March 1990 LIDS-P-1960

# ONE–WAY MULTIGRID ALGORITHMS AND THEIR OPTIMALITY FOR A CLASS OF FIXED–POINT PROBLEMS[1]

**Chee–Seng Chow**[2]
**John N. Tsitsiklis**[3]

## Abstract

We develop an abstract framework for the study of multigrid algorithms for the approximate solution of a general fixed-point problem that can be discretized at various levels of accuracy ("grids"). We assume that at each grid–level, we have an iterative algorithm for solving the resulting discretized problem. There are three grid-level dependent parameters: (i) the rate of convergence of the iterative algorithm, (ii) the distance of the solution of the discretized problem from the solution of the original problem, and (iii) the cost per iteration. The objective is to find a most cost-efficient algorithm for computing an approximation to the solution of the original problem, by performing iterations on a sequence of different grid-levels. We derive an optimal algorithm and evaluate its complexity. Furthermore, under our assumptions, we establish the optimality of one–way multigrid algorithms whereby iterations proceed from coarser to finer grids.

1. Research supported by the NSF under Grant ECS-8552419, with matching funds from Bellcore and Du Pont, and by the ARO under Grant DAAL03-86-K-0171.

2. Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass. 02139. Current address: I.B.M. T. J. Watson Research Center, P.O. Box 704, Yorktown Heights, NY 10598.

3. Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, Mass. 02139.

## I. INTRODUCTION

There are several computational problems whose exact solution is difficult or impossible, but which can be solved approximately when they are discretized or otherwise approximated. (Examples abound in image processing [T84] or in the numerical solution of partial differential equations [H85].) When choosing a problem discretization, one is always faced with the dilemma that a finer discretization can improve solution accuracy, at the expense of higher computational costs. This generic tradeoff has led to *multigrid* algorithms, whereby an approximate solution on a coarser grid is used as a starting point for the computations on a finer grid. The key idea is that iterations at finer grids, being more costly, could greatly benefit from the availability of a good starting point. Such algorithms, that move from coarser to finer grids, are sometimes called *one–way multigrid* algorithms.

In this paper, we address the problem of designing an "optimal" one–way multigrid iterative algorithm. That is, we take into account the grid–level dependence of the convergence rate, the solution accuracy, and the cost per iteration at each grid–level, to decide how many iterations should be performed at each grid–level, and at what sequence of grid–levels. This is done within a general framework that captures the essence of several interesting problems.

Within our framework, it is shown that there is no advantage in considering multigrid algorithms that are not of the one–way type. This seems contrary to what is known for the numerical solution of certain classes of partial differential equations where sophisticated multigrid methods that move back and forth between coarse and fine grids have been very succesful [H85]. The explanation for this discrepancy is that, in numerical analysis, multigrid methods are designed to exploit the special structure of certain PDEs, whereas our framework is in a sense too general to capture such special structure. In fact, our results are based on worst–case considerations and on an assumption that our knowledge of the problem's structure is somewhat limited. Thus, our results are not necessarily useful for the solution of certain classes of PDEs, but can be applicable to the solution of less structured, possibly nonlinear, problems, as they arise in image processing, stochastic control, and other contexts.

We now give an informal overview of the basic results in this paper. Consider a fixed-point equation $A_0x = x$, where $A_0$ is a contraction mapping [on a metric space $(M, d)$] with fixed point $x_0^*$. Suppose that we can discretize $A_0$ and obtain a family of fixed-point equations $A_hx = x$, $h \in (0, 1]$, where $A_h$ is a contraction mapping [on the same metric space $(M, d)$] with fixed point $x_h^*$. (Our convention is that $h$ denotes the grid-level with a smaller $h$ corresponding to a more accurate discretization.)

We make the following assumptions:

1. The discretization error $d(x_0^*, x_h^*)$ is bounded above by a function $D(h)$, where $D(h) \sim h^s$ for some known $s > 0$.
2. The contraction factor $\alpha(h)$ of $A_h$ satisfies $\alpha(h) \sim 1 - h^q$ for some known $q \geq 0$.
3. The computational cost of an iteration at grid–level $h$ satisfies $C(h) \sim h^{-r}$ for some known $r \geq 0$.

We are interested in computing an approximation of $x_0^*$, within some given accuracy $\epsilon$, by applying a sequence of iteration mappings drawn from the family $\{A_h \mid h \in (0, 1]\}$.

(Different values of $h$ can be used at different iterations.) The objective is to design such a multigrid algorithm that has minimal computational cost.

We show that the complexity of a single-grid algorithm is $O\big((1/\epsilon)^{(q+r)/s} \cdot \log(1/\epsilon)\big)$ and that the complexity of a particular one-way multigrid algorithm is $O\big((1/\epsilon)^{(q+r)/s}\big)$. We also establish the optimality of the one-way multigrid algorithm by proving a lower bound on the complexity of any multigrid algorithm within the class of algorithms that we consider. (Of course, this does not preclude algorithms with better complexity if more were known on the structure of the problem under consideration.)

Most available lower bounds on the algorithmic complexity of problems involving continuous variables are based on the so called "information–based" approach ([NY79], [TWW88]) and hold for fairly arbitrary classes of algorithms. In contrast, our lower bound exploits in an important manner the assumed limitations on the class of algorithms under consideration.

Our results are actually developed for a more general model. In particular, we let $\alpha(h) = 1 - f_1 h^q$, $C(h) = f_2 h^{-r}$, and $D(h) = f_3 h^s$, for some known positive constants $f_1$, $f_2$, and $f_3$, and we analyze the dependence of the algorithms and their complexity on these constants as well. One of the reasons for introducing this generalization is to account for the discount-factor dependence of the complexity in discrete-time stochastic control problems (Section 7). Another generalization is that we only require that for each $h$ and $x$, the sequence $\{A_h^t x\}$ converges to $x_h^*$ geometrically, thus weakening the contraction assumption.

**Outline**

In Section 2, we introduce our assumptions on the available family of iteration mappings. In Section 3, we provide a formal definition of the class of algorithms to be considered and describe the problem to be addressed. In Section 4, we evaluate the complexity of a single–grid algorithm. In Section 5, we introduce a one–way multigrid algorithm and upper bound its complexity. In Section 6, we develop a lower bound on the worst–case complexity of an arbitrary multigrid algorithm and establish the optimality of the algorithm of Section 5. In Section 7, we apply our results to the numerical solution of the Bellman equation for discrete–time stochastic control problems and some other examples. Section 8 contains our conclusions.

## II. FAMILIES OF ITERATION MAPPINGS

Let $(M, d)$ denote a metric space with metric $d$. Elements of $M$ will be referred to as *points*. We use Z to denote the set of nonnegative integers. For any mapping $A : M \mapsto M$, and any $t \in Z$, $A^t$ denotes the composition of $t$ copies of $A$. (In particular, $A^0$ is the identity mapping.)

**Definition 2.1:** A mapping $A : M \mapsto M$ is called a *pseudocontraction* if there exists some $\alpha \in [0, 1)$, some $K \in [1, \infty)$, and a point $x^* \in M$, such that

$$d(A^t x, x^*) \leq K \alpha^t d(x, x^*), \qquad \forall t \in Z,\ \forall x \in M. \tag{2.1}$$

We call $\alpha$ the *contraction factor, K* the *delay factor* (because it determines the number of iterations needed before the distance from $x^*$ is guaranteed to be reduced), and $x^*$ the *fixed point* of $A$. [Note that the uniqueness of $x^*$ follows from Eq. (2.1).]

Some interesting special cases are the following. If $A$ is a contraction mapping, then it is automatically a pseudocontraction, with $K = 1$. Suppose now that $A$ has the following two properties: a) $d(Ax, Ay) \leq d(x, y)$, and b) $d(A^m x, A^m y) \leq \alpha d(x, y)$, for all $x, y \in M$ and for some specific positive integer $m$. (Such a mapping $A$ is usually called an *m-step contraction.*) We then have

$$d(x^*, A^t x) \leq \alpha^{-1} \alpha^{t/m} d(x^*, x), \qquad \forall t \in Z,\ \forall x \in M,$$

and $A$ is a pseudocontraction with contraction factor $\alpha^{1/m}$ and delay factor $K = \alpha^{-1}$.

Our objective is to compute (approximately) a certain element $x_0^*$ of $M$. To this effect, it is assumed that we have available a family $\{A_h \mid h \in (0,1]\}$ of pseudocontractions that map $M$ into itself. Here, the parameter $h$ is meant to represent the *grid-level,* with the convention that a smaller $h$ represents a finer grid. Each mapping $A_h$ has a fixed point $x_h^*$, and it is assumed that $\lim_{h \downarrow 0} x_h^* = x_0^*$. For example, $x_0^*$ could be the solution to a PDE, and each $x_h^*$ could represent the solution to a difference equation $A_h x = x$ obtained by discretizing the original PDE.

We assume that each one of the pseudocontractions $A_h$ has a grid–dependent discount factor $\alpha(h) \in [0, 1)$ and the same delay factor $K$. In particular, for every $t \in Z$, $h \in (0, 1]$, $x \in M$, we have

$$d(A_h^t x, x_h^*) \leq K \alpha(h)^t d(x, x^*). \tag{2.2}$$

To model computational costs, we assume that we are given a function $C : (0, 1] \mapsto [0, \infty)$, and that $C(h)$ represents the computational effort involved in applying the mapping $A_h$ once. Thus, $C(h)$ is the cost per iteration at grid–level $h$.

If we wish to compute $x_0^*$ within a desired accuracy, we need some bounds for the distance of the solutions $x_h^*$ of the discretized problems from $x_0^*$. We thus assume that we are given a function $D : (0, 1] \mapsto [0, \infty)$ such that

$$d(x_0^*, x_h^*) \leq D(h), \qquad \forall h \in (0, 1]. \tag{2.3}$$

We call $d(x_0^*, x_h^*)$ the *discretization error* and $D(h)$ the *discretization error bound.*

While a theory can be developed when the functions $\alpha$, $C$, and $D$ are allowed to be fairly general [C89], the results are most powerful and informative when a special form is assumed, as we now proceed to do.

**Assumption 2.1:** (a) $\alpha(h) = 1 - f_1 h^q$, where $f_1 \in (0, 1]$ and $q \geq 0$ are constants independent of $h$.

(b) $C(h) = f_2 h^{-r}$, where $f_2 > 0$ and $r \geq 0$ are constants independent of $h$.

(c) $D(h) = f_3 h^s$, where $f_3 \geq 1$ and $s > 0$ are constants independent of $h$.

Our interest in Assumption 2.1 is justified because it corresponds to the most commonly encountered cases of problem discretization. For example, if a PDE in $r$ dimensions is discretized and $h$ indicates the grid spacing, then the number of grid points is of the order of $h^{-r}$. Assuming a constant computational effort per grid point and per iteration, the assumption $C(h) \sim h^{-r}$ is justified. Similarly, for the discretization of linear elliptic PDEs, the spectral radius $\alpha(h)$ of an iteration matrix typically satisfies $\alpha(h) \sim 1 - h^q$ for some $q > 0$, which motivates our assumption on $\alpha(h)$. Also, notice that by letting $q = 0$, we obtain the interesting case where the contraction factor is grid–independent. Finally, the case where $d(x_0^*, x_h^*) \sim f_3 h^s$ for some $f_3$ and $s$ is quite common.

## III. PROBLEMS, INSTANCES, AND ALGORITHMS

In this section, we specify a class of algorithms based on the iteration mappings $A_h$, that can be used to compute an approximation to the desired point $x_0^*$. Loosely speaking, the algorithms we consider have the following capabilities (and only those):

1. For any $x \in M$ and $h \in (0,1]$, we are able to compute $A_h x$, at a cost of $C(h)$.
2. The only knowledge available is the delay factor $K$, the parameters $q, r, s, f_1, f_2, f_3$ of Assumption 2.1, and an accuracy parameter $\epsilon > 0$,

Given the above capabilities, we are to apply a finite sequence of iteration mappings $A_h$ (different iterations can use different values of $h$) and terminate with a point $\hat{x} \in M$ satisfying $d(x_0^*, \hat{x}) \leq \epsilon$. We view such a sequence of iterations as an algorithm. Note that an iterative algorithm always needs a starting point, and we will assume that the point $x_1^*$ is available for free. This is not a significant loss of generality because $h = 1$ corresponds to the coarsest possible grid and, in practical problems, an exact solution at a very coarse grid is usually easily computable.

It should be emphasized that the algorithms to be considered are free to apply the mappings $A_h$ but do not have any insights into the internal structure of the mappings $A_h$. For example, a very intelligent algorithm, could look into the subroutine defining $A_h$, perform some mathematical analysis and come up with a closed–form formula for $x_0^*$. Our model precludes such forms of intelligence. In essence, we assume that the subroutines corresponding to $A_h$ are given to us in the form of black boxes.

We now formalize the above described notions. Following common practice in the theory of computation [LP81] it is important to distinguish between *problems* and *problem instances*.

**Definition 3.1:** Let us fix a metric space $(M, d)$.
(a) A *problem* $P$ [defined on $(M, d)$] consists of a collection of constants $K > 0$, $q \geq 0$, $r \geq 0$, $s > 0$, $f_1 \in (0,1]$, $f_2 > 0$, and $f_3 \geq 1$. Symbolically, $P = (K, q, r, s, f_1, f_2, f_3)$.
(b) An *instance* of the problem $P = (K, q, r, s, f_1, f_2, f_3)$ is a point $x_0^* \in M$ and a collection of mappings $A_h : M \mapsto M$, $h \in (0,1]$, such that the following are true for every $h \in (0,1]$:
(i) $A_h$ is a pseudocontraction with delay factor $K$ and contraction factor $1 - f_1 h^q$.
(ii) The fixed-point $x_h^*$ of $A_h$ satisfies $d(x_h^*, x_0^*) \leq f_3 h^s$.

Notice that condition (b)(ii) in this definition has the implication that $d(x_h^*, x_{\hat{h}}^*) \leq 2 f_3 (h^s + \hat{h}^s)$, a fact that will be often used later.

**Definition 3.2:** Let there be given a problem $P = (K, q, r, s, f_1, f_2, f_3)$ and a positive scalar $\epsilon > 0$. An *$\epsilon$-approximation algorithm* for this problem is a finite sequence $((t_1, h_1), \ldots, (t_i, h_i))$ such that

$$d(x_0^*, A_{h_i}^{t_i} \cdots A_{h_1}^{t_1} x_1^*) \leq \epsilon,$$

for all instances of the problem. The *complexity* of such an algorithm is defined to be $\sum_{j=1}^{i} t_j C(h_j)$. An algorithm is called *single–grid* if $i = 1$; it is called *one–way* if $h_1 >$

$h_2 > \cdots > h_i$. The *complexity* $C(P,\epsilon)$ of the problem is defined as the infimum of the costs of all $\epsilon$–approximation algorithms for this problem.

In our development, we view $K, q, r, s$ as given absolute constants and we will investigate the dependence of the complexity $C(P,\epsilon)$ on $\epsilon, f_1, f_2, f_3$. As we cannot hope to obtain a closed–form formula for the complexity, it is convenient to introduce the following order–of–magnitude notational convention. If $P = (K, q, r, s, f_1, f_2, f_3)$, we will write $C(P,\epsilon) = O(g(f_1, f_2, f_3, \epsilon))$ [respectively, $C(P,\epsilon) = \Omega(g(f_1, f_2, f_3, \epsilon))$] to indicate the fact that there exist constants $c > 0$ and $\epsilon_0 \in (0,1]$ (possibly depending on $K, q, r, s$), such that $C(P,\epsilon) \le cg(f_1, f_2, f_3, \epsilon)$ [respectively, $C(P,\epsilon) \ge cg(f_1, f_2, f_3, \epsilon)$], for all $f_1 \in (0,1]$, $f_2 > 0$ and $f_3 \ge 1$, and for all $\epsilon \in (0, \epsilon_0)$.

## IV. SINGLE–GRID ALGORITHMS

For the purpose of comparison with subsequent results, we evaluate here the complexity of a single–grid algorithm.

A single–grid $\epsilon$–approximation algorithm is easily described. Given $\epsilon > 0$, we choose a grid–level $h_\ell \in (0,1]$ so that $d(x_0^*, x_{h_\ell}^*) \le \epsilon/2$. This is certainly the case if we choose $h_\ell$ so that $f_3 h_\ell^s \le \epsilon/2$. Furthermore, we require that $h_\ell = 2^{-i}$ for some integer $i$ and we choose the largest possible $h_\ell$ subject to this restriction. Thus, the chosen $h_\ell$ satisfies

$$2^{-s}\frac{\epsilon}{2} \le f_3 h_\ell^s \le \frac{\epsilon}{2}. \tag{4.1}$$

Next, we find the smallest $t$ that satisfies

$$2K\alpha(h_\ell)^t f_3 \le \frac{\epsilon}{2}. \tag{4.2}$$

The single–grid algorithm consists of $t$ iterations at grid–level $h_\ell$, initialized with $x_1^*$ (which is assumed to be available for free).

We verify that this is indeed an $\epsilon$–approximation algorithm. We have

$$\begin{aligned} d(x_0^*, A_{h_\ell}^t x_1^*) &\le d(x_0^*, x_{h_\ell}^*) + d(x_{h_\ell}^*, A_{h_\ell}^t x_1^*) \le \frac{\epsilon}{2} + K\alpha(h_\ell)^t d(x_{h_\ell}^*, x_1^*) \\ &\le \frac{\epsilon}{2} + K\alpha(h_\ell)^t f_3 (h_\ell^s + 1^s) \le \frac{\epsilon}{2} + 2f_3 K\alpha(h_\ell)^t \le \epsilon, \end{aligned} \tag{4.3}$$

where the last inequality follows from our choice of $t$ [cf. Eq. (4.2)].

The complexity of this algorithm is $tC(h_\ell)$. Using Eq. (4.1), we have

$$C(h_\ell) = \frac{f_2}{h_\ell^r} \le f_2 2^r \left(\frac{2f_3}{\epsilon}\right)^{r/s}. \tag{4.4}$$

Furthermore, Eq. (4.2) yields

$$t \le \frac{\log(4Kf_3/\epsilon)}{|\log \alpha(h_\ell)|} + 1 = \frac{\log(4Kf_3/\epsilon)}{|\log(1 - f_1 h_\ell^q)|} + 1 \le \frac{\log(4Kf_3/\epsilon)}{f_1 h_\ell^q} + 1, \tag{4.5}$$

where the last inequality follows from $\log(1-\delta) \leq -\delta$ for all $\delta$. Using Eqs. (4.4)–(4.5), some algebra, and ignoring the constants that do not depend on $f_1, f_2, f_3, \epsilon$, we conclude that the complexity of the single–grid algorithm is

$$O\left(\frac{f_2}{f_1}\left(\frac{f_3}{\epsilon}\right)^{(q+r)/s}\log(f_3/\epsilon)\right). \tag{4.6}$$

## V. A ONE–WAY MULTIGRID ALGORITHM

We now introduce a one-way multigrid algorithm and analyze its complexity. This algorithm has better complexity than the single-grid algorithm of the preceding section and is based on the following:

**Multigrid Principle:** *Iterate at each level h, and obtain an approximation $\hat{x}_h$ of $x_h^*$, until the "approximation error" $d(\hat{x}_h, x_h^*)$ is comparable to the "discretization error" $d(x_h^*, x_0^*)$. Then, continue on a finer grid, using $\hat{x}_h$ as a starting point.*

The above stated multigrid can be considered part of folk knowledge. Interestingly enough, within our framework, it can be shown to lead to optimal algorithms.

We now provide a precise description of the algorithm. The algorithm employs the grid levels $h_i = 1/2^i$, $i = 1, 2, \ldots, \ell$, where $\ell$ is the smallest positive integer such that $f_3 h_\ell^s \leq \epsilon/2$. In particular, the finest grid-level $h_\ell$ is chosen exactly as in the single–grid algorithm, and Eq. (4.1) still holds. At each grid–level $h_i$, the algorithm starts with a point $\hat{x}_{i-1}$ computed at the preceding grid–level, and performs $t_i$ iterations, thus computing

$$\hat{x}_i = A_{h_i}^{t_i} \hat{x}_{i-1}. \tag{5.1}$$

For the first grid–level ($h_1 = 1$), the starting point is taken to be $\hat{x}_0 = x_1^*$ (assumed to be available).

The number $t_i$ of iterations at grid–level $h_i$ is chosen so that $d(\hat{x}_i, x_{h_i}^*) \leq f_3 h_i^s$. This ensures the correctness of the algorithm because at the final grid–level $h_\ell$ we have

$$d(\hat{x}_\ell, x_0^*) \leq d(\hat{x}_\ell, x_{h_\ell}^*) + d(x_{h_\ell}^*, x_0^*) \leq 2 f_3 h_\ell^s \leq \epsilon, \tag{5.2}$$

where the last inequality follows from our choice of $h_\ell$.

We now estimate the number of iterations needed at each grid–level. Assuming that $d(\hat{x}_{i-1}, x_{h_{i-1}}^*) \leq f_3 h_{i-1}^s$, we have

$$\begin{aligned} d(\hat{x}_i, x_{h_i}^*) &\leq K\alpha(h_i)^{t_i} d(\hat{x}_{i-1}, x_{h_i}^*) \leq K\alpha(h_i)^{t_i}[d(\hat{x}_{i-1}, x_{h_{i-1}}^*) + d(x_{h_{i-1}}^*, x_0^*) + d(x_0^*, x_{h_i}^*)] \\ &\leq K\alpha(h_i)^{t_i} f_3 [h_{i-1}^s + h_{i-1}^s + h_i^s] = K\alpha(h_i)^{t_i} f_3 (1 + 2^{s+1}) h_i^s, \end{aligned} \tag{5.3}$$

where the last equality follows because $h_i = h_{i-1}/2$. Since we wish to have $d(\hat{x}_i, x_{h_i}^*) \leq f_3 h_i^s$, it is sufficient to choose $t_i$ large enough so that

$$K\alpha(h_i)^{t_i} f_3 (1 + 2^{s+1}) h_i^s \leq f_3 h_i^s.$$

Thus,

$$t_i \le \frac{\log\left[K(1+2^{s+1})\right]}{|\log \alpha(h_i)|} + 1 \le \frac{\log\left[K(1+2^{s+1})\right]}{f_1 h_i^q} + 1, \tag{5.4}$$

where we have used the inequality $\log(1 - f_1 h_i^q) \le -f_1 h_i^q$.

We now evaluate the complexity of the one–way multigrid algorithm. Note that, by our choice of grid-levels, we have

$$\sum_{i=1}^{\ell} \frac{1}{h_i^p} = \sum_{i=1}^{\ell} \left(\frac{1}{2^{i-1} h_\ell}\right)^p = \frac{1}{h_\ell^p} \sum_{i=0}^{\ell-1} 2^{-pi} \le \frac{1}{h_\ell^p} \frac{1}{1-2^{-p}}, \quad \forall p > 0. \tag{5.5}$$

Using Eqs. (5.4)–(5.5), we obtain the following bound on the total complexity:

$$\begin{aligned}
\sum_{i=1}^{\ell} t_i C(h_i) &\le \sum_{i=1}^{\ell} \left(\frac{\log\left[K(1+2^{s+1})\right]}{f_1 h_i^q} + 1\right) \frac{f_2}{h_i^r} \\
&\le 2 f_2 \frac{\log\left[K(1+2^{s+1})\right]}{f_1} \sum_{i=1}^{\ell} \frac{1}{h_i^{(q+r)}} \\
&\le 2 f_2 \frac{\log\left[K(1+2^{s+1})\right]}{f_1} \frac{1}{h_\ell^{q+r}} \frac{1}{1-2^{-(q+r)}} \\
&\le 2 f_2 \frac{\log\left[K(1+2^{s+1})\right]}{f_1} \left(\frac{2 f_3}{\epsilon}\right)^{(q+r)/s} \frac{2^{(q+r)}}{1-2^{-(q+r)}} \\
&= O\left(\frac{f_2}{f_1}\left(\frac{f_3}{\epsilon}\right)^{(q+r)/s}\right).
\end{aligned} \tag{5.6}$$

We note, for future reference, that in the special case where the discount factor $\alpha(h)$ is equal to a constant $\beta$, for all grid–levels $h$, we can let $q = 0$, $f_1 = 1 - \beta$, to see that the complexity of our algorithm is

$$O\left(\frac{f_2}{1-\beta}\left(\frac{f_3}{\epsilon}\right)^{r/s}\right). \tag{5.7}$$

In our algorithm, we have used the grid–levels $h_i = 2^{-i}$. In general, we could have used $h_i = z^{-i}$, for any $z > 1$. Both the single–grid and the multigrid algorithms are easily adapted to this choice of grid-levels and their respective complexities remain unchanged. In practice, there is a preference for choosing $z$ an integer.

We notice that our multigrid algorithm improves upon the complexity of its single–grid counterpart by a factor of $\log(f_3/\epsilon)$. The natural question at this point is to inquire whether there exists an alternative multigrid algorithm with better complexity. For example, could it be advantageous to consider multigrid algorithms that are not of the one–way type? In the next section, we provide a negative answer to these questions.

## VI. LOWER BOUNDS AND THE OPTIMALITY OF ONE–WAY MULTIGRID ALGORITHMS

In this section, we show that for any fixed $K, q, r, s$, no algorithm (within our model of computation) can have better complexity than that of the one–way multigrid algorithm of the preceding section (at least as far as the dependence on $f_1, f_2, f_3, \epsilon$ is concerned). There is one technical assumption that will be needed. In particular, it is not possible to prove a lower bound on the computational complexity of a fixed–point problem if that problem is trivial, which would be the case if the metric space $(M, d)$ were an extremely simple one. For this reason, we have to assume that the underlying metric space is rich enough.

**Assumption 6.1:** The interval $[0,1]$ (with its usual metric) is isometrically imbedded in the metric space $(M, d)$.

**Theorem 6.1:** Suppose that the metric space $(M, d)$ satisfies Assumption 6.1. Then,

$$C(P,\epsilon) = \Omega\left(\frac{f_2}{f_1}\left(\frac{f_3}{\epsilon}\right)^{(q+r)/s}\right).$$

**Proof:** We first prove the result for the case where $M = [0,1]$ and $d$ is the usual metric. We will later indicate how the proof can be generalized.

For any problem $P = (K, q, r, s, f_1, f_2, f_3)$, we construct an instance as follows. We let $x_0^* = 0$, $x_h^* = \min\{1, f_3 h^s\}$, and

$$A_h x = \min\{1,\ x - f_1 h^q x + f_1 f_3 h^{q+s}\}, \qquad \forall x \in [0,1].$$

(It is easily seen that this is a legitimate instance satisfying all of our requirements.)

Consider an arbitrary $\epsilon$–approximation algorithm. Let $x(t)$ be the value of $x$ after $t$ iterations, and let $h(t)$ be the grid–level employed for the $t$th iteration. We then have $x(0) = x_1^* = 1$ and, since $x_0^* = 0$, we also have $x(T) \le \epsilon$, where $T$ is the number of iterations. Furthermore,

$$x(t) = \min\{1,\ x(t-1) - f_1 h(t)^q x(t-1) + f_1 f_3 h(t)^{q+s}\}, \qquad t = 1, \ldots, T. \tag{6.1}$$

It remains to derive a lower bound for the expression $\sum_{t=1}^{T} f_2 h(t)^{-r}$, which is the complexity of the algorithm.

**Lemma 6.1:** There exists some $\epsilon_0 \le 1/5$, depending only on $q, r, s$, such that if $\epsilon \le \epsilon_0$ and $x(t-1) \ge 5\epsilon$, then $x(t) \ge 4\epsilon$.

**Proof:** Suppose that $x(t-1) \ge 5\epsilon$ and that $\epsilon \le \epsilon_0 \le 1/5$. If $x(t) = 1$, then $x(t) \ge 5\epsilon_0 \ge 4\epsilon$. If $x(t) < 1$, then Eq. (6.1) yields

$$x(t) \ge 5\epsilon - f_1 h(t)^q 5\epsilon + f_1 f_3 h(t)^{q+s}. \tag{6.2}$$

If $5\epsilon \le f_3 h(t)^s$, then $x(t) \ge 5\epsilon$ and we are done. So, let us assume that $5\epsilon \ge f_3 h(t)^s$. Then, $h(t)^q \le (5\epsilon/f_3)^{q/s}$, and Eq. (6.2) becomes

$$x(t) \ge 5\epsilon - f_1 h(t)^q 5\epsilon \ge 5\epsilon\left(1 - f_1\left(\frac{5\epsilon}{f_3}\right)^{q/s}\right) \ge 5\epsilon(1 - (5\epsilon)^{q/s}),$$

where the last inequality follows because $f_1 \leq 1$ and $f_3 \geq 1$. If we now choose $\epsilon_0$ so that $1-(5\epsilon_0)^{q/s} \geq 4/5$, we obtain $x(t) \geq 4\epsilon$, as desired. **Q.E.D.**

Let us assume that $\epsilon \leq \epsilon_0$, where $\epsilon_0$ is the constant of Lemma 6.1. Let $\tau$ be the last value of $t$ for which $x(t) \in [4\epsilon, 5\epsilon]$. Such a $\tau$ exists because $x(0) = 1 \geq 5\epsilon$, $x(T) \leq \epsilon$, and because of Lemma 6.1. Let

$$J = \{t \mid \tau < t \leq T \text{ and } x(t) - x(t-1) < 0\}.$$

For every $t \in J$, Eq. (6.1) and the fact $x(t-1) \leq 5\epsilon$ imply that

$$5\epsilon \geq x(t-1) \geq f_3 h(t)^s, \qquad t \in J. \tag{6.3}$$

From Eq. (6.1), we also obtain

$$x(t) \geq \left(1 - f_1 h(t)^q\right) x(t-1),$$

which, combined with Eq. (6.3) yields

$$x(t) \geq \left(1 - f_1 (5\epsilon/f_3)^{q/s}\right) x(t-1), \qquad t \in J. \tag{6.4}$$

If $t > \tau$ but $t \notin J$, we have $x(t) \geq x(t-1)$, by the definition of $J$. This observation, together with Eq. (6.4) yields

$$\epsilon \geq x(T) \geq \left(1 - f_1(5\epsilon/f_3)^{q/s}\right)^{|J|} x(\tau) \geq \left(1 - f_1(5\epsilon/f_3)^{q/s}\right)^{|J|} 4\epsilon.$$

Thus, $\left(1 - f_1(5\epsilon/f_3)^{q/s}\right)^{|J|} \leq 1/4$. Let us first consider the case $|J| \geq 2$. Then, we can use the inequality $(1 - 1/|J|)^{|J|} \geq 1/4$ to conclude that

$$|J| \geq \frac{1}{f_1}\left(\frac{f_3}{5\epsilon}\right)^{q/s}. \tag{6.5}$$

Using Eq. (6.3), for every $t \in J$, the cost $f_2 h(t)^{-r}$ of the $t$th iteration is bounded below by $f_2(f_3/5\epsilon)^{r/s}$. Using this and Eq. (6.5), we conclude that the complexity of the algorithm is bounded below by

$$f_2\left(\frac{f_3}{5\epsilon}\right)^{r/s} \frac{1}{f_1}\left(\frac{f_3}{5\epsilon}\right)^{q/s} = \Omega\left(\frac{f_2}{f_1}\left(\frac{f_3}{\epsilon}\right)^{(q+r)/s}\right).$$

As for the case $|J| = 1$, it is easily shown to be impossible when $\epsilon_0$ (and therefore $\epsilon$) is chosen sufficiently small.

We have thus proved the result for the case $M = [0,1]$. In the general case, we have assumed that $M$ contains a set $Q$ which we can identify with $[0,1]$. We then construct an instance by defining $A_h$ on $Q$ exactly as before, and by letting $A_h x = x_h^*$ for all $x \notin Q$.

It is easily verified that this is an instance of the problem $P$, and the proof of the lower bound is identical. **Q.E.D.**

The lower bound of Theorem 6.1 coincides (up to a constant depending only on $K, q, r, s$) with the complexity of the one–way multigrid algorithm of Section 5 [cf. Eq. (5.7)]. Thus, if we are only concerned with the order of magnitude dependence of the complexity on the parameters $f_1, f_2, f_3, \epsilon$, the algorithm of Section 5 is an optimal algorithm. The lower bound of this section and the just stated optimality result should be interpreted with some care. It does not preclude the existence of better algorithms for a given problem. Rather, it should be interpreted as saying that if the only useful or available information about a given problem is captured by Assumption 2.1, then there cannot be any multigrid algorithm that works correctly and has lower complexity. This allows the possibility that non–multigrid algorithms could have better complexity, but such algorithms would have to rely on additional structural assumptions on the problem at hand. Similarly, it is also possible that better multigrid algorithms can be found, but they should again exploit some additional information on the problem.

## VII. EXAMPLES

We illustrate our results by considering the problem of the numerical solution of Bellman's equation in discrete–time stochastic control. Some other examples are also discussed. The discussion in this section is brief and occasionally informal, because our aim is only to highlight the connections with our framework.

### Solving the Bellman equation

Let $S = [0,1]^n$ and $U = [0,1]^m$ be the state and control spaces of a discrete–time stochastic control problem. Let $g : S \times U \mapsto [0,1]$ be a cost function, and let $p : S \times S \times U \mapsto [0,\infty)$ be another function satisfying $\int_S p(y,x,u)\,dy = 1$, for all $x \in S$ and $u \in U$. Intuitively, $p(\cdot,x,u)$ is the probability density function of the next state when the current state is $x$ and control $u$ is applied. Finally, let $\beta \in [0,1)$ be a discount factor. We assume that the functions $p$ and $g$ are Lipschitz continuous and that a bound $L$ on their Lipschitz constants is available. The objective is to minimize the infinite horizon discounted expected cost, starting from a given state.

Let $M$ be the set of all Borel measurable functions $J : S \mapsto \Re$, and let $d$ be the $L_\infty$ norm on $M$. Solving the above described stochastic control problem amounts to finding a function $J^* \in M$ (the optimal cost–to–go function) that solves the Bellman equation $TJ = J$, where $T : M \mapsto M$ is the dynamic programming operator defined by [BS78]

$$(TJ)(x) = \inf_{u \in U} E\left[g(x,u) + \beta \int_S p(y,x,u)J(y)\,dy\right] \qquad (7.1).$$

Under our assumptions, it is well known that $T$ is a contraction mapping, with contraction factor $\beta$. Furthermore, the problem can be approximated by one involving finite state and control spaces, as follows. We discretize the problem using $1/h$ points per dimension, thus partitioning $S$ into $(1/h)^n$ cubes and $U$ into $(1/h)^m$ cubes. There is a dynamic programming operator $T_h$ associated with the discretized problem which is also a contraction mapping with contraction factor $\beta$, and has a unique fixed point $J_h^*$ satisfying [W78, W79, CT89a]

$$d(J^*, J_h^*) \leq Bh/(1-\beta)^2,$$

where $B$ is a constant depending only on $L$, $m$, and $n$. Given a function $J \in M$ which is constant on each of the above mentioned boxes, $T_h J$ can be evaluated using $c/h^{2n+m}$ arithmetic operations, where $c$ is some given constant [CT89a]. [This is because the integral in Eq. (7.1) is replaced by a summation.]

The classical successive approximation (value iteration) algorithm for approximating $J^*$ chooses a grid–level $h$ and an initial function $J^0$, and applies $T_h$ a number of times until $J_h^*$ is closely approximated. If $h$ is chosen suitably small, this provides an approximation to $J^*$. In a multigrid version of this algorithm, different iterations apply the operator $T_h$ with different choices of $h$. We notice that the problem of finding a good multigrid algorithm is a special case of the problem considered in the preceding sections of this paper, with the following identifications:
(a) $1 - f_1 h^q = \beta$, which yields $q = 0$ and $f_1 = 1 - \beta$;
(b) $f_2 h^{-r} = ch^{-2n-m}$, which yields $r = 2n + m$ and $f_2 = c$.
(c) $f_3 h^s = Bh/(1-\beta)^2$, which yields $s = 1$ and $f_3 = B/(1-\beta)^2$;

According to the results of Sections 5–6, and disregarding the dependence on $B, c$, the one–way multigrid algorithm of Section 5 has complexity

$$O\Big(\frac{1}{1-\beta}\Big[\frac{1}{(1-\beta)^2\epsilon}\Big]^{2n+m}\Big), \tag{7.2}$$

and is optimal within the class of multigrid algorithms. (This is the same as the algorithm proposed in [CT89a].) In [CT89b], it is shown that every algorithm (not necessarily of the multigrid type) for the problem under consideration has complexity

$$O\Big(\Big[\frac{1}{(1-\beta)^2\epsilon}\Big]^{2n+m}\Big)$$

which differs by a factor of $1/(1-\beta)$ from the upper bound of Eq. (7.2). Whether this gap can be closed is an open problem. However, as the results of Section 6 show, if that gap is ever closed, this will be accomplished by a more sophisticated algorithm that exploits properties of the problem other than the facts $C(h) \sim h^{-2n-m}$ and $D(h) \sim h/(1-\beta)^2$.

### Stochastic approximation

Consider the iteration

$$y(t+1) = y(t) - h(t)\big(y(t) + w(t)\big), \tag{7.3}$$

where $y(t)$ is a scalar, $w(t)$ is a white noise sequence of unit variance, and $h(t)$ is a stepsize parameter. Equation (7.3) is the stochastic gradient algorithm for the minimization of the cost function $y^2$ on the basis of noisy observations. Let $V(t)$ be the mean square error $E[y(t)^2]$. Then, Eq. (7.3) yields

$$V(t+1) = V(t)\big(1-h(t)\big)^2 + h(t)^2. \tag{7.4}$$

The stepsize is under our control, and we can pose the problem of finding a stepsize sequence so as to minimize the number of iterations until we have $V(t) \leq \epsilon$.

It is clear that Eq. (7.4) is a contracting iteration with contraction factor $(1-h(t))^2$, which is close to $1-2h(t)$ when $h(t)$ is very small. In terms of our formalism, we have $q=1$, $s=1$, and $r=0$. Thus, Eq. (5.6) shows that an optimal choice of the sequence $h(t)$ requires $O(1/\epsilon)$ iterations. In contrast, if we were to let $h(t)$ be a constant (this would correspond to what we call a "single-grid algorithm"), $h$ should be chosen to be of the order of $\epsilon$ and the number of iterations would be $O\big((1/\epsilon)\cdot\log(1/\epsilon)\big)$.

Furthermore, our "one–way multigrid algorithm" indicates an appropriate choice for the stepsize sequence $h(t)$. The sequence $h(t)$ takes the values $2^{-i}$, $i=0,1,\ldots$, and the value $2^{-i}$ is used for $O(2^i)$ iterations. Thus, we have (up to a constant multiplicative factor) $h(t) \sim 1/t$ which coincides with the most common choice of stepsize in practice. Of course, the optimality of a stepsize sequence of the form $h(t) \sim 1/t$ can be derived from first principles (or from the Kalman filter equations). Our point is that there is a more general principle at play.

**Other examples**

In [L90], it is shown that the backpropagation algorithm for learning in feedforward neural networks attains a desired accuracy much faster if a slowly decreasing learning rate (stepsize parameter) is used, in contrast to a constant learning rate. The principle at play here is similar to the one in our stochastic approximation example, and we believe that the results of [L90] have a similar interpretation within our framework.

For another example, consider simulated annealing. If we view different temperatures as different "grid–levels", a cooling schedule corresponds to a "multigrid algorithm". As it turns out [C89], under our formalism, the one–way multigrid algorithm of Section 5 corresponds to the the popular logarithmic schedule [H85a]. Unfortunately, there is one technical assumption (the assumption that the delay factor is the same at all grid–levels) that does not seem to hold. Nevertheless, the analogy between our framework and simulated annealing is quite suggestive.

Finally, let us consider the celebrated Full Multigrid V–cycle algorithm [H85]. This algorithm is usually described as moving up and down between different grid-levels. There is however an alternative description whereby the algorithm is viewed as a sequence of V–cycles performed on a sequence of successively finer grids. If we view a V–cycle at grid-level $h$ as an iteration of $A_h$, then the Full Multigrid algorithm corresponds to the one-way multigrid algorithm of Section 5, with its attendant optimality properties. In contrast, if each $V$–cycle were to be performed at the same grid-level, we would be in a situation analogous to the single-grid algorithm of Section 4, which is non–optimal.

For most of the examples mentioned here, our formalism does not lead to any new results. On the other hand, we find it conceptually satisfying to realize that a common principle is at play in so many different types of algorithms.

## VIII. DISCUSSION

We have described an abstract general framework for the design of multigrid algorithms for a class of fixed–point problems, and have demonstrated that a particular one–way multigrid algorithm has certain optimality properties. In particular, our framework establishes rigorously the principle stating that computations at a grid–level should be carried out only until the solution accuracy balances the discretization error.

In our model, we have assumed specific functional forms for the contraction factor, the discretization error bound, and the cost per iteration. A related theory can be carried out in much greater generality. In particular, it can be shown [C89] that under some reasonable and fairly general assumptions, one–way multigrid algorithms are no worse than more general multigrid algorithms. Furthermore, similar results are also possible when one also considers a somewhat enlarged class of algorithms where the next iterate is a function of all previous iterates at the current grid–level [C89].

Regarding further research directions, it would be interesting to identify interesting application areas that can be cast within our framework.

## REFERENCES

[BS78] Bertsekas, D. P. and S. E. Shreve (1978). *Stochastic Optimal Control: The Discrete Time Case.* Academic Press, New York.

[C89] Chow, C.–S. (1989). Multigrid Algorithms and Complexity Results for Discrete–Time Stochastic Control and Related Fixed–Point Problems. Ph.D. Thesis, Technical Report LIDS–TH–1934, Laboratory for Information and Decision Systems, M.I.T., Cambridge, Massachusetts.

[CT89a] Chow, C.-S. and J. N. Tsitsiklis (1989). An Optimal Multigrid Algorithm for Discrete-Time Stochastic Control, Technical Report LIDS-P1864, Laboratory for Information and Decision Systems, M.I.T., Cambridge, Massachusetts, April. (To appear in the *IEEE Trans. Automatic Control.*)

[CT89b] Chow, C.-S. and J. N. Tsitsiklis (1989a). The Complexity of Dynamic Programming, *J. Complexity,* 5, pp. 466–488.

[H85] Hackbusch, W. (1985). *Multi-Grid Methods and Applications.* Springer-Verlag, New York.

[H85a] Hajek, B., "A Tutorial Survey of Theory and Applications of Simulated Annealing", *Proceedings of the 24th IEEE Conference on Decision and Control,* 1985, pp. 755–760.

[LP81] Lewis, H. R. and C. H. Papadimitriou (1981). *Elements of The Theory of Computation.* Prentice-Hall, Englewood Cliffs, New Jersey.

[L90] Luo, Z-Q., "On the Convergence of the Backpropagation algorithm with Adaptive Learning Rate", unpublished manuscript, 1990.

[NY83] Nemirovsky, A. S. and D. B. Yudin (1979). *Problem Complexity and Method Efficiency Optimization.* John Wiley & Sons, New York.

[T84] Terzopoulos, D. (1984). Multigrid Relaxation methods and the analysis of lightness, shading, and flow, *MIT AI Laboratory A.I. Memo No. 803.,* MIT, Cambridge, Massachusetts.

[TWW88] Traub, J. F., G. W. Wasilkowski, and H. Wozniakowski (1988). *Information-Based Complexity.* Academic Press, New York.

[W78] Whitt, W. (1978). Approximations of Dynamic Programs I, *Math. Oper. Res.* **3**, 231–243.

[W79] Whitt, W. (1979). Approximations of Dynamic Programs II, *Math. Oper. Res.* **4**, 179–185.