$ constraints. We define the following set of pairs

$$\mathcal{V} = \{e = (i,j) \mid i, j \in [D],\ i < j\}$$

which are ordered (that is, for example we take $(1,2)$ and not $(2,1)$, to avoid counting constraints twice). Clearly to each pair $e = (i,j)$ corresponds the constraint $\|\boldsymbol{t}_i - \boldsymbol{t}_j\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2$.

Let now $\boldsymbol{t}_{i,e}$ and $\boldsymbol{t}_{j,e}$ be copies of $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ associated to the e-th constraint; then we can equivalently reformulate the OpReg (14) as

$$\min_{\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}} \frac{1}{2(D-1)} \sum_{e \in \mathcal{V}} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 \tag{17a}$$

$$\text{s.t. } \|\boldsymbol{t}_{i,e} - \boldsymbol{t}_{j,e}\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 \tag{17b}$$

$$\boldsymbol{t}_{i,e} = \boldsymbol{t}_{i,e'} \ \forall e, e' | i \sim e, e'. \tag{17c}$$

Clearly (17) is a strongly convex problem with convex constraints defined in the variables $\boldsymbol{t}_{i,e}$.

A.1. PRS solver

Let $\boldsymbol{\xi}$ be the vector stacking all the $\boldsymbol{t}_{i,e}$, then the problem is equivalent to

$$\min_{\boldsymbol{\xi}} f(\boldsymbol{\xi}) + g(\boldsymbol{\xi})$$

where

$$f(\boldsymbol{\xi}) = \frac{1}{2(D-1)} \|\boldsymbol{\xi} - \boldsymbol{y}\|^2 + f_1(\boldsymbol{\xi})$$

with f_1 the indicator function imposing (17b) and g the indicator function imposing the "consensus" constraints (17c). The problem can then be solved using the Peaceman-Rachford splitting (PRS) characterized by the following updates $\ell \in \mathbb{N}$:

$$\boldsymbol{\xi}^\ell = \operatorname{prox}_{\rho f}(\boldsymbol{z}^\ell) \tag{18a}$$

$$\boldsymbol{v}^\ell = \operatorname{prox}_{\rho g}(2\boldsymbol{\xi}^\ell - \boldsymbol{z}^\ell) \tag{18b}$$

$$\boldsymbol{z}^{\ell+1} = \boldsymbol{z}^\ell + \boldsymbol{v}^\ell - \boldsymbol{\xi}^\ell. \tag{18c}$$

The proximal of g corresponds to the projection onto the consensus space, and thus can be characterized simply by

$$\boldsymbol{v}_{i,e}^\ell = \frac{1}{D-1} \sum_{e' | i \sim e'} \left(2\boldsymbol{t}_{i,e'}^\ell - \boldsymbol{z}_{e'}^\ell\right).$$

Regarding the proximal of f, is is clear that f is separable, in the sense that it can be written as

$$f(\boldsymbol{\xi}) = \sum_{e \in \mathcal{V}} \left[\frac{1}{2(D-1)} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 + \iota_e(\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}) \right]$$

where ι_e denotes the indicator function of (17b). Therefore, the update (18a) can be solved by solving (possibly in parallel) the problems

$$(\boldsymbol{t}_{i,e}, \boldsymbol{t}_{j,e}) = \arg\min \left\{ \frac{1}{2(D-1)} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} \right\|^2 + \frac{1}{2\rho} \left\| \begin{bmatrix} \boldsymbol{t}_{i,e} \\ \boldsymbol{t}_{j,e} \end{bmatrix} - \boldsymbol{z}_e^\ell \right\| \right\}$$
$$\text{s.t.} \quad \|\boldsymbol{t}_{i,e} - \boldsymbol{t}_{j,e}\|^2 \leq \zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2. \tag{19}$$

A.2. Local updates

The problems (19) are quadratic programs with quadratic constraints, that is, they can be written in the form

$$\min_{\boldsymbol{\xi}} \frac{1}{2} \boldsymbol{\xi}^\top \boldsymbol{P}_0 \boldsymbol{\xi} + \langle \boldsymbol{q}_0, \boldsymbol{\xi} \rangle \tag{20a}$$

$$\text{s.t. } \frac{1}{2} \boldsymbol{\xi}^\top \boldsymbol{P}_1 \boldsymbol{\xi} + \langle \boldsymbol{q}_1, \boldsymbol{\xi} \rangle + r_1 \leq 0. \tag{20b}$$

In particular, for the cost function we have

$$\boldsymbol{P}_0 = \left(\frac{1}{D-1} + \frac{1}{\rho} \right) \boldsymbol{I}_{2n}, \quad \boldsymbol{q}_0 = -\left(\frac{1}{D-1} \begin{bmatrix} \boldsymbol{y}_i \\ \boldsymbol{y}_j \end{bmatrix} + \frac{1}{\rho} \boldsymbol{z}_e^\ell \right)$$

and for the constraint

$$\boldsymbol{P}_1 = 2 \begin{bmatrix} \boldsymbol{I}_n & -\boldsymbol{I}_n \\ -\boldsymbol{I}_n & \boldsymbol{I}_n \end{bmatrix}, \quad \boldsymbol{q}_1 = 02n, \quad r_1 = -\zeta^2 \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2.$$

B. Interpolation Using MAP