Dual Hashing-based Algorithms for Discrete Integration (Extended Abstract) ⋆

Alexis de Colnet[1] and Kuldeep S. Meel[2]

[1] CNRS, CRIL UMR 8188, Lens, France
[2] School of Computing, National University of Singapore, Singapore

Abstract. Given a boolean formula F and a weight function ρ, the problem of discrete integration seeks to compute the weight of F, defined as the sum of the weights of satisfying assignments. Discrete integration, also known as weighted model counting, is a fundamental problem in computer science with wide variety of applications ranging from machine learning and statistics to physics and infrastructure reliability. Given the intractability of the exact variant, the problem of approximate weighted model counting has been subject to intense theoretical and practical investigations over the years.

The primary contribution of this paper is to investigate development of algorithmic approaches for discrete integration. Our framework allows us to derive two different algorithms: WISH, which was already discovered by Ermon et al [8], and a new algorithm: SWITCH. We argue that these algorithms can be seen as dual to each other, in the sense that their complexities differ only by a permutation of certain parameters. Indeed we show that, for F defined over n variables, a weight function ρ that can be represented using p bits, and a confidence parameter δ, there is a function f and an NP oracle such that WISH makes $\mathcal{O}\left(f(n,p,\delta)\right)$ calls to NP oracle while SWITCH makes $\mathcal{O}\left(f(p,n,\delta)\right)$ calls. We find $f(x,y,\delta)$ polynomial in x, y and $1/\delta$, more specifically $f(x,y,\delta) = x\log(y)\log(x/\delta)$. We first focus on striking similarities of both the design process and structure of the two algorithms but then show that despite this quasi-symmetry, the analysis yields time complexities dual to each other. Another contribution of this paper is the use of 3-wise property independence of XOR based hash functions in the analysis of WISH and SWITCH. To the best of our knowledge, this is the first usage of 3-wise independence in deriving stronger concentration bounds and we hope our usage can be generalized to other applications.

⋆ The author list has been sorted alphabetically by last name; this order should not be used to determine the extent of authors' contributions.
The work was performed during first author's stay at NUS.
The full version along with Appendix is available at https://github.com/meelgroup/dualhashing

1 Introduction

Given a set of constraints F and a weight function ρ that assigns a non-negative weight to every assignment of values to variables, the problem of discrete integration seeks to compute the weight of F, defined as the sum of weights of its satisfying assignments. If every assignment has weight 1, the corresponding problem is often simply called *model counting*. For clarity of presentation, we use *unweighted model counting* to denote this variant. Discrete integration is a fundamental problem in computer science. A wide variety of problems such as probabilistic inference [14], partition function of graphical models, permanent of a matrix [18], un-reliability of a network [13] can be reduced to discrete integration.

In his seminal work, Valiant [18] established the complexity of discrete integration as #P-complete for all polynomially computable weight functions, where #P is the complexity class comprised of counting problems whose decision variant lies in NP. Given the computational intractability of discrete integration, approximate variants have been subject of intense theoretical and practical investigations over the past few decades.

Approaches to discrete integration can be classified into three categories: variational techniques, sampling techniques, and hashing-based techniques. Inspired from statistical physics, variational methods often scale to large instances but do not provide guarantees on the computed estimates [19,17]. Sampling-based techniques focus on approximation of the discrete integral via sampling from the probability distribution induced by the boolean formula and the weight function [11]. The estimation of rigorous bounds, however, requires exponential mixing times for the underlying chains and therefore, practical implementations such as those based on Markov Chain Monte Carlo methods [2] or randomized branching choices [9] fail to provide rigorous estimates [7,12]. Recently, hashing-based techniques have emerged as a promising alternative to variational and sampling techniques to provide rigorous approximation guarantees [8,5,4]. The hashing-based algorithm WISH seeks to utilize progress made in combinatorial solving over the past two decades and to this end, the problem of discrete integration is reduced to linear number of optimization queries subject to randomly generated parity constraints [8].

The primary contribution of this paper is to investigate the development of algorithmic approaches for discrete integration. Our framework allows us to derive two different algorithms, which can be seen as dual to each other: WISH, which was already discovered by Ermon et al [8], and a new algorithm: SWITCH. In particular, WISH reduces the problem of discrete integration to optimization queries while SWITCH proceeds via reduction to unweighted model counting. Both WISH and SWITCH compute constant factor approximations with arbitrarily high probability $1 - \delta$ via usage of universal hash functions, a concept invented by Carter and Wegman in their seminal work [3]. We first

focus on the design process of WISH and SWITCH. We study discrete integration through the framework of general integration and reduce the task to optimization and counting subproblems. Then we present WISH and SWITCH as hashing-based algorithms solving the aformentionned subproblems to approximate a discrete integral. Finally we analyse these algorithms, proving that both compute constant factor approximations of the integral with high probability. However we show that they have dual time complexities in the sense that, for F defined over n variables and a weight function ρ that can be represented using p bits, there is a function f and an NP oracle such that WISH makes $\mathcal{O}\left(f(n,p,\delta)\right)$ calls to NP oracle while SWITCH makes $\mathcal{O}\left(f(p,n,\delta)\right)$. We find $f(x,y,\delta)$ polynomial in x, y and $1/\delta$, more specifically $f : x,y,\delta \mapsto x\log(y)\log(x/\delta)$.

Another contribution of this paper is the use of 3-wise property independence of XOR based hash functions in the analysis of WISH and SWITCH. To the best of our knowledge, this is the first usage of 3-wise independence in deriving stronger concentration bounds. The hardness of usage of 3-independence for concentration bounds is well documented by absence of such analyses (c.f.: wonderful blogpost by Mihai Pătraşcu: [3]).

The duality obtained may not seem surprising in retrospect but such has not been the case for the past few years. The prior work has often, without complete evidence, asserted that the corresponding dual approach would be inferior both theoretically and empirically [8,4]. Our work, in turn, contradicts such assertions and shows that the two approaches indeed have dual time complexity from theoretical perspective and empirical analysis will be key in determining their usefulness. Since the work on development of MaxSAT solvers that support XORs and SAT solvers that support XORs and Pseudo-Boolean (PB) constraints is in its infancy; our work provides a strong argument for the need and potential of both of these solvers as queries generated by WISH require MaxSAT solvers with the ability to handle XORs while the queries by SWITCH requires SAT solvers that support XORs and PB constraints.

The rest of the paper is organized as follows. We introduce notations and preliminaries in Section 2. We then provide general framework for discrete integration in Section 3, which is employed to derive the aforementioned algorithms, WISH and SWITCH, in Section 4. We finally conclude in Section 5.

2 Preliminaries and Notations

Let F be a boolean formula over n variables. Let X be the set of variables appearing in F. A literal is a variable x or its negation $\neg x$. An assignment σ of all n variables is a *satisfying assignment* or *witness* of F if it makes F evaluate to *true*, which we note $\sigma \models F$. We note $\#F$ the number of witnesses of F.

[3] http://infoweekly.blogspot.com/2010/01/moments.html

Weight function. Let $\rho : \{0,1\}^n \to \mathbb{Q}_+$ be the *weight function* mapping each truth assignment to a positive value such that

- $\forall \sigma \in \{0,1\}^n$, weight $\rho(\sigma)$ is computable in polynomial time
- $\forall \sigma \in \{0,1\}^n$, weight $\rho(\sigma)$ is written in binary representation with less than p bits.

We extend the weight function to sets of truth assignments and boolean formulas. Let Y be a subset of $\{0,1\}^n$, the weight of Y is defined as the cumulative weight of the truth assignments in Y: $\rho(Y) = \sum_{\sigma \in Y} \rho(\sigma)$. By definition the weight of the empty set is 0. The weight of a formula F is defined as the cumulative weight of its witnesses $\rho(F) = \sum_{\sigma \models F} \rho(\sigma)$. For notational clarity, we overload ρ to indicate weight of an assignment, set of assignments, and formula depending on the context.

Given a formula F and weight function ρ, we define the *effective weight function* w as the restriction of ρ to the witnesses of F

$$w(\sigma) = \begin{cases} \rho(\sigma) & \text{if } \sigma \models F \\ 0 & \text{otherwise} \end{cases}$$

We will note $w_{\min} = \min_{\sigma \models F} w(\sigma)$ and $w_{\max} = \max_{\sigma \models F} w(\sigma)$ the minimum and maximum weights of a witness of F. Due to the hypothesis on ρ we have $w_{\max} \leq 2^p$ and $w_{\min} \geq 2^{-p}$ if F is satisfiable. Note that the expression for the weight of a formula can be rewritten $\rho(F) = \sum_{\sigma \in \{0,1\}^n} w(\sigma)$.

Tail function. Dual to the effective weight function is the *tail function* τ. It is defined from the space of weights to $\mathbb{N}$. The tail function on some weight u counts the number of truth assignments heavier than u (i.e. of weight greater than u).

$$\tau(u) = \left|\{\sigma \in \{0,1\}^n : w(\sigma) \geq u\}\right|$$

For notational clarity we extend the tail function to truth assignments using the notation $\tau(\sigma)$ for $\tau(w(\sigma))$. Note that

1. The tail function is non-increasing.
2. The maximum tail is $\tau(0) = 2^n$.
3. For any $0 < u \leq w_{\min}$ there is $\tau(u) = \#F$.
4. If $u > w_{\max}$ then $\tau(u)$ evaluates to 0, but the minimal non-zero tail $\tau(w_{\max})$ is not necessarily 1 since more than one truth assignment can weight $w_{\max}$.

MPE-MAP queries. Following standard definitions, MPE (*most probable explanation*) corresponds to solving $\max(\rho(\sigma) : \sigma \models F)$, which is to find $w_{\max}$. It is worth noting that MPE is related to another query: MAP (*maximum a posteriori*), and different communities use different definitions for MAP and MPE, to the extent that what one community calls MAP is called MPE by another [8,4].

(ε, δ)-approximation algorithms. Given computational intractability of computing $\rho(F)$, we are interested in approximation schemes. For a tolerance $\varepsilon > 0$ and a confidence $\delta > 0$, an algorithm $\mathcal{A}$ generates a (ε, δ)-approximation of W if it returns a quantity in $\left[W(1+\varepsilon)^{-1}, (1+\varepsilon)W\right]$ with probability at least $1-\delta$.

$$\Pr\left[(1+\varepsilon)^{-1}W \leq \mathcal{A}(F, \rho, \varepsilon, \delta) \leq (1+\varepsilon)W\right] \geq 1-\delta$$

3-universal hash functions. We focus on hashing-based methods to approximate $\rho(F)$. We use particular classes of hash functions based on parity constraints. A constraint specifies a set of indices S from $[n]$ and a bucket value β in $\{0,1\}$. The assignment σ is said to satisfy the constraint if the xored value of its coordinates on S matches β, or more formally if $\bigoplus_{i \in S} \sigma[i] = \beta$, where $\oplus$ denotes the "xor" operation. Using the binary vector representation of subsets S in $\{0,1\}^n$, one can rewrite the left hand side of the constraint as a scalar product in the field $\mathbb{F}_2^n$ which addition and product operations are, respectively, the "xor" and the "and" operations. Therefore we will use matrix representations when applying several constraints. For m given constraints represented with the matrix $A \in \{0,1\}^{m \times n}$ and the vector of bucket values $b \in \{0,1\}^m$, σ satisfies all m constraints if $A\sigma = b$, or equivalently $A\sigma \oplus b = 0$. A hash function h from $\{0,1\}^n$ to $\{0,1\}^m$ is defined by a collection of m constraints embedded in A and b. An assignment σ is hashed through h to $h(\sigma) = A\sigma \oplus b$. So the i-th component of $h(\sigma)$ is

$$h(\sigma)[i] = b_i \oplus \bigoplus_{j=1}^{n} A[i,j]\sigma[j]$$

Let $H_{\text{xor}}(n,m)$ be the class of all such hash functions from $\{0,1\}^n$ to $\{0,1\}^m$.

$$H_{\text{xor}}(n,m) = \left\{\sigma \mapsto A\sigma \oplus b : A \in \{0,1\}^{m \times n}, b \in \{0,1\}^m\right\}$$

We note $h \xleftarrow{R} H_{\text{xor}}(n,m)$ the action of choosing a hash function uniformly at random from $H_{\text{xor}}(n,m)$, which is equivalent to sampling A from $\mathcal{B}_{1/2}(m,n)$ and b from $\mathcal{B}_{1/2}(m)$. Hash functions in $H_{\text{xor}}(n,m)$ have uniformity property, meaning that for all y in $\{0,1\}^m$ and σ in $\{0,1\}^n$, there is

$$\Pr\left[h \xleftarrow{R} H_{\text{xor}}(n,m) : h(\sigma) = y\right] = \frac{1}{2^m}$$

It was also shown in [10] that they display 3-wise independence property, meaning that for all three images y_1, y_2, y_3 in $\{0,1\}^m$ and for all three distinct assignments $\sigma_1, \sigma_2, \sigma_3$ in $\{0,1\}^n$, there is

$$\Pr\left[h \xleftarrow{R} H_{\text{xor}}(n,m) : h(\sigma_1) = y_1 \text{ and } h(\sigma_2) = y_2 \text{ and } h(\sigma_3) = y_3\right] = \frac{1}{2^{3m}}$$

They do not display independence at higher order. For instance for 4-wise independence, consider three assignments $\sigma_1, \sigma_2, \sigma_3$ and four images y_1, y_2, y_3, y_4. Define $\sigma_4 = \sigma_1 \oplus \sigma_2 \oplus \sigma_3$ and see that if $h(\sigma_i) = y_i$ for $i \in \{1,2,3\}$, then $h(\sigma_4) = y_1 \oplus y_2 \oplus y_3$. So the probability for all four assignments to be projected on their respective images is null when $y_4 \neq y_1 \oplus y_2 \oplus y_3$.

3 A Framework for Discrete Integration

This section presents a framework for discrete integration. Methods from this framework follow a two-steps strategy:

1. translate the task of discrete integration into an integration problem for a real non-increasing function
2. apply a method to approximate the integral of a real function

In the first step, we specifically ask for a non-increasing function so that we can ensure constant factor approximations when estimating its integral. Examples of approximation methods for real function integrals are the upper and lower rectangles approximations or Monte Carlo integrators.

3.1 From discrete integration to real function integration

Given F a boolean formula and a weight function ρ, let $u_1, \cdots, u_K$ be all possible weights taken by the satisfying assignments of F. To obtain the discrete integral $\rho(F)$, i.e. the sum of weights of satisfying assignments of F, one can gather assignments in packets of same effective weight and sum over these packets. For the weight u_i, the pre-image $w^{-1}(u_i)$ is the set of all witnesses of F mapped to u_i by ρ. So the discrete integral can be written

$$\rho(F) = \sum_{i=1}^{K} u_i |w^{-1}(u_i)| \tag{1}$$

We observe the following *tail transformation*:

- For $i < K$, there is $|w^{-1}(u_i)| = \tau(u_i) - \tau(u_{i+1})$
- In the case $i = K$, there is $|w^{-1}(u_K)| = \tau(u_K)$

Applying this transformation to Eq (1) gives:

$$\rho(F) = u_K \tau(u_K) + \sum_{i=1}^{K-1} u_i \left(\tau(u_i) - \tau(u_{i+1})\right) \tag{2}$$

and after rearranging the terms:

$$\rho(F) = u_1 \tau(u_1) + \sum_{i=1}^{K-1} \tau(u_{i+1}) \left(u_{i+1} - u_i\right) \tag{3}$$

These two representations of the discrete integral have a graphical interpretation: draw the curve of τ as a function of the weight, and observe that both $\tau(u_{i+1})(u_{i+1} - u_i)$ and $u_i(\tau(u_i) - \tau(u_{i+1}))$ are areas of rectangles under the curve as illustrated in figure 1. Eq (2) decomposes the integral into rectangles built along the τ axis while Eq (3) is a decomposition into rectangles built along the w axis .

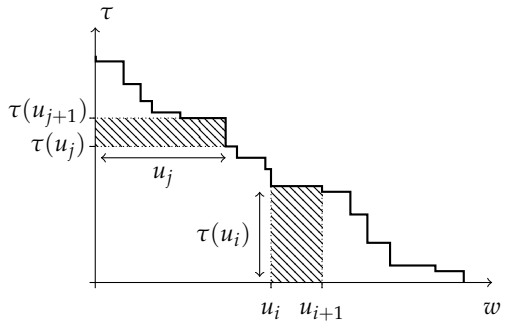

Fig. 1: Decomposition into rectangle areas

The discrete integral $\rho(F)$ is the area under the curve of τ.

$$\rho(F) = \int \tau(u)du \tag{4}$$

The effective weight function w can be expressed as a function of the tails which extension to $\mathbb{R}_+$ is $w : t \mapsto \max_\sigma(w(\sigma) : \tau(\sigma) \geq t)$. Graphically, one can just rotate the graph of τ to obtain that of w and see that:

$$\rho(F) = \int w(t)dt \tag{5}$$

Both (4) and (5) are integrals of non-increasing functions defined over $\mathbb{R}_+$ and of finite support.

3.2 From discrete integration to optimization

Direct computation of any form previously obtained is intractable. We resort to approximations of $\rho(F)$ when it is written as (4) or (5). Given that τ and w are staircase functions, rectangles approximation seems to be the only method fitted to approximate their integrals. First we apply the method on Eq (4). The first step is the partition of the weight axis into linearly many intervals. We split the axis at the quantile weights, defined as followed:

Definition 1. *The 2^i-th quantile weight of the weight distribution is the maximal weight q_i such that $\tau(q_i) \geq 2^i$.*

The quantile weights $q_0, \cdots, q_n$ are all well-defined, and form a non-increasing sequence. Consecutive quantile weights can be equal. For instance if F has $< 2^m$ witnesses for some $m < n$, then $q_m = q_{m+1} = \cdots = q_n = 0$. Note that for each quantile weight q_i, there exists some truth assignment σ such that $q_i = w(\sigma)$. The partition of integral (4) at the quantile weights gives:

$$\rho(F) = q_n 2^n + \sum_{i=1}^{n} \int_{q_i}^{q_{i-1}} \tau(u)du$$

where $\int_{q_i}^{q_{i-1}}$ represents the integral on $]q_i, q_{i-1}]$. Since the weight q_n does not lie in any interval we add the term $q_n\tau(q_n) = q_n 2^n$ manually.
If u is in $]q_i, q_{i-1}]$, then $\tau(u) < 2^i$, otherwise q_i would not be the maximal weight of tail $\geq 2^i$. Furthermore $\tau(u) \geq \tau(q_{i-1})$ which is $\geq 2^{i-1}$ by definition. So for each weight in $]q_i, q_{i-1}]$ we bound the corresponding tail within a factor of 2. Figure 2 illustrates this rectangle approximation on the interval $]q_{n-1}, q_{n-2}]$.

$$2^{i-1}\int_{q_i}^{q_{i-1}} du \leq \int_{q_i}^{q_{i-1}} \tau(u)du \leq 2^i \int_{q_i}^{q_{i-1}} du$$
$$2^{i-1}\left(q_{i-1} - q_i\right) \leq \int_{q_i}^{q_{i-1}} \tau(u)du \leq 2^i\left(q_{i-1} - q_i\right)$$

Note that the bound holds when $]q_i, q_{i-1}]$ is empty ($q_i = q_{i-1}$). Summing all bounds together and rearranging the terms to get rid of differences of quantiles, we obtain:

$$q_0 + \sum_{i=0}^{n-1} q_{i+1}2^i \leq \rho(F) \leq q_0 + \sum_{i=0}^{n-1} q_i 2^i$$

The two bounds are within a ratio of 2 of each other because the integral on each interval was bounded within a ratio of 2. Let us choose the lower bound to be our first estimate of $\rho(F)$ and name it $W_1 = q_0 + \sum_{i=0}^{n-1} q_{i+1}2^i$. We have

$$W_1 \leq \rho(F) \leq 2W_1 \tag{6}$$

Given $q_0, \cdots, q_n$, the estimate W_1 can be computed in polynomial time. For all i, the weight q_i is, by definition, the solution of the following optimization problem:

$$q_i = \max\left\{w(\sigma) : \tau(w(\sigma)) \geq 2^i\right\}$$

So the approximation of the discrete integral $\rho(F)$ has been reduced to $n+1$ optimization sub-problems.

3.3 From discrete integration to counting

To find W_1 we have done rectangles approximation on Eq (4). In this section we investigate the estimate resulting from a similar approximation on Eq (5). The first step is the partition of the tail axis. We will assume, for notational clarity, that $w_{\max} \leq 1$. This bound is legitimate in the context of probabilistic inferences [14], and the results of this paper can be extended to any arbitrary but fixed bound. Recall that the weights are written with p bits in binary representation, so the bounds $w_{\max} \leq 2^p$ and $w_{\min} \geq 2^{-p}$ are always valid. For our partition, we define the splitting tails as followed:

Definition 2. *The i-th splitting tail τ_i is the tail at weight $1/2^i$: $\tau_i = \tau(1/2^i)$.*

Given the assumption on the range value of w, the interesting tails are $\tau_0, \cdots, \tau_p$. They form a non-decreasing sequence. The partition of integral (5) at the splitting tails gives:

$$\rho(F) = \tau_0 + \sum_{i=0}^{p-1} \int_{\tau_i}^{\tau_{i+1}} w(t)dt$$

where $\int_{\tau_i}^{\tau_{i+1}}$ represents the integral on $]\tau_i, \tau_{i+1}]$. Since the tail τ_0 does not lie in any interval we add the term $\tau_0 w(\tau_0)$ manually. τ_0 is the number of assignments heavier than weight 1. Either there are no such assignment and $\tau_0 = 0$, or there are some, in which case $w(\tau_0) = w_{\max} = 1$. In both cases we find that $\tau_0 w(\tau_0) = \tau_0$. If t is in $]\tau_i, \tau_{i+1}]$, then $2^{-i-1} \leq w(t) \leq 2^{-i}$. So for each tail in $]\tau_i, \tau_{i+1}]$, we bound the corresponding weight within a factor of 2. Figure 3 illustrates this rectangle approximation on the interval $]\tau_1, \tau_2]$.

$$2^{-i-1} \int_{\tau_i}^{\tau_{i+1}} dt \leq \int_{\tau_i}^{\tau_{i+1}} w(t)dt \leq 2^{-i} \int_{\tau_i}^{\tau_{i+1}} dt$$
$$2^{-i-1} (\tau_{i+1} - \tau_i) \leq \int_{\tau_i}^{\tau_{i+1}} w(t)dt \leq 2^{-i} (\tau_{i+1} - \tau_i)$$

Note that the bound holds when $]\tau_i, \tau_{i+1}]$ is empty ($\tau_i = \tau_{i+1}$). Summing all bounds together and rearranging the terms to get rid of differences of tails, we obtain:

$$\tau_p 2^{-p} + \sum_{i=0}^{p-1} \tau_i 2^{-(i+1)} \leq \rho(F) \leq \tau_p 2^{-p} + \sum_{i=0}^{p-1} \tau_{i+1} 2^{-(i+1)}$$

The two bounds are within a ratio of 2 of each other because the integral on each interval was bounded within a ratio of 2. Let us choose the lower bound to be our first estimate of $\rho(F)$ and name it $W_2 = \tau_p 2^{-p} + \sum_{i=0}^{p-1} \tau_i 2^{-(i+1)}$. We have

$$W_2 \leq \rho(F) \leq 2W_2 \tag{7}$$

Given $\tau_0, \cdots, \tau_p$, the estimate W_2 can be computed in polynomial time. For all i, the tail τ_i is, by definition, the solution of the following counting problem:

$$\tau_i = \left|\{\sigma : w(\sigma) \geq 2^{-i}\}\right|$$

So the approximation of the discrete integral $\rho(F)$ has been reduced to $p + 1$ counting sub-problems.

3.4 On the limitations of the estimates

The two estimates W_1 and W_2 are not only similar in terms of construction but also in terms of theoretical guarantees and limitations. Both are lower bounds of $\rho(F)$ and approximate $\rho(F)$ within a ratio of 2. Furthermore, both W_1 and W_2 use some unknown quantities, respectively the weights $q_0, \cdots, q_n$ and the tails $\tau_0, \cdots, \tau_p$. These are to be approximated.

Assuming that for positive some ε and δ we have an algorithm $\mathcal{A}$ returning C, a (δ, ε)-approximations of W_1 (resp. W_2). Then with probability at least $1 - \delta$, C is a bounded estimate of $\rho(F)$ such that $\frac{\rho(F)}{2(1+\varepsilon)} \leq C \leq (1 + \varepsilon)\rho(F)$. In any case, the quality of the estimate is capped: the best approximation interval possible is $[\rho(F)/2, \rho(F)]$. However, note that we obtained 2-approximations of $\rho(F)$ using

base-2 partitions of the tail and weight axis for the rectangles approximations. If we use base-β partitions instead, with $\beta < 2$, we can improve our estimates. For instance for $\beta = 1 + \varepsilon$, we partition the weight axis at the $(1+\varepsilon)^i$-th quantile weights and the tail axis at the tails $\tau((1+\varepsilon)^{-i})$. Rectangles approximations W_1' and W_2' are then both in $[\rho(F), (1+\epsilon)\rho(F)]$. Now, algorithm $\mathcal{A}$ returns some quantity in $\left[\rho(F)/(1+\varepsilon)^2, (1+\varepsilon)\rho(F)\right]$ with probability at least $1-\delta$. So $\mathcal{A}$ computes a $(3\varepsilon, \delta)$-approximation of $\rho(F)$ (for $\varepsilon < 1$ we have $(1+\varepsilon)^{-2} \geq (1+3\varepsilon)^{-1}$).

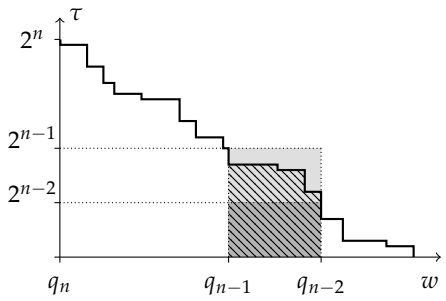

Fig. 2: Rectangles approximation on Eq (2)

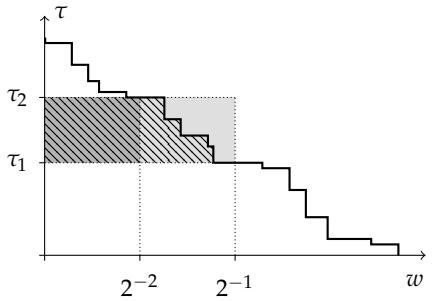

Fig. 3: Rectangles approximation on Eq (3)

4 Algorithms

In this section we present two algorithms to approximate the discrete integral $\rho(F)$. The first one approximates W_1. It uses a hashing-based approach to approximate solutions for the optimization/MPE sub-problems described in section 3.2. Given the lack of any approach to find 2^i-th quantiles, hashing is used to reduce the task to that of standard optimization. The second algorithm approximates W_2 and also implements a strategy based on hashing functions to approximately solve the counting sub-problems described in section 3.3, this choice is motivated by the success of hashing-based technique for model counting [5]. Since it is known that $(1+\varepsilon)$-approximations can be obtained from constant factor approximations by standard amplification techniques [8], we will focus on obtaining constant factor approximations. Possibilities of extension of the algorithms to reach arbitrary precision approximations following the strategy of section 3.4 will be discussed in section 4.4.

4.1 An NP Oracle

The procedure for discrete integration via optimization was first discovered by Ermon et al [8]. They expressed the complexity of their algorithm as the number of calls to an MPE oracle. It is customary to express complexity with respect to oracles corresponding to decision problems. Therefore, we express

complexities in terms of invocations of an NP oracle. The oracle is a system capable of solving a decision problem in $\mathcal{O}(1)$ time. In this paper, the oracle is given a boolean formula F, a weight function ρ computable in polynomial time and a real number u, and returns *YES* if and only if there exists a satisfying assignment of F of weight greater than u. More formally, it solves the problem $\mathsf{SAT}(F \wedge \{\rho(\sigma) \geq u\})$ where the constraint $\rho(\sigma) \geq u$ is not necessarily boolean. Given a solution σ, the condition $\sigma \models F$ can be tested in polynomial time, and so is $\rho(\sigma) \geq u$ by hypothesis on ρ. So the decision problem is in NP.

4.2 Discrete integration by optimization: WISH

We present a modified version of the algorithm of Ermon et al [8]: WISH (Weighted Integration and Sum by Hashing). When comparing our version of WISH to the original, we will refer to the latter as WISH_EGSS, from the initials of its authors. WISH takes in a formula F, a weight function ρ, and a confidence parameter δ, and returns an estimate for W_1.

Algorithm 1 WISH(F, ρ, δ)

$T \leftarrow \lceil 128 \ln(2n/\delta) \rceil$
for all $1 \leq t \leq T$ **do**
  $A_0 \leftarrow [\,]$, $b_0 \leftarrow [\,]$
  for all $0 \leq i \leq n$ **do**
    Sample constraint C_i in $\{0,1\}^n$ and β_i in $\{0,1\}$
    $A_{i+1} \leftarrow \text{concat}(A_i, C_i)$, $b_{i+1} \leftarrow \text{concat}(b_i, \beta_i)$
    Let m_i^t be $\max_\sigma(w(\sigma) : A_{i+1}\sigma \oplus b_{i+1} = 0)$
    $\widehat{m}_i^t \leftarrow 2^\kappa$ where $\kappa = \max(k : 2^k \leq m_i^t)$
  end for
end for
$\forall i$, $\widehat{q}_i \leftarrow \text{Median}(\widehat{m}_i^1, \cdots, \widehat{m}_i^T)$
return $\sqrt{2}\left(\widehat{q}_0 + \sum_{i=0}^{n-1} \widehat{q}_{i+1} 2^i\right)$

WISH's main task is to estimate the quantiles weights q_i. The general idea is to use hash functions projecting the truth assignments into 2^i buckets and to take the heaviest assignment in a random bucket. Hash functions are built adding xor constraints incrementally: $\text{concat}(A, C)$ adds the line C to the matrix of constraints A. By uniformity property, an arbitrary bucket contains in expectation 2^{n-i} truth assignments after i constraints. Since there are roughly 2^{i-1} assignments heavier than q_{i-1}, the expected amount mapped to the chosen bucket should be close to zero. So hopefully the heaviest weight of the bucket (noted m_i^t for the t-th run) is in $[q_i, q_{i-1}]$, and it is chosen as candidate for the estimate $\widehat{q}_i$. The following lemma gives guarantees on the range of the heaviest weight of a bucket (proof is deferred to appendix).

Lemma 1. *For all t in $[\![1, T]\!]$ and all i in $[\![1, n]\!]$, there is*

$$Pr\left[m_i^t \geq q_i\right] \geq \left(\frac{3}{4}\right)^2 \text{ and } Pr\left[m_i^t \leq q_{i-1}\right] \geq \left(\frac{3}{4}\right)^2$$

Iterating this process T times and taking the median candidate amplifies the confidence of the estimation.

Lemma 2. *For $i > 0$, let $\widehat{q}_i$ be the median of $m_i^1, \cdots, m_i^T$ resulting from the T independent iterations. And let I_i be the interval $[q_i, q_{i-1}]$. There is:*

$$\Pr\left[\widehat{q}_i \in I_i\right] \geq 1 - 2\exp\left(-\frac{T}{\alpha}\right)$$

where $\alpha = 2^7 = 128$.

In the algorithm, $\widehat{q}_i$ is actually not the median of $m_i^1, \cdots, m_i^T$ but the median of their 2-approximations $\widehat{m}_i^1, \cdots, \widehat{m}_i^T$. With this modification, the statement of the lemma holds for $I_i = [q_i/2, q_{i-1}]$. An estimate of the integral is finally generated using the formula for W_1 and replacing the q_i by their estimates $\widehat{q}_i$.

We make several contributions to WISH_EGSS in WISH. We first reduce the MPE queries employed in WISH_EGSS to find the weights m_i^t to binary searches using the NP oracle queries. Weights are written with p bits so finding m_i^t takes $\mathcal{O}(p)$ oracle queries. However we prefer the approximate variant in which the binary search explores $[\![0, p]\!]$ to find $\kappa = \lfloor \log(m_i^t) \rfloor$ and returns 2^κ. This variant reduces the cost to $\mathcal{O}(\log(p))$ queries while approximating m_i^t within a factor of 2. A second contribution is the usage of dependence among different hash functions: hash functions of $i+1$ constraints are no longer sampled independently but built upon hash functions of i constraints (hence the concat$(\cdot,\cdot)$ function). Our last contribution is the significant improvement of WISH_EGSS's guarantees: the original analysis used only the pairwise independence of hash functions, but we use 3-wise independence to obtain the improved lemma 1. This lemma ultimately allows us to prove that WISH approximates $\rho(F)$ within a factor of 8, while the initial factor was 256. The reduce factor is still quite large but greatly accelerates the amplification process described in [8].

Theorem 1. *For any $\delta > 0$, $\mathsf{WISH}(F, \rho, \delta)$ makes $\mathcal{O}\left(n \log(p) \log(n/\delta)\right)$ calls to NP oracle and returns an approximation of $\rho(F)$ within $\left[\rho(F)/(2\sqrt{2}), 2\sqrt{2}\rho(F)\right]$ with probability at least $1 - \delta$.*

4.3 Discrete integration by counting: SWITCH

We now describe an algorithm for discrete integration that utilizes the reduction to counting sub-problems. We call the algorithm SWITCH (Sum of Weights and Integral via Threshold Counting and Hashing). SWITCH takes in a formula F, a weight function ρ, and a confidence parameter δ, and returns an estimate

for W_2. SWITCH's main task is to estimate the tails $\tau_i = \tau(2^{-i})$. The core idea is to view tails as cardinals of some subsets of witnesses of F and use hashing to estimate these cardinalities. The approximation method is very similar to previous hashing-based techniques [16,5]. For a given subset of size τ_i, we successively apply constraints until its projection on an arbitrary bucket is empty. Each new randomly sampled constraint halves the remaining set in expectation, so the number of constraints necessary to reach the empty set can be viewed as a good approximation of $l_i = \log(\tau_i)$ and its power of 2 approaches τ_i.

Algorithm 2 SWITCH(F, ρ, δ)

1: $T \leftarrow \lceil 128 \ln(4p/\delta) \rceil$
2: **for all** $1 \leq t \leq T$ **do**
3: Sample A in $\{0,1\}^{n \times n}$ and b in $\{0,1\}^n$
4: **for all** $0 \leq i \leq p$ **do**
5: $\widehat{l}_i^t \leftarrow \max \left\{ k \mid \exists\, \sigma \text{ such that } \sigma \models F, \rho(\sigma) \geq 2^{-i} \text{ and } A_k \sigma \oplus b_k = 0 \right\}$ where $A_k \leftarrow A[1..k]$ and $b_k \leftarrow b[1..k]$
6: **end for**
7: **end for**
8: $\forall i$, $\widehat{l}_i \leftarrow \text{Median}(\widehat{l}_i^1, \cdots, \widehat{l}_i^T)$, $\widehat{\tau}_i = 2^{\widehat{l}_i}$ (handle cases $\tau_i = 0$ and $\tau_i = 1$ exactly)
9: **return** $\sqrt{2}\left(\widehat{\tau}_p 2^{-p} + \sum_{i=0}^{p-1} \widehat{\tau}_i 2^{-i-1}\right)$

Lemma 3. *If $l_i > 0$ ($\tau_i > 1$), then there is for all t in $[\![1, T]\!]$:*

$$\Pr\left[\widehat{l}_i^t \leq \lceil l_i \rceil\right] \geq \left(\frac{3}{4}\right)^2 \quad \textit{and} \quad \Pr\left[\widehat{l}_i^t \geq \lfloor l_i \rfloor\right] \geq \left(\frac{3}{4}\right)^2$$

One may note that τ_i is not necessarily a power of 2, so our method of approximating logarithms by integers is imprecise and the estimation error is amplified as a power of 2. Furthermore, there are two cases not handled by the lemma

- The case $\tau_i = 0$ ($l_i = -\infty$): there are no witness of F of weight greater than 2^{-i}. One call to the NP oracle is enough to spot this case.
- The case $\tau_i = 1$ ($l_i = 0$): a set of 1 element stays intact after 1 constraint with probability $1/2$, so we overestimate is size with probability $1/2$. This case is spotted with two NP oracle queries (adding a block clause before the second query).

Assuming we are not in any such cases, we amplify the confidence on the estimates of l_i repeating the process T times and choosing the median candidate.

Lemma 4. *Let $\widehat{l}_i$ be the median of the T independent $\widehat{l}_i^1, \cdots, \widehat{l}_i^T$. Assume $\tau_i > 1$ and let J_i be the interval $[\lfloor \log(\tau_i) \rfloor, \lceil \log(\tau_i) \rceil]$. There is:*

$$\Pr\left[\widehat{l}_i \in J_i\right] \geq 1 - 2\exp\left(-\frac{T}{\alpha}\right)$$

where $\alpha = 2^7 = 128$. *Therefore* $\widehat{\tau_i} = 2^{\widehat{l_i}}$ *is an estimate of* τ_i *that lies in* $\left[\frac{\tau_i}{2}, 2\tau_i\right]$ *with same probability.*

The tails estimates are finally used to compute an estimate of W_2.

The NP oracle is called to check if a set is empty after application of constraints. When approximating l_i line 5, the constraints are taken from the same set of n constraints stored in A and b (A_k and b_k representing the first k lines of A and b). Consequently, if there are witnesses satisfying $A_j\sigma \oplus b_j = 0$ for some j, they also satisfy $A_i\sigma \oplus b_i = 0$ for all $i \leq j$. Similarly if no witness satisfy $A_j\sigma \oplus b_j = 0$, none satisfy $A_i\sigma \oplus b_i = 0$ for $i \geq j$. So to find how many constraints are enough to empty the set of witnesses of F heavier than 2^{-i}, one can proceed by binary search in $[\![0, n]\!]$. Following this idea, the procedure line 5 makes $\mathcal{O}(\log(n))$ calls to the NP oracle.

Theorem 2. *For any* $\delta > 0$, $\mathsf{SWITCH}(F, \rho, \delta)$ *makes* $\mathcal{O}(p \log(n) \log(p/\delta))$ *calls to NP oracle and returns a approximation of* $\rho(F)$ *within* $\left[\rho(F)/(2\sqrt{2}), 2\sqrt{2}\rho(F)\right]$ *with probability at least* $1 - \delta$.

Theorems 1 and 2 show that WISH and SWITCH approximate the discrete integral within a factor of 8 but have dual complexities, in the sense that there is a function f, such that WISH makes $\mathcal{O}(f(n, p, \delta))$ NP oracle calls against $\mathcal{O}(f(p, n, \delta))$ calls for SWITCH. We have found this function to be $f(n, p, \delta) = n \log(p) \log(n/\delta)$. Furthermore, the analysis shows that the constants hidden by the $\mathcal{O}$ notation are of same order of magnitude. So depending on the value of n and p, one may prefer one algorithm to the other.

4.4 On the extension to arbitrary precision algorithms

In the discussion on the limitation of the estimates section 3.4, we pointed out that generating approximations of W_1 or W_2, one could not hope for better than approximations of the discrete integral within a factor 2. This capped approximation factor comes from doing base 2 partitions of the integration axis when defining the quantiles weights q_i and the splitting tails τ_i. We explained that using base β (β in $]0, 1[$) partitions, one could easily find estimates of arbitrarily close approximations.

Typically WISH and SWITCH should be adapted so as to ensure arbitrarily close approximations of the quantiles weights and splitting tails defined from base β partitions. Both algorithm rely on hash functions which particularity is to halve cardinality with each new constraint added. For WISH, such hash functions are fitted when computing base 2 quantile weights, because the tails are also halved from one quantile to the next. But for base β quantile weights, we have yet to find how to adapt the algorithm. Another alternative would be to use Stockmeyer's trick for converting an algorithm $\mathcal{A}$ returning constant factor approximation into arbitrary precision algorithm by invoking $\mathcal{A}$ on multiple copies of F [1,8].

For SWITCH, there exists hashing-base algorithms for approximate model counting which return (ε, δ)-approximations [5,6,15]. SWITCH can be adapted taking inspiration from these algorithms so as to generate arbitrarily close approximations of all base β splitting tails.

5 Conclusion

In this paper, we provide a framework for developing algorithms for approximate discrete integration. In this framework, discrete integrals are transformed into integrals of non-increasing real functions which are subsequently approximated using classical methods. We build two algorithms from this framework: we demonstrate how transformations over the discrete integral give rise to two different algorithmic approaches. One approach, WISH, relies on usage of optimization queries while the other, SWITCH, reduces the problem of discrete integration to that of several unweighted counting problems. The analysis that lead to these two reductions were shown to be very alike, in that they follow similar steps in the transformation of the discrete integral. The similarity extends to the algorithms as we have shown that SWITCH makes $\mathcal{O}(p \log(n) \log(p/\delta))$ calls to NP oracle in contrast to $\mathcal{O}(n \log(p) \log(n/\delta))$ calls in the context of WISH, so that the two complexities are dual on n and p. This result provides insight on deciding which approach to use depending on the context, as the approach expected to do fewer oracle queries depends on n and p.

It would be of interest to understand empirical performance comparison of WISH and SWITCH and we hope that the aforementioned algorithmic approaches will motivate practitioners to develop the underlying required solvers: (i) SAT solvers capable of handling XOR and PB constraints, and (ii) MaxSAT solvers capable of handling XOR constraints. The current MaxSAT solvers and the CNF-PB solvers handle these XOR constraints blasting them into CNF after performing top-level Gaussian elimination. The recent success of BIRD framework owing to a tighter integration of CNF and XOR solving for CNF-XOR formulas motivates the tighter integration of (i) XOR and PB constraints , and (ii) MaxSAT solving with XOR constraints [15].

Acknowledgements This research has been supported in part by the National Research Foundation Singapore under its AI Singapore Programme [R-252-000-A16-490] and the NUS ODPRT Grant [R-252-000-685-133].

References

1. Bellare, M., Petrank, E.: Making zero-knowledge provers efficient. In: Proceedings of the 24th Annual Symposium on the Theory of Computing, ACM. Citeseer (1992)
2. Brooks, S., Gelman, A., Jones, G., Meng, X.L.: Handbook of markov chain monte carlo. Chapman & Hall/CRC (2011)
3. Carter, J.L., Wegman, M.N.: Universal classes of hash functions. Journal of Computer and System Sciences (1977)
4. Chakraborty, S., Fremont, D.J., Meel, K.S., Seshia, S.A., Vardi, M.Y.: Distribution-aware sampling and weighted model counting for sat. In: Proc. of AAAI. pp. 1722–1730 (2014)
5. Chakraborty, S., Meel, K.S., Vardi, M.Y.: A scalable approximate model counter. In: Proc. of CP. pp. 200–216 (2013)
6. Chakraborty, S., Meel, K.S., Vardi, M.Y.: Algorithmic improvements in approximate counting for probabilistic inference: From linear to logarithmic SAT calls. In: Proc. of IJCAI (2016)
7. Ermon, S., Gomes, C., Sabharwal, A., Selman, B.: Embed and project: Discrete sampling with universal hashing. In: Proc. of NIPS. pp. 2085–2093 (2013)
8. Ermon, S., Gomes, C., Sabharwal, A., Selman, B.: Taming the curse of dimensionality: Discrete integration by hashing and optimization. In: Proc. of ICML. pp. 334–342 (2013)
9. Gogate, V., Dechter, R.: Approximate counting by sampling the backtrack-free search space. In: Proc. of the AAAI. vol. 22, p. 198 (2007)
10. Gomes, C., Sabharwal, A., Selman, B.: Near-uniform sampling of combinatorial spaces using xor constraints. In: Proc. of NIPS. pp. 481–488 (2006)
11. Jerrum, M.R., Sinclair, A.: The Markov chain Monte Carlo method: an approach to approximate counting and integration. Approximation algorithms for NP-hard problems pp. 482–520 (1996)
12. Kitchen, N., Kuehlmann, A.: Stimulus generation for constrained random simulation. In: Proc. of ICCAD. pp. 258–265 (2007)
13. Paredes, R., Duenas-Osorio, L., Meel, K.S., Vardi, M.Y.: Network reliability estimation in theory and practice. Reliability Engineering and System Safety (2018)
14. Roth, D.: On the hardness of approximate reasoning. Artificial Intelligence (1996)
15. Soos, M., Meel, K.S.: Bird: Engineering an efficient cnf-xor sat solver and its applications to approximate model counting. In: Proceedings of AAAI Conference on Artificial Intelligence (AAAI)(2019) (2019)
16. Stockmeyer, L.: The complexity of approximate counting. In: Proc. of STOC. pp. 118–126 (1983)
17. Tzikas, D.G., Likas, A.C., Galatsanos, N.P.: The variational approximation for Bayesian inference. In: IEEE Signal Processing Magazine. pp. 131–146 (Nov 2008)
18. Valiant, L.G.: The complexity of computing the permanent. Theoretical Computer Science **8**, 189–201 (1977)
19. Wainwright, M.J., Jordan, M.I.: Graphical models, exponential families, and variational inference. Found. Trends Machine Learning **1**(1-2), 1–305 (2008)