A Multilevel Proximal Gradient Algorithm for Large Scale Optimization

Panos Parpas

June 30, 2016

Abstract Composite optimization models consist of the minimization of the sum of a smooth (not necessarily convex) function and a non-smooth convex function. Such models arise in many applications where, in addition to the composite nature of the objective function, a hierarchy of models is readily available. It is common to take advantage of this hierarchy of models by first solving a low fidelity model and then using the solution as a starting point to a high fidelity model. We adopt an optimization point of view and show how to take advantage of the availability of a hierarchy of models in a consistent manner. We do not use the low fidelity model just for the computation of promising starting points but also for the computation of search directions. We establish the convergence and convergence rate of the proposed algorithm. Our numerical experiments on large scale image restoration problems and the transition path problem suggest that, for certain classes of problems, the proposed algorithm is significantly faster than the state of the art.

Keywords Composite Optimization · Multigrid · Nonsmooth Optimization

1 Introduction

It is often possible to exploit the structure of large-scale optimization models in order to develop algorithms with lower computational complexity. We consider the case when the fidelity by which the optimization model captures the underlying application can be controlled. Typical examples include the discretization of Partial Differential Equations in computer vision and optimal control [5], the number of features in machine learning applications [30], the number of states in a Markov Decision Processes [27], and nonlinear inverse problems [25]. Indeed anytime a
finite dimensional optimization model arises from an infinite dimensional model it is straightforward to define such a hierarchy of optimization models. In many areas, it is common to take advantage of this structure by solving a low fidelity (coarse) model and then use the solution as a starting point in the high fidelity (fine) model. In this paper we adopt an optimization point of view and show how to take advantage of the availability of a hierarchy of models in a consistent manner. We do not use the coarse model just for the computation of promising starting points but also for the computation of search directions. We consider optimization models that consist of the sum of a smooth but not necessarily convex function and a non-smooth convex function. This class of problems is referred to as composite optimization models.

The algorithm we propose is similar to the Proximal Gradient Method (PGM). There is a substantial amount of literature related to proximal algorithms and we refer the reader to [26] for a review of recent developments. The main difference between PGM and the algorithm we propose is that we use both gradient information and a coarse model in order to compute a search direction. This modification of PGM for the computation of the search direction is akin to multigrid algorithms developed recently by a number of authors. There exists a considerable number of papers exploring the idea of using multigrid methods in optimization [5]. However, the large majority of these are concerned with solving the linear system of equations to compute a search direction using linear multigrid methods (both geometric and algebraic). A different approach and the one we adopt in this paper is the class of multigrid algorithms proposed in [20] and further developed in [18]. The framework proposed in [20] was used for the design of a first order unconstrained line search algorithm in [32], and a trust region algorithm in [11]. The trust region framework was extended to deal with box constraints in [10]. The general equality constrained case was discussed in [21], but no convergence proof was given. Numerical experiments with multigrid are encouraging and a number of numerical studies have appeared so far, see e.g. [9,22]. The algorithm we develop combines elements from PGM (gradient proximal steps) and the multigrid framework (coarse correction steps) developed in [20] and [32]. We call the proposed algorithm Multilevel Proximal Gradient Method (MPGM).

The literature in multilevel optimization is largely concerned with models where the underlying dynamics are governed by differential equations and convergence proofs exist only for the smooth case and with simple box or equality constraints. When the functions involved in the optimization problem are convex, the PGM method is identical to the so-called, Iterative Shrinkage Thresholding Algorithm (ISTA). In this case, it is possible to accelerate the algorithm to achieve an improved rate of convergence. We refer the interested reader to [1] for the accelerated version of ISTA, and to our recent work [14] for accelerated multilevel methods for composite convex optimization models. The main contribution of this work is the extension of the multilevel framework to composite optimization problems when the smooth function is not convex. In addition, our algorithm also allows some amount of non-smoothness in the objective function. For example, our algorithm allows the use of indicator functions to model certain types of constraints. Theoretically, the algorithm is valid for any convex constraint but the algorithm is computationally feasible when the proximal projection step can be performed in closed form or when it has a low computational cost. Fortunately, many problems in machine learning, computer vision, statistics and computational chemistry
do satisfy our assumptions. Apart from the work in [10] that addresses box constraints, the general constrained case has not been addressed before. Existing approaches assume that the objective function is twice continuously differentiable, while the proximal framework we develop in this paper allows for a large class of non-smooth optimization models. In addition, our convergence proof is different from the one given in [20] and [4] in that we do not assume that the algorithm used in the finest level performs one iteration after every coarse correction step. Our proof is based on analyzing the whole sequence generated by the algorithm and does not rely on asymptotic results as in previous works [11,32].

Outline The rest of the paper is structured as follows: in the next section we introduce our notation and assumptions. We also discuss the role of quadratic approximations in composite optimization models and describe our algorithm. The convergence of the algorithm is established in Section 3 and we report numerical results in Section 4.

2 Problem Formulation and Algorithm Description

The main difference between the proposed algorithm, Multilevel Proximal Gradient Method (MPGM), and existing algorithms such as the Proximal Gradient Method (PGM) is that we do not use a quadratic approximation for all iterations. Instead, we use a coarse model approximation for some iterations. In this section, we first describe the optimization problem under investigation and introduce our notation and assumptions. We then briefly review the use of quadratic approximations in the conventional PGM. We then specify how information can be transferred from the low dimensional model (coarse model) to the fine model and vice versa. Finally, we end this section by describing the proposed algorithm.

2.1 Optimization Model and Assumptions

We will assume that the optimization model can be formulated using two levels of fidelity, a fine model, and a coarse model. It is important to stress that the proposed algorithm aims to find the optimal solution of the (original) fine model. The coarse model is only used as an auxiliary model to construct a direction of search. The coarse model plays a similar role that the quadratic approximation model plays in the PGM.

We use $h$ and $H$ to indicate whether a particular quantity/property is related to the fine and coarse model respectively. It is easy to generalize the algorithm to more levels but with only two levels the notation is simpler. The fine model is assumed to have the following form,

$$
\min_{x_h \in \mathbb{R}^h} \left\{ F_h(x) \triangleq f_h(x_h) + g_h(x_h) \right\},
$$

We make the following assumptions regarding the fine model above.

**Assumption 1** The function $f_h : \mathbb{R}^h \to \mathbb{R}$ is a smooth (not necessarily convex) function with a Lipschitz continuous gradient. The Lipschitz constant will be denoted by $L_h$. 

Assumption 2. The function \( g_h : \mathbb{R}^h \to \mathbb{R} \) is a continuous convex function that is possibly non-smooth but admits a smooth approximation (in the sense of Definition 2.1 in \[2\], see also Definition 1 below).

Assumption 3. The set \( X_h^* = \arg \min \ F_h(x) \) is not empty.

All the assumptions above are standard except the part of Assumption 2 stating that \( g_h \) admits a smooth approximation. This later statement requires some further clarification. Firstly the definition of a smoothable convex function is given below. We refer the interested reader to \[2\] for further properties regarding smoothable convex functions.

Definition 1 (Smoothable convex function) \[2\] The function \( g_h : \mathbb{R}^h \to \mathbb{R} \) is called \((\alpha, \beta, K)\)-smoothable if there exist \( \beta_1, \beta_2 \) satisfying \( \beta_1 + \beta_2 = \beta > 0 \) such that for every \( \mu > 0 \) there exists a continuously differentiable convex function \( g^\mu_h : \mathbb{R}^h \to \mathbb{R} \) such that,

(i) \( g_h(x_h) - \beta_1 \mu \leq g^\mu_h(x_h) \leq g_h(x_h) + \beta_2 \mu, \forall x_h \in \mathbb{R}^h \).

(ii) The function \( g^\mu_h \) has a Lipschitz continuous gradient over \( \mathbb{R}^h \) and a Lipschitz constant that is less than or equal to \( K + \alpha/\mu \).

The assumption above is not particularly restrictive in terms of models PGM or MPGA is typically applied to. Any one of the techniques reviewed in \[2\] can be used to construct a smooth approximation. For future reference, we define the smooth approximation of \( F_h \) as follows,

\[
F^\mu_h(x) \triangleq f_h(x_h) + g^\mu_h(x_h)
\]  

The incumbent solution at iteration \( k \) in resolution \( h \) is denoted by \( x_{h,k} \). We use \( f_{h,k} \) and \( \nabla f_{h,k} \) to denote \( f_h(x_{h,k}) \) and \( \nabla f_h(x_{h,k}) \) respectively. Unless otherwise specified we use \( \| \cdot \| \) to denote \( \| \cdot \|_2 \).

2.2 Quadratic Approximations and the Proximal Gradient Method

In order to update the current solution at iteration \( k \), PGM employs a quadratic approximation of the smooth component of the objective function, and then solves the proximal subproblem,

\[
x_{h,k+1} = \arg \min_y \ f_{h,k} + \langle \nabla f_{h,k}, y - x_{h,k} \rangle + \frac{L_h}{2} \|x_{h,k} - y\|^2 + g(y).
\]  

Note that the above can be rewritten as follows,

\[
x_{h,k+1} = \arg \min_y \frac{L_h}{2} \left\| y - \left( x_{h,k} - \frac{1}{L_h} \nabla f_{h,k} \right) \right\|^2 + g(y).
\]

When the Lipschitz constant is known, the PGM keeps updating the solution vector by solving the optimization problem above. For later use, we define the generalized proximal operator as follows,

\[
\text{prox}_h(x) = \arg \min_y \frac{1}{2} \| y - x \|^2 + g(y).
\]
The gradient mapping at the point $D_{h,k}$ is defined as follows,

$$D_h(L_k, x_{h,k}) \triangleq L_k \left[ x_{h,k} - \text{prox}_h(x_{h,k} - \frac{1}{L_k} \nabla f_{h,k}) \right].$$

Using our notation, the PGM updates the current solution as follows,

$$x_{h,k+1} = x_{h,k} - \frac{1}{L_k} D_h(L_k, x_{h,k}).$$

Note that if $g(y)$ is not present then the PGM reduces to the classical gradient method. If $g(y)$ is an indicator function then the PGM reduces to the projected gradient method. The direction of search ($D_{h,k}$) is obtained by solving an optimization model either in closed form or using an iterative method. It is shown in Section 3 of [8] (see Corollary 1) that the PGM algorithm described in this Section converges in the sense that,

$$\min_{k=1,\ldots,N} \|D_h(L_k, x_{h,k})\| \leq A/N$$

For some constant $A$ that depends only on the Lipschitz constant of $f_h$, and the initial point $x_{h,0}$. We note that taking the minimum of the norm of the optimality conditions is by far the most common convergence criterion for first order projected gradients methods (see also Theorem 3 in [24]).

Since the non-convex function $f_h$ is linearized and $g_h$ is convex, it follows that the proximal sub-problem is convex. This convex approximation is solved in every iteration in order to obtain the next iterate. The algorithm we propose in this paper constructs the search direction using a different optimization model that is not necessarily convex but has a reduced number of dimensions. Before we introduce the coarse model, we need first to address the issue of how to transfer information between different levels.

2.3 Information transfer between levels

Multilevel algorithms require information to be transferred between levels. In the proposed algorithm we need to transfer information concerning the incumbent solution, proximal projection and gradient around the current point. At the fine level, the design vector $x_h$ is a vector in $\mathbb{R}^h$. At the coarse level the design vector is a vector in $\mathbb{R}^H$ and $H < h$. At iteration $k$, the proposed algorithm projects the current solution $x_{h,k}$ from the fine level to coarse level to obtain an initial point for the coarse model denoted by $x_{H,0}$. This is achieved using a suitably designed matrix ($I^H_h$) as follows,

$$x_{H,0} = I^H_h x_{h,k}.$$ 

The matrix $I^H_h \in \mathbb{R}^{H \times h}$ is called a restriction operator and its purpose is to transfer information from the fine to the coarse model. There are many ways to define this operator and we will discuss some possibilities in Section 3. This is a standard technique in multigrid methods both for solutions of linear and nonlinear equations and for optimization algorithms [7,20]. In addition to the restriction operator, we also need to transfer information from the coarse model to the fine model. This is done using the prolongation operator $P^H_h \in \mathbb{R}^{h \times H}$. The standard assumption in multigrid literature [7] is to assume that $I^H_h = c(I^H_h)^\top$, where $c$ is some positive scalar.
2.4 Coarse Model Construction

The construction of the coarse models in multilevel algorithms is a subtle process. It is this process that sets apart rigorous multilevel algorithms with performance guarantees from other approaches (e.g., kriging methods) used in the engineering literature. A key property of the coarse model is that locally (i.e., at the initial point of the coarse model, $x_{H,0}$) the optimality conditions of the two models match in a certain sense. In the unconstrained case, this is achieved by adding a linear term in the objective function of the coarse model [11, 20, 32]. In the constrained case the linear term is used to match the gradient of the Lagrangian [20]. However, the theory for the constrained case of multilevel algorithms is less developed, and the non-smooth case has received even less attention.

For non-smooth optimization problems, we propose a coarse model that has the following form,

$$F_H(x_H) = f_H(x_H) + g_H(x_H) + \langle v_H, x_H \rangle.$$  \hfill (3)

We make similar assumptions for the coarse model as we did for the fine model.

**Assumption 4** The functions $f_H : \mathbb{R}^H \to \mathbb{R}$, and $g_H : \mathbb{R}^H \to \mathbb{R}$ are smooth functions with a Lipschitz continuous gradient. In addition the function $g_H$ is convex. The Lipschitz constant for $f_H + g_H$ will be denoted by $L_H$.

**Assumption 5** The set $X_H^* = \arg\min_{x_H \in \mathbb{R}^H} F_H(x)$ is not empty.

As is standard in the multigrid literature we presume that when $f_h$ and $g_h$ are given then the construction of $f_H$ and $g_H$ is possible. Finally, the third term in (3) will be used to ensure that the gradient of the smooth model is consistent with the gradient of the smooth fine model (1). The $v_H$ term changes every time the coarse model is used and it is given by,

$$v_H = I_h^H \nabla F_{h,k} - (\nabla f_{H,0} + \nabla g_{H,0}).$$  \hfill (4)

It is easy to see that with the definition of $v$ given above, the following first order coherency condition holds,

$$I_h^H \nabla F_{h,k} = \nabla F_{H,0}.$$  

To establish convergence of the algorithm the main assumptions on the coarse model Assumption 4 and 5 above together with definition 4 are enough. In particular, this means that the functions $f_H$ and $g_H$ may not be just reduced order versions of $f_h$ and $g_h$. As an example, one can choose $f_H$ to be a reduced order quadratic approximation of $f_h$. In this case, the direction obtained from the coarse model will contain information from the Hessian of $f_h$ without having to solve a large system of equations in $\mathbb{R}^h$. This approximation is called the Galerkin approximation in the multigrid literature and has been shown to be effective in optimization in [9]. Of course, for the coarse model to be useful in practice it is important to exploit the hierarchical structure of the underlying application. Many applications that include discretized partial differential equations, images processing applications or any application that includes an approximation of an infinite dimensional object are good candidates for the application of the proposed
algorithm. Obviously, the choice of the lower dimensional model will have a great bearing on the success of the algorithm. If the underlying model has no hierarchical structure then a multilevel method may not be appropriate as at best it may provide a good starting point.

2.5 Multilevel Proximal Gradient Method (MPGM)

Rather than computing a search direction using a quadratic approximation, we propose to construct an approximation with favorable computational characteristics for at least some iterations. In the context of optimization, favorable computational characteristics means reducing the dimensions of the problem and increasing its smoothness. This approach facilitates the use of non-linear approximations around the current point. The motivation behind this class of approximations is that the global nature of the approximation would reflect global properties of the model that would (hopefully) yield better search directions.

At iteration $k$ the algorithm can compute a search direction using one of two approximation techniques. The first possibility is to construct a quadratic approximation for $f_h$ around $x_{h,k}$ in order to obtain $x_{h,k+1}$ from the solution of the proximal subproblem in (2). We then say that the algorithm performed a gradient correction iteration. The classical PGM only performs gradient correction iterations. The second possibility for the proposed algorithm is to use the coarse model in (3) in lieu of a quadratic approximation. When this type of step is performed we say that the algorithm performs a coarse correction iteration. The conditions that need to be satisfied for the algorithm to perform a coarse iteration are given below.

**Condition 1** A coarse correction iteration is performed when both conditions below are satisfied,

$$
||I_H D_{h,k}|| > \kappa ||D_{h,k}|| \\
||x_{h,k} - \tilde{x}_h|| > \eta ||\tilde{x}_h||
$$

where $\tilde{x}_h$ is the last point to trigger a coarse correction iteration, and $\kappa, \eta \in (0,1)$.

The conditions where based on our own numerical experiments and the results in [11,20,32]. The first condition in (5) prevents the algorithm from performing coarse iterations when the first order optimality conditions are almost satisfied. If the current fine level iterate is close to being optimal the coarse model constructs a correction term that is nearly zero. Typically, $\kappa$ is the tolerance on the norm of the first-order optimality condition of (the fine) level $h$ or alternatively $\kappa \in (0, \min(1, \min ||I_H||))$. The second condition in (5) prevents a coarse correction iteration when the current point is very close to $\tilde{x}_h$. The motivation is that performing a coarse correction at a point $x_{h,k}$ that satisfies both the above conditions will yield a new point close to the current $x_{h,k}$.

Suppose for now that the algorithm decides to perform a coarse correction iteration at the current point $x_{h,k}$. Then the coarse model in (3) is constructed using the initial point $x_{H,0} = I_{h,k} x_{h,k}$. Note that the linear term in (4) changes every time the algorithm deploys the coarse model approximation. The algorithm then performs $m > 0$ iterations of the PGM algorithm on the coarse model (3).
Note that for the convergence of the algorithm it is not necessary to use PGM on the coarse model. A faster algorithm can be used, but for the sake of consistency we perform the analysis with PGM. After \( m \) iteration we obtain the coarse correction term,

\[
e_{H,m} = x_{H,0} - x_{H,m}.
\]

After the coarse correction term is computed, it is projected to the fine level using the prolongation operator and it is used as a search direction. We denote the direction of search obtained from the coarse model as \( d_{h,k} \) and is defined as follows,

\[
d_{h,k} \triangleq \pi^h_{H} e_{H,m}.
\]

The current solution at the fine level is updated as follows,

\[
x_{h,k+1} = \text{prox}\left(x_{h,k}^+ - \beta_k \nabla f_h(x_{h,k}^+)\right),
\]

where

\[
x_{h,k}^+ = x_{h,k} - \alpha_k d_{h,k}.
\]

The two step sizes \( \alpha_k \) and \( \beta_k \) are selected according to an Armijo step size strategy and a backtracking strategy respectively. Both of these step size strategies are standard and are specified in Algorithm 1 below.

The algorithm is fully specified in the caption below (see Algorithm 1), but a few remarks are in order. Firstly, if Condition 1 is never satisfied then the algorithm reduces to the proximal gradient method. Its convergence with the specific step size strategy used above will be established in the next section. Of course, Condition 1 will be satisfied for all cases of practical interest. In this case, the main difference between the proposed algorithm and the proximal gradient algorithm is the computation of the search direction \( d_{h,k} \) which is used to compute an intermediate point \( x_{h,k}^+ \). To compute the coarse correction term \( d_{h,k} \), the proposed algorithm performs \( m \) iterations of a first order algorithm. In fact, the only difference between the steps performed at the coarse level and the steepest descent algorithm applied to (3), is the step size strategy in (6). The right-hand side of (6) is nothing but the standard Armijo condition. The left hand side of (6) is the line search condition used in [32].

Due to the requirement to perform \( m \) iterations of a first order algorithm in the coarse model, the computational cost of the coarse correction step in (5) can be more expensive than the gradient correction step in (10). From a theoretical point of view, if we assume that \( m = 1 \) and that the Lipschitz constants are known then the computational cost of PGM and MPGM are approximately the same. For example, if we assume that each application of the gradient mapping operator costs \( \psi(h) \) for a model in \( h \) dimensions, then the cost of MPGM is \( m\psi(H) + \psi(h) \). If \( m = 1 \), then since the computational cost of a single application of the gradient mapping operator is typically very low the cost per iteration of the two algorithms is about the same. Another point that needs to be discussed is the number of parameters that need to be specified for MPGM. Once the coarse model is specified, the only other requirement is to specify the smooth line search parameters (see Line [3]). These parameters are the ones that appear in standard Armijo line search procedures and are in general not too difficult to specify.
Algorithm 1: Multilevel Proximal Gradient Method (MPGM)

1. **Input:**
   - Initial point: $x_{h,0}$
   - Line search parameters: $0 < \kappa_1 < \frac{1}{2}, 1 - \kappa_1 \leq \kappa_2 \leq 1$
   - Coarse smoothing parameters: $\mu > 0$ and $0 < \gamma < 1$
   - Initialization: Set iteration counters $k = 0, r = 0$

2. If Condition 1 is satisfied at $x_{h,k}$ then
   - Set $x_{H,0} = I_H^h x_{h,k}$, and use the coarse model (3) with the following linear term,
     \[ v_H = I_H^h \nabla F_{h,k}^\mu - (\nabla f_{H,0} + \nabla g_{H,0}). \]
     where $\mu_k = \mu \gamma^r$.
   - For $l = 0, \ldots, m$
     \[ x_{H,l+1} = x_{H,l} - \beta_{H,l} \nabla F_H(x_{H,l}) \]
   - Where $\beta_{H,l}$ is chosen so that,
     \[ F_{H,0} + \kappa_2 (\nabla F_H(x_{H,0}), x_{H,l+1} - x_{H,0}) < F_H(x_{H,l+1}) \leq F_{H,l} - \kappa_1 \beta_{H,l} \| \nabla F_{H,l} \|^2 \]
   - Set $d_{h,k} = I_H^h (x_{H,0} - x_{H,m}), r \leftarrow r + 1$ and compute:
     \[ x_{h,k+1} = x_{h,k} - \alpha_{h,k} d_{h,k} \]
   - Where $\alpha_{h,k}$ is chosen such that,
     \[ F_h(x_{h,k}^+ \leq F_h - \alpha_{h,k} (\nabla F_{h,k}^\mu, d_{h,k}) \]
   - Update current solution,
     \[ x_{h,k+1} = x_{h,k}^+ - \frac{1}{L_k} D_h(L_k, x_{h,k}^+) \]
   - Where $L_k$ is chosen such that,
     \[ f(x_{h,k+1}) \geq f(x) + (\nabla f(x_{h,k}), x_{h,k+1} - x_{h,k}) + \frac{L_k}{2} \| x_{h,k+1} - x_{h,k} \|^2. \]
   - Set $k \leftarrow k + 1$ and go to Step 6.

3. Else
   - $x_{h,k+1} = x_{h,k} - D_h(L_k, x_{h,k})$
   - Where $L_k$ is chosen to satisfy the condition in (9).
   - Set $k \leftarrow k + 1$ and go to Step 6.

3 Global convergence rate analysis

In this section we establish the convergence of the algorithm under different assumptions. The first result (Theorem 1) covers the general case when $f_h$ has a Lipschitz continuous gradient (but is not necessarily convex), and $g_h$ is convex but is not necessarily smooth. In Theorem 1 the coarse model is not assumed to be convex. As in the single level case (see e.g. Theorem 3 in [23]) convergence is
in terms of the following optimality criterion,

\[ R_N = \min_{0 \leq k \leq N} \| D(x_{h,k}, L_{h,k}) \|^2. \]

We remind the reader that the parameter \( \mu_k \) appearing in the Theorem below is the smoothing parameter used to specify the smooth fine model in (1).

**Theorem 1**  (a) Suppose that Assumptions 1-5 hold, and that \( L_{h,k} \geq L_h/2 \). Then the iterates of Algorithm 1 satisfy,

\[ R_N \leq 2L_h N \left( F_{h,0} - F^*_h + \beta \sum_{k=0}^{N} \mu_k \right) \]

(11)

where \( \mu_k \) is a positive scalar if Condition 1 was satisfied at iteration \( k \) and zero otherwise.

(b) Suppose that \( \mu_k \leq \mu \gamma^k \), for some \( \mu > 0 \) and \( 0 < \gamma < 1 \), then

\[ R_N \leq 2L_h N \left( F_{h,0} - F^*_h + \beta \mu \frac{1 - \gamma^{N+1}}{1 - \gamma} \right) \]

Proof (a) Note that if a coarse correction is performed then according to (8), \( x_{h,k+1} \) is the solution of the following optimization problem,

\[ x_{h,k+1} = \arg \min_y \langle \nabla f_h(x_{h,k}^+), y - x_{h,k}^+ \rangle + \frac{L_{h,k}}{2} \| y - x_{h,k}^+ \|^2 + g_h(y). \]

It follows from the optimality condition of the problem above,

\[ \langle \nabla f_h(x_{h,k}^+), x_{h,k+1} - x_{h,k}^+ \rangle \leq \langle g_h(x_{h,k+1}), x_{h,k+1} - x_{h,k}^+ \rangle \geq 0, \]

where \( g_h(x_{h,k+1}) \in \partial g(x_{h,k+1}) \). Rearranging the inequality above and specializing it to \( y = x_{h,k}^+ \) we obtain,

\[ \langle \nabla f_h(x_{h,k}^+), x_{h,k+1} - x_{h,k}^+ \rangle \leq \langle g_h(x_{h,k+1}), x_{h,k}^+ - x_{h,k+1} \rangle - L_{h,k} \| x_{h,k}^+ - x_{h,k+1} \|^2 \]

\[ \leq g(x_{h,k}^+) - g(x_{h,k+1}) - L_{h,k} \| x_{h,k}^+ - x_{h,k+1} \|^2. \]

(12)

where the second inequality above follows from the subgradient inequality on \( g \),

\[ g(x_{h,k+1}^+) \geq g(x_{h,k+1}) + \langle g_h(x_{h,k+1}), x_{h,k+1}^+ - x_{h,k+1} \rangle. \]

Since the gradient of \( f_h \) is Lipschitz continuous the descent Lemma [3] yields the following inequality,

\[ f_h(x_{h,k+1}) \leq f_h(x_{h,k}^+) + \langle \nabla f_h(x_{h,k}^+), x_{h,k+1} - x_{h,k}^+ \rangle + \frac{L_h}{2} \| x_{h,k}^+ - x_{h,k+1} \|^2. \]

Using the inequality above in (12) we obtain,

\[ F_h(x_{h,k+1}) \leq F_h(x_{h,k}^+) + \left( \frac{L_h}{2} - L_{h,k} \right) \| x_{h,k}^+ - x_{h,k+1} \|^2 \]

(13)
It follows from Lemma 2.7 in [32], that there exists a $\beta_{H,l}$, $l = 0, \ldots, m$ such that the two conditions in (9) are simultaneously satisfied, i.e. after $m$ coarse iterations in the coarse model we have,

$$F_H(x_{H,m}) > F_{H,0} + \kappa_2 \langle \nabla F_{H,0}, x_{H,m} - x_{H,0} \rangle = F_{H,0} - \kappa_2 \langle \nabla F_{H,k}^\mu, d_{h,k} \rangle$$

The rhs of (6) implies that $F_{H,m} - F_{H,0} < 0$ and therefore, $d_{h,k}$ is a descent direction for $F_{h}^\mu$ at $x_{h,k}$. It follows from standard arguments (see [3]) that there exists a stepsize $\alpha_{h,k}$ that satisfies (7) such that,

$$F_h(x_{h,k+1}) \leq F_h(x_{h,k}) + \left( \frac{L_h}{2} - L_{h,k} \right) \| x_{h,k}^+ - x_{h,k+1} \|^2 + \beta \mu_k$$  (14)

Summing up all the inequalities in (14) we obtain,

$$F_h(x_{h,N+1}) \leq F_h(x_{h,0}) + \sum_{k=0}^{N} \left( \frac{L_h}{2} - L_{h,k} \right) \| x_{h,k}^+ - x_{h,k+1} \|^2 + \beta \sum_{k=0}^{N} \mu_k$$

Using our assumption that $L_{h,k} \geq L_h/2$ and the definition of $R_N$ we obtain,

$$R_N \frac{N}{2L_h} \leq F_h(x_{h,0}) - F_h(x_h^*) + \beta \sum_{k=0}^{N} \mu_k.$$  

as required.

(b) This follows from the fact that the last term in (11) is a geometric series.

Note that if the algorithm does not perform any coarse correction iterations then the rate becomes,

$$R_N \leq \frac{2L_h}{N}.$$  

The rate above is the same rate obtained in Theorem 3 of [23] for the non-convex case of the gradient algorithm with a constant step size strategy. In the multilevel setting, we get the same rate but with a slightly worse constant. The augmented constant is due to the smoothing approximation. As shown in part (b) of the Theorem above, when $\mu_k = \mu_{\gamma}^k$ this constant is small and known in advance. In addition this constant is controllable by the user, and in most applications of interest it is negligible.

In order to simplify the derivations above we assumed that $L_{h,k} \geq L_h/2$. Of course in practice we may not know the Lipschitz constant and this assumption may seem too restrictive. However, this can be easily addressed by using a backtracking strategy. To be precise, we start with $L_0$ and set $L_j = \eta^j L_{j-1}$ until [9].
is satisfied. It follows from the descent Lemma that such a constant exists. The convergence argument remains the same with $\eta L_h$ replacing $L_h$.

We end this section by establishing the convergence rate for the strongly convex case. In particular we change Assumptions 1 and 4 as follows.

**Assumption 6** The function $f_h : \mathbb{R}^h \to \mathbb{R}$ is a smooth strongly convex function with a Lipschitz continuous gradient. The Lipschitz constant will be denoted by $L_h$ and the strong convexity constant by $\gamma_f$.

**Assumption 7** The functions $f_H : \mathbb{R}^H \to \mathbb{R}$, and $g_H : \mathbb{R}^H \to \mathbb{R}$ are smooth functions with a Lipschitz continuous gradient. In addition the function $g_H$ is convex and $f_H$ is strongly convex. The Lipschitz constant for $f_H + g_H$ will be denoted by $L_H$, and the strong convexity constant is given by $\gamma_F$.

**Theorem 2** Suppose that the same assumptions as in Theorem 1 hold, except Assumption 1 and 7 as follows.

(a) If $\gamma_f/2L_h \leq 1$ and $\mu_i \leq \mu \gamma^i$ is chosen such with $0 < \gamma \leq 1/4$, then,

$$F_h(x_{h,k+1}) - F_h(x^*) \leq \left( \frac{1}{2} \right)^k \left( F_h(x_{h,0}) - F_h(x^*) + 2\beta \mu \left( 1 - \frac{1}{2\lambda} \right) \right).$$

(b) If $\gamma_f/2L_h \geq 1$ and $\mu_i \leq \mu \gamma^i$ is chosen such with $0 < \gamma < \delta$, then,

$$F_h(x_{h,k+1}) - F_h(x^*) \leq \left( 1 - \frac{\gamma_f}{4L_h} \right)^k \left( F_h(x_{h,0}) - F_h(x^*) + \beta \mu \left( 1 - \frac{\delta}{1 - \delta} \right) \right),$$

where $\delta \triangleq (4\gamma_f L_h)/(4L_h - \gamma_f)$.

**Proof** (a) Suppose that at iteration $k$ a coarse correction direction is computed using Steps 7 and 8. Using the descent lemma, and the fact that $x_{h,k+1}$ solves the quadratic subproblem around $x_{h,k}$, we obtain the following,

$$F_h(x_{h,k+1}) \leq f(x_k^h) + \langle \nabla f(x_k^h), x_{h,k+1} - x_{h,k}^+ \rangle + \frac{L_h}{2} \| x_{h,k+1} - x_{h,k}^+ \|^2 + g_h(x_{h,k+1})$$

$$= \min_z f(x_k^h) + \langle \nabla f(x_k^h), z - x_{h,k}^+ \rangle + \frac{L_h}{2} \| z - x_{h,k}^+ \|^2 + g_h(z)$$

$$\leq \min_z f_h(z) + g_h(z) + \frac{L_h}{2} \| z - x_{h,k}^+ \|^2$$

Let $z = \lambda x^* + (1 - \lambda)x_{h,k}^+$, for $0 \leq \lambda \leq 1$, and use the fact that $F_h(x) = f_h(x) + g_h(x)$ is strongly convex to obtain the following estimate,

$$F_h(x_{h,k+1}) \leq \min_{0 \leq \lambda \leq 1} \lambda F_h(x^*) + (1 - \lambda)F_h(x_k^h) + \frac{L_h}{2} \| \lambda(x^* - x_{h,k}^+) \|^2$$

$$\leq \min_{0 \leq \lambda \leq 1} F(x_k^h) - \lambda \left( 1 - \frac{\lambda L_h}{\gamma_f} \right) \left( F_h(x_k^h) - F_h(x^*) \right)$$

(15)

Where to obtain the last inequality we used the fact that for a strongly convex function, the following holds,

$$F_h(x_k^h) - F(x^*) \geq \frac{\gamma_f}{2} \| x_k^+ - x^* \|^2.$$
The minimum in (15) is obtained for either $\lambda = 1$ or $\lambda = \gamma f / 2L_h$. If $\lambda = 1$ (i.e. $\gamma f / 2L_h \leq 1$) then,

$$F_h(x_{h,k+1}) - F_h(x^*) \leq \frac{L_f}{\gamma f} \left( F_h(x_{h,k}) - F_h(x^*) \right)$$

$$\leq \frac{1}{2} \left( F_h(x_{h,k}^+) - F_h(x^*) \right)$$

(16)

Using the same arguments used in Theorem 1, we obtain

$$F(x_{h,k+1}) \leq F(x_{h,k}) + \beta \mu_k,$$

(17)

Using (17) in (16) we obtain,

$$F_h(x_{h,k+1}) - F_h(x^*) \leq \frac{1}{2} \left( F_h(x_{h,k}) - F_h(x^*) \right) + \frac{1}{2} \beta \mu_k.$$

If a coarse correction term was not done then we use the convention that $\mu_k$ is zero and that $x_{h,k} = x_{h,k}^+$. Therefore after $k$ iterations and with $r$ coarse correction steps, we obtain

$$F_h(x_{h,k+1}) - F_h(x^*) \leq \left( 1 - \frac{\gamma f}{4L_h} \right)^k \left( F_h(x_{h,0}) - F_h(x^*) \right) + \beta \mu \sum_{i=0}^{k} \left( 1 - \frac{\gamma f}{4L_h} \right)^{k-i} \mu_i$$

$$\leq \left( 1 - \frac{\gamma f}{4L_h} \right)^k \left( F_h(x_{h,0}) - F_h(x^*) \right) + \beta \mu \sum_{i=0}^{k} \left( 4L_h \gamma \right) \left( 1 - \frac{\gamma f}{4L_h} \right)^{i-1}$$

$$\leq \left( 1 - \frac{\gamma f}{4L_h} \right)^k \left( F_h(x_{h,0}) - F_h(x^*) \right) + \beta \mu \frac{1 - \delta^k}{1 - \delta},$$

as required.

4 Numerical experiments

In this section we illustrate the numerical performance of the algorithm on both convex and non-convex problems. For the convex problem we chose the image restoration problem and for the non-convex case we chose the problem of finding transition paths in energy landscapes. Both problems have been used as benchmark problems and are important applications in image processing and computational chemistry respectively. The two problem classes are quite different. In particular, the image restoration problem is convex and unconstrained but has more than a million variables, whereas the saddle point problem is non-convex, constrained but is lower dimensional. We compare the performance of the proposed MPGM against the state of the art for image restoration problems (FISTA) and against the Newton algorithm for the transition path problem.
4.1 The Image Restoration Problem

The image restoration problem consists of the following composite convex optimization model,

$$\min_{x_h \in \mathbb{R}^h} \| A_h x_h - b_h \|^2_2 + \mu_h \| W(x_h) \|_1,$$

where $b_h$ is the vectorized version of the input image, $A_h$ is the blurring operator based on the point spread function (PSF) and reflexive boundary conditions, and $W(x_h)$ is the wavelet transform of the image. The two dimensional version of the input image and the restored image are denoted by $X_h$ and $B_h$ respectively. The first term in the objective function aims to find an image that is as close to the original image as possible, and the second term enforces a relationship between the pixels and ensures that the recovered image is neither blurred nor noisy. The regularization parameter $\lambda_h$ is used to balance the two objectives. In our implementation of the fine model we used $\lambda_h = 10e^{-4}$. Note that the first term is convex and differentiable, the second term is also convex but non-smooth. The blurring operator $A_h$, is computed by utilizing an efficient implementation provided in the HNO package [12]. In particular, we rewrite the expensive matrix computation $A_h x_h - b_h$ in the reduced form,

$$A_h^c X_h (A_h^r)^\top - B_h,$$

where $A_h^c, A_h^r$ are the row/column blurring operators and $A_h = A_h^c \otimes A_h^r$. We illustrate the problem of image restoration using the widely used cameraman image. Figure 1(a) is the corrupted image, and the restored image is shown in Figure 1(b). The restored image was computed with MPGM. The image restoration problem fits exactly the framework of convex composite optimization. In addition it is easy to define a hierarchy of models by varying the resolution of the image. We discuss the issue of coarse model construction next.

![Corrupted Image](image1.png) ![Restored Image](image2.png)

Fig. 1 (a) Corrupted cameraman image used as the input vector $b_h$, (b) Restored image.
4.1.1 The Coarse Model in the Image Restoration Problem

We described MGPM as a two level algorithm but it is easy to generalize it to many levels. In our computations we used the fine model described above and two coarse models, one with resolution $512 \times 512$ and its coarse version i.e. a model with $256 \times 256$. Each model on the hierarchy has a quarter of the variables of the model above it. We used the smoothing approach to construct the coarse models (see Section 2.4). Following the smoothing approach we used the following objective function,

$$\min_{x_H \in \Omega} \| A_H x_H - b_H \|_2^2 + \langle v_H, x_H \rangle + \lambda_H \sum_{i \in \Omega} \sqrt{W(x_H)_i^2 + \mu^2} - \mu$$

where $\mu = 0.2$ is the smoothing parameter, $v_H$ was defined in (4), and $\lambda_H$ is the regularizing parameter for the coarse model. Since the coarse model has less dimensions, the coarse problem is smoother, therefore the regularizing parameter should be reduced, we used $\lambda_H = \lambda_h/2$. The information transfer between levels is done via a simple linear interpolation technique to group four fine pixels into one coarse pixel. This is a standard way to construct the restriction and prolongation operators and we refer the reader to [7] for the details. The input image, and the current iterate are restricted to the coarse scale as follows,

$$x_{H,0} = I_{h}^H x_{h,k}, \quad b_H = I_{h}^H b_h.$$  

The standard matrix restriction $A_H = I_{h}^H A_h (I_{h}^H)^\top$ is not performed explicitly as we never need to store the large matrix $A_h$. Instead, only column and row operators $A_h^c, A_h^r$ are stored in memory. As a decomposition of the restriction operator is available for our problem, in particular $I_{h}^H = R_1 \otimes R_2$, we can obtain the coarse blurring matrix by,

$$A_H = A_{H}^c \otimes A_{H}^r$$

where $A_{H}^c = R_2 A_h R_2^\top$ and $A_{H}^r = R_1 A_h R_1^\top$.

The condition to use the coarse model in MGPM is specified in (5), and we used the parameters $\kappa = 0.5$ and $\eta = 1$ in our implementation.

4.1.2 Performance comparison

We compare the performance of our methods with FISTA and ISTA using a representative set of corrupted images (blurred with 0.5% additive noise). FISTA is considered to be a state of the art method for the image restoration problem [1], and has a theoretically better convergence rate than the proposed algorithm. ISTA has the same convergence rate as the proposed algorithm. We compare the CPU time required to achieve convergence of MPGM against ISTA and FISTA. We chose to report CPU times since the computational complexity of MPGM per iteration can be larger than ISTA or FISTA. We tested the algorithm on several images, and below we report results on a representative set of six images. All our test images have the same size, $1024 \times 1024$. At this resolution, the optimization model at the fine scale has more than $10^6$ variables (1048576, to be precise). We implemented the ISTA and FISTA algorithms with the same parameter settings.
Panos Parpas

Fig. 2. (a) Comparison of the three algorithms in terms of function value. MPGM clearly outperforms the other algorithms and converges in essentially 5 iterations, while others have not converged even after 100 iterations. CPU time required to find a solution within 2% of the optimum for the three algorithms. (b) Results for blurred images with 0.5% noise. (c) Results for blurred images with 1% noise. Higher levels of noise lead to more ill-conditioned problems. The figures in (b) and (c) compare CPU times and suggest that MPGM is on average ten times faster than ISTA and three/four times than FISTA.

For the fine model we used the standard backtracking line strategy for ISTA as in [1]. All algorithms were implemented in MATLAB and run on a standard desktop PC. Due to space limitations, we only report detailed convergence results from the widely used cameraman image. The images we used, the source code for MGPM, and further numerical experiments can be obtained from the web-page of the author [www.doc.ic.ac.uk/~pp500](http://www.doc.ic.ac.uk/~pp500).
In Figure 2(a) we compare the three algorithms in terms of the progress they make in function value reduction. In this case we see that MPGM clearly outperforms ISTA. This result is not surprising since MPGM is a more specialized algorithm with the same convergence properties. However, what is surprising is that MPGM still outperforms the theoretically superior FISTA. Clearly, MPGM outperforms FISTA in early iterations and is comparable in latter iterations.

Figure 2 gives some idea of the performance of the algorithm but of course what matters most is the CPU time required to compute a solution. This is because an iteration of MPGM requires many iterations in a coarse model, and therefore comparing the algorithms in terms of the number of iterations is not fair. In order to level the playing field, we compare the performance of the algorithms in terms of CPU time required to find a solution that satisfies the optimality conditions within 2%. Two experiments were performed on a set of six images. The first experiment takes as input a blurred image with 0.5% additive Gaussian noise and the second experiment uses 1% additive noise. We expect the problems with the 1% additive noise to be more difficult to solve than the one with 0.5% noise. This is because the corrupted image is more ill-conditioned. Figure 2(b) shows the performance of the three algorithms on blurred images with 0.5% noise. We can see that MPGM outperforms both ISTA and FISTA by some margin. On average MPGM is four times faster than FISTA and ten times faster than ISTA. In Figure 2(c), we see an even greater improvement of MPGM over ISTA/FISTA. This is expected since the problem is more ill-conditioned (with 1% noise as opposed to 0.5% noise in Figure 2(c)), and so the fine model requires more iterations to converge. Since ISTA/FISTA perform all their computation with the ill-conditioned model, CPU time increases as the amount of noise in the image increases. On the other hand, the convergence of MPGM depends less on how ill-conditioned the model is since one of the effects of averaging is to decrease ill conditioning.

4.2 The Transition Path Problem

The computational of transition paths is a fundamental problem in computational chemistry, material science and biology. Most methods for computing fall into broad classes. Chain of state methods and surface walking methods. For a recent review and a description of several methods in the latter category we refer the interested reader to [33]. We approach the problem using a chain-of-state method. In these methods we are given two initial points $x_a$ and $x_b$ that are stable points on a potential energy surface. Usually these two points are local minima. The transition path problem in this case is to find how the geometry and the energy changes from one stable state to the next.

The mathematical formulation of the transition path problem using an optimization problem was established in [19] through the mountain pass theorem. We will adopt the formulation from [19]. We will review the formulation of the optimization problem below, but for the precise properties of the model we refer the reader to [19]. The method (also known as the ‘elastic string algorithm’) is derived from the solution of the following infinite dimensional saddle point problem,

$$
\inf_{p \in P} \max_{t \in [0, 1]} f(p(t)),
$$

(18)
where $f$ is the non-convex potential energy function, and $\Gamma$ is the set of all paths connecting $x_a$ and $x_b$. We also impose the initial and final conditions $p(0) = x_a$ and $p(1) = x_b$. By only considering piecewise linear paths with $m$ breakpoints, the elastic string algorithm (described in detail in [19]) reduces the infinite dimensional problem in (18) to the following finite problem,

$$\min \{ v(x) ||x_{k+1} - x_k||_1 \leq h_k, \ 0 \leq k \leq m, \ x_0 = x_a, \ x_m = x_b \},$$

(19)

where $v(x) = \max(f(x_0), \ldots, f(x_m))$. In order to place (19) in the same format as the optimization models we consider in this paper we use entropic smoothing for the max function [28] and replace the constraints with an indicator function. The entropic smoothing function is given by,

$$\phi_\lambda(x) = \frac{1}{\lambda} \log \left( \sum_{k=0}^{m} \exp(\lambda f_k(x)) \right).$$

This is a frequently used function to smooth the max function appearing in min-max problems, for its properties see [28,31]. To smooth the indicator function we used the Moreau-Yosida regularization approach [13]. To be precise, let $g(x)$ denote the indicator function of the problem in (19), the Moreau-Yosida regularization of $g$ (also referred to as the Moreau envelope) is given by,

$$g_\mu(x) = \inf_{w} \left\{ g(w) + \frac{1}{2\mu} \|w - x\|^2 \right\},$$

Let $\hat{x}$ denote the unique minimizer of the problem above, then the gradient of $g_\mu(x)$ is Lipschitz continuous and is given by,

$$\nabla g_\mu(x) = \frac{1}{\mu} [x - \hat{x}].$$

We used the following smoothing parameters $\lambda = \mu = 0.01$. For the fine model we used $2^h$ breakpoints with $h = 10$. This means that for a potential energy function in $n$ dimensions the optimization problem has $n \times 2^{10}$ variables, and $2^{10}$ constraints. We solved the fine model using the same method as in [19], i.e. the Newton’s method with an Armijo line search. Before we compare the Newton method with the proposed algorithm we describe the construction of the coarse model.

4.2.1 The Coarse Model in the Transition Path Problem

As for the image restoration problem, the coarse model for the transition path problem is very easy to construct. We used right levels, with $2^2, 2^3, \ldots, 2^9$ breakpoints. For the prolongation and restriction operators we used the same simple linear interpolation technique as in the image restoration problem. Finally, we used $\kappa = 0.5$ and $\eta = 1$ for the coarse correction criteria (these are the same parameters we used in the image restoration problem).
Table 1 Test Problems in Numerical Experiments

<table>
<thead>
<tr>
<th>Problem #</th>
<th>Name</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Camel Function</td>
<td>[19]</td>
</tr>
<tr>
<td>2</td>
<td>LEPS potential</td>
<td>[19]</td>
</tr>
<tr>
<td>3</td>
<td>LEPS/Gaussian potential</td>
<td>[19]</td>
</tr>
<tr>
<td>4</td>
<td>2-d energy function</td>
<td>[33]</td>
</tr>
<tr>
<td>5</td>
<td>Minyaev–Quapp surface</td>
<td>[33]</td>
</tr>
<tr>
<td>6</td>
<td>Stingray function</td>
<td>[33]</td>
</tr>
<tr>
<td>7</td>
<td>Eckhardt surface</td>
<td>[33]</td>
</tr>
</tbody>
</table>

4.2.2 Performance comparison

We tested the performance of MPGM against the Newton algorithm on seven widely used benchmark problems. We used the three test problems from [19]. For these three potentials we used the same parameters and functional form described in [19], and for the interest of space we omit the details here. We also used four test problems from [33]. For later reference we tabulate the name of the problem and the appropriate reference in Table 1. As in the previous case study, comparing the performance of the proposed algorithm in terms of CPU time is the fairest metric. In Figure 4.2.2 we plot the results. It can be seen from this figure that on all but two problems the proposed algorithm is far superior than the Newton algorithm.

Fig. 3 Comparison of MPGM with the Newton algorithm in terms of CPU time.

5 Conclusions

We developed a multilevel proximal gradient method (MPGM) for composite optimization models. The key idea behind MPGM is, for some iterations, to replace
the quadratic approximation with a coarse approximation. The coarse model is used to compute search directions that are often superior to the search directions obtained using just gradient information. We showed how to construct coarse models in the case where the objective function is non-differentiable. For the convex case, our initial numerical experiments show that the proposed MISTA algorithm is on average ten times faster than ISTA, and three-four times faster (on average) than the theoretically superior FISTA algorithm. For the non-convex case, we tested the performance of the algorithm on a problem arising in computational chemistry, and we showed that the proposed method is significantly faster than the Newton algorithm.

The initial numerical results are promising but still the algorithm can be improved in a number of ways. For example, we only considered the most basic prolongation and restriction operators in approximating the coarse model. The literature on the construction of these operators is quite large and there exist more advanced operators that adapt to the problem data and current solution (e.g. bootstrap AMG [6]). We expect that the numerical performance of the algorithm can be improved if these advanced techniques are used instead of the naive approach proposed here. In the last few years several algorithmic frameworks for large scale composite convex optimization have been proposed. Examples include active set methods [17], stochastic methods [15], Newton type methods [16] as well as block coordinate descent methods [29]. In principle, all these algorithmic frameworks could be combined with the multilevel framework developed in this paper. Based on the theoretical and numerical results obtained from the multilevel version of the proximal gradient method we are hopeful that the multilevel framework can improve the numerical performance of many of the recent algorithmic developments in large scale composite optimization.

Acknowledgements The author acknowledges the help of Ryan Duy Luong with the numerical experiments in Section 4.

References


