Optimally Adaptive Integration of Univariate Lipschitz Functions

Ilya Baran[1], Erik D. Demaine[1], and Dmitriy A. Katz[2]

[1] MIT Computer Science and Artificial Intelligence Laboratory,
32 Vassar Street, Cambridge, MA 02139, USA,
{ibaran,edemaine}@mit.edu

[2] Sloan School of Management, Massachusetts Institute of Technology,
50 Memorial Drive, Cambridge, MA 02142, USA,
dimdim@mit.edu

Abstract. We consider the problem of approximately integrating a Lipschitz function f (with a known Lipschitz constant) over an interval. The goal is to achieve an error of at most ϵ using as few samples of f as possible. We use the adaptive framework: on all problem instances an adaptive algorithm should perform almost as well as the best possible algorithm tuned for the particular problem instance. We distinguish between DOPT and ROPT, the performances of the best possible deterministic and randomized algorithms, respectively. We give a deterministic algorithm that uses $O(\mathrm{DOPT}(f,\epsilon)\cdot\log(\epsilon^{-1}/\mathrm{DOPT}(f,\epsilon)))$ samples and show that an asymptotically better algorithm is impossible. However, any deterministic algorithm requires $\Omega(\mathrm{ROPT}(f,\epsilon)^2)$ samples on some problem instance. By combining a deterministic adaptive algorithm and Monte Carlo sampling with variance reduction, we give an algorithm that uses at most $O(\mathrm{ROPT}(f,\epsilon)^{4/3} + \mathrm{ROPT}(f,\epsilon)\cdot\log(1/\epsilon))$ samples. We also show that any algorithm requires $\Omega(\mathrm{ROPT}(f,\epsilon)^{4/3} + \mathrm{ROPT}(f,\epsilon)\cdot\log(1/\epsilon))$ samples in expectation on some problem instance (f,ϵ), which proves that our algorithm is optimal.

1 Introduction

We consider the problem of approximating a definite integral of a univariate Lipschitz function (with known Lipschitz constant) to within ϵ using the fewest possible samples. The function is given as a black box: sampling it at a parameter value is the only allowed operation. It is easy to show that $\Theta(\epsilon^{-1})$ samples are necessary and sufficient for a deterministic algorithm in the worst case (see, e.g., [1]). The results in [2] imply a Monte-Carlo method that requires only $\Theta(\epsilon^{-2/3})$ samples in the worst case.

The Adaptive Framework. The univariate Lipschitz integration problem becomes more interesting in the adaptive setting. The motivation is that, for a given ϵ, some problem instances have much lower complexity than others. For example, if $f(x) = Lx$, where L is the Lipschitz constant, then evaluating f at the endpoints of the interval over which the integral is taken is sufficient to solve the problem for any ϵ. Thus, it is desirable to have an algorithm that is guaranteed to use fewer samples on easier problem instances. Such an algorithm is called *adaptive*. We formalize this notion by defining the difficulty of a problem as the performance of the best possible algorithm on that problem:

Definition 1. *Let $\mathcal{P}$ be a class of problem instances. Let $\mathcal{A}$ be the set of all correct algorithms for $\mathcal{P}$ (among some reasonable class of algorithms). Let* $\mathrm{COST}(A, P)$ *be the performance of algorithm $A \in \mathcal{A}$ on problem instance $P \in \mathcal{P}$. Define* $\mathrm{OPT}(P) = \min_{A \in \mathcal{A}} \mathrm{COST}(A, P)$*. We use* DOPT *when $\mathcal{A}$ is the set of deterministic algorithms and* ROPT *when $\mathcal{A}$ is the set of randomized algorithms that are correct on each $P \in \mathcal{P}$ with probability at least* $2/3$*.*

By definition, for every problem instance P, there is an algorithm whose cost on P is $\mathrm{OPT}(P)$. A good adaptive algorithm is a single algorithm whose cost is not much greater than $\mathrm{OPT}(P)$ for *every* problem instance P. Therefore, an adaptive guarantee is in general much stronger than a worst-case guarantee.

The ultimate goal of investigating a problem in the adaptive framework is to design an "optimally adaptive" algorithm. Suppose $\mathcal{P}$ is the set of problem instances and each problem instance $P \in \mathcal{P}$ has certain natural parameters, $v_1(P), \ldots, v_k(P)$, with the first parameter $v_1(P) = \mathrm{OPT}(P)$. An algorithm is *optimally adaptive* if its performance on every problem instance $P \in \mathcal{P}$ is within a constant factor of every algorithm's worst-case performance on the family of instances with the same values for the parameters: $\{P' \in \mathcal{P} \mid v_i(P') = v_i(P) \text{ for all } i\}$. Note that this definition depends on the choice of parameters, so in addition to OPT, we need to choose reasonable parameters, such as ϵ, the desired output accuracy.

Related Work. While approximate definite integration is well-studied both in numerical analysis (see, e.g., [3]) and in information-based complexity [4], those algorithms do not have provable guarantees about adaptivity. In that literature, the term "adaptive" typically refers to an algorithm that is allowed to pick samples based on previous sample values, which is quite different from our meaning.

For other problems, optimally adaptive algorithms have been previously designed in the context of set operations [5], aggregate ranking [6], and independent set discovery in [7]. Lipschitz functions also lend themselves well to adaptive algorithms. It is shown in [8] that Piyavskii's algorithm [9] for minimizing a univariate Lipschitz function performs $O(\mathrm{OPT})$ samples. [10] gives an adaptive algorithm for minimizing the distance from a point to a Lipschitz curve that is within a logarithmic factor of OPT. [11] gives adaptive algorithms for several problems on Lipschitz functions.

Our Results. In this paper we give a deterministic algorithm that makes at most $O(\mathrm{DOPT} \cdot \log(\epsilon^{-1}/\mathrm{DOPT}))$ samples. We also prove a matching lower bound on deterministic algorithms. When comparing to ROPT, however, we show that any deterministic adaptive algorithm uses $\Omega(\mathrm{ROPT}^2)$ samples on some problem instance. We present a randomized adaptive algorithm, LIPSCHITZ-MC-INTEGRATE, that always uses $O(\mathrm{ROPT}^{4/3} + \mathrm{ROPT} \cdot \log(\epsilon^{-1}))$ samples and prove a matching lower bound.

We therefore give optimally adaptive algorithms for the Lipschitz integration problem in the deterministic and randomized settings. Although the algorithms are simple, in both cases analyzing their adaptive performance is nontrivial. To our knowledge, LIPSCHITZ-MC-INTEGRATE is the first randomized optimally adaptive algorithm. Also, a simple corollary of the randomized lower bound is that the non-adaptive algorithm based on the results in [2] is optimal in the worst case.

Some of the results in this paper, primarily in Sections 3 and 4, are based on the first author's master's thesis [11]. Many of the proofs are omitted from this extended abstract.[3]

2 Problem Basics

We start by giving a precise formulation of the problem we consider:
Problem LIPSCHITZ-INTEGRATION:

Given: (f, a, b, L, ϵ)

Such that: $f\colon [a, b] \to \mathbb{R}$
and for $x_1, x_2 \in [a, b]$, $|f(x_2) - f(x_1)| \leq L|x_2 - x_1|$

Compute: $I \in \mathbb{R}$ such that $\left| I - \int_a^b f(x)\, dx \right| \leq \epsilon$

A randomized algorithm needs to be correct with probability at least $2/3$.

Some input parameters can be eliminated without loss of generality. The problem instance (f, a, b, L, ϵ) is equivalent to the problem instance $(\hat{f}, 0, 1, 1, \epsilon/L(b-a)^2)$ where $\hat{f}(x) = f\left(\frac{x-a}{b-a}\right)/L(b-a)$, so we can assume without loss of generality that $a = 0$, $b = 1$, and $L = 1$.

We now develop some basic tools we will need for discussing and analyzing the algorithms. Essentially, we show how to make use of the Lipschitz condition to bound the error of our estimates.

The Lipschitz condition allows an algorithm that has sampled f at two points to bound the value of the integral of f on the interval between them. We call the quality of this bound *area looseness*, and it depends on both the length of the interval and the values of f at the sampled points. A greater difference between values of f (a steeper function) results in a smaller area looseness. We define area looseness as follows (see Figure 1):

Definition 2. *Given a Lipschitz function f on $[0, 1]$, define the* area looseness *of a subinterval $[x_1, x_2]$ of $[0, 1]$ as $AL_f(x_1, x_2) = ((x_2 - x_1)^2 - (f(x_1) - f(x_2))^2)/2$. When it is clear which f we are talking about, we simply write $AL(x_1, x_2)$.*

Our analysis relies on area looseness being well behaved. The following proposition shows that it has the properties one would expect a bound on integration error to have and that an additional sample in the middle of the interval decreases total area looseness quickly.

[3] The full version of this paper is available at
`http://www.mit.edu/~ibaran/papers/intfull.{pdf,ps}`

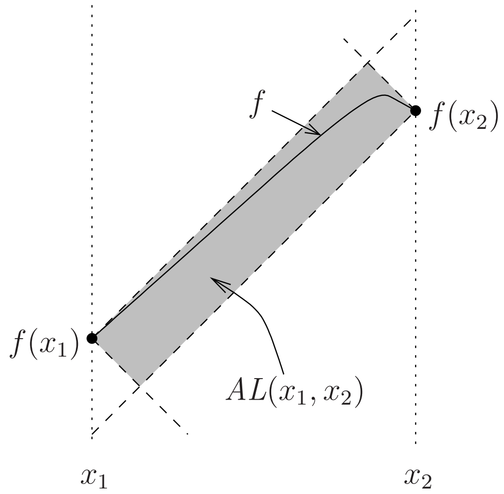

Fig. 1. Illustration of area looseness. Lipschitz bounds are dashed.

Proposition 1. *Area-looseness has the following properties:*
(1) $0 \le AL(x_1, x_2) \le (x_2 - x_1)^2/2$.
(2) If $x_1' \le x_1 < x_2 \le x_2'$ *then* $AL(x_1, x_2) \le AL(x_1', x_2')$.
(3) If $x \in [x_1, x_2]$, *then* $AL(x_1, x) + AL(x, x_2) \le AL(x_1, x_2)$.
(4) $AL\big(x_1, \frac{x_1+x_2}{2}\big) + AL\big(\frac{x_1+x_2}{2}, x_2\big) \le AL(x_1, x_2)/2$.

For the lower bounds, both on OPT and on adaptive algorithms, we need "extremal" Lipschitz functions, whose integral is either maximal or minimal, given the samples. We call these functions HI and LO. We also define *looseness*, the maximum difference between HI and LO over an interval.

Definition 3. *Given a Lipschitz function* f, *and* $0 \le a < b \le 1$, *define the Lipschitz functions* HI_a^b *and* LO_a^b *on* $[a, b]$ *as:* $HI_a^b(x) = \min(f(a) + x - a, f(b) + b - x)$ *and* $LO_a^b(x) = \max(f(a) - x + a, f(b) - b + x)$. *Also define* L_f *as* $L_f(a, b) = b - a - |f(b) - f(a)|$.

Proposition 2. *Given a Lipschitz function* f, *the functions* HI_a^b *and* LO_a^b *have the following properties:*

(1) If g *is Lipschitz,* $g(a) = f(a)$, *and* $g(b) = f(b)$, *then for* $x \in [a, b]$, $HI_a^b(x) \ge g(x) \ge LO_a^b(x)$.

(2) $AL(a, b)/(b-a) \le \max_{x \in [a,b]} (HI_a^b(x) - LO_a^b(x)) = L(a, b) \le 2AL(a, b)/(b-a)$

(3) $\int_a^b HI_a^b(x)\, dx = (b - a)\frac{f(a)+f(b)}{2} + AL(a, b)/2$ *and* $\int_a^b LO_a^b(x)\, dx = (b - a)\frac{f(a)+f(b)}{2} - AL(a, b)/2$.

Proposition 3. *Given a Lipschitz function f, looseness has the following properties:*
(1) $0 \le L(a,b) \le b-a$
(2) If $a' \le a \le b \le b'$, *then* $L(a,b) \le L(a',b')$.
(3) If $x_1 \le x_2 \le \cdots \le x_n$, *then* $\sum_{i=1}^{n-1} L(x_i, x_{i+1}) \le L(x_1, x_n)$.

3 Proof Sets

In order to compare the running time of an algorithm on a problem instance to DOPT, we define the concept of a proof set for a problem instance. A set P of points in $[0,1]$ is a *proof set* for problem instance (f,ϵ) and output x if for every f' that is equal to f on P, x is a correct output on (f',ϵ). In other words, sampling f at a proof set proves the correctness of the output. We say that a set of samples is a proof set for a particular problem instance without specifying the output if some output exists for which it is a proof set.

It is clear from the definition that sampling a proof set is the only way a deterministic algorithm can guarantee correctness: if an algorithm doesn't sample a proof set for some problem instance, we can feed it a problem instance that has the same value on the sampled points, but for which the output of the algorithm is incorrect. Conversely an algorithm can terminate as soon as it has sampled a proof set and always be correct. Thus, DOPT is equal to the size of a smallest proof set.

In order to analyze the deterministic algorithm, we will compare the number of samples it makes to the size of a proof set P. We will need some tools for doing this.

Let P be a nonempty finite set of points in $[0,1]$. Consider the execution of an algorithm which samples a function at points on the interval $[0,1)$ (if it samples at 1, ignore that sample). Let $s_1, s_2, \ldots, s_n$ be the sequence of samples that the algorithm performs in the order that it performs them. Let I_t be the set of unsampled intervals after sample s_t, i.e., the connected components of $[0,1) - \{s_1, \ldots, s_t\}$, except make each element of I_t half-open by adding its left endpoint, so that the union of all the elements of I_t is $[0,1)$. Let $[l_t, r_t)$ be the element of I_{t-1} that contains s_t.

Then sample s_t is a:

split	if	$[l_t, s_t) \cap P \ne \emptyset$ and $[s_t, r_t) \cap P \ne \emptyset$
squeeze	if	$[l_t, s_t) \cap P \ne \emptyset$ or $[s_t, r_t) \cap P \ne \emptyset$, but not both
fizzle	if	$[l_t, r_t) \cap P = \emptyset$.

These definitions are, of course, relative to P. See Figure 2. We can now bound the number of samples of different types:

Proposition 4. *The number of splits is at most* $|P| - 1$.

Proposition 5. *Suppose that for all i and j with $i \ne j$, $|s_i - s_j| > \epsilon$ and that for all t, $s_t = (l_t + r_t)/2$. Then if $|P| \le \epsilon^{-1}/2$, the number of squeezes is at most $|P| \log_2(\epsilon^{-1}/|P|)$.*

We now characterize proof sets for LIPSCHITZ-INTEGRATION.

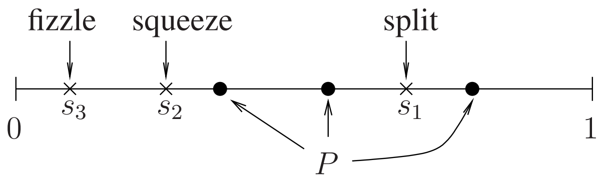

Fig. 2. Different types of samples.

Proposition 6. *Let* $P = \{x_1, x_2, \ldots, x_n\}$ *such that* $0 \le x_1 < x_2 < \cdots < x_n \le 1$. *Then* P *is a proof set for problem instance* (f, ϵ) *if and only if* $x_1^2 + (1 - x_n)^2 + \sum_{i=1}^{n-1} AL(x_i, x_{i+1}) \le 2\epsilon$.

4 Deterministic Algorithm and Analysis

Proposition 6, together with Proposition 1 immediately shows the correctness of a trivial algorithm. Let $n = \lceil \epsilon^{-1}/4 \rceil$ and let the algorithm make n samples, at $\frac{1}{2n}, \frac{3}{2n}, \ldots, \frac{2n-1}{2n}$ and output the integral M as in the proof of Proposition 6. It is correct because the area-looseness of every interval is at most $(1/n)^2/2$. Because there are $n - 1$ intervals, the total area-looseness of all of them is at most $(n - 1)/(2n^2)$. Also, $x_1^2 = (1 - x_n)^2 = 1/(2n)^2$, so $x_1^2 + (1 - x_n)^2 + \sum_{i=1}^{n-1} AL(x_i, x_{i+1}) = n/(2n^2) \le 2\epsilon$. Therefore, $\Theta(\epsilon^{-1})$ samples are always sufficient (and if, for instance, f is a constant, necessary).

We now give a deterministic adaptive algorithm. The algorithm maintains the total area-looseness of the current unsampled intervals, the unsampled intervals themselves in a linked list, and uses a priority queue to choose the unsampled interval with the largest area-looseness at every step and sample in the middle of it.

Let L be a linked list of $(\text{PARAMETER}, \text{VALUE})$ pairs and let Q be a priority queue of (AL, ELEM) pairs where the first element is a real number (and defines the order of Q) and the second element is a pointer into an element of L. The algorithm follows:

Algorithm LIPSCHITZ-INTEGRATE

1. Add $(0, f(0))$ and $(1, f(1))$ to L and insert $(AL(0, 1), (0, f(0)))$ into Q
2. A-LOOSENESS $\leftarrow AL(0, 1)$.
3. Do while A-LOOSENESS $> 2\epsilon$:
  4. $(\text{AL}, P_1) \leftarrow$ EXTRACT-MAX$[Q]$
  5. $P_2 \leftarrow$ NEXT$[L, P_1]$
  6. $x \leftarrow (\text{PARAMETER}[P_1] + \text{PARAMETER}[P_2])/2$
  7. $\text{AL}_1 \leftarrow AL(\text{PARAMETER}[P_1], x)$, $\text{AL}_2 \leftarrow AL(x, \text{PARAMETER}[P_2])$
  8. Insert $(x, f(x))$ into L after P_1 and insert (AL_1, P_1) and $(\text{AL}_2, (x, f(x)))$ into Q
  9. A-LOOSENESS $\leftarrow$ A-LOOSENESS $-$ AL $+$ AL_1 $+$ AL_2
10. Compute and output M using the values stored in L as described in Proposition 6.

The correctness of the algorithm is clear from Proposition 6: the algorithm stops precisely when the total area-looseness of the unsampled intervals is no more than 2ϵ. We need to analyze the algorithm's performance.

Theorem 1. *Algorithm* LIPSCHITZ-INTEGRATE *makes* $O(\mathrm{DOPT} \cdot \log(\epsilon^{-1}/\mathrm{DOPT}))$ *samples on problem instance* (f, ϵ).

Proof: We will actually compare the number of samples to $\mathrm{DOPT}(f, \epsilon/2)$ rather than to $\mathrm{DOPT}(f, \epsilon)$. We can do this because if we take a proof set for $\mathrm{DOPT}(f, \epsilon)$ and sample in the middle of every unsampled interval, then by Proposition 1 (4), we will obtain a proof set for $\mathrm{DOPT}(f, \epsilon/2)$. Thus, $\mathrm{DOPT}(f, \epsilon/2) \leq 2 \cdot \mathrm{DOPT}(f, \epsilon) + 1$. So let P be a proof set for $(f, \epsilon/2)$ of size $\mathrm{DOPT}(f, \epsilon/2)$.

First, we argue that no interval of length smaller than 4ϵ is ever subdivided. Suppose for contradiction that among n intervals $I_1, \ldots, I_n$ of lengths $a_1, \ldots, a_n$, interval I_k with $a_k < 4\epsilon$ is chosen for subdivision. By Proposition 1 (1), $AL(I_i) \leq a_i^2/2$, so $\sqrt{AL(I_k)} \leq 2\epsilon$. On the other hand, $\sum a_i = 1$, so $\sum \sqrt{AL(I_i)} \leq 1$. Multiplying the inequalities, we get $\sum AL(I_i) \leq \sum \sqrt{AL(I_i) AL(I_k)} \leq 2\epsilon$. But this implies that the algorithm should have terminated, which is a contradiction.

Now, we count the number of samples relative to P. The number of splits is $O(|P|)$ by Proposition 4. The above paragraph shows that we can use Proposition 5 to conclude that there are $O(|P| \log(\epsilon^{-1}/|P|))$ squeezes. We now show that there are $O(|P|)$ fizzles and so prove the theorem.

A fizzle occurs when an interval not containing a point of P is chosen for subdivision. Consider the situation after n points have been sampled. Let the sampled points be $0 = x_1 \leq x_2 \leq \cdots \leq x_n = 1$. Because the total area-looseness of intervals between points of P is at most ϵ, by repeated application of Proposition 1 (2,3), we have $\sum_{[x_i, x_{i+1}) \cap P = \emptyset} AL(x_i, x_{i+1}) \leq \epsilon$. The algorithm has not terminated, so the total area-looseness must be more than 2ϵ, which implies that $\sum_{[x_i, x_{i+1}) \cap P \neq \emptyset} AL(x_i, x_{i+1}) > \epsilon$. Because there are at most $|P|$ elements in the sum on the left hand side, the largest element must be greater than $\epsilon/|P|$. Therefore, there exists a k such that $[x_k, x_{k+1})$ contains a point of P and $AL(x_k, x_{k+1}) > \epsilon/|P|$. So if a fizzle occurs, the area-looseness of the chosen interval must be at least $\epsilon/|P|$.

Now let S_t be the set of samples made by the algorithm after time t. Define A_t as follows: let $\{y_1, y_2, \ldots, y_n\} = S_t \cup P$ with $0 = y_1 \leq y_2 \leq \cdots \leq y_n$ and let $A_t = \sum_{i=1}^{n-1} AL(y_i, y_{i+1})$. Clearly, $A_t \geq 0$, $A_t \geq A_{t+1}$ (by Proposition 1 (3)), and therefore, $A_t \leq A_0 \leq 2\epsilon$. Every fizzle splits an interval between adjacent y's into two. Because the area-looseness of the interval before the split was at least $\epsilon/|P|$, by Proposition 1 (4), A_t decreases by at least $\epsilon/(2|P|)$ as a result of every fizzle. Therefore, there can be at most $4|P|$ fizzles during an execution. □

We prove a matching lower bound, showing that the logarithmic factor is necessary and that LIPSCHITZ-INTEGRATE is optimally adaptive:

Theorem 2. *For any deterministic algorithm and for any* $\epsilon > 0$ *and any integer* k *such that* $0 < k < \epsilon^{-1}/2$, *there exists a problem instance* (f, ϵ) *of* LIPSCHITZ-INTEGRATION *with* $\mathrm{DOPT}(f, \epsilon) = O(k)$ *on which that algorithm performs* $\Omega(k \log(\epsilon^{-1}/k))$ *samples.*

5 Algorithm LIPSCHITZ-MC-INTEGRATE

A standard strategy in a Monte Carlo integration algorithm is to sample at a point picked uniformly at random from an interval. The expected value of such a sample, scaled by the length of the interval, is precisely the value of the integral over the interval, so the goal is to minimize the variance. When the function is Lipschitz, the variance of the integral estimate based on such a sample can be as high as a constant times the fourth power of the length of the interval. However, if we use the fact that when the area looseness of an interval is low, we approximately know the function, we can adjust the sample to get an unbiased estimator of the integral over that interval whose variance is the square of the area looseness in the worst case. Procedure MC-SAMPLE shows how to do this.

Procedure MC-SAMPLE(x_1, x_2):

1. Let x be a random number, uniformly chosen from $[x_1, x_2]$
2. If $f(x_1) \le f(x_2)$, then SAMPLE $\leftarrow \left(f(x) - x + \frac{x_1+x_2}{2}\right)$
3. Else SAMPLE $\leftarrow \left(f(x) + x - \frac{x_1+x_2}{2}\right)$
4. Return SAMPLE $\cdot\,(x_2 - x_1)$

Proposition 7. MC-SAMPLE(x_1, x_2) *returns an unbiased estimator of* $\int_{x_1}^{x_2} f(x)\,dx$ *that has variance at most* $AL^2(x_1, x_2)$.

In order to compute the integral over $[0, 1]$, we would like an estimator for that integral with low variance. If we split $[0, 1]$ into intervals whose total AL^2 is small and run MC-SAMPLE on each interval, we will get such an estimator, as shown in the following corollary.

Corollary 1. *Let* $0 = x_1 < x_2 < \cdots < x_n = 1$ *and suppose* $\sum_{i=1}^{n-1} AL^2(x_i, x_{i+1}) \le \epsilon^2/3$. *Let* $\hat{I} = \sum_{i=1}^{n-1}$ MC-SAMPLE(x_i, x_{i+1}). *Let* $I = \int_0^1 f(x)\,dx$. *Then* $\Pr[|\hat{I} - I| \ge \epsilon] \le 1/3$.

The remaining difficulty is to find a small number of intervals whose total AL^2 is smaller than $\epsilon^2/3$. Note that the deterministic adaptive algorithm in Section 4 finds a small number of intervals whose total AL is smaller than ϵ. We show that we can use the same idea here. Thus, to obtain a randomized adaptive algorithm, we use a deterministic adaptive algorithm to get a rough idea of the function and then use Monte Carlo sampling with variance reduction (MC-SAMPLE) to improve our estimate of the integral.

Let L be a linked list of (PARAMETER, VALUE) pairs and let Q be a priority queue of (AL, ELEM) pairs where the first element is a real number (and defines the order of Q) and the second element is a pointer into an element of L. The algorithm is as follows:

Algorithm LIPSCHITZ-MC-INTEGRATE:

1. Add $(0, f(0))$ and $(1, f(1))$ to L and insert $(AL^2(0, 1), (0, f(0)))$ into Q
2. ALSQ $\leftarrow AL^2(0, 1)$.
3. Do while ALSQ $> \epsilon^2/3$:
  4. (AL, P_1) $\leftarrow$ EXTRACT-MAX$[Q]$
  5. $P_2 \leftarrow$ NEXT$[P_1]$
  6. $x \leftarrow$ (PARAMETER$[P_1]$ + PARAMETER$[P_2]$)/2
  7. AL$_1 \leftarrow AL^2$(PARAMETER$[P_1], x$), AL$_2 \leftarrow AL^2(x,$ PARAMETER$[P_2]$)

8. Insert $(x, f(x))$ into L after P_1 and insert (AL_1, P_1) and $(\text{AL}_2, (x, f(x)))$ into Q
9. $\text{ALSQ} \leftarrow \text{ALSQ} - \text{AL} + \text{AL}_1 + \text{AL}_2$
10. $\hat{I} \leftarrow 0$.
11. For each element P of L except the last:
12. $\hat{I} \leftarrow \hat{I} + \text{MC-SAMPLE}(\text{PARAMETER}[P], \text{PARAMETER}[\text{NEXT}[P]])$
13. Output $\hat{I}$

Correctness is guaranteed by Corollary 1 because the algorithm exits the loop in lines 3-9 only when the total AL^2 of intervals between points in L is no more than $\epsilon^2/3$.

6 Performance Analysis

For the analysis of the algorithm, let f be the Lipschitz function input to LIPSCHITZ-MC-INTEGRATE.

Lemma 1. *Given f, there exists a set of points $0 = x_1 < x_2 < \cdots < x_n = 1$ such that for $1 \leq i \leq n-2$, $AL(x_i, x_{i+1}) = 3\epsilon$, and $AL(x_{n-1}, x_n) \leq 3\epsilon$. Furthermore, $\text{ROPT}(f, \epsilon) \geq (n-2)/3$.*

Proof: We begin by constructing a set of points that satisfies the conditions. Obviously, x_1 should be 0. Suppose we have constructed the first k points and $x_k \neq 1$. If $AL(x_k, 1) \leq 3\epsilon$, set $x_{k+1} = 1$ and we are done. Otherwise, notice that f is continuous, so AL is also continuous. By Proposition 1 (1), $AL(x_k, x_k) = 0$. Therefore, by the intermediate value theorem, there is an $x \in [x_k, 1]$ such that $AL(x_k, x) = 3\epsilon$ and we set x_{k+1} to be that x.

Consider an algorithm A that is correct with probability at least $2/3$ on all inputs and consider its executions on f. Let e_i for $1 \leq i \leq n-2$ be the expected number of samples A performs in (x_i, x_{i+1}). We claim that in order for A to be correct, it must have $e_i \geq 1/3$ for all i and therefore, the total expected number of samples is $\sum_{i=1}^{n-2} e_i \geq (n-2)/3$.

Suppose for contradiction, that $e_i < 1/3$ for some i. Then, by Markov's inequality, the probability that A samples in (x_i, x_{i+1}) is less than $1/3$. Now consider two functions defined as follows: $\hat{f}_1(x) = \hat{f}_2(x) = f(x)$ everywhere except (x_i, x_{i+1}) and $\hat{f}_1(x) = LO_{x_i}^{x_{i+1}}(x)$ and $\hat{f}_2(x) = HI_{x_i}^{x_{i+1}}(x)$ on (x_i, x_{i+1}). By Proposition 2 (3), $\int_0^1 \hat{f}_2(x)dx - \int_0^1 \hat{f}_1(x) = AL(x_i, x_{i+1}) = 3\epsilon$, so no output is correct for both $\hat{f}_1$ and $\hat{f}_2$. Suppose, that we feed $\hat{f}_1$ and $\hat{f}_2$ with probability $1/2$ each as input to A. Conditioned on A not sampling in (x_i, x_{i+1}), the output of A is independent of which function was input. Therefore, conditioned on A not sampling in (x_i, x_{i+1}), the probability of error is at least $1/2$. Because $\hat{f}_1 = \hat{f}_2 = f$ not on (x_i, x_{i+1}), the probability of A not sampling on (x_i, x_{i+1}) is greater than $2/3$, so the probability of error is greater than $1/3$, which implies that A is invalid. □

Because the number of samples in steps 11–13 is smaller (by 1) than the number of samples in steps 1–9, we only focus on the samples in steps 1-9. For the analysis, we split the execution of the algorithm into two phases. The algorithm is in Phase 1 while there is a pair of adjacent elements x_i and x_{i+1} in L for which $AL(x_i, x_{i+1}) > 3\epsilon$. When all pairs of adjacent elements have AL at most 3ϵ, the algorithm is in Phase 2. Note that by Proposition 1 (2), area looseness between adjacent points in L never increases as the algorithm executes, so once it enters Phase 2, it never goes back to Phase 1. We now bound the number of samples made in steps 1–9 in the phases.

Lemma 2. *In Phase 1,* LIPSCHITZ-MC-INTEGRATE *makes* $O(\mathrm{ROPT}(f, \epsilon) \log(1/\epsilon))$ *samples on problem instance* (f, ϵ).

Proof: Let X be the set of x_i's constructed as in Lemma 1. We count the samples made by LIPSCHITZ-MC-INTEGRATE relative to X. By Proposition 4, there are at most $O(|X|)$ splits. We now need a lower bound on the size of intervals in Phase 1 to count the number of squeezes. We note that an interval whose length is smaller than $\sqrt{6\epsilon}$ has area looseness at most 3ϵ (by Proposition 1 (1)) and will therefore never be chosen for subdivision in Phase 1. Therefore, in Phase 1, every interval has length at least $\sqrt{6\epsilon}/2$. So by Proposition 5, there are at most $|X| \log((\sqrt{6\epsilon}/2)^{-1}/|X|) = O(|X| \log(1/\epsilon))$ squeezes. There are no fizzles because any interval whose area looseness is greater than 3ϵ must have a point of X (by Proposition 1 (2) and by construction of X). By Lemma 1, $|X| = O(\mathrm{ROPT}(f, \epsilon))$, so we have the claimed bound. □

Lemma 3. *In Phase 2,* LIPSCHITZ-MC-INTEGRATE *uses at most* $O(\mathrm{ROPT}(f, \epsilon)^{4/3} + \mathrm{ROPT}(f, \epsilon) \log(1/\epsilon))$ *samples on problem instance* (f, ϵ).

Proof: After Phase 1 is complete, L consists of points such that the area looseness between adjacent pairs is at most 3ϵ. Let $0 = y_1 < y_2 < \cdots < y_m = 1$ be the smallest subset of points in L (including 0 and 1) such that $AL(y_i, y_{i+1}) \leq 3\epsilon$ for all y. We claim that $m \leq 6 \cdot \mathrm{ROPT}(f, \epsilon)$. Consider the set of x_i's constructed as in Lemma 1. If y_i's are a minimal set of points with area looseness no greater than 3ϵ between adjacent ones, then every interval of the form $[x_i, x_{i+1}]$ has at most two y_i's (if there are three, the middle one is unnecessary). Therefore there are at most twice as many y_i's as x_i's.

Now assume the algorithm makes more samples in Phase 2 than in Phase 1 because otherwise, it makes $O(\mathrm{ROPT}(f, \epsilon) \log(1/\epsilon))$ samples and we are done. We apply Propostion 8 to prove this lemma. Let Y be the set of y_i's, let $Z^{(0)}$ be the set of points in L at the end of Phase 1 and let $t_0 = 550 \cdot \mathrm{ROPT}^{4/3}$. We have $A = \sum_{i=1}^{m-1} AL(y_i, y_{i+1}) \leq 18 \cdot \mathrm{ROPT} \cdot \epsilon$. By Proposition 8, after t_0 samples, the total AL^2 will be at most $\frac{4608 \cdot (6 \cdot \mathrm{ROPT})^2 \cdot (18 \cdot \mathrm{ROPT})^2 \epsilon^2}{550^3 \mathrm{ROPT}^4} \leq \epsilon^2/3$ so the algorithm will stop after t_0 steps. □

The following proposition shows that as our algorithm samples, the total squared area looseness declines as the cube of the number of samples. We prove it by associating a number with each interval that is an upper bound on its area looseness. We then show that these numbers are within a factor of four of each other and use this to show that that the sum of their squares decreases as the cube of the number of samples.

Proposition 8. *Let* $Y = \{y_1, \ldots, y_m\}$ *with* $0 = y_1 < \cdots < y_m = 1$*, and let* $A = \sum_{i=1}^{m-1} AL(y_i, y_{i+1})$*. Consider the sequence* $Z^{(0)}, Z^{(1)}, Z^{(2)}, \ldots$ *of sets of samples where* $Z^{(0)} \supseteq Y$ *is an arbitrary superset of* Y *and, for each* $t \geq 1$*,* $Z^{(t)} =$

$Z^{(t-1)} \cup \{z^{(t)}\}$ *where* $z^{(t)}$ *is the midpoint* $(x^{(t)} + y^{(t)})/2$ *of the interval* $(x^{(t)}, y^{(t)})$ *of* $Z^{(t-1)}$ *with the largest area looseness* $AL(x^{(t)}, y^{(t)})$*. Then, for any* $t_0 \geq |Z_0|$, $\sum_{(x,y)\in\mathcal{I}(Z^{(t)})} AL^2(x, y) \leq (4608m^2A)/t_0^3$.

The upper bound follows immediately from the two lemmas we have shown.

Theorem 3. *On problem instance* (f, ϵ) *algorithm* LIPSCHITZ-MC-INTEGRATE *performs* $O(\mathrm{ROPT}^{4/3}(f, \epsilon) + \mathrm{ROPT}(f, \epsilon)\log(1/\epsilon))$ *samples.*

7 Randomized Lower Bounds

We first show that Lemma 1 is actually a tight (to within a constant factor) lower bound on ROPT by proving the following upper bound.

Lemma 4. *Given a Lipschitz function* f*, there is a set of points* $0 = x_1 < x_2 < \cdots < x_k = 1$ *such that for* $1 \leq i \leq k-2$, $AL(x_i, x_{i+1}) = \epsilon/4$*, and* $AL(x_{k-1}, x_k) \leq \epsilon/4$*. Furthermore,* $\mathrm{ROPT}(f, \epsilon) \leq 2k - 1$.

The above lemma implies that deterministic algorithms are not very powerful relative to ROPT. For instance, if $f(x) = 0$ for all x, $\mathrm{ROPT}(f, \epsilon) = O(\epsilon^{-1/2})$ by Lemma 4, but DOPT is $\Theta(\epsilon^{-1})$. Therefore every deterministic algorithm requires $\Omega(\mathrm{ROPT}^2)$ samples on some instances.

Theorem 4. *Given an* $\epsilon > 0$ *and an integer* k *such that* $0 < k < \epsilon^{-1}/2$*, there is a family of problem instances such that* $\mathrm{ROPT} = O(k)$ *on every member on the family, but any algorithm requires* $\Omega(k^{4/3} + k\log(1/\epsilon))$ *samples in expectation on some member of that family.*

A simple corollary shows that the nonadaptive method in [2] is optimal.

Corollary 2. *Any algorithm requires* $\Omega(\epsilon^{-2/3})$ *samples on some problem instance.*

8 Conclusion

We gave optimally adaptive deterministic and randomized algorithms for LIPSCHITZ-INTEGRATION. To simplify the analysis, we have been lax with constant factors in the randomized algorithm and the related proofs. Thus, it is possible to improve both the algorithm's performance and its analysis by constant factors.

A more interesting open problem is to design adaptive algorithms for definite integration over two or higher-dimensional domains or to prove that good adaptive algorithms do not exist. Although simple Monte Carlo methods readily extend to higher dimensions, designing and analyzing adaptive algorithms seems difficult.

References

1. Werschulz, A.G.: An overview of information-based complexity. Technical Report CUCS-022-02, Computer Science Department, Columbia University (2002)
2. Barabesi, L., Marcheselli, M.: A modified monte carlo integration. International Mathematical Journal **3**(5) (2003) 555–565
3. Davis, P.J., Rabinowitz, P.: Methods of Numerical Integration. second edn. Academic Press, San Diego (1984)
4. Traub, J., Wasilkowski, G., Woźniakowski, H.: Information-Based Complexity. Academic Press, New York (1988)
5. Demaine, E.D., López-Ortiz, A., Munro, J.I.: Adaptive set intersections, unions, and differences. In: Proceedings of the 11th Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, California (2000) 743–752
6. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. Journal of Computer and System Sciences **66**(4) (2003) 614–656
7. Biedl, T., Brejova, B., Demaine, E.D., Hamel, A.M., López-Ortiz, A., Vinař, T.: Finding hidden independent sets in interval graphs. Theoretical Computer Science **310**(1–3) (2004) 287–307
8. Hansen, P., Jaumard, B., Lu, S.H.: On the number of iterations of piyavskii's global optimization algorithm. Mathematics of Operations Research **16**(2) (1991) 334–350
9. Piyavskii, S.: An algorithm for finding the absolute extremum of a function. USSR Computational Mathematics and Mathematical Physics **12** (1972) 57–67
10. Baran, I., Demaine, E.D.: Optimal adaptive algorithms for finding the nearest and farthest point on a parametric black-box curve. In: Proceedings of the 20th Annual ACM Symposium on Computational Geometry, Brooklyn, NY (2004) To appear.
11. Baran, I.: Adaptive algorithms for problems involving black-box lipschitz functions. Master's thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts (2004) At `http://www.mit.edu/~ibaran/papers/mthesis.{pdf,ps}`.