# End-to-End Learning to Warm-Start for Real-Time Quadratic Optimization

Rajiv Sambharya, Georgina Hall , Brandon Amos, and Bartolomeo Stellato

presented by Francesco Demelas

# Table of Contents

# Table of Contents

# Main Ideas of the Approach

- **First-order methods** to solve **convex quadratic programs** (QPs), **Douglas-Rachfors** Splitting:
  - *low per-iteration cost*.
  - suffer from *slow convergence* to accurate solutions.

  **Learn a warm-start** for a popular first-order method (DR splitting), **two modules**:
  - *feedforward neural network* block (input the parameters of the QP) and outputs a warm-start,
  - block which performs a *fixed number of iterations of DR splitting* and outputs a candidate solution.
- A key feature is the d**differentiation through the DR iterations**.
- Provide **generalization bounds** that improve with the number of training problems and the number of iterations simultaneously.
- **Applied to three real-time applications** and they are able to significantly reduce the number of iterations required to obtain high-quality solutions

# Literature Review

**Other Papers**:

- Learning to warmstart solvers
- Unrolling to learn Step Sizes for an Algorithm
- Learning Surrogate Model

**This Paper**:

- With a Loss that takes into account some iterations of the solver
- Unrolling to learn warmstart
- No approximation and no surrogates model

# Table of Contents

# Quadratic Problems

## Formulation Parametric (convex) QP

$$\min(1/2)x^T Px + c^T x$$
$$s.t. Ax + s = b$$
$$s \geq 0$$

with

- **parameter** $\theta = (vec(P), vec(A), c, b) \in \mathbb{R}^d$,
- decision variables $x \in \mathbb{R}^n$ and $s \in \mathbb{R}^m$,
- P is a positive semidefinite matrix in $S_+^{n \times n}$,
- b and c are vectors in $\mathbb{R}^m$ and $\mathbb{R}^n$ respectively,
- For a matrix Y, vec(Y ) denotes the vector obtained by stacking the columns of Y .

## **Goal**:

quickly solve the QP with $\theta$ randomly drawn from a distribution $\mathcal{D}$ with compact support set $\Theta$, assuming that it admits an optimal solution for any $\theta \in \Theta$.

# Optimality conditions

The KKT optimality conditions of the QP problem (primal and dual feasibility, and complementary slackness) are

## KKT optimality conditions

$$
\begin{aligned}
Ax + s &= b, \\
ATy + Px + c &= 0, \\
s &\geq 0, \\
y &\geq 0, \\
s \perp y &= 0,
\end{aligned}
$$

where $y \in \mathbb{R}^m$ is the dual variable to QP problem.

# Reformulation

We can compactly write these conditions as a linear complementarity problem, i.e., the problem of finding

$$u = (x, y) \in \mathbb{R}^{m+n}$$

such that

$$\mathcal{C} \ni u \perp Mu + q \in \mathcal{C}^*,$$

where

$$M = \begin{pmatrix} P & A^T \\ A & 0 \end{pmatrix} \in \mathbb{R}^{(m+n)\times(m+n)}$$

and $q = (c, b) \in \mathbb{R}^{m+n}$, $\mathcal{C} = \mathbb{R}^n \times \mathbb{R}^m_+$ and $\mathcal{C}^* = \{0\}^n \times \mathbb{R}^m_+$ is the dual cone of $\mathcal{C}$.

# Another Reformulation

This problem is equivalent to:

finding $u \in \mathbb{R}^m + m$ that satisfy the following inclusion

$$0 \in Mu + q + \mathcal{N}_C(u)$$

where $\mathcal{N}_C(u)$ is the normal cone of $\mathcal{C}$ defined as
$\mathcal{N}_C(u) = \{x | (y-u)^T x \leq 0,\ \forall y \in \mathcal{C}\}$ is $u \in \mathcal{C}$ and  otherwise.

To **ensure convergence** of the algorithm (that we define next) is the fact that $Mu + q + \mathcal{N}_C(u)$ is the **maximal monotone**.
This follows from $P \succeq 0$, $\mathcal{C}$ **convex polyhedron**, and the starting QP **admitting an optimal solution**.

# Table of Contents

# Douglas-Rachfors (DR) splitting

For an operator $F$ the *resolvent* is $(I + F)^{-1}$
DR consists on evaluating the *resolvent* of the operators $Mu + q$ and $\mathcal{N}_{\mathcal{C}}$.
The resolvent of $Mu + q$ is $(M + I)^{-1}(z - q)$
and for $\mathcal{N}_{\mathcal{C}}$ is $\Pi_{\mathcal{C}}(z)$, i.e. the projection onto $\mathcal{C}$.
Hence we obtain the DR algorithm:

**Algorithm 1** The DR Splitting algorithm for $k$ iterations to solve problem (4).

**Inputs:** initial point $z^0$, problem data $(M, q)$, tolerance $\epsilon$, $k$ number of iterations
**Output:** approximate solution $z^k$
**for** $i = 0, \ldots, k-1$ **do**

$$u^{i+1} = (M + I)^{-1}(z^i - q)$$
$$\tilde{u}^{i+1} = \Pi_{\mathcal{C}}(2u^{i+1} - z^i)$$
$$z^{i+1} = z^i + \tilde{u}^{i+1} - u^{i+1}$$

**end**

**Algorithm 1** The DR Splitting algorithm for $k$ iterations to solve problem (4).

**Inputs:** initial point $z^0$, problem data $(M, q)$, tolerance $\epsilon$, $k$ number of iterations
**Output:** approximate solution $z^k$
**for** $i = 0, \ldots, k-1$ **do**

$$u^{i+1} = (M + I)^{-1}(z^i - q)$$
$$\tilde{u}^{i+1} = \Pi_{\mathcal{C}}(2u^{i+1} - z^i)$$
$$z^{i+1} = z^i + \tilde{u}^{i+1} - u^{i+1}$$

**end**

- The linear system defined in the first step is **always solvable**
- The **projection** onto $\mathcal{C}$ simply clips negative values to zero and leaves non-negative values unchanged.

## Compact Formulation DR:

$$z^{i+1} = T_\theta(z^i) \quad \text{where} \quad T_\theta(z) = z + \Pi_{\mathcal{C}}(2(M+I)^{-1}(z-q) - z) - (M+I)^{-1}(z-q)$$

## Compact Formulation DR:

$$z^{i+1} = T_\theta(z^i) \quad \text{where} \quad T_\theta(z) = z + \Pi_{\mathcal{C}}(2(M+I)^{-1}(z-q) - z) - (M+I)^{-1}(z-q)$$

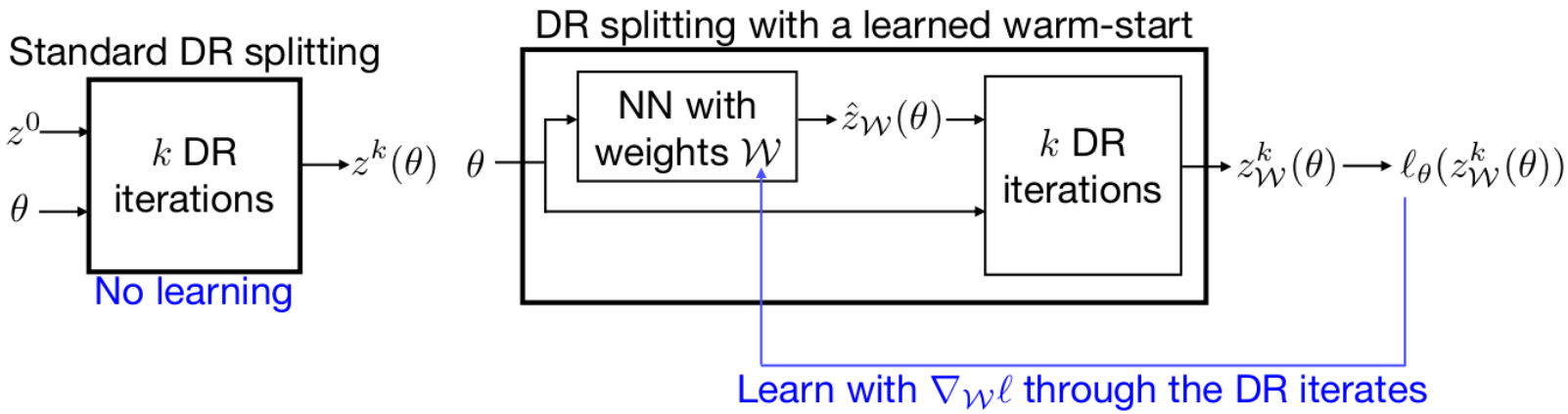
- DR splitting is **guaranteed to converge** to a fixed point $z^* \in \mathbf{fix}\, T_\theta$ such that $T_\theta(z^*) = z^*$.
- DR splitting returns an approximate solution $z^k$ from which we can **recover an approximated primal-dual solution** of the original QP by computing

$$(x^k, y^k) = u^k = (M + I)^{-1}(z^k - q)$$

and $s^k = b - Ax^k$

# Table of Contents

# Architecture




**Two Modules**:

- Standard Multi-Layer Perceptron with weigths $\mathcal{W}$:

$$z_{\mathcal{W}}(\theta) = h_L(h_{L-1}(\cdots h_1(\theta)))$$

  with $h_i(y_l) = (W_l y_l + b_l)_+$.
- Optimization Block: k-iterations of DR splitting

$$T^k_\theta(z_{\mathcal{W}}(\theta)) = z^k_{\mathcal{W}}(\theta)$$

# Learning Task

The loss function is the fixed-point residual of the operator $T_\theta$:

$$l_\theta(z) = ||T_\theta(z) - z||_2$$

The learning problem consists then in minimizing, w.r.t $\mathcal{W}$, the risk:

$$\mathbb{E}_{\theta \sim \mathcal{D}} \left[ l_\theta( T_\theta^k(h_\mathcal{W}(\theta)) \right]$$

To differentiate through the second block we have just to observe that one iteration simply consists of solving linear systems and projections onto $\mathcal{C}$: linear systems are differentiable as they always have a unique solution and the projection is differentiable (everywhere less than in zero) as it simply consists in clipping non-negative values to zero.

# Table of Contents

# Generalization Bounds

Upper bound one the expected loss $R(h_{\mathcal{W}})$ of our framework for any $h_{\mathcal{W}} in \mathcal{H}$. DR splitting achieves a **linear convergence rate**, i.e.

$$dist_{fix T_\theta} T_\theta(z) \leq \beta_\theta dist_{fix T_\theta} z$$

where $dist_S(x) = \min\{||x - y|| \quad | \, y \in S\}$ and $\beta_\theta \in (0, 1)$

**Theorem 1.** *Let* $\beta = \max_{\theta \in \Theta} \beta_\theta$ *for* $\beta_\theta$ *as in* (8). *Assume that* $\mathcal{H}$ *is the set of mappings defined in Section* 3 *with the additional assumption that for any* $h_{\mathcal{W}} \in \mathcal{H}$, $\mathbf{dist}_{\mathbf{fix}\, T_\theta}(h_{\mathcal{W}}(\theta)) \leq B$ *for some* $B > 0$ *and any* $\theta \in \Theta$. *Then, with probability at least* $1 - \delta$ *over the draw of i.i.d samples,*

$$R(h_{\mathcal{W}}) \leq \hat{R}(h_{\mathcal{W}}) + 2\sqrt{2}\beta^k \left(2\mathbf{rad}(\mathcal{H}) + B\log(1/\delta)/(2N)\right), \quad \forall h_{\mathcal{W}} \in \mathcal{H},$$

*where* $k$ *is the number of iterations of DR splitting in the second module,* $N$ *is the number of training samples,* $\mathbf{rad}(\mathcal{H})$ *is the Rademacher complexity of* $\mathcal{H}$, *and* $\beta \in (0, 1)$.

In the case of NNs linear functions with the bounded norm or 2-layer NNs with ReLU activation function, we can provide a bound on the generalization error of our framework which makes the dependence of k and N more explicit.

**Corollary 2.** *Let $\mathcal{H}$ be the set of linear functions with bounded norm, i.e., $\mathcal{H} = \{h \mid h(\theta) = W\theta\}$ where $\theta \in \mathbf{R}^d$, $W \in \mathbf{R}^{(m+n)\times d}$ and $(1/2)||W||_F^2 \leq B$ for some $B > 0$. Then, with probability at least $1-\delta$ over the draw of i.i.d samples,*

$$R(h_{\mathcal{W}}) \leq \hat{R}(h_{\mathcal{W}}) + 2\sqrt{2}\beta^k \left(2\rho_2(\theta)\sqrt{2d/N} + B\log(1/\delta)/(2N)\right), \quad \forall h_{\mathcal{W}} \in \mathcal{H},$$

*where $k$ and $N$ are as defined in Theorem 1, and $\rho_2(\theta) = \max_{\theta\in\Theta} ||\theta||_2$ (Mohri et al., 2012, Thm. 5.10).*

# Table of Contents

- Example problems are QPs problem repeatedly solved in control and portfolio optimization.
- 10000 training problems, 2000 test
- MLP with 3 hidden layers of size 500 each.

# Oscillating Masses

$$
\begin{array}{ll}
\text{minimize} & x_T^T Q_T x_T + \sum_{t=1}^{T-1} x_t^T Q x_t + u_t^T R u_t \\
\text{subject to} & x_{t+1} = A x_t + B u_t \quad t = 0, \ldots, T-1, \\
& u_{\min} \leq u_t \leq u_{\max} \quad t = 0, \ldots, T-1 \\
& x_{\min} \leq x_t \leq x_{\max} \quad t = 1, \ldots, T, \\
& x_0 = x_{\text{init}}
\end{array}
$$

where the states $x_t \in \mathbf{R}_x^n$ and the inputs $u_t \in \mathbf{R}^{n_u}$ are subject to lower and upper bounds. Matrices $A \in \mathbf{R}^{n_x \times n_x}$ and $B \in \mathbf{R}^{n_x \times n_u}$ define the system dynamics. The horizon length is $T$ and the parameter $\theta$ is initial state $x_{\text{init}}$. Matrices $Q \in \mathbf{S}_+^{n_x}$ and $R \in \mathbf{S}_{++}^{n_u}$ define the state and input costs at each stage, and $Q_T \in \mathbf{S}_+^{n_x}$ the final stage cost.

- states $x_t \in \mathbb{R}_x^n$
- inputs $u_t \in \mathbb{R}^{n_u}$
- the matrices $A$ and $B$ define the system dynamics.
- Q and S defines the state and input costs at each stage.
- **parameter** $\theta$ is the initial state $x_{init}$

## Time horizon T=50

Table 1: Oscillating masses problem. We compare the number of iterations of DR splitting required to reach different levels of accuracy with different warm-starts (learned warm-start with $k = 5, 15, 50$, no warm-start, and a nearest neighbor warm-start). The reduction columns are the iterations reduced as a fraction of the no learning iterations.

| | no learning | nearest neighbor | | train $k = 5$ | | train $k = 15$ | | train $k = 50$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | iters | iters | reduction | iters | reduction | iters | reduction | iters | reduction |
| 0.01 | 381 | 353 | 0.07 | 279 | 0.27 | 176 | 0.54 | 127 | 0.67 |
| 0.001 | 651 | 616 | 0.05 | 555 | 0.15 | 438 | 0.33 | 338 | 0.48 |
| 0.0001 | 1019 | 973 | 0.05 | 932 | 0.09 | 816 | 0.20 | 663 | 0.35 |

# Vehicle dynamics control problem

$$
\begin{array}{ll}
\text{minimize} & (y_T - y_T^{\text{ref}})^T Q (y_T - y_T^{\text{ref}}) + \sum_{t=1}^{T-1} (y_t - y_t^{\text{ref}})^T Q_T (y_t - y_t^{\text{ref}}) + u_t^T R u_t \\
\text{subject to} & x_{t+1} = A(v)x_t + B(v)u_t + E(v)\delta_t \quad t = 0, \ldots, T-1 \\
& |u_t| \leq \bar{u}, \quad |u_t - u_{t-1}| \leq \overline{\Delta u}, \quad t = 0, \ldots, T-1 \\
& y_t = C x_t, \quad t = 0, \ldots, T-1 \\
& x_0 = x_{\text{init}},
\end{array}
$$

- state $x_t \in \mathbb{R}^4$
- input $u_t \in \mathbb{R}^3$
- $\sigma_t \in \mathbb{R}$ driver steering input
- $y_t \in \mathbb{R}^3$ output
- $y_t^{ref} \in \mathbb{R}^3$ reference trajectory
- $Q, Q_T$ defines the states costs, $R$ the input costs and $C$ the output costs.
- $v \in \mathbb{R}$ longitudinal velocity
- $A, B, E$ parametrize $v$
- $\bar{u}, \bar{\nabla} u$ bound the magnitude of inputs and change with inputs.
- **parameter** $\theta = (x_{init}, v, u_{.-1}, (y_t^{ref})_{t=0}^T, (\sigma_t)_{t=0}^T)$

## Time horizon T=30

Table 2: Vehicle problem. We compare the number of iterations of DR splitting required to reach different levels of accuracy with different warm-starts (learned warm-start with $k = 5, 15, 50$, no warm-start, and a nearest neighbor warm-start). The reduction columns are the iterations reduced as a fraction of the no learning iterations.

| | no learning | nearest neighbor | | train $k = 5$ | | train $k = 15$ | | train $k = 50$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | iters | iters | reduction | iters | reduction | iters | reduction | iters | reduction |
| 0.01 | 639 | 520 | 0.19 | 203 | 0.68 | 48 | 0.92 | 48 | 0.92 |
| 0.001 | 1348 | 1163 | 0.14 | 895 | 0.34 | 351 | 0.74 | 299 | 0.78 |
| 0.0001 | 2126 | 1948 | 0.08 | 1653 | 0.22 | 1006 | 0.53 | 882 | 0.59 |

# Portfolio Optimization

$$\begin{array}{ll} \text{maximize} & \rho\mu^T x - x^T \Sigma x \\ \text{subject to} & \mathbf{1}^T x = 1, \quad x \geq 0, \end{array}$$

- $x \in \mathbb{R}^n$ portfolio
- $\mu \in \mathbb{R}^n$ expected returns
- $1/\rho > 0$ the risk-aversion parameter
- $\sum \in S_+^n$ the covariance.
- **parameter** $\theta = \mu$

Table 3: Markowitz problem. We compare the number of iterations of DR splitting required to reach different levels of accuracy with different warm-starts (learned warm-start with $k = 5, 15, 50$, no warm-start, and a nearest neighbor warm-start). The reduction columns are the iterations reduced as a fraction of the no learning iterations.

| | no learning | nearest neighbor | | train $k = 5$ | | train $k = 15$ | | train $k = 50$ | |
|---|---|---|---|---|---|---|---|---|---|
| $\epsilon$ | iters | iters | reduction | iters | reduction | iters | reduction | iters | reduction |
| 0.01 | 14 | 7 | 0.5 | 7 | 0.5 | 9 | 0.36 | 11 | 0.21 |
| 0.001 | 54 | 24 | 0.56 | 22 | 0.59 | 16 | 0.7 | 19 | 0.65 |
| 0.0001 | 186 | 148 | 0.2 | 147 | 0.21 | 72 | 0.61 | 61 | 0.67 |

# The three problems together

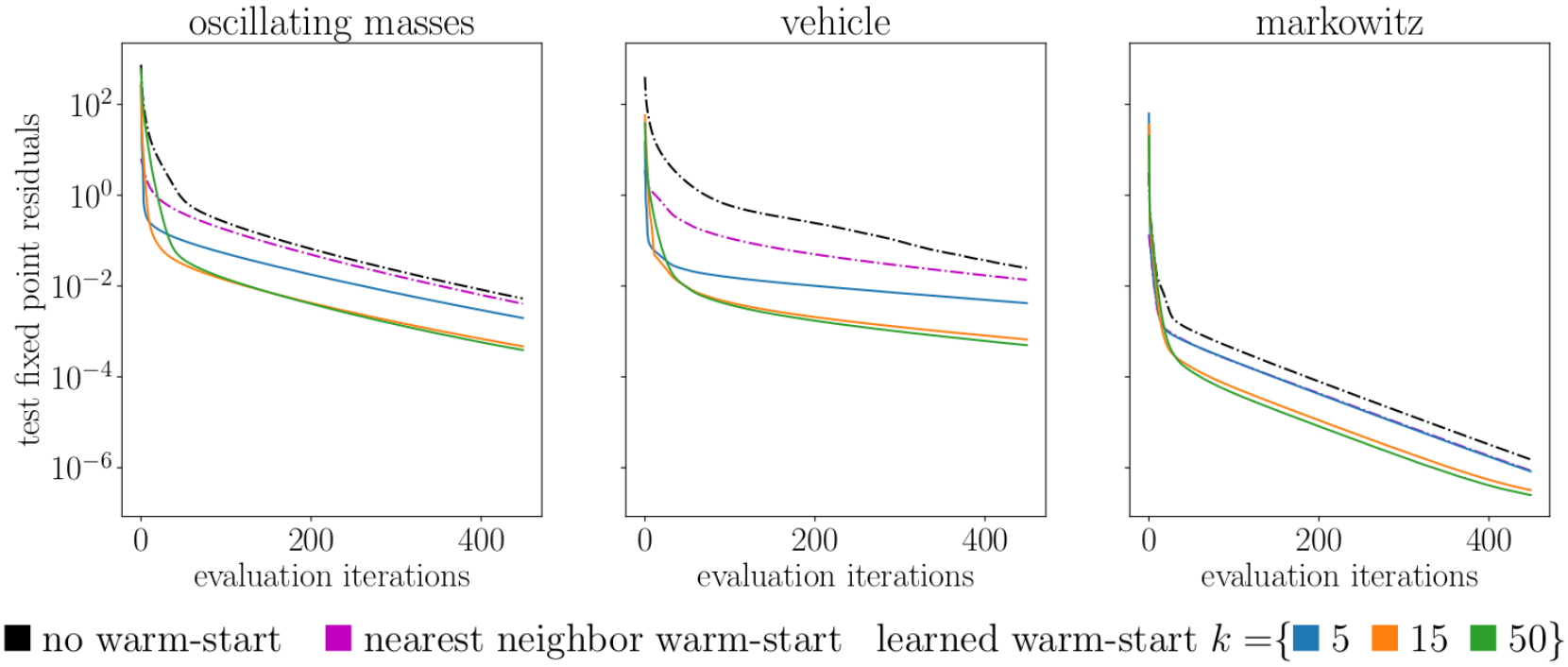



Figure 2: We plot the test fixed point residuals for different warm-starts of DR splitting. We train our architecture with $k = 5, 15$, and 50 DR iterations with loss function (7). We compare our results against a random initialization (black) and against warm-starting DR splitting with the nearest neighbor from the train set (magenta). Left: oscillating masses example. Middle: vehicle dynamics example. Right: portfolio optimization example.

# Table of Contents

# What we can do in a more general context?

In order to use a similar approach for the SGD or for the Bundle method we first need each iteration in order to:

- deduce a fixed point equation
- assures the differentiability or at least the sub-differentiability

We can start with the easier algorithm the Gradient Descent.

In this case if we denote by $z \in \mathbb{R}^m$ the Lagrangian multipliers vector and we want minimize $l(z)$, An iteration is given by

$$z^{i+1} = z^i - \alpha \partial_z l(z)$$

hence the fixed point iteration is

$$z = z - \alpha \partial_z l(z)$$

This means that

$$T(z) = z - \alpha \partial_z l(z)$$

and we are minimizing

$$||T(z) - z||_2 = ||z - z + \alpha \partial_z l(z)||_2 = ||\alpha \partial_z l(z)||_2$$

and this is not necessarily a good idea as we do not necessarily calculate the second derivative $\partial_z \partial_z l(z)$.
For example if $l(z) = \min_x z^t x$, then $\partial l(z) = \arg\min_x z^t x$ and we cannot calculate $\partial_z^2 l(z) = \partial_z \arg\min_x z^t x$ The only possibility is to **approximate** the derivative of the argmin.

For the bundle method, we can write an iteration as:

$$z^{i+1} = \bar{z}^i + \arg\max_d(\min_{j \in B_i}(g_j^t d + \alpha_j))$$

Here $\bar{z}^i$ is the stabilization point and is given by

$$\bar{z}^i = \begin{cases} \bar{z}^{i-1} & \text{if} l(z^i) < l(\bar{z}^{i-1}) \\ z^i & \text{otherwise} \end{cases}$$

Clearly it is not possible to write $\bar{z}^i$ without taking into account all the trajectory of $z$ composed by the algorithm. Anyway we can observe that near to the optimum $\bar{z} \simeq z$ and so, also in this case consider a functor $T$ of the form

$$T(z) = z + \partial_z \arg\max_d(\min_{j \in B_i}(g_j^t d + \alpha_j))$$

. Hence, also in this case we are minimizing $||T(z) - z||_2 = ||z - z + \partial_z \arg\max_d(\min_{j \in B_i}(g_j^t d + \alpha_j)||_2 = ||\partial_z \arg\max_d(\min_{j \in B_i}(g_j^t d + \alpha_j)||_2$. In this case is even "worst" than the SGD as we need to compute $\partial_z \arg\max_d(\min_{j \in B_i}(g_j^t d + \alpha_j)$ and $\partial_z^2 \arg\max_d(\min_{j \in B_i}(g_j^t d + \alpha_j)$. Both these sub-gradients need an approximation.