The Cost of False Alarms in Hellman and Rainbow Tradeoffs

Jin Hong

Abstract Cryptanalytic time memory tradeoff algorithms are generic one-way function inversion techniques that utilize pre-computation. Even though the online time complexity is known up to a small multiplicative factor for any tradeoff algorithm, false alarms pose a major obstacle in its accurate assessment.

In this work, we study the expected pre-image size for an iteration of functions and use the result to analyze the cost incurred by false alarms. We are able to present the expected online time complexities for the Hellman tradeoff and the rainbow table method in a manner that takes false alarms into account. We also analyze the effects of the checkpoint method in reducing false alarm costs.

The ability to accurately compute the online time complexities will allow one to choose their tradeoff parameters more optimally, before starting the expensive pre-computation process.

Keywords time memory tradeoff, Hellman, rainbow table, checkpoint

Mathematics Subject Classification (2000) 94A60 · 68W40 · 68Q25

1 Introduction

Cryptanalytic time memory tradeoffs were first introduced by Hellman [11] as an attack on blockciphers. Since then, much progress has been made, with [1, 3–8, 10, 12, 13, 16] being a very partial list of works contributing to the theoretic aspects of the tradeoff algorithms. There are also numerous works on their applications and implementations. Today, the tradeoff algorithms are understood as techniques for quickly inverting generic one-way functions, and since much of cryptanalysis can be expressed as the process of inverting an appropriate one-way function, time memory tradeoff technique is a valuable tool for cryptography.

Jin Hong
Department of Mathematical Sciences and ISaC, Seoul National University, Seoul 151-747, Korea
E-mail: jinhong@snu.ac.kr

Any cryptanalytic tradeoff algorithm works in two stages. In the pre-computation phase, the one-way function is iteratively computed and a digest of the computation is stored in tables of total size smaller than the complete dictionary. When a specific image point to be inverted is given, the online phase algorithm returns a pre-image of the target in time shorter than what is taken by an exhaustive search.

An analysis of the storage and online time complexities is usually provided with any published tradeoff algorithm. For example, the tradeoff curve $TM^2 = N^2$ gives the complexity of the original Hellman tradeoff, where N is the size of space on which the one-way function is defined. That is, for any T and M satisfying the tradeoff curve, the Hellman tradeoff algorithm, when run with appropriate parameters, enables one to finds a pre-image of the inversion target in online time T, utilizing a pre-computed table of size M.

Tradeoff curves for various tradeoff algorithms are typically true only up to a small multiplicative factor. This is largely due to the effects of what are called false alarms, which make it difficult to give an exact expression for the time complexity T. The online phase is an iterative process that is occasionally interrupted when a certain collision is found. Sometimes the collision leads to the target inversion, but this does not happen in most of the cases. Such a non-useful interruption is said to be a false alarm and resolving whether the collision is useful or a false alarm requires time.

Most previous analyses of the online time complexities concentrated on the main iterative process and the added cost of dealing with false alarms was either neglected or roughly argued as being relatively small. For example, the original work by Hellman gave a heuristic argument that false alarms can increase the online computation by 50% at the most. On the other hand, the paper presenting the rainbow table method [16], which is the tradeoff method most widely known to the public, cites experiment results of observing 75% of cryptanalytic effort spent on false alarms. There, concerning false alarms, only a heuristic comparison to the distinguished point tradeoff method is given. Thus, it is apparent that the extra cost incurred by false alarms is not well understood, and this contributes a certain degree of uncertainty to the online time complexity.

Obtaining an accurate assessment of the online time complexity is meaningful for at least two reasons. The first concerns parameter selection for tradeoff implementation. Running time of the online phase cannot be measured through tests until a tradeoff table is ready, and since table creation is a time consuming process, experimenting with various parameters and tweaking them for maximum performance is very uncomfortable, to say the least. This is especially true with the rainbow table method, where the huge table cannot be broken down into smaller pieces for testing. One cannot measure the online time complexity with fake tables either, because these show different false alarm behavior. Thus, to find implementation parameters that would optimally obtain the online performance one needs, without going through the expensive trial and error process, one needs to be able to compute and predict the online complexity corresponding to a given set of parameters.

The second need for understanding the online complexities concerns comparisons between tradeoff algorithms. To choose the tradeoff algorithm that is optimal for the situation in hand and the available resources, one needs to be able to compare tradeoff algorithms. This is not a straightforward task with our current knowledge of the online complexity. For example, the tradeoff curve for the rainbow table method [16] is $TM^2 = \frac{1}{2}N^2$ and this was argued as evidence of its superiority by a factor of two over the Hellman tradeoff, whose tradeoff curve is $TM^2 = N^2$. But the later work [4] asserted otherwise, mentioning differences in applicable storage reduction techniques. So, even though the behaviors of various tradeoff algorithms are known in terms of rough tradeoff curves, as the performance of these tradeoff algorithms differ mostly by a small factor, a fair comparison between algorithm performances is not an easy task. The apparent difficulty is that of not knowing what the optimal storage technique is for each tradeoff algorithm, but analysis into such matters quickly reveals that this is intimately related to the online time complexity. Aggressive storage reduction through ending point truncation results in increased online time, in a manner that resembles the workings of the false alarms.

In this work, we give a more accurate assessment of the online time complexity, taking the effects of false alarms into account. The original Hellman tradeoff and both the perfect and non-perfect versions of the rainbow table method are considered. We do not require the rainbow tables to be maximal. A corresponding analysis under the application of the checkpoint technique [1], that reduces the effects of false alarms, is also given.

To the best of our knowledge, the only previous attempt at a rigorous analysis of false alarms appears in [1]. That work presents results on the very special case of maximal perfect rainbow tables, which is difficult to use in practice, due to its high pre-computation cost. The current work will provide a more realistic view of the cost incurred by false alarms and the resulting total online time complexity.

The rest of the paper is organized as follows. We start with a very brief review of the main concepts surrounding time memory tradeoffs in the next section. In Section 3, we consider an image set corresponding to a random input set under an iteration of functions and study its expected pre-image size. These results are used to analyze the cost of false alarms in Section 4. This is the main contribution of this paper. Section 5 deals with the checkpoint technique. Test results that support our theory are provided in Section 6. The final section summarizes this paper and provides comments on possible future developments of this work.

2 Time memory tradeoff algorithms

Let us start by making the inversion problem more explicit. A one-way function $F : \mathcal{N} \rightarrow \mathcal{N}$ acting on a finite set $\mathcal{N}$ is fixed. Then, one is given a target image point $F(\mathbf{x}) \in \mathcal{N}$ and asked to find $\mathbf{x}$. There are also situations where one is given an image point $\mathbf{y} \in \mathrm{Im}(F)$ and asked to find an $\mathbf{x} \in \mathcal{N}$ such that $F(\mathbf{x}) = \mathbf{y}$. The difference between the two problems, which has mostly been ignored in the time memory tradeoff literature, is whether the objective is to obtain *the* solution or *any* solution to the inversion problem. For example, consider the problem of inverting the password hash of a user on a specific computer system. Obtaining *any* pre-image would suffice in breaking into the user's account on the system, but obtaining *the* password would allow breakage into other systems on which the same user is suspected of using the same password. In this paper, we focus on the *the* version of the inversion problem, but the *any* version can also be approached in a similar manner.

We shall not explain the explicit tradeoff algorithms and ask readers to refer to the original papers on the Hellman tradeoff [11] and the rainbow table method [16]. Here, we only list and fix the basic terminologies. Notation of this section will be used throughout the paper and, from now on, we shall refer to the rainbow table method simply as the *rainbow tradeoff*.

A *Hellman chain*, for a one-way function $F : \mathscr{N} \to \mathscr{N}$, is of the form

$$\mathbf{sp}_i = \mathbf{x}_{i,0} \xrightarrow{F} \mathbf{x}_{i,1} \xrightarrow{F} \cdots \xrightarrow{F} \mathbf{x}_{i,t} = \mathbf{ep}_i \quad (1 \le i \le m). \tag{1}$$

The *Hellman table* $\{(\mathbf{sp}_i, \mathbf{ep}_i)\}_{i=1}^{m}$ is sorted on the *ending points* $\mathbf{ep}_i$, or a hash table technique is used, to make table lookups easier. We have omitted the *reduction functions*, as we shall mostly deal with a single Hellman table. The complete set of m chains, consisting of m *rows* and $t+1$ *columns*, is a *Hellman matrix*. The positive integer parameters m and t are chosen to satisfy $mt^2 = c_{\mathtt{H}}|\mathscr{N}|$, with positive constant $c_{\mathtt{H}}$ neither very large nor too close to zero. The *matrix stopping rule* refers to the $c_{\mathtt{H}} = 1$ case, which is what is almost always used.

Likewise, we have the notions of a *rainbow chain*

$$\mathbf{sp}_i = \mathbf{x}_{i,0} \xrightarrow{F_1} \mathbf{x}_{i,1} \xrightarrow{F_2} \cdots \xrightarrow{F_t} \mathbf{x}_{i,t} = \mathbf{ep}_i \quad (1 \le i \le m), \tag{2}$$

a *rainbow table*, and a *rainbow matrix*. For rainbow tradeoffs, it is usual to take $mt = c_{\mathtt{R}}|\mathscr{N}|$ with constant $c_{\mathtt{R}}$, once again, close to 1. A rainbow table is *perfect* if all its ending points are distinct. A *maximal* perfect rainbow table is one containing the maximal number of possible rows. A rainbow tradeoff using perfect rainbow tables will be referred to as a *perfect rainbow tradeoff*.

Our *inversion target* will always be $\mathbf{y} = F(\mathbf{x})$ throughout the paper. If the current end $F^k(\mathbf{y})$ of the Hellman tradeoff's *online chain*

$$\mathbf{y} \xrightarrow{F} F(\mathbf{y}) \xrightarrow{F} F^2(\mathbf{y}) \xrightarrow{F} \cdots \xrightarrow{F} F^k(\mathbf{y})$$

of length k matches an ending point $\mathbf{ep}_i$, we have an *alarm*. If an alarm involving $\mathbf{ep}_i$ shows the property $F^{t-k-1}(\mathbf{sp}_i) \neq \mathbf{x}$, it is said to be a *false alarm*. Notice that $F^{t-k-1}(\mathbf{sp}_i) = \mathbf{x}$ is not guaranteed by the weaker condition $F^{t-k}(\mathbf{sp}_i) = \mathbf{y}$. The weaker condition is checked first and the stronger condition is checked outside the tradeoff algorithm, only when the weaker condition is satisfied. There is a corresponding notion of false alarms for rainbow tradeoffs. False alarms are caused by *merges* between the online chain and a *pre-computed chain*.

Checkpoint [1] is a technique for resolving false alarms without regenerating the pre-computed chain, applicable to both Hellman and rainbow tradeoffs. The idea is to choose a fixed, say k-th, column of the pre-computation matrix and supplement a 1-bit information about $\mathbf{x}_{i,k}$, for each i, in the pre-computed table. When an alarm sounds during the online phase, the corresponding information is computed for the online chain. If the online chain and the pre-computed chain had merged somewhere between the checkpoint and the ending point, there is a possibility that a comparison of checkpoint information will filter out the false alarm.

Let us briefly return to the *the* versus *any* pre-image versions of the inversion problem. Given a pre-computed tradeoff table and the inversion target $\mathbf{y} = F(\mathbf{x})$, the former problem of looking for $\mathbf{x}$ will end successfully if and only if $\mathbf{x}$ appears among the pre-computation matrix entries, excluding the ending points. The latter problem is solved if and only if $\mathbf{y}$ is found among matrix entries excluding the starting points. This difference needs to be kept in mind for any rigorous analysis of tradeoff algorithms.

Throughout this paper, the one-way function $F : \mathscr{N} \to \mathscr{N}$ being considered will always act on a set $\mathscr{N}$ of size N. We always assume N to be large. A Hellman matrix is denoted by $\mathtt{HM}$ and a rainbow matrix by $\mathtt{RM}$. Any Hellman matrix or rainbow matrix will consist of m rows and $t+1$ columns. The parameters m and t are to satisfy an appropriate $mt^2 = c_{\mathtt{H}} N$ or $mt = c_{\mathtt{R}} N$ and are assumed to be reasonable in the sense that $1 \ll m, t \ll N$. The k-th column $\{\mathbf{x}_{i,k}\}_{i=1}^{m}$ of the pre-computation matrix will be written as $\mathtt{HM}_k$ or $\mathtt{RM}_k$.

3 Pre-image under function iteration

In this section, we present information concerning the size of a pre-image set under an iteration of functions. Throughout this section $F : \mathscr{N} \to \mathscr{N}$ will be a random function.

3.1 Preliminary definitions and notation

We denote the u-times iteration of F by $F^u = F \circ \cdots \circ F$. It is well known [9, 15] that if m_0-many distinct random inputs are subject to F^u, the expected image size m_u can be computed through the recursion

$$\frac{m_{k+1}}{N} = 1 - \exp\Bigl(-\frac{m_k}{N}\Bigr) \qquad (k = 0, 1, \ldots, u-1). \tag{3}$$

For example, setting $m_0 = N$, one can conclude that $\frac{m_1}{N} = 1 - \frac{1}{e}$ of the total space is expected to belong to the image space of F. We will use the closed form approximation

$$\frac{m_k}{N} \approx \frac{1}{N/m_0 + k/2}, \tag{4}$$

which can be found[1] in [2].

For a Hellman table with $m_0 t^2 = c_{\mathtt{H}} N$, the above approximation shows

$$m_t \approx \frac{N}{N/m_0 + t/2} = \frac{m_0}{1 + c_{\mathtt{H}}/2t} = m_0 \Bigl\{ 1 - \frac{c_{\mathtt{H}}}{2t} + \bigl(\frac{c_{\mathtt{H}}}{2t}\bigr)^2 - \cdots \Bigr\} \approx m_0. \tag{5}$$

This implies that the number of collisions among ending points is very small compared to the total number of rows. Thus the behavior of the Hellman tradeoff will depend very little on whether or not we replace the merging chains with new chains.

[1] The statement in the referenced paper is different, but this it due to multiple typographic errors. The version presented here can easily be obtained by following through their proofs.

Hence, in our further discussions of the Hellman tradeoff, we shall assume that the Hellman matrix was created with $m_0 = m$ distinct starting points and that the resulting $m_t = m$ ending points are distinct. This insures, in particular, that all points within each column $\mathtt{HM}_k$ are distinct.

An i-node for a mapping is an element of the range space with exactly i-many pre-images. Still referring to the random function $F : \mathscr{N} \to \mathscr{N}$, it is known [9] that the ratio of i-nodes among $\mathscr{N}$ is expected to be

$$p_i = \frac{1}{e} \cdot \frac{1}{i!}. \tag{6}$$

Notice that $p_0 = \frac{1}{e}$, the ratio of nodes with no pre-images, is in good agreement with our previous figure of $1 - \frac{1}{e}$ for the total image space ratio.

The i-node ratio expectation (6) is only guaranteed to be a good approximation only when the set size N is large compared to i. In fact, there cannot be any i-nodes for $i > N$, so that these p_i should be zero. But since $\frac{1}{i!}$ is very small at these large i, we shall use p_i as given by (6), for all $i \in \mathbf{Z}_{\geq 0}$. Similarly, our various future definitions will be vacuous for large i, and arguments not strictly correct, but we shall ignore them, and they will not cause any trouble in the end result.

For each non-negative integers i and k, let us write

$$\mathscr{R}_{k,i}(F) = \big\{\mathbf{y} \in \mathscr{N} \mid \mathbf{y} \text{ is an } i\text{-node under } F^k\big\}, \tag{7}$$

$$\mathscr{D}_{k,i}(F) = \big\{\mathbf{x} \in \mathscr{N} \mid F^k(\mathbf{x}) \in \mathscr{R}_{k,i}(F)\big\} \tag{8}$$

to denote the set of i-nodes and pre-images of i-nodes, associated to the mapping F^k. The characters $\mathscr{D}$ and $\mathscr{R}$ reflect the role of $\mathscr{N}$ as domain and range, respectively. The dependence of these two sets on F will usually be omitted.

An elements of $\mathscr{R}_{k,i}(F)$ will be referred to as an (F^k, i)-node and we shall say that an element of $\mathscr{D}_{k,i}(F)$ produces an (F^k, i)-node. The probability of a random point from the range space $\mathscr{R} = \mathscr{N}$ to be an (F^k, i)-node is denoted by

$$p_{k,i} = \frac{|\mathscr{R}_{k,i}|}{N}. \tag{9}$$

Note that the values of $p_{1,i}$ has already been stated as p_i in (6). Since every point of $\mathscr{R}_{k,i}$ has i-many pre-images under F^k, the probability of a random point from the domain $\mathscr{D} = \mathscr{N}$ to produce an (F^k, i)-node is $i\, p_{k,i}$.

A more rigorous definition for the $p_{k,i}$ would express this as an average of $\frac{|\mathscr{R}_{k,i}(F)|}{N}$ over all functions $F : \mathscr{N} \to \mathscr{N}$, but we refrained from going into this complicated expression and simply took F to be a random function, at the beginning of this section. Many of our future statements presenting explicit values should be understood in the same context. They are averages over all function $F : \mathscr{N} \to \mathscr{N}$, or equivalently, what we can expected from a random function, rather than for any fixed function.

For each non-negative integer k, we fix a notation

$$\mathscr{P}_k(x) = \sum_{i=0}^{\infty} p_{k,i}\, x^i,$$

for the formal power series relating to (F^k, i)-node ratios.

Lemma 1 *We have* $\mathscr{P}_0(x) = x$ *and* $\mathscr{P}_1(x) = \mathrm{Exp}(x-1)$, *where* $\mathrm{Exp}(x)$ *is the formal power series* $\sum_{i=0}^{\infty} \frac{1}{i!}x^i$.

Proof As F^0 is the identity map, we have $p_{0,1} = 1$ and $p_{0,i} = 0$ for all other i. This implies the first statement. The second statement follows from the value of $p_{1,i}$ as given by (6) and the identity $\frac{1}{e}\,\mathrm{Exp}(x) = \mathrm{Exp}(x-1)$ of formal power series. $\square$

3.2 Node size correlation under function iteration

The generating function $\mathscr{P}_k(x)$ was defined to be a formal power series. As we know that each $p_{k,i}$, being a probability value, is bounded, it is clear that $\mathscr{P}_k(x)$ converges for any $|x| < 1$. Before showing that $\mathscr{P}_k(x)$ is convergent for all real number inputs x, we shall present a certain conditional probability.

Proposition 1 *The probability for a random* (F^v, j)*-node to be an* (F^{u+v}, i)*-node is approximately*

$$p\big(\overset{u}{\to}\overset{v}{\to}_i \,\big|\cdot \quad \overset{v}{\to}_j *\big) \;=\; \sum_{i_1+\cdots+i_j=i} p_{u,i_1}\, p_{u,i_2,} \cdots p_{u,i_j}.$$

Here, the summation indices $i_1, \ldots, i_j$ *are to run over non-negative integers. When* $j = i = 0$, *the degenerate sum is interpreted to be* 1. *When* $j = 0$ *with* $i > 0$, *the empty sum is interpreted to be zero.*

Proof Fix any single j-node $\mathbf{z}$ for F^v and consider its j-many pre-images, which we name $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_j$. Suppose each $\mathbf{y}_k$ is an (F^u, i_k)-node. Then $\mathbf{z}$ is an i-node for F^{u+v} if and only if $i_1 + \cdots + i_j = i$. When $i \ll N$, the events of each $\mathbf{y}_k$ being an (F^u, i_k)-node may be seen as being independent of each other, and we may write $p_{u,i_1} p_{u,i_2,} \cdots p_{u,i_j}$ as an approximation for the probability of the product event. For large i, both sides of the equation are very small and these will not cause trouble later. By summing the probability for these product events over all j-many indices that add up to i, we obtain the full conditional probability. $\square$

By the law of alternatives, it is clear that

$$p_{u+v,i} = \sum_{j=0}^{\infty} p\big(\overset{u}{\to}\overset{v}{\to}_i \,\big|\cdot \quad \overset{v}{\to}_j *\big) \cdot p_{v,j}. \tag{10}$$

The next result, relying heavily on this observation, allows us to compute the formal power series $\mathscr{P}_k(x)$ inductively from $\mathscr{P}_1(x)$.

Proposition 2 *The generating functions for node ratios satisfy the recurrence relation*

$$\mathscr{P}_{u+v}(x) = \mathscr{P}_v\big(\mathscr{P}_u(x)\big).$$

Proof Combining the definition of $\mathscr{P}_{u+v}(x)$ with identity (10), one can state that

$$\begin{aligned}
\mathscr{P}_{u+v}(x) &= \sum_i p_{u+v,i}\, x^i \\
&= \sum_i \Big\{ \sum_j p\big(\xrightarrow{u}\xrightarrow{v}{}_i \,\big|\, \cdot \quad \xrightarrow{v}{}_j *\big) \cdot p_{v,j} \Big\}\, x^i \\
&= \sum_j p_{v,j} \Big\{ \sum_i p\big(\xrightarrow{u}\xrightarrow{v}{}_i \,\big|\, \cdot \quad \xrightarrow{v}{}_j *\big) x^i \Big\}.
\end{aligned}$$

Here, the last expression is an infinite sum of formal power series, which is not defined in general, but the condition $\sum_j p_{v,j} = 1$ justifies its use.

Now, notice that the coefficient of x^i in $\mathscr{P}_u(x)^j$ is exactly the right hand side of the equation appearing in Proposition 1, so that the above is equal to

$$\sum_j p_{v,j}\, \mathscr{P}_u(x)^j = \mathscr{P}_v\big(\mathscr{P}_u(x)\big).$$

This completes the proof. $\square$

During the proof of this theorem, we referred to the coefficient of a certain power series. As this notion will appear frequently in this paper, we fix a notation for this. Given any power series $\mathscr{A}(x) = \sum_i a_i x^i$, we shall write $\lceil \mathscr{A}(x) \rfloor_i$ for a_i, the coefficient of x^i in $\mathscr{A}(x)$.

The above theorem, together with Lemma 1, shows that $\mathscr{P}_k(x)$ is equal to a k-times iterated composition of $\mathscr{P}_1(x) = \mathrm{Exp}(x-1)$. Since the formal power series $\mathrm{Exp}(x-1)$ converges to the real number value $\exp(x-1)$ for any real number x, we know $\mathscr{P}_k(x)$ is convergent for any real number x and non-negative integer k. This allows us to view $\mathscr{P}_k$ as a function defined on the set of all real numbers.

It is clear that each function $\mathscr{P}_k$ is injective. Hence, they have corresponding inverse functions, defined on suitable subsets of the real field. Since $\mathscr{P}_0$ is the identity function, in view of the relation $\mathscr{P}_{u+v}(x) = \mathscr{P}_v\big(\mathscr{P}_u(x)\big)$, we shall write $\mathscr{P}_{-k}$ for the inverse function of $\mathscr{P}_k$.

The following lemma is an easy corollary to the above proposition and shows how iterated image size is related to the function $\mathscr{P}_k$.

Lemma 2 *Let $D_0 \subset \mathscr{N}$ be a set consisting of m_0 randomly chosen points. Set $D_k = F^k(D_0)$ and let m_k be the expected size of D_k. Then the set sizes are related by*

$$1 - \frac{m_{k+u}}{N} = \mathscr{P}_u\big(1 - \frac{m_k}{N}\big)$$

for all non-negative k and all $u \geq -k$.

Proof Recalling Lemma 1, we can rewrite (3) in the form

$$1 - \frac{m_{k+1}}{N} = \exp\Big(\big(1 - \frac{m_k}{N}\big) - 1\Big) = \mathscr{P}_1\big(1 - \frac{m_k}{N}\big)$$

As Proposition 2 shows that $\mathscr{P}_u$ is equal to u-times iterations of $\mathscr{P}_1$, the result is true for non-negative integers u. The negative case is obtained by applying the inverse function $\mathscr{P}_{-u}$ to both sides of the positive case equation. $\square$

The proof for the next lemma is almost identical to the proof for approximation (4) appearing in [2].

Lemma 3 *The formal power series $\mathscr{P}_k(x)$, seen as a function, can be approximated by*

$$\mathscr{P}_k(x) \approx 1 - \frac{2(1-x)}{2+k(1-x)}.$$

The approximation is also valid for negative integers k.

Proof Let us treat the non-negative k first. By Proposition 2 and Lemma 1, we may write

$$\mathscr{P}_{k+1}(x) = \mathrm{Exp}(\mathscr{P}_k(x)-1) \approx 1 + (\mathscr{P}_k(x)-1) + \frac{1}{2}(\mathscr{P}_k(x)-1)^2 \quad \text{and} \quad \mathscr{P}_0(x) = x.$$

Let us temporarily use the notation $\mathscr{Q}_x(k) = 1 - \mathscr{P}_k(x)$. Notice that we have interchanged the position of k and x so that the new object seems more like a function of k rather than x. We rewrite the above as

$$\mathscr{Q}_x(k+1) - \mathscr{Q}_x(k) \approx -\frac{1}{2}\mathscr{Q}_x(k)^2 \quad \text{with} \quad \mathscr{Q}_x(0) = 1-x. \tag{11}$$

The Euler method states that the sequence in k, defined by the above difference equation, approximates the solution to the differential equation

$$\frac{d}{dk}\mathscr{Q}_x(k) = -\frac{1}{2}\mathscr{Q}_x(k)^2 \quad \text{with} \quad \mathscr{Q}_x(0) = 1-x. \tag{12}$$

We can use this the other way around and state that the solution

$$\mathscr{Q}_k(x) = \frac{1}{1/\mathscr{Q}_x(0) + k/2} \tag{13}$$

to the differential equation (12) approximates the sequence defined through (11). The solution (13) is equivalent to the claimed first statement.

To treat the negative case, note that the function

$$\tilde{\mathscr{P}}_k(x) := 1 - \frac{2(1-x)}{2+k(1-x)}$$

satisfies the relation $\tilde{\mathscr{P}}_{u+v}(x) = \tilde{\mathscr{P}}_v(\tilde{\mathscr{P}}_u(x))$, for any integers u and v. As $\tilde{\mathscr{P}}_0$ is the identity function, this implies that $\tilde{\mathscr{P}}_{-k}$ is a good approximation for the inverse function $\mathscr{P}_{-k}$, where it is defined. This is the second statement of this lemma. □

We have obtained a closed form approximation for function $\mathscr{P}_k(x)$. Even though each i-node ratio $p_{k,i}$ can be explicitly computed from the recursion formula (10) and Proposition 1, the power series $\mathscr{P}_k(x)$ and this lemma secures this information in a more useable state.

Recall from freshman calculus that for any given power series $\mathscr{A}(x)$, within its radius of convergence, the function $\mathscr{A}(x)$ is differentiable and the derivative is given by

$$\mathscr{A}'(x) = \sum_{k=1}^{\infty} k \lceil \mathscr{A}(x) \rfloor_k x^{k-1}, \tag{14}$$

where we are using the notation introduced below Proposition 2. In the other direction, one must also have

$$\sum_{k=1}^{\infty} \frac{1}{k} \big(\beta^k - \alpha^k\big) \lceil \mathscr{A}(x) \rfloor_{k-1} = \int_{\alpha}^{\beta} \mathscr{A}(x)\, dx, \tag{15}$$

when the integration boundaries α and β lie within the radius of convergence.

Note that since $\mathscr{P}_k(x)$ is convergent for all x, we may freely differentiate this function any number of times or integrate over any finite interval. For later use, we state the following two lemmas. Both can be shown true using Proposition 2 and Lemma 1, through induction on k. We can also check the second of these more directly, but approximately, with Lemma 3.

Lemma 4 *We have $\mathscr{P}_k'(x) = \mathscr{P}_1(x)\, \mathscr{P}_2(x) \cdots \mathscr{P}_k(x)$, for all k.*

Lemma 5 *We have $\mathscr{P}_k(1) = 1$, $\mathscr{P}_k'(1) = 1$, and $\mathscr{P}_k''(1) = k$, for all k.*

We now state conditional probabilities of slightly different nature from that of Proposition 1. The subtle difference is in whether the random selection is done at the domain or at the range space. Comparing these probabilities with those given in Appendix B will show how careful we should be in treating random selection.

Proposition 3 *Let $\mathbf{x}$ be a random point from domain $\mathscr{N}$. Set $\mathbf{y} = F^u(\mathbf{x})$ and $\mathbf{z} = F^v(\mathbf{y})$. Note that $\mathbf{z}$ is never a 0-node for F^{u+v}.*

1. *Assuming $\mathbf{z}$ to be an (F^{u+v}, i)-node, the probability for $\mathbf{y}$ to be an (F^u, k)-node is*

$$p\big(\overset{u}{\to}_k \quad \cdot \,\big|\, * \overset{u}{\to}\overset{v}{\to}_i\big) = \frac{k\, p_{u,k}}{i\, p_{u+v,i}} \sum_{j=1}^{\infty} j\, p_{v,j} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2} \cdots p_{u,i_j}.$$

2. *Assuming $\mathbf{z}$ to be an (F^{u+v}, i)-node, the probability for $\mathbf{z}$ to be an (F^v, j)-node is*

$$p\big(\cdot \quad \overset{v}{\to}_j \,\big|\, * \overset{u}{\to}\overset{v}{\to}_i\big) = \frac{j\, p_{v,j}}{i\, p_{u+v,i}} \sum_{k=1}^{\infty} k\, p_{u,k} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2} \cdots p_{u,i_j}.$$

In the statements above, the summation indices are to be taken from the non-negative integers. Any sum with an empty index set is interpreted to be zero and any sum with a degenerate index description is interpreted to be one. For example, $k + i_2 + \cdots + i_j = i$ gives an empty index set when $k > i$ with $j \geq 1$. The same condition is degenerate when $k = i$ with $j = 1$.

Proof Assuming $\mathbf{y}$ to be an (F^u, k)-node and $\mathbf{z}$ to be an (F^v, j)-node at the same time, the probability for $\mathbf{z}$ to be an (F^{u+v}, i)-node is

$$p\left(\overset{u}{\to}\overset{v}{\to}_i \,\middle|\, \begin{matrix} * \overset{u}{\to}_k & \cdot \\ \cdot & \overset{v}{\to}_j \end{matrix}\right) = \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2} \cdots p_{u,i_j} = \lceil \mathscr{P}_u(x)^{j-1} \rfloor_{i-k}.$$

Recall from discussion below (9) that the probabilities for a random input to create an (F^u, k)-node or an (F^v, j)-node are $k\, p_{u,k}$ and $j\, p_{v,j}$, respectively. Hence the probability for a random input to create an (F^{u+v}, i)-node can be expressed as the sum

$$i\, p_{u+v,i} = \sum_{k,j} k\, p_{u,k} \cdot j\, p_{v,j} \cdot p\left(\xrightarrow{u} \xrightarrow{v}_i \,\middle|\, \begin{matrix} * \xrightarrow{u}_k & \cdot \\ \cdot & \xrightarrow{v}_j \end{matrix} \right). \tag{16}$$

In fact, as a secondary check, one may follow through the next sequence of equalities.

$$\begin{aligned}
\text{RHS of (16)} &= \sum_{k,j} \lceil \mathscr{P}_u'(x) \rfloor_{k-1} \cdot \lceil \mathscr{P}_v'(x) \rfloor_{j-1} \cdot \lceil \mathscr{P}_u(x)^{j-1} \rfloor_{i-k} \\
&= \sum_{k} \lceil \mathscr{P}_u'(x) \rfloor_{k-1} \cdot \left\lceil \sum_j \lceil \mathscr{P}_v'(x) \rfloor_{j-1} \mathscr{P}_u(x)^{j-1} \right\rfloor_{i-k} \\
&= \sum_{k} \lceil \mathscr{P}_u'(x) \rfloor_{k-1} \cdot \lceil \mathscr{P}_v'(\mathscr{P}_u(x)) \rfloor_{i-k} \\
&= \lceil \mathscr{P}_u'(x)\, \mathscr{P}_v{}'(\mathscr{P}_u(x)) \rfloor_{i-1} = \lceil \{ \mathscr{P}_v(\mathscr{P}_u(x)) \}' \rfloor_{i-1} \\
&= \lceil \mathscr{P}_{u+v}'(x) \rfloor_{i-1} = i\, p_{u+v,i}.
\end{aligned}$$

Here, we have applied (14) multiple times and also used Propositions 2.

The two claimed probabilities are suitable partial sums of (16), divided by the probability of creating an (F^{u+v}, i)-node. □

3.3 Equivalence under function iterations

Let us define two points $\mathbf{x}, \mathbf{x}' \in \mathscr{N}$ to be F^k-equivalent, if $F^k(\mathbf{x}) = F^k(\mathbf{x}')$. This is trivially an equivalence relation. As an example use of this notion, recalling notation (8), one can say that any point of $\mathscr{D}_{k,i}$ is F^k-equivalent to i-many points, including itself.

In this subsection, we will work out the number of points that are F^k-equivalent to a set of m randomly chosen points. The single point case is relatively easy.

Lemma 6 *Given a random point $\mathbf{x} \in \mathscr{N}$ from the domain, the point $\mathbf{y} = F^u(\mathbf{x})$ is expected to be F^v-equivalent to $v+1$ points.*

Proof Given a random $\mathbf{x} \in \mathscr{N}$, the image $F^{u+v}(\mathbf{x})$ is an (F^{u+v}, i)-node with probability $i\, p_{u+v,i}$. Thus the probability for $F^u(\mathbf{x})$ to be F^v-equivalent to j-many nodes can be written as

$$\sum_i p\big(\cdot \quad \xrightarrow{v}_j \,\big|\, * \xrightarrow{u} \xrightarrow{v}_i \big) \cdot i\, p_{u+v,i},$$

through the law of alternatives, and the expectation we seek is

$$\sum_j j \left\{ \sum_i p\big(\cdot \quad \xrightarrow{v}_j \,\big|\, * \xrightarrow{u} \xrightarrow{v}_i \big) \cdot i\, p_{u+v,i} \right\}.$$

Applying Proposition 3, we see that this is equal to

$$\sum_{i,j} j^2 p_{v,j} \sum_k k\, p_{u,k} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2} \cdots p_{u,i_j}$$

$$\begin{aligned}
&= \sum_{i,j} \lceil \{x\mathscr{P}_v'(x)\}' \rfloor_{j-1} \sum_k \lceil \mathscr{P}_u'(x) \rfloor_{k-1} \cdot \lceil \mathscr{P}_u(x)^{j-1} \rfloor_{i-k} \\
&= \sum_{i,j} \lceil \{x\mathscr{P}_v'(x)\}' \rfloor_{j-1} \cdot \lceil \mathscr{P}_u'(x)\, \mathscr{P}_u(x)^{j-1} \rfloor_{i-1} \\
&= \sum_i \Big\lceil \mathscr{P}_u'(x) \sum_j \lceil \{x\mathscr{P}_v'(x)\}' \rfloor_{j-1} \mathscr{P}_u(x)^{j-1} \Big\rfloor_{i-1} \\
&= \sum_i \Big\lceil \mathscr{P}_u'(x) \big[\{x\mathscr{P}_v'(x)\}'\big]_{x\leftarrow \mathscr{P}_u(x)} \Big\rfloor_{i-1} \\
&= \sum_i \Big\lceil \{\mathscr{P}_u(x)\, \mathscr{P}_v'(\mathscr{P}_u(x))\}' \Big\rfloor_{i-1} \\
&= \Big[\{\mathscr{P}_u(x)\, \mathscr{P}_v'(\mathscr{P}_u(x))\}' \Big]_{x\leftarrow 1} \\
&= v+1.
\end{aligned}$$

Multiple applications of Lemma 5 is needed for the last equality. □

Let $\mathbf{x} \in \mathscr{N}$ be random and consider $\mathbf{y} = F^k(\mathbf{x})$. By substituting $m_0 = N$ into (4), we can state that the ratio of the complete F^k-image space size over the input space size is $\frac{2}{k+2}$. It is tempting to say that, on average, image point $\mathbf{y}$ will have $(\frac{2}{k+2})^{-1} = \frac{k}{2}+1$ pre-images, implying that $\mathbf{x}$ is F^k-equivalent to $\frac{k}{2}+1$ points. On the other hand, restricting Lemma 6 to the $u=0$ case tells us that $\mathbf{x}$ is expected to be F^k-equivalent to $k+1$ points.

The apparent contradiction arises from the careless use of random selection. The random choice of $\mathbf{x}$ leads to non-uniform choice among the image points. Those image points with larger pre-image sets are more likely to be taken than other image points. Lemma 6 presents the pre-image size of an image point $\mathbf{y} = F^k(\mathbf{x})$ that was obtained with random input $\mathbf{x}$. This is different from the pre-image size of a random point taken from the image space.

This dependance on where the random choice is being made is even more evident in the following lemma.

Lemma 7 *Let $D_0 \subset \mathscr{N}$ be a set of m_0 randomly chosen points. The expected number of (F^k, i)-nodes in $F^k(D_0)$ is*

$$N p_{k,i} \Big\{ 1 - \Big(1 - \frac{m_0}{N}\Big)^i \Big\}.$$

Proof Consider the process of choosing points of the F^k-image space through random selection of points in its domain. As discussed earlier, since the probability of an image point being selected will depend on the number of its pre-images, this will not produce a random distribution of image points. On the other hand, note that random selection within $\mathscr{D}_{k,i}$ will lead to uniform distribution in the corresponding image space $\mathscr{R}_{k,i}$.

As already mentioned multiple times, we have $\frac{|\mathscr{D}_{k,i}|}{N} = i\, p_{k,i}$, and the number of elements belonging to $D_0 \cap \mathscr{D}_{k,i}$ is expected to be $m_0\, i\, p_{k,i}$. Modeling F as a random function, we can interpret

$$F^k(D_0 \cap \mathscr{D}_{k,i}) = F^k(D_0) \cap \mathscr{R}_{k,i}$$

as a set obtained by drawing $(m_0\, i\, p_{k,i})$-many points from $\mathscr{R}_{k,i}$, uniformly at random, with replacements. Thus the size of this image set is expected to be

$$\begin{aligned}
&|\mathscr{R}_{k,i}|\Big\{1-\Big(1-\frac{1}{|\mathscr{R}_{k,i}|}\Big)^{m_0\, i\, p_{k,i}}\Big\}\\
&\qquad = Np_{k,i}\Big\{1-\Big(1-\frac{1}{Np_{k,i}}\Big)^{m_0\, i\, p_{k,i}}\Big\} \approx Np_{k,i}\Big\{1-\Big(1-\frac{m_0}{N}\Big)^{i}\Big\},
\end{aligned}$$

which is the stated claim. □

Remark 1 During the proof of Lemma 7 we have utilized the explicit expected values $E(D_0 \cap \mathscr{D}_{k,i}) = m_0\, i\, p_{k,i}$ and $E(|\mathscr{R}_{k,i}|) = Np_{k,i}$ as the number of certain objects in computing another expected value. Such an argument is not strictly correct. A correct proof would involve conditional expectations and compute an iterated expectation to arrive at the claim, but we shall not do this here. Our final evidence that this proof reflects the reality is the simulation result given in Section 6. The current remark will apply also to many of our later proofs.

Since Lemma 6 show that a single random point is F^k-equivalent to $k+1$ points, we may conclude that for small m_0, the number of points F^k-equivalent to m_0 random points will be $m_0(k+1)$. For large m_0, this will no longer be true, as some of the m_0 points will be equivalent to each other. Let us discuss this in more detail.

Lemma 8 *Let $D_0 \subset \mathscr{N}$ be a set of m_0 randomly chosen points. The number of points that are F^v-equivalent to points of $D_u = F^u(D_0)$ is expected to be*

$$N\Big\{1-\mathscr{P}_u\big(1-\frac{m_0}{N}\big)\,\mathscr{P}_v'\Big(\mathscr{P}_u\big(1-\frac{m_0}{N}\big)\Big)\Big\} = N\Big\{1-\prod_{k=u}^{u+v}\mathscr{P}_k\big(1-\frac{m_0}{N}\big)\Big\}.$$

Proof By Lemma 7, the number of (F^{u+v}, i)-nodes in $F^{u+v}(D_0)$ is expected to be

$$Np_{u+v,i}\Big\{1-\Big(1-\frac{m_0}{N}\Big)^{i}\Big\}.$$

Thus, the expected equivalence size we are looking for can be computed as

$$\begin{aligned}
&\sum_i Np_{u+v,i}\Big\{1-\Big(1-\frac{m_0}{N}\Big)^{i}\Big\}\sum_j j\cdot p\big(\cdot \quad \xrightarrow{v}_j \,\big|\, * \xrightarrow{u}\xrightarrow{v}_i\big)\\
&\qquad = N\sum_i \frac{1}{i}\Big\{1-\Big(1-\frac{m_0}{N}\Big)^{i}\Big\}\sum_j j^2 p_{v,j}\sum_k k\, p_{u,k}\sum_{k+i_2+\cdots+i_j=i} p_{u,i_2}\cdots p_{u,i_j}\\
&\qquad = N\sum_i \frac{1}{i}\Big\{1-\Big(1-\frac{m_0}{N}\Big)^{i}\Big\}\Big\lceil\big\{\mathscr{P}_u(x)\,\mathscr{P}_v'(\mathscr{P}_u(x))\big\}'\Big\rfloor_{i-1}.
\end{aligned}$$

Here, the first equality was obtained through Proposition 3 and computation for the second equality is identical to that done in the proof of Lemma 6. Now, by (15), this is equal to

$$N \int_{1-\frac{m_0}{N}}^{1} \{ \mathscr{P}_u(x)\, \mathscr{P}'_v(\mathscr{P}_u(x)) \}' \, dx$$

$$= N \Big[\mathscr{P}_u(x)\, \mathscr{P}'_v(\mathscr{P}_u(x)) \Big]_{1-\frac{m_0}{N}}^{1}$$

$$= N - N\, \mathscr{P}_u\big(1 - \frac{m_0}{N}\big)\, \mathscr{P}_v{}'\Big(\mathscr{P}_u\big(1 - \frac{m_0}{N}\big)\Big).$$

Here, the last equality relies on Lemma 5. We can complete the proof by applying Lemma 4. □

For applications to tradeoffs, it is more useful to have the above equivalence count given in terms of the image space size rather than the input size.

Proposition 4 *Let $D_0 \subset \mathscr{N}$ be a set of randomly chosen points. If the number of distinct elements in $D_t = F^t(D_0)$ is m_t, then the pre-image of D_t under F^k is expected to be of size*

$$m_t(1+k)\big(1 - \frac{m_t k}{4N}\big).$$

Proof With m_0 set to the size of D_0, Lemma 8 implies that the pre-image size we are looking for is

$$N\Big\{1 - \prod_{i=t-k}^{t} \mathscr{P}_i\big(1 - \frac{m_0}{N}\big)\Big\}. \tag{17}$$

The condition of the proposition implies

$$\mathscr{P}_t\big(1 - \frac{m_0}{N}\big) = 1 - \frac{m_t}{N}.$$

Recalling the notation $\mathscr{P}_{-k}$ for the inverse function of $\mathscr{P}_k$, the above can be rewritten as

$$\mathscr{P}_i\big(1 - \frac{m_0}{N}\big) = \mathscr{P}_{i-t}\big(1 - \frac{m_t}{N}\big).$$

and (17) becomes

$$N\Big\{1 - \prod_{j=-k}^{0} \mathscr{P}_j\big(1 - \frac{m_t}{N}\big)\Big\}. \tag{18}$$

When Lemma 3 is applied, there is a series of cancelations within the product term and this simplifies to

$$m_t(1+k)\Big(1 - \frac{m_t k}{2(2N - m_t)}\Big).$$

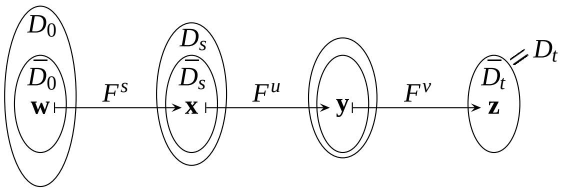
The stated result is an approximation of this value. □

Once again, we emphasize that the randomness in this proposition refers to the selection of inputs rather than to the selection within the image space. Another proof for this proposition is given in Appendix C.

Lemma 9 *Let $t = s + u + v$ be a sum of non-negative integers and choose a set of random points $D_0 \subset \mathscr{N}$ of size m_0. For each $\mathbf{z} \in D_t = F^t(D_0)$, randomly choose a $\mathbf{w} \in D_0$ such that $F^t(\mathbf{w}) = \mathbf{z}$. The collection of the chosen pre-images $\mathbf{w}$ will be called $\bar{D}_0$. Then, the number of points that are F^u-equivalent to points of $\bar{D}_s = F^s(\bar{D}_0)$ is expected to be*

$$N \int_{\mathscr{P}_s(1-\frac{m_0}{N})}^{1} \big(x\,\mathscr{P}_u'(x)\big)'\,\mathscr{P}_v'\big(\mathscr{P}_u(x)\big)\,dx.$$

Proof The diagram below may be of help in following through this proof.



Consider the intermediate image set $D_s = F^s(D_0)$ and let m_s be its size. Since we are taking F to be a random function, D_s is a set of random points from $\mathscr{N}$. So, by Lemma 7, the expected number of (F^{u+v}, i)-nodes in $D_t = F^{u+v}(D_s)$ is

$$N p_{u+v,i}\Big\{1 - \Big(1 - \frac{m_s}{N}\Big)^i\Big\}.$$

Now suppose $\mathbf{z} \in \bar{D}_t = F^t(\bar{D}_0) = D_t$ is an (F^{u+v}, i)-node, let $\mathbf{w} \in \bar{D}_0$ be the corresponding input element that was chosen. Set $\mathbf{x} = F^s(\mathbf{w})$ and $\mathbf{y} = F^u(\mathbf{x})$. Then $\mathbf{x} \in \bar{D}_s$ is a random input that produces the (F^{u+v}, i)-node $\mathbf{z}$. By Proposition 3, the probability for $\mathbf{y}$ to be an (F^u, k)-node is

$$p\big(\overset{u}{\rightarrow}_k\ \cdot\,\big|\,*\overset{u}{\rightarrow}\overset{v}{\rightarrow}_i\big) = \frac{k\,p_{u,k}}{i\,p_{u+v,i}} \sum_j j\,p_{v,j} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2}\cdots p_{u,i_j}.$$

The expected equivalence size we want to find is

$$\sum_i \Big(\#\text{ of }(F^{u+v}, i)\text{-nodes in }\bar{D}_t\Big) \sum_k k \cdot p\big(\overset{u}{\rightarrow}_k\ \cdot\,\big|\,*\overset{u}{\rightarrow}\overset{v}{\rightarrow}_i\big),$$

and our discussion so far shows that this may be computed through

$$\sum_i N p_{u+v,i}\Big\{1 - \Big(1 - \frac{m_s}{N}\Big)^i\Big\} \sum_k k \cdot \frac{k\,p_{u,k}}{i\,p_{u+v,i}} \sum_j j\,p_{v,j} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2}\cdots p_{u,i_j}.$$

This is equal to the following sequence of equations.

$$\begin{aligned}
&N \sum_i \frac{1}{i}\Big\{1 - \Big(1 - \frac{m_s}{N}\Big)^i\Big\} \sum_k \big\lceil \big(x\,\mathscr{P}_u'(x)\big)' \big\rfloor_{k-1} \sum_j \big\lceil \mathscr{P}_v'(x) \big\rfloor_{j-1} \cdot \big\lceil \mathscr{P}_u(x)^{j-1} \big\rfloor_{i-k} \\
&= N \sum_i \frac{1}{i}\Big\{1 - \Big(1 - \frac{m_s}{N}\Big)^i\Big\} \sum_k \big\lceil \big(x\,\mathscr{P}_u'(x)\big)' \big\rfloor_{k-1} \cdot \big\lceil \mathscr{P}_v'(\mathscr{P}_u(x)) \big\rfloor_{i-k}
\end{aligned}$$

$$= N \sum_i \frac{1}{i} \Big\{ 1 - \Big(1 - \frac{m_s}{N}\Big)^i \Big\} \Big[\big(x \mathscr{P}'_u(x)\big)' \mathscr{P}'_v\big(\mathscr{P}_u(x)\big) \Big]_{i-1}$$
$$= N \int_{1-\frac{m_s}{N}}^{1} \big(x \mathscr{P}'_u(x)\big)' \mathscr{P}'_v\big(\mathscr{P}_u(x)\big)\, dx$$

The last equality follows from (15). To arrive at our claim, it now suffices to substitute

$$1 - \frac{m_s}{N} = \mathscr{P}_s\big(1 - \frac{m_0}{N}\big),$$

as given by Lemma 2. □

Proposition 5 *Let $t = s+u+v$ be a sum of non-negative integers and choose a set of random points $D_0 \subset \mathscr{N}$. Suppose $D_t = F^t(D_0)$ is of size m_t, and for each $\mathbf{z} \in D_t$, randomly choose $\underline{a}$ $\mathbf{w} \in D_0$ such that $F^t(\mathbf{w}) = \mathbf{z}$. The collection of chosen pre-images $\mathbf{w}$ will be called $\bar{D}_0$. Then, the pre-image of $F^{s+u}(\bar{D}_0)$ under F^u is expected to be of size*

$$m_t(u+1) + \frac{u(u+2)}{v^2} \Big\{ m_t v + 2N \ln\Big(1 - \frac{m_t v}{2N}\Big)\Big\}.$$

Proof Combining Lemma 9 and Lemma 2, we can state the pre-image size as

$$\int_{\mathscr{P}_{-(u+v)}(1-\frac{m_t}{N})}^{1} N \big(x \mathscr{P}'_u(x)\big)' \mathscr{P}'_v\big(\mathscr{P}_u(x)\big)\, dx.$$

When the approximation given by Lemma 3 is applied, the above indefinite integral becomes

$$\frac{4N(u^2+uv+2u+v)}{v(u+v)\big(2+(u+v)(1-x)\big)} - \frac{2Nu(u+2)}{v^2} \ln\Big\{ \frac{2+u(1-x)}{2+(u+v)(1-x)} \Big\}.$$

After the integration boundaries are placed in this equation with $\mathscr{P}_{-(u+v)}(1-\frac{m_t}{N})$ as given by Lemma 3, the result simplifies to

$$\frac{m_t(u^2+uv+2u+v)}{v} + \frac{2Nu(u+2)}{v^2} \ln\Big(1 - \frac{m_t v}{2N}\Big).$$

This is equal to what is claimed. □

4 Cost of false alarms

Let us use results of the previous section to quantify the effects of false alarms. We shall show that, under typical parameters, the Hellman, perfect rainbow, and non-perfect rainbow tradeoffs, spend 14.3%, 25.8%, and 27.6% of the total online time in resolving false alarms, respectively.

As with any analysis of tradeoff algorithms, we shall model F as a random function during our arguments.

4.1 Hellman tradeoff

Hellman tradeoff utilizes multiple tables with usually one of these containing the correct answer. Unless all the tables are processed in parallel, we will end up producing the full length online chain for most of the Hellman tables that we start search on. So we shall state our cost claims in terms of single table processing, rather than for the whole online phase. If needed, one can readily obtain the expected total cost from the success probability of a single table and the cost per table.

Recall that the k-th column of a Hellman matrix is denoted by $\mathtt{HM}_k$. We shall write $F^{-j}(\mathtt{HM}_t)$ for the set of pre-images under F^j of the ending points $\mathtt{HM}_t$.

Lemma 10 *For each $0 \le k \le t$, the expected size of the pre-image set $F^{-k}(\mathtt{HM}_t)$ is approximately $m(1+k)$.*

Proof Proposition 4 states $m(1+k)\left(1-\frac{mk}{4N}\right)$ as the expected pre-image size. Since our assumptions of $c_{\mathtt{H}} \sim 1$ and $1 \ll t$ imply $0 \le \frac{mk}{4N} = \frac{c_{\mathtt{H}}}{4t}\frac{k}{t} \le \frac{c_{\mathtt{H}}}{4t} \ll 1$, the conclusion follows.

Lemma 6, combined with our discussion on ending point distinctness, stated below (5), also leads to the same conclusion. □

It is now possible to compute the extra work incurred by false alarms.

Theorem 1 *During the full online processing of a single Hellman table, the online chain creation requires about t applications of F. The false alarms are expected to cause approximately $\frac{c_{\mathtt{H}}}{6}t$ extra applications of F.*

Proof The complete online chain creation requires $t-1$ invocations of F. Thus the first claim is evident.

Let $\mathbf{y} = F(\mathbf{x})$ be given as the target image point. For each $0 < k \le t$, the k-th iteration of the online phase, i.e., the search of $F^{k-1}(\mathbf{y})$ among the ending points, causes an alarm if and only if $\mathbf{x} \in F^{-k}(\mathtt{HM}_t)$. The alarm is a false one, if $\mathbf{x} \notin \mathtt{HM}_{t-k}$. By Lemma 10 and the assumption on ending point distinctness, the probability of such an incident happening is $\frac{m(1+k)}{N} - \frac{m}{N}$.

Each alarm causes $(t-k+1)$ iterations of F to be executed before it can be dismissed as being false. Thus, the expected number of F iterations spent on false alarm treatment throughout the full processing of a single Hellman table can be calculated as

$$\sum_{0<k\le t} (t-k+1)\frac{mk}{N} = \frac{t(t+1)(t+2)}{6} \cdot \frac{m}{N}.$$

It now suffices to bring out the highest term in t from this equation and apply the condition $mt^2 = c_{\mathtt{H}}N$. □

Under the matrix stopping rule $mt^2 = N$, approximately $\frac{1}{6}t$ iterations of F are spent on resolving false alarms per Hellman table. In comparison, the full online chain generation requires approximately t iterations. So if the multiple Hellman tables are processed sequentially, false alarms will cause $\frac{1}{6} \approx 16.7\%$ increase in online time. In other words, $\frac{1}{7} \approx 14.3\%$ of the total online time is related to false alarms. In general,

Table 1 Expected iteration counts per table (unit: t) and relative cost of false alarms for Hellman tradeoffs (OC=online chain, FA=false alarm, itr=iterations)

$c_{\mathtt{H}}$	1/4	1/3	1/2	1	2	3	4
OC itr	1.000	1.000	1.000	1.000	1.000	1.000	1.000
FA itr	0.042	0.056	0.083	0.167	0.333	0.500	0.667
total itr	1.042	1.056	1.083	1.167	1.333	1.500	1.667
FA/total	4.0%	5.3%	7.7%	14.3%	25.0%	33.3%	40.0%

$\frac{c_{\mathtt{H}}}{6+c_{\mathtt{H}}}$ of the online time is spent on resolving false alarms. As can be seen through Table 1, the relative cost of false alarms is sensitive to $c_{\mathtt{H}}$.

Notice that $(t-k+1)\frac{mk}{N}$, the term being summed in the proof of Theorem 1, viewed as a function of k, attains its maximum value near $k=\frac{t}{2}$. So, the unwanted effect of false alarms is at its greatest when the online chain has reached half of its full length. Assuming that the correct answer is found after going through half of the pre-computed data, this implies that we get the same $\frac{c_{\mathtt{H}}}{6}$ factor increase in online time due to false alarms, even when all the Hellman tables are processed in parallel.

4.2 Perfect rainbow tradeoff

We shall concentrate on analyzing the cost of processing a single perfect rainbow table. At the end of this subsection, we briefly explain how to work with multiple tables. In our discussion, the perfect rainbow table is not assumed to (but may) contain the maximal number of possible rows. While maximal perfect tables are easier to analyze, building them are very costly and are seldom used. In practice, a near-maximal table, or a non-perfect table, considered in the next subsection, are used.

Recall the notation (2). We shall write $F_\star$ when referring to multiple F_j without explicitly specifying their, possibly different, indices. To deal with rainbow tradeoffs, we redefine $F^k = F_{\star+k-1}\circ\cdots\circ F_{\star+1}\circ F_\star$ to be any k iterations of consecutive $F_\star$. As long as each F_j is a random function, all the proofs of Section 3 remain valid for this new F^k and, in particular, Proposition 4 is true for the new F^k.

Disregarding the effects of false alarms, the rainbow tradeoff requires approximately $\frac{1}{2}t^2$ applications of $F_\star$ in fully processing a single table. But unlike the Hellman tradeoff, the complete table is not likely to be fully processed. Because a rainbow table is huge, early exit from the online phase algorithm, which occurs when the correct answer to the inversion problem is found, is quite likely. So we need a measure of the online chain creation cost that reflects the realistic use of rainbow tradeoffs. This is presented in the following theorem, where the cost of false alarms is also given.

We shall write $F^{-j}(\mathtt{RM}_t)$ for the set of pre-images under F^j of the ending points.

Theorem 2 *The online processing of a single perfect rainbow table is expected to require approximately*

$$\left\{1-\frac{1+c_{\mathtt{R}}}{e^{c_{\mathtt{R}}}}\right\}\left(\frac{t}{c_{\mathtt{R}}}\right)^2$$

applications of $F_\star$, in creating the online chain. Approximately

$$\left\{2(c_{\mathtt{R}}-1)+\frac{2-c_{\mathtt{R}}^2}{e^{c_{\mathtt{R}}}}\right\}\left(\frac{t}{2c_{\mathtt{R}}}\right)^2$$

Table 2 Expected iteration counts (unit: t^2) and relative cost of false alarms for perfect rainbow tradeoffs (OC=online chain, FA=false alarm, itr=iterations)

$c_{\mathtt{R}}$	0.25	0.5	1	1.5	1.8	1.9	1.95	2
OC itr	0.4240	0.3608	0.2642	0.1965	0.1658	0.1569	0.1526	0.1485
FA itr	0.0357	0.0614	0.0920	0.1049	0.1076	0.1080	0.1081	0.1081
total itr	0.4597	0.4222	0.3562	0.3014	0.2734	0.2648	0.2607	0.2566
FA/total	7.77%	14.55%	25.82%	34.80%	39.37%	40.77%	41.45%	42.12%

additional invocations of $F_\star$ are expected to be needed, in dealing with false alarms.

Proof Let us say $\mathbf{y} = F(\mathbf{x})$ is given as the inversion target. Since we are dealing with a perfect table, $\mathbf{x}$ is in the rainbow matrix column $\mathtt{RM}_{t-1}$ with probability $\frac{m}{N}$. Checking for this is done by searching for $\mathbf{y}$ in $\mathtt{RM}_t$, which requires no $F_\star$ computation. This first iteration fails to return the answer $\mathbf{x}$ with probability $1-\frac{m}{N}$. In the second iteration, a single application of F_t is required. At the third iteration, which will be reached with probability $(1-\frac{m}{N})^2$, application of $F_{t-1}\circ F_t$, or two applications of $F_\star$, is required.

Thus, the expected number of $F_\star$ iterations can be written as

$$\sum_{0<k\le t}(k-1)\Big(1-\frac{m}{N}\Big)^{k-1}. \tag{19}$$

Simplification of this, using the approximation $(1-\frac{m}{N})^t \approx \exp(-\frac{mt}{N}) = e^{-c_{\mathtt{R}}}$, results in

$$\Big\{\Big(1-\frac{c_{\mathtt{R}}}{t}\Big)-\Big(1-\frac{c_{\mathtt{R}}}{t}+c_{\mathtt{R}}\Big)\frac{1}{e^{c_{\mathtt{R}}}}\Big\}\Big(\frac{t}{c_{\mathtt{R}}}\Big)^2.$$

Collection of the highest terms in t from this equation is the claimed F invocation count for online chain creation.

Let us now look into the second claim. As was already argued, the k-th $(0<k\le t)$ iteration of the online phase is executed with probability $(1-\frac{m}{N})^{k-1}$. The k-th iteration sounds a false alarm if and only if $\mathbf{x}\in F^{-k}(\mathtt{RM}_t)\setminus \mathtt{RM}_{t-k}$. Proposition 4 implies that this event happens with probability $\frac{m}{N}(1+k)(1-\frac{mk}{4N})-\frac{m}{N}$. Each verification of whether we are dealing with a false alarm requires $(t-k+1)$ iterations of $F_\star$.

The expected number of extra $F_\star$ iterations required to deal with false alarms is thus

$$\sum_{0<k\le t}(t-k+1)\cdot\frac{m}{N}\Big(k-\frac{mk}{4N}-\frac{mk^2}{4N}\Big)\cdot\Big(1-\frac{m}{N}\Big)^{k-1}. \tag{20}$$

This simplifies to

$$\Big\{2(c_{\mathtt{R}}-1)+\frac{2-c_{\mathtt{R}}^2}{e^{c_{\mathtt{R}}}}\Big\}\frac{t^2}{4c_{\mathtt{R}}^2}+\Big\{2+\frac{c_{\mathtt{R}}^2-c_{\mathtt{R}}-2}{e^{c_{\mathtt{R}}}}\Big\}\frac{t}{4c_{\mathtt{R}}}+\Big\{\frac{c_{\mathtt{R}}}{e^{c_{\mathtt{R}}}}\Big\}\frac{1}{4},$$

when the approximation $(1-\frac{m}{N})^t\approx e^{-c_{\mathtt{R}}}$ is used. The highest terms in t from this equation is what our claim states as the cost of false alarms. □

When $c_{\mathtt{R}} = 1$, which is typical for perfect rainbow tradeoffs, the number of F invocations spent on the online chain creation is $(1 - \frac{2}{e})t^2$ and $\frac{1}{4e}t^2$ additional invocations due to false alarms are expected. This translates to 34.8% extra $F_\star$ iterations due to false alarms. In other words, 25.8% of the online time is spent in resolving false alarms. More such information for specific $c_{\mathtt{R}}$ can be found in Table 2. Values for $c_{\mathtt{R}}$ of only up to 2 are listed, because it can be argued from (4) that, for any choice of parameters, we always have $c_{\mathtt{R}} < 2$.

We advise caution in interpreting data from Table 2. The iteration counts are given in multiples of t^2, so that, for example, the total iteration $0.3562t^2$ for $c_{\mathtt{R}} = 1$ being greater than $0.2566t^2$ for $c_{\mathtt{R}} = 2$ does not imply that the $c_{\mathtt{R}} = 1$ parameter takes longer. To the contrary, for the same m, the t for $c_{\mathtt{R}} = 2$ is twice as large as the t for $c_{\mathtt{R}} = 1$, and the net result is that $c_{\mathtt{R}} = 2$ takes much longer. That said, this alone does not automatically imply superiority of the parameter $c_{\mathtt{R}} = 1$ over $c_{\mathtt{R}} = 2$, since the two brings about different (easily computable) success probabilities.

It is easy to translate results of this section to the case when ℓ tables are processed in parallel. To obtain the expected $F_\star$ iteration counts per table, it suffices to replace the $(1 - \frac{m}{N})^{k-1}$ factors appearing in (19) and (20) with $(1 - \frac{m}{N})^{\ell(k-1)}$. Resulting formulas are no longer simple, but computable. If multiple tables are processed sequentially, counts for the ℓ-th table may be obtained by multiplying the factor $(1 - \frac{m}{N})^{t(\ell-1)} \approx \exp\big(-c_{\mathtt{R}}(\ell-1)\big)$ to the corresponding values given for a single table.

4.3 Non-perfect rainbow tradeoff

Let us now consider non-perfect rainbow tradeoffs, i.e, the case where none of the colliding pre-computed chains are removed. Partial results for this case, obtained in a manner different from the current work, appears in [14]. The case where one removes only some of the colliding chains is also conceivable, but such a case does not appear in the literature and will not be dealt with here.

When a non-perfect rainbow table is in use, the expected number of $F_\star$ iterations for online chain creation can be written as

$$\sum_{0<k\leq t} (k-1) \prod_{j=1}^{k-1} \Big(1 - \frac{m_{t-j}}{N}\Big),$$

where each m_j is given by (3) with $m_0 = m$. When the approximation (4) is applied to this, we see cancelations within the product term and the above simplifies into

$$\begin{aligned}
&\sum_{0<k\leq t} (k-1) \cdot \frac{2N + m(t-k)}{2N + m(t-1)} \cdot \frac{2N + m(t-k-1)}{2N + m(t-2)} \\
&\qquad = \frac{t(t-1)\{(24 + 8c_{\mathtt{R}} + c_{\mathtt{R}}^2)t^2 - c_{\mathtt{R}}(28 + 5c_{\mathtt{R}})t + 6c_{\mathtt{R}}^2\}}{12\{(2 + c_{\mathtt{R}})^2 t^2 - 3c_{\mathtt{R}}(2 + c_{\mathtt{R}})t + 2c_{\mathtt{R}}^2\}} \\
&\qquad \approx \frac{1}{12} \cdot (24 + 8c_{\mathtt{R}} + c_{\mathtt{R}}^2) \cdot \Big(\frac{t}{2 + c_{\mathtt{R}}}\Big)^2.
\end{aligned}$$

Table 3 Expected iteration counts (unit: t^2) and relative cost of false alarms for non-perfect rainbow tradeoffs (OC=online chain, FA=false alarm, itr=iterations)

$c_{\mathtt{R}}$	0.25	0.5	1	2	5	10	100
OC itr	0.4290	0.3767	0.3056	0.2292	0.1514	0.1181	0.0867
FA itr	0.0372	0.0677	0.1167	0.1917	0.3656	0.6250	5.1326
total itr	0.4662	0.4443	0.4222	0.4208	0.5170	0.7431	5.2193
FA/total	7.98%	15.23%	27.63%	45.54%	70.72%	84.11%	98.34%

To deal with the cost of false alarms, note that if the current end of the online chain matches a common ending point of two pre-computed chains, both of them will have to be regenerated for false alarm verification. Instead of doubling the work factor for this case, we can simply let the probability of false alarms be added twice, once for each of the colliding chain. Thus, for the purpose of analyzing false alarm costs, we can simply ignore collisions and treat colliding pre-computed chains as independent chains.

Using Lemma 6, the expected number of extra $F_\star$ iterations required to deal with false alarms can be written as

$$\sum_{0<k\le t}(t-k+1)\cdot\Big\{\frac{m}{N}(1+k)-\frac{m}{N}\Big\}\cdot\prod_{j=1}^{k-1}\Big(1-\frac{m_{t-j}}{N}\Big)$$
$$\approx\frac{c_{\mathtt{R}}(t+1)\left\{\begin{matrix}(40+20c_{\mathtt{R}}+3c_{\mathtt{R}}^2)t^3-(160+80c_{\mathtt{R}}+13c_{\mathtt{R}}^2)t^2\\ +6c_{\mathtt{R}}(20+3c_{\mathtt{R}})t-8c_{\mathtt{R}}^2\end{matrix}\right\}}{60\{(2+c_{\mathtt{R}})^2t^2-3c_{\mathtt{R}}(2+c_{\mathtt{R}})t+2c_{\mathtt{R}}^2\}}$$
$$\approx\frac{c_{\mathtt{R}}}{60}\cdot(40+20c_{\mathtt{R}}+3c_{\mathtt{R}}^2)\cdot\Big(\frac{t}{2+c_{\mathtt{R}}}\Big)^2,$$

where m_j are to be defined as before. The first approximation is, once again, based on (4). We gather what we have discussed so far in the next theorem.

Theorem 3 *The online processing of a single non-perfect rainbow table is expected to require approximately*

$$\frac{1}{12}\cdot(24+8c_{\mathtt{R}}+c_{\mathtt{R}}^2)\cdot\Big(\frac{t}{2+c_{\mathtt{R}}}\Big)^2$$

applications of $F_\star$, in creating the online chain, and

$$\frac{c_{\mathtt{R}}}{60}\cdot(40+20c_{\mathtt{R}}+3c_{\mathtt{R}}^2)\cdot\Big(\frac{t}{2+c_{\mathtt{R}}}\Big)^2.$$

additional invocations of $F_\star$, in resolving false alarms.

Some example values for expected online time complexities are given in Table 3. Note that the cost of false alarms quickly dominates the main online chain creation cost as $c_{\mathtt{R}}$ is increased.

Remark 2 The arguments of this subsection used $\prod_{j=1}^{k-1}(1-\frac{m_{t-j}}{N})$ as the probability for the k-th online iteration to happen. We already saw through Remark 1 that this is not strictly correct. Unlike the perfect rainbow case, there is an added complication to be mentioned here. The number of distinct points $\mathtt{RM}_j$ will be highly correlated to that of $\mathtt{RM}_{j+1}$, so that such events are far from being independent. Thus the product expression stands on shaky grounds. Still, we know that this will be roughly correct for large m, and our proof should reflect the true picture, as the test results of Section 6 agree with the theory.

5 Checkpoints

An analysis of the online time complexities for the Hellman and rainbow tradeoffs was provided in the previous section. But since there is a technique called checkpoints, that can be applied to tradeoff algorithms to reduce the cost of false alarms, our online time complexity analysis would not be complete without measuring their effects.

5.1 Hellman tradeoff

Let us first deal with the Hellman tradeoff case. While we can give analysis for any number of checkpoints, the results do not simplify into one uniform formula, and the content of this section is best explained with examples. Furthermore, our analysis shows that using a large number of checkpoints is uncalled-for in most situations.

Let us say that $\mathbf{y}=F(\mathbf{x})$ is given as the target image point. We assume that a 1-bit checkpoint has been placed at the $(t-c)$-th column, i.e., c iterations from the ending point. We shall write $F^{-j}(\mathtt{HM}_k)$ for the set of pre-images under F^j of the k-th column $\mathtt{HM}_k$ of the Hellman matrix.

It is clear that, for $k<c$, false alarms at the k-th iteration of the online phase cannot be filtered out with the checkpoint. Since the online chain available to the tradeoff operator starts from $\mathbf{y}=F(\mathbf{x})$, rather than from $\mathbf{x}$, the checkpoint information becomes useful starting from the $(c+1)$-th iteration.

Suppose that we observed an alarm in the k-th iteration, with $k>c$. This implies $\mathbf{x}\in F^{-k}(\mathtt{HM}_t)$. Notice that

$$\mathtt{HM}_{t-k}\subset F^{-(k-c)}(\mathtt{HM}_{t-c})\subset F^{-k}(\mathtt{HM}_t).$$

If $\mathbf{x}\in\mathtt{HM}_{t-k}$, the correct answer has been found. If $\mathbf{x}\in F^{-(k-c)}(\mathtt{HM}_{t-c})\setminus\mathtt{HM}_{t-k}$, we have a false alarm, but the online chain starting from $\mathbf{x}$ will merge with the pre-computed Hellman chain before passing over the checkpoint and the checkpoint information is useless in resolving false alarms. Finally, for the remaining case of $\mathbf{x}\in F^{-k}(\mathtt{HM}_t)\setminus F^{-(k-c)}(\mathtt{HM}_{t-c})$, the 1-bit checkpoint information will resolve a false alarm with probability $1/2$. Thus, for $k>c$, false alarms which are unresolved by the checkpoint occurs at the k-th iteration with probability

$$\frac{1}{N}\Big\{\big(|F^{-(k-c)}(\mathtt{HM}_{t-c})|-|\mathtt{HM}_{t-k}|\big)+\frac{1}{2}\big(|F^{-k}(\mathtt{HM}_t)|-|F^{-(k-c)}(\mathtt{HM}_{t-c})|\big)\Big\}.$$

It remains to fill in the various set sizes. We know $|F^{-k}(\mathtt{HM}_t)| = m(1+k)$ from Lemma 10. To deal with $|F^{-(k-c)}(\mathtt{HM}_{t-c})|$, we give a version of Proposition 5 specialized for Hellman tables.

Lemma 11 *For each $0 \le u \le t-c$, the expected size of the pre-image set $F^{-u}(\mathtt{HM}_{t-c})$ is approximately $m(1+u)$.*

Proof By Proposition 5, the pre-image size is expected to be

$$m(u+1) + \frac{u(u+2)}{c^2}\Big\{mc + 2N\ln\Big(1-\frac{mc}{2N}\Big)\Big\}.$$

Under normal parameters for Hellman tradeoffs, we have $mc \ll N$, so that we may approximate $\ln(1-\frac{mc}{2N})$ simply by $-\frac{mc}{2N}$. The second term disappears and the result follows.

It is also possible to obtain the same result from Lemma 10 or Lemma 6 and our assumption, stated below (5), that no chains were removed in creating a Hellman table. □

Using the set size expectations that are now available, the probability of false alarm at the k-th iteration can be written as

$$\mathrm{Prob}_{\mathrm{FA}}(k) = \begin{cases} \frac{mk}{N} & \text{for } 1 \le k \le c, \\ \frac{mk}{N} - \frac{mc}{2N} & \text{for } c < k \le t. \end{cases}$$

Finally, we can state that

$$\sum_{0<k\le t}(t-k+1)\frac{mk}{N} - \sum_{c<k\le t}(t-k+1)\frac{mc}{2N}$$

is the expected number of F iterations caused by false alarms. Below, we summarize what has so far been discussed and add a few more simple computational consequences.

Example 1 By placing a single 1-bit checkpoint at the $(t-c)$-th column of a Hellman matrix, one can expect to remove

$$\sum_{c<k\le t}(t-k+1)\frac{mc}{2N} \;=\; \frac{m}{N}\Big\{\frac{c(t-c)(t-c+1)}{4}\Big\}$$

of the F applications caused by false alarms, per table. For large t, the maximum effect is obtained with $c \approx \frac{t}{3}$. Of the $\frac{c_{\mathtt{H}}}{6}t$ extra function iterations caused by false alarms, per table, $\frac{c_{\mathtt{H}}}{27}t$ iterations can be removed through a single 1-bit checkpoint at the optimal position $c = \frac{t}{3}$.

Table 4 Optimal 1-bit checkpoint positions for Hellman tradeoffs (distance from ending point in units of t)

# of CP	1-st CP	2-nd CP	3-rd CP	4-th CP
1 CP	0.33333			
2 CP	0.25760	0.41920		
3 CP	0.21166	0.33617	0.48067	
4 CP	0.18029	0.28273	0.39601	0.52748

One can easily extend the above results to more than one checkpoint. For example, when 1-bit checkpoints are placed at the $(t-c_2)$-th and $(t-c_1)$-th columns $(c_2 > c_1)$, the work induced by false alarms can be written as follows.

$$\begin{aligned}&\sum_{0<k\le c_1}(t-k+1)\{k\}\frac{m}{N}\\&\quad+\sum_{c_1<k\le c_2}(t-k+1)\Big\{(k-c_1)+\frac{c_1}{2}\Big\}\frac{m}{N}\\&\quad+\sum_{c_2<k\le t}(t-k+1)\Big\{(k-c_2)+\frac{c_2-c_1}{2}+\frac{c_1}{4}\Big\}\frac{m}{N}.\end{aligned}$$

Notice that alarms related to columns situated to the left of both checkpoints are filtered twice. We summarize this in a simplified form below.

Example 2 By placing 1-bit checkpoints at the $(t-c_2)$-th and $(t-c_1)$-th columns $(c_2 > c_1)$ of a Hellman matrix, one can expect to remove

$$\frac{1}{8}\frac{m}{N}\big\{(2c_1^3-c_1c_2^2+2c_2^3)-(2c_1^2-c_1c_2+2c_2^2)(2t+1)+(c_1+2c_2)(t^2+t)\big\}$$

of the F iterations caused by false alarms, per table. Assuming c_1 and c_2 to be $O(t)$, this is approximately

$$\frac{1}{8}\frac{m}{N}\big\{(2c_1^3-c_1c_2^2+2c_2^3)-(2c_1^2-c_1c_2+2c_2^2)2t+(c_1+2c_2)t^2\big\}.$$

We can compute that the maximum effect is obtained when using

$$c_1 \approx \frac{34-3\sqrt{46}}{53}t \approx 0.2576t \quad\text{and}\quad c_2 \approx \frac{29-\sqrt{46}}{53}t \approx 0.4192t.$$

Using these parameters, one can remove about $\frac{181+23\sqrt{46}}{5618}c_{\mathtt{H}}t \approx 0.05998c_{\mathtt{H}}t$ of the $\frac{c_{\mathtt{H}}}{6}t \approx 0.1667c_{\mathtt{H}}t$ function iterations related to false alarms.

It is now clear how to approach any number of checkpoints. The computation will be more complicated, but clearly feasible for anyone that needs the information. We have done the explicit computations and some optimal checkpoint positions are given in Table 4, where the values indicate distance from the ending points in units of t. The optimal positions are independent of $c_{\mathtt{H}} = mt^2/N$, assuming t to be large.

Table 5 lists the reduction in total F iterations that are expected when checkpoints are placed at the optimal positions. For example, the value 2.03% in the $c_{\mathtt{R}} = 1$ table

Table 5 Online cost reduction for Hellman tradeoffs through checkpoint usage

$c_{\mathtt{H}} = 1/2$	1 CP	2 CP	3 CP	4 CP
0 CP	1.71%	2.77%	3.50%	4.03%
1 CP		1.08%	1.82%	2.36%
2 CP			0.75%	1.30%
3 CP				0.55%

$c_{\mathtt{H}} = 1$	1 CP	2 CP	3 CP	4 CP
0 CP	3.17%	5.14%	6.49%	7.49%
1 CP		2.03%	3.43%	4.45%
2 CP			1.43%	2.47%
3 CP				1.06%

$c_{\mathtt{H}} = 2$	1 CP	2 CP	3 CP	4 CP
0 CP	5.56%	9.00%	11.37%	13.10%
1 CP		3.64%	6.15%	8.00%
2 CP			2.60%	4.51%
3 CP				1.96%

Table 6 Time reduction equivalent to increase in storage by single bit per entry

$b \to b+1$	$25 \to 26$	$30 \to 31$	$35 \to 36$	$40 \to 41$	$45 \to 46$	$50 \to 51$	$55 \to 56$
$1-(\frac{b}{b+1})^2$	7.54%	6.35%	5.48%	4.82%	4.30%	3.88%	3.54%

$b \to b+1$	$60 \to 61$	$65 \to 66$	$70 \to 71$	$75 \to 76$	$80 \to 81$	$85 \to 86$	$90 \to 91$
$1-(\frac{b}{b+1})^2$	3.25%	3.01%	2.80%	2.61%	2.45%	2.31%	2.19%

$b \to b+1$	$95 \to 96$	$100 \to 101$	$105 \to 106$	$110 \to 111$	$115 \to 116$	$120 \to 121$	$125 \to 126$
$1-(\frac{b}{b+1})^2$	2.07%	1.97%	1.88%	1.79%	1.72%	1.65%	1.58%

$b \to b+1$	$130 \to 131$	$135 \to 136$	$140 \to 141$	$145 \to 146$	$150 \to 151$	$155 \to 156$	$160 \to 161$
$1-(\frac{b}{b+1})^2$	1.52%	1.47%	1.41%	1.37%	1.32%	1.28%	1.24%

indicates that, compared to using a single checkpoint, using two checkpoints will result in 2.03% decrease in total online time.

Adding more checkpoints will require more storage space and may make the reduction in online time meaningless in view of the time memory tradeoff $TM^2 \propto N^2$. Table 6 translates storage disadvantage of a single bit per table entry increase into equivalent reduction in online time. Let us explain this with an example. We shall consider the "$95 \to 96$: 2.07%" entry. This meas that, if each $(\mathbf{sp}, \mathbf{ep})$ entry of the current tradeoff table consumes 95 bits, the storage disadvantage of adding one more checkpoint is justifiable only if you expect more that 2.07% decrease in online time. Thus, restricting our discussion to the $c_{\mathtt{H}} = 1$ case, use of a single 1-bit checkpoint, which is expected to bring 3.17% time reduction, is justifiable. But, if the 95 bits already contained a single checkpoint information, as adding one more checkpoint will reduce time only by 2.03%, doing so is not advisable. This example shows that, with Hellman tables at $c_{\mathtt{H}} = 1$, only a small number of checkpoints will be practical.

There are issues concerning storage that forces us to use Table 6 only as rough guidelines. We have already seen that, due to effects of false alarms, $TM^2 \propto N^2$ is only true when the parameters t and m are restricted to those that give constant $c_{\mathtt{H}} = \frac{mt^2}{N}$

(Hellman) or $c_{\mathtt{R}} = \frac{mt}{N}$ (rainbow) values. Furthermore, even if $TM^2 \propto N^2$ were strictly true, the M in this equation refers to the number of table entries, and not the real-world storage size. When M is increased, the number of bits that need to be allocated to each entry also increases, so the real-world storage size needed is not linear in M. In other words, an increase in storage, for example, by a factor of γ, does not translate exactly to disadvantage in time by a factor of γ^2. So these numbers should only be taken as a rough guideline. Of course, in practice, storage for checkpoints could come essentially for free, due to properties concerning natural word size of the implementation platform.

5.2 Perfect rainbow tradeoff

The general line of argument for analyzing the effects of checkpoints on perfect rainbow tradeoffs are exactly the same with the Hellman tradeoff case. We shall deal only with a single 1-bit checkpoint on a single perfect rainbow table. Extensions to multiple checkpoints and multi-bit checkpoints are straightforward.

If a single checkpoint is placed at the $(t-c)$-th column, for each $k > c$, the probability of meeting a false alarm on the k-th iteration is

$$\begin{aligned}
&\frac{1}{N}\Big\{ \big(|F^{-(k-c)}(\mathtt{RM}_{t-c})| - |\mathtt{RM}_{t-k}|\big) + \frac{1}{2}\big(|F^{-k}(\mathtt{RM}_t)| - |F^{-(k-c)}(\mathtt{RM}_{t-c})|\big) \Big\} \\
&\quad = \frac{1}{N}\Big\{ \big(|F^{-k}(\mathtt{RM}_t)| - |\mathtt{RM}_{t-k}|\big) - \frac{1}{2}\big(|F^{-k}(\mathtt{RM}_t)| - |F^{-(k-c)}(\mathtt{RM}_{t-c})|\big) \Big\}.
\end{aligned}$$

The second term within the braces,

$$\frac{1}{2N}\Big\{ |F^{-k}(\mathtt{RM}_t)| - |F^{-(k-c)}(\mathtt{RM}_{t-c})| \Big\}, \tag{21}$$

corresponds to the false alarms that are filtered out through the single checkpoint.

According to Proposition 4, we have

$$|F^{-k}(\mathtt{RM}_t)| = m(1+k)\big(1 - \frac{mk}{4N}\big) \tag{22}$$

and Proposition 5 gives

$$\begin{aligned}
&|F^{-(k-c)}(\mathtt{RM}_{t-c})| \\
&\quad = m(k-c+1) + \frac{(k-c)(k-c+2)}{c^2}\Big\{ mc + 2N \ln\Big(1 - \frac{mc}{2N}\Big) \Big\}.
\end{aligned} \tag{23}$$

When (22) and (23) are substituted into (21), it becomes

$$\frac{cm}{2N} - \Big\{ \frac{cm}{2N} + \ln\big(1 - \frac{cm}{2N}\big) \Big\} \frac{1}{c^2}(k-c)(k-c+2) - \frac{m^2}{8N^2} k(k+1) \tag{24}$$

After multiplying the work factor and the probability to reach the k-th iteration, we can state

$$\sum_{c<k\le t} (t-k+1)\Big(1 - \frac{m}{N}\Big)^{k-1} \cdot \big(\text{Equation (24)}\big), \tag{25}$$

Table 7 Expected iteration counts (unit: t^2) and effects of a single 1-bit checkpoint on perfect rainbow tradeoffs (FA=false alarm, itr=iterations)

$c_{\mathtt{R}}$	0.25	0.5	1	1.5	1.8	1.9	1.95	2
optimal c/t	0.3192	0.3048	0.2757	0.2472	0.2311	0.2259	0.2234	0.2209
filtered FA itr	0.0077	0.0128	0.0182	0.0197	0.0197	0.0197	0.0196	0.0195
FA itr	0.0357	0.0614	0.0920	0.1049	0.1076	0.1080	0.1081	0.1081
total itr	0.4597	0.4222	0.3562	0.3014	0.2734	0.2648	0.2607	0.2566
filtered/FA	21.5%	20.9%	19.8%	18.8%	18.3%	18.2%	18.1%	18.1%
filtered/total	1.67%	3.04%	5.10%	6.55%	7.22%	7.42%	7.52%	7.62%

as the number of F iterations that can be removed through a single 1-bit checkpoint at the $(t-c)$-th column.

Unlike the Hellman tradeoff case, (25) does not simplify into a nice formula. Noting that $c = O(t)$, let us write $c = c_{\mathtt{C}}t$. When this is used on (25), together with the condition $mt = c_{\mathtt{R}}N$ and the approximation $(1-\frac{c_{\mathtt{R}}}{t})^t \approx e^{-c_{\mathtt{R}}}$, it becomes a polynomial in t of degree two, with functions of $c_{\mathtt{R}}$ and $c_{\mathtt{C}}$ as coefficients. By concentrating on the coefficient of the most dominant t^2 term, we can find the optimal checkpoint position for each fixed $c_{\mathtt{R}}$. More information on this process is given in Appendix D.

The optimal $(t-c)$-th column to place a single 1-bit checkpoint is given in the first row of Table 7, for some values of $c_{\mathtt{R}}$, spread over its range of interest. The table also shows the reduction in online F invocations a checkpoint brings, when they are placed at the optimal position. Values such as iterations reduced through checkpoints, iterations due to false alarms, and expected numbers of total F iterations, per table, are listed. The count values have been approximated to multiples of t^2, but are quite accurate for large t.

Since the relative cost of false alarms is high on rainbow tradeoffs, the effects of a single checkpoint are better here, compared to the Hellman tradeoffs. As before, storage disadvantage of checkpoint addition, translated through Table 6, should be compared with time advantage. Because rainbow tables contain more entries than the naturally corresponding Hellman table, the bits per entry count will usually be larger with rainbow tables. Hence, use of multiple checkpoints will be easier to justify with rainbow tables.

5.3 Non-perfect rainbow tradeoff

Let us consider the non-perfect rainbow tradeoff. Recalling the arguments of Section 4.3, we must treat any colliding chains as totally independent.

The expected number of F iterations reduced through a single checkpoint at the $(t-c)$-th column can be computed as

$$\sum_{c<k\le t} (t-k+1)\cdot\frac{m}{N}\frac{c}{2}\cdot\prod_{j=1}^{k-1}\Big(1-\frac{m_{t-j}}{N}\Big).$$

Table 8 Expected iteration counts (unit: t^2) and effects of a single 1-bit checkpoint at the optimal position for non-perfect rainbow tradeoffs (FA=false alarm, itr=iterations)

$c_{\mathtt{R}}$	0.25	0.5	1	2	5	10	100
optimal c/t	0.3218	0.3117	0.2952	0.2724	0.2410	0.2234	0.2026
filtered FA itr	0.0082	0.0147	0.0250	0.0403	0.0755	0.1283	1.0512
FA itr	0.0372	0.0677	0.1167	0.1917	0.3656	0.6250	5.1326
total itr	0.4662	0.4443	0.4222	0.4208	0.5170	0.7431	5.2193
filtered/FA	21.9%	21.7%	21.4%	21.0%	20.7%	20.5%	20.5%
filtered/total	1.75%	3.31%	5.91%	9.57%	14.61%	17.27%	20.14%

Use of approximation (4) brings about cancelations within the last product term and the above becomes

$$\frac{m}{N}\frac{c}{2}\sum_{c<k\le t}(t-k+1)\cdot\frac{2N+m(t-k)}{2N+m(t-1)}\cdot\frac{2N+m(t-k-1)}{2N+m(t-2)}$$
$$\approx \frac{c_{\mathtt{R}}}{24}\cdot\frac{c}{t}\cdot\left(1-\frac{c}{t}\right)^2\cdot\left\{3c_{\mathtt{R}}^2\left(1-\frac{c}{t}\right)^2+16c_{\mathtt{R}}\left(1-\frac{c}{t}\right)+24\right\}\cdot\left(\frac{t}{c_{\mathtt{R}}+2}\right)^2.$$

One now has all the tools necessary to compute the optimal position to place a single 1-bit checkpoint. Table 8 lists the optimal $(t-c)$-th checkpoint column and various iteration counts for a non-perfect rainbow tradeoff. Note that, as $c_{\mathtt{R}}$ is increased, the cost of false alarms quickly dominates the main online chain creation cost and checkpoints become correspondingly effective.

6 Simulation Results

We tested the theory presented in this paper with experiments using small parameters. In all cases, the experiment data matched our theory very well.

The key to ciphertext mapping of the blockcipher AES-128 under a fixed random plaintext was used as the one-way function. We chose to fix the search space size to a manageable $N=2^{20}$. This was done by zero-padding the 20-bit input to a 128-bit key and truncating the 128-bit ciphertext to 20 bits.

In each of the tests described below, multiple tables were used and multiple inversion targets were tried per table. The one-way function was changed every time a new table was generated, by choosing another random plaintext.

Our theory on the probability of false alarms at each iteration and also that on probability of false alarm removal by checkpoints were tested. Each randomly generated tradeoff table was made to contain the information of a single 1-bit checkpoint. With the table ready, a random input $\mathbf{x}$ was generated and the online chain starting from the inversion target $\mathbf{y}=F(\mathbf{x})$ was created.

A set of t-many counters were prepared. For every false alarm observed during the online chain creation, the current online chain length was observed and the counter corresponding to the length was incremented. Dividing each counter content by the total number of tests that were run gives the probability of false alarm occurring at each iteration separately.

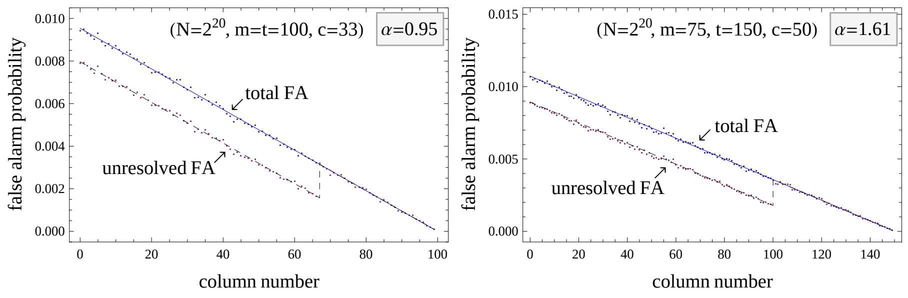


Fig. 1 Probability of false alarms at each column for Hellman tradeoffs

In addition to the usual counter incrementing, those false alarms that were resolved through the checkpoint information were made to increment a separate set of t-many counters. This allowed us to later present both the false alarm probability and the unresolved false alarm probability, both separately for each iteration.

In all the graphs below, the slightly irregular dots represent experiment data, averaged over multiple tables and inversion targets, and the lines, barely visible under the dots, represent our theory. Column number $t-k$ corresponds to false alarm probability at the k-th iteration. In each diagram, the upper graph, with theory given by a thin solid line, represents the total false alarm probability at each column. The lower graph, in a dashed line, corresponds to false alarms that remained unresolved even after checkpoint information was utilized. These two coincide to the right of the checkpoint.

6.1 Hellman

Refer to Figure 1. Each graph represents data averaged over 500 Hellman tables and 1000 random inversion targets per table. Even though infrequent, ending point collisions did occur during table generation. These were not removed and when an online chain matched the end of a pair of colliding chains, only one of them was regenerated to check for false alarm and the counter was incremented just once.

6.2 Perfect rainbow

Refer to Figure 2. Rather than fixing the table size m, or, equivalently, the number of collision free ending points, we chose to fix initial input m_0. The left hand side graph used $m_0 = 21020$ and the average of m_t observed over 30 tables was $m = 10471$, which translates to $c_{\mathtt{R}} \approx 1.0$. The right hand side graph used $m_0 = 502418$. The average of m_t over 30 tables was $m = 19902$ and we have $c_{\mathtt{R}} \approx 1.9$. Each of the tradeoff tables were subject to 5000 random inversion targets.

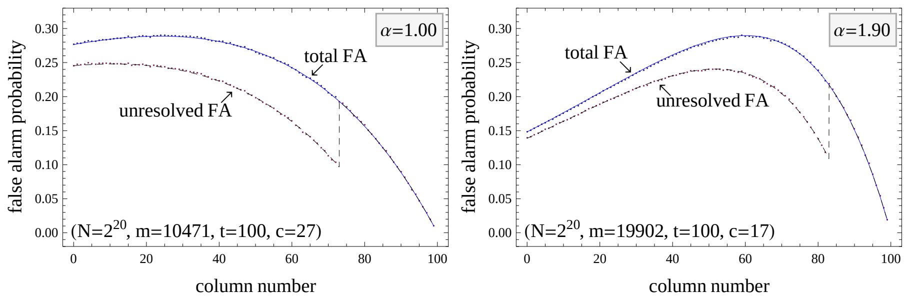


Fig. 2 Probability of false alarms at each column for perfect rainbow tradeoffs

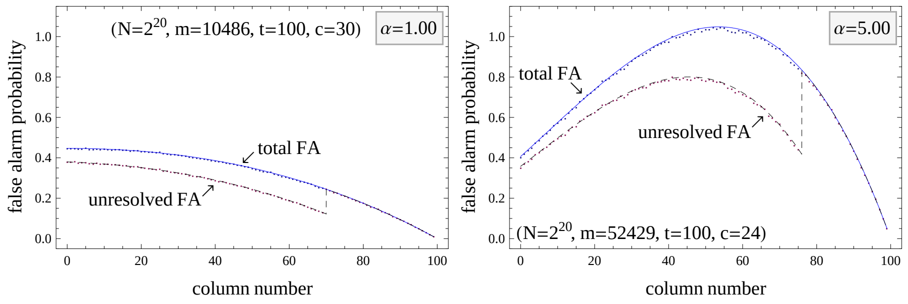


Fig. 3 Probability of false alarms at each column for non-perfect rainbow tradeoffs

6.3 Non-perfect rainbow

Refer to Figure 3. Both graphs represent data averaged over 30 tables with 5000 random inversion targets per table. Readers may notice that the right hand side graph shows false alarm probability slightly greater than 1 at the center columns and consider this to be strange. This is due to colliding ending points. Some false alarms will cause more than one pre-computation chain to be regenerated and the counters were incremented correspondingly many times.

7 Conclusion

Previous analyses of the online time complexities for cryptanalytic time memory tradeoff algorithms were usually based on the worst case operation of just the online chain creation process and the added cost of dealing with false alarms was either neglected or roughly argued as being relatively small.

In this work, we presented accurate measures of the expected online time complexities, for the Hellman and rainbow tradeoffs. By studying the size of pre-image sets under an iteration of random functions, the cost induced by false alarms were analyzed and taken into account. An analysis of the workings of the checkpoint method, a technique for reducing the effects of false alarms, was also provided. We have computed their optimal positions and quantified the resulting reduction in false alarm costs.

For those familiar with the distinguished point method, we remark that more work needs to be done in order to give an analysis similar to the current work. There are many complications arising from the existence of distinguished points within the pre-image tree. This is one direction the current work may be extended to.

In the introduction, we briefly mentioned the issue of finding the optimal storage technique for each tradeoff algorithm. Unlike the time complexity analysis given in this paper, which was largely independent of storage issues, storage reduction techniques affect both storage and time complexities in a way that is not accessible through the tradeoff curves. The less understood technique of storage reduction through aggressive ending point truncation brings about extra work in a probabilistic manner, just as with false alarms. Analysis of its effects and its optimal usage will only be possible in conjunction with time complexity arguments provided in this work. Note that the tradeoff curves available today present a relation between the *online chain iteration count* and the *table entry count*. Further research in this direction will lead to the more useful tradeoff curve connecting the *total online iteration count* and the *real-world storage size*.

References

1. G. Avoine, P. Junod, and P. Oechslin, Time-memory trade-offs: False alarm detection using checkpoints. *Indocrypt 2005,* LNCS 3797, pp.183–196, Springer-Verlag, 2005.
2. G. Avoine, P. Junod, and P. Oechslin, Time-memory trade-offs: False alarm detection using checkpoints (extended version). Technical Report LASEC-REPORT-2005-002, Swiss Federal Institute of Technology (EPFL), Security and Cryptography Laboratory (LASAC), Lausanne, Switzerland, September 2005.
3. S. H. Babbage, Improved exhaustive search attacks on stream ciphers. *European Convention on Security and Detection,* IEE Conference publication No. 408, pp.161–166, IEE, 1995.
4. E. Barkan, E. Biham, and A. Shamir, Rigorous bounds on cryptanalytic time/memory tradeoffs. *Crypto 2006,* LNCS 4117, pp.1–21, Springer-Verlag, 2006.
5. A. Biryukov, S. Mukhopadhyay, and P. Sarkar, Improved time-memory trade-offs with multiple data. *SAC 2005,* LNCS 3897, pp.110–127, Springer-Verlag, 2006.
6. A. Biryukov and A. Shamir, Cryptanalytic time/memory/data tradeoffs for stream ciphers. *Asiacrypt 2000,* LNCS 1976, pp.1–13, Springer-Verlag, 2000.
7. D. E. Denning, *Cryptography and Data Security* (p.100, Ron Rivest's distinguished points observation). Addison-Wesley, 1982.
8. A. Fiat and M. Naor, Rigorous time/space tradeoffs for invering functions. *SIAM J. on Computing,* **29**, no 3, pp.790–803, SIAM, 1999.
9. P. Flajolet and A. M. Odlyzko, Random mapping statistics. *Eurocrypt 1989,* LNCS 434, pp.329–354, Springer-Verlag, 1990.
10. J. Dj. Golić, Cryptanalysis of alleged A5 stream cipher. *Eurocrypt 1997,* LNCS 1233, pp.239–255, Springer-Verlag, 1997.
11. M. E. Hellman, A cryptanalytic time-memory trade-off. *IEEE Trans. on Infor. Theory,* **26** (1980), pp.401–406.
12. J. Hong, K. C. Jeong, E. Y. Kwon, I.-S. Lee, and D. Ma, Variants of the distinguished point method for cryptanalytic time memory trade-offs. *ISPEC 2008,* LNCS 4991, Springer-Verlag, pp.131–145, 2008.
13. I.-J. Kim and T. Matsumoto, Achieving higher success probability in time-memory trade-off cryptanalysis without increasing memory size. *IEICE Trans. Fundamentals,* **E82-A**, pp.123–129, 1999.
14. D. Ma, Studies on the cryptanalytic time memory trade-offs. Ph.D. Thesis, Seoul National University, Aug., 2008.
15. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.
16. P. Oechslin, Making a faster cryptanalytic time-memory trade-off. *Crypto 2003,* LNCS 2729, pp.617–630, 2003.

A Another approach to $\mathscr{P}_u(x)$ as a function

In this section, we give another interpretation for some of the contents that start out Section 3.2.

For the random function F on $\mathscr{N}$, its u-times iterated composition F^u is a function with the property that the number of its i-nodes is $N\,p_{u,i}$. Let D be a random set of m_0 elements and consider the image set $F^u(D)$. Given any specific i-node, it does not belong to $F^u(D)$ if and only if none of its i-many pre-images belong to the input set. Thus the probability of an i-node not being in $F^u(D)$ is $(1-\frac{m_0}{N})^i$. The number of i-nodes that are non-images may thus be written as $(N\,p_{u,i})\cdot\big(1-\frac{m_0}{N}\big)^i$.

Equating two ways to express the number of non-images under F^u, we obtain

$$N - m_u = \sum_i (N\,p_{u,i})\cdot\big(1-\frac{m_0}{N}\big)^i,$$

where m_u is the expected size of $F^u(D)$. This equation is equivalent to the $k=0$ case of Lemma 2. The positive k case of Lemma 2 also follows by simply taking the above D to be of size m_k.

So far we have shown

$$1-\frac{m_u}{N} = \mathscr{P}_u\big(1-\frac{m_0}{N}\big)$$

to be true, when $\frac{m_0}{N}$ is non-negative and reasonably small. If we can agree that enlarging the input set to F^u will always result in strictly larger output set, at least on the expectation level, we may conclude that the function $\mathscr{P}_u(x)$ is injective in some small range $1-\varepsilon < x \le 1$. Thus a suitable inverse function can be defined and we can state the negative k case of Lemma 2.

With Lemma 2 in our hands, we have a strong connection between function $\mathscr{P}_u$ and image sizes under function iterations. This allows us to use (3) to states that, as a function, $\mathscr{P}_1(x)$ is identical to $\exp(1-x)$ in the range discussed above. Furthermore (3) also shows that $\mathscr{P}_u$ is the u-times iterated composition of $\mathscr{P}_1$. We have thus reobtained Lemma 1 and Proposition 2.

Now, combining (10), which is true by definition, with Proposition 2, we can state

$$\begin{aligned} p_{u+v,i} &= \sum_j p_{v,j}\cdot p\big(\overset{u}{\to}\overset{v}{\to}_i \,\big|\, \cdot \quad \overset{v}{\to}_j *\big) \\ &= \sum_j p_{v,j} \sum_{i_1+\cdots+i_j=i} p_{u,i_1}\,p_{u,i_2},\cdots p_{u,i_j}\,. \end{aligned}$$

This equation is a strong indication that Proposition 1 and Proposition 3 are true, but does not directly imply them in the logical sense. This seems to be as far as we can go, starting with the approach of this section.

B More on node size correlation

We state two conditional probabilities that resemble Proposition 1, in the sense that the random choice concerns the image point, rather than the input. Both statements are consequences of the observation (10). We ask readers to compare these with probabilities given by Proposition 3.

Proposition 6 *Some conditional probabilities concerning node structures are as follows, where the summation indices are to be taken from non-negative integers.*

1. *Fix $i>0$ and let a random (F^{u+v},i)-node be given. If one of its i-many pre-images under F^v is chosen at random, the probability for this to be an (F^u,k)-node is*

$$p\big(\overset{u}{\to}_k * \quad \cdot \,\big|\, \overset{u}{\to}\overset{v}{\to}_i *\big) = \frac{1}{p_{u+v,i}} \sum_{j=1}^{\infty} \Big\{ \sum_{k+i_2+\cdots+i_j=i} p_{u,k}p_{u,i_2}\cdots p_{u,i_j} \Big\}\cdot p_{v,j}\,.$$

When $k<i$, the second summation is empty for $j=1$. Likewise, when $k>i$, the second summation is empty for each j. These empty sums are interpreted to be zero. When $k=i$, the second summation is degenerate for $j=1$ and is interpreted to be one.

2. *The probability for a random (F^{u+v}, i)-node to be an (F^v, j)-node is*

$$p\big(\cdot \quad \overset{v}{\rightarrow}_j \,\big|\, \overset{u}{\rightarrow}\overset{v}{\rightarrow}_i *\big) = \frac{p_{v,j}}{p_{u+v,i}} \cdot \sum_{i_1+\cdots+i_j=i} p_{u,i_1} p_{u,i_2} \cdots p_{u,i_j}.$$

   When $i = j = 0$, the degenerate sum is interpreted to be one. When $i > 0$ with $j = 0$, the empty sum is interpreted to be zero.

The following proposition is only provided for completeness. This should be easy to prove when the proof of Proposition 3 is understood.

Proposition 7 *Let $\mathbf{x}$ be a random point from $\mathscr{N}$. Set $\mathbf{y} = F^u(\mathbf{x})$ and $\mathbf{z} = F^v(\mathbf{y})$. In the statements below, the summation indices are to be taken from the non-negative integers.*

1. *Assuming $\mathbf{y}$ to be an (F^u, k)-node, the probability for $\mathbf{z}$ to be an (F^{u+v}, i)-node is*

$$p\big(\overset{u}{\rightarrow}\overset{v}{\rightarrow}_i \,\big|\, *\overset{u}{\rightarrow}_k \quad \cdot\big) = \sum_{j=1}^{\infty} j\, p_{v,j} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2} \cdots p_{u,i_j}.$$

2. *Assuming $\mathbf{z}$ to be an (F^v, j)-node, the probability for $\mathbf{z}$ to be an (F^{u+v}, i)-node is*

$$p\big(\overset{u}{\rightarrow}\overset{v}{\rightarrow}_i \,\big|\, * \quad \overset{v}{\rightarrow}_j\big) = \sum_{k=1}^{\infty} k\, p_{u,k} \sum_{k+i_2+\cdots+i_j=i} p_{u,i_2} \cdots p_{u,i_j}.$$

   Note that $j \neq 0$.

Any sum with an empty index set is interpreted to be zero and any sum with a degenerate index description is interpreted to be one.

C Alternative derivation for the equivalence set size

In this section, we present another proof for Proposition 4, giving the expected size of an F^k-equivalence set. The line of argument follows that of [1], where the special case of maximal perfect rainbow tradeoff was considered.[2]

Let `TM` be a tradeoff matrix of size $m \times (t+1)$. It may be either Hellman or rainbow. In the rainbow case, it may be either perfect, non-perfect, or anywhere in between. That is, we assume `TM` was created with random inputs and if any of the chains were removed, at least one chain with the same ending point remains in the matrix. The k-th column of `TM` is denoted by $\mathtt{TM}_k$.

Let $\mathbf{y} \in \mathscr{N}$ be random and consider the online chain

$$\mathbf{y} = \mathbf{x}_{t-k} \xrightarrow{F} \mathbf{x}_{t-k+1} \xrightarrow{F} \cdots \xrightarrow{F} \mathbf{x}_t$$

of length k. We assume that the right end of this chain is aligned to the ending points of `TM`, in the sense that, for rainbow tables, the *colors* of the iterating functions are in match with the right end of the rainbow matrix. Let us say that this online chain *merges* with one of the chains from `TM`, if $\mathbf{x}_j \in \mathtt{TM}_j$ for some j.

Proposition 8 *The probability of online chain merging with one of the chains from* `TM` *is given by*

$$1 - \prod_{j=0}^{k} \big(1 - \frac{m_{t-j}}{N}\big),$$

where m_t is the number of distinct elements of $\mathtt{TM}_t$ and other m_i are defined recursively by following through (3) *in the reverse direction.*

[2] That the argument of [1] should be valid under more general circumstances was brought to the author's attention through an early version of [14].

We can argue with Lemma 2 and (18) that this proposition is equivalent to Proposition 4. Proof of the current version, to be given below, is a tricky exercise on what a random function is. We mention that issues discussed in Remark 1 and Remark 2 also apply to the current proof.

Note that m_0 is the number of random inputs, that would have naturally produced the m_t distinct ending points under the random function, and the intermediate m_i give the intermediate image sizes.

Suppose we are in the process of constructing a random function F. We start with D_0, a set of m_0 random starting points. The function value on each point of D_0 is defined to be a randomly chosen element of $\mathcal{N}$. The image set D_1 will contain m_1 distinct elements. We then define the value of F on each point of D_1 and so on. When t and the size of D_i are assumed to be small enough, previous definitions will not interfere with later definitions of the random function F. We stop after t steps, ending with m_t distinct ending points D_t.

After the random function F has been constructed this far, we continue defining F further with another random chain. Probability for the randomly chosen starting point $\mathbf{y} = \mathbf{x}_{t-k}$ to be not in D_{t-k} is $1 - \frac{m_{t-k}}{N}$. The probability of the randomly chosen image $F(\mathbf{x}_{t-k}) := \mathbf{x}_{t-k+1}$ to be not in D_{t-k+1} is $1 - \frac{m_{t-k+1}}{N}$. Since each iterative definition of F during the creation of the online chain consists of choosing a random point from $\mathcal{N}$, the iterations are independent of each other. Thus, the probability of the online chain not meeting with the pre-constructed chains, all the way to the end, is as claimed and the proof is complete.

One should not confuse D_{t-j} with the pre-image $F^{-j}(D_t)$, which would be much larger. In fact, during the process of constructing the random function, the pre-image is not fully defined. Also note that once the online chain ends without merging with $\mathtt{TM}$ chains, completing the construction of F by defining its value on all yet undefined inputs will have no effect on the non-merging state.

Even though the proof given here requires less machinery than the proof in the main body of this paper, the longer proof is more constructive in that it deals with the true pre-image sizes. In addition, a parallel proof for Proposition 5 does not seem to be in reach. For the perfect rainbow tradeoff, an input set that naturally produces the intermediate column $\mathtt{RM}_k$ is hard to devise without considering probabilities related to colliding chain removals.

D Optimal checkpoint position for the perfect rainbow tradeoff

In this section, we show how to obtain the optimal position for a single 1-bit checkpoint in the perfect rainbow tradeoff case.

Recall that (25) gives the number of online iterations that can be removed through a single 1-bit checkpoint at the $(t-c)$-th column. Under the conditions $mt = c_{\mathtt{R}}N$, $c = c_{\mathtt{C}}t$, and the approximation $(1 - \frac{c_{\mathtt{R}}}{t})^t \approx e^{-c_{\mathtt{R}}}$, the sum (25) becomes the following polynomial in t of degree two.

$$
\left\{
\begin{aligned}
&\frac{1}{8c_{\mathtt{C}}c_{\mathtt{R}}}\Big(-\frac{8+c_{\mathtt{C}}^2c_{\mathtt{R}}^2}{e^{c_{\mathtt{C}}c_{\mathtt{R}}}}+\frac{8+8c_{\mathtt{R}}-8c_{\mathtt{C}}c_{\mathtt{R}}+c_{\mathtt{C}}c_{\mathtt{R}}^2}{e^{c_{\mathtt{R}}}}\Big)\\
&-\frac{2}{c_{\mathtt{C}}^2c_{\mathtt{R}}^2}\Big(\frac{1}{e^{c_{\mathtt{C}}c_{\mathtt{R}}}}+\frac{-1-c_{\mathtt{R}}+c_{\mathtt{C}}c_{\mathtt{R}}}{e^{c_{\mathtt{R}}}}\Big)\ln\Big(1-\frac{c_{\mathtt{C}}c_{\mathtt{R}}}{2}\Big)
\end{aligned}
\right\}
$$
$$
+\left\{
\begin{aligned}
&\frac{1}{8c_{\mathtt{C}}c_{\mathtt{R}}^2}\left(\begin{aligned}&-\frac{16+4c_{\mathtt{R}}+2c_{\mathtt{C}}c_{\mathtt{R}}-c_{\mathtt{C}}^2c_{\mathtt{R}}^2+c_{\mathtt{C}}^2c_{\mathtt{R}}^3}{e^{c_{\mathtt{C}}c_{\mathtt{R}}}}\\&+\frac{16+8c_{\mathtt{R}}-2c_{\mathtt{C}}c_{\mathtt{R}}+4c_{\mathtt{R}}^2-5c_{\mathtt{C}}c_{\mathtt{R}}^2+c_{\mathtt{C}}c_{\mathtt{R}}^3}{e^{c_{\mathtt{R}}}}\end{aligned}\right)\\
&+\frac{1}{c_{\mathtt{C}}^2c_{\mathtt{R}}^3}\left(\begin{aligned}&\frac{-4-c_{\mathtt{R}}+c_{\mathtt{C}}c_{\mathtt{R}}}{e^{c_{\mathtt{C}}c_{\mathtt{R}}}}\\&+\frac{4+2c_{\mathtt{R}}-2c_{\mathtt{C}}c_{\mathtt{R}}+c_{\mathtt{R}}^2-2c_{\mathtt{C}}c_{\mathtt{R}}^2+c_{\mathtt{C}}^2c_{\mathtt{R}}^2}{e^{c_{\mathtt{R}}}}\end{aligned}\right)\ln\Big(1-\frac{c_{\mathtt{C}}c_{\mathtt{R}}}{2}\Big)
\end{aligned}
\right\}\times t
$$

$$+\left\{\begin{array}{l} \dfrac{1}{8c_{\mathtt{C}}c_{\mathtt{R}}^3}\left(\dfrac{\left(\begin{array}{c}24-8c_{\mathtt{R}}+14c_{\mathtt{C}}c_{\mathtt{R}}-2c_{\mathtt{C}}c_{\mathtt{R}}^2+2c_{\mathtt{C}}^2c_{\mathtt{R}}^2\\ +2c_{\mathtt{C}}^2c_{\mathtt{R}}^3-c_{\mathtt{C}}^3c_{\mathtt{R}}^3-c_{\mathtt{C}}^3c_{\mathtt{R}}^4+c_{\mathtt{C}}^4c_{\mathtt{R}}^4\end{array}\right)}{e^{c_{\mathtt{C}}c_{\mathtt{R}}}} - \dfrac{24+16c_{\mathtt{R}}-10c_{\mathtt{C}}c_{\mathtt{R}}+4c_{\mathtt{R}}^2-4c_{\mathtt{C}}c_{\mathtt{R}}^2+c_{\mathtt{C}}c_{\mathtt{R}}^3}{e^{c_{\mathtt{R}}}}\right) \\ +\dfrac{1}{c_{\mathtt{C}}^2c_{\mathtt{R}}^4}\left(\dfrac{6-2c_{\mathtt{R}}+2c_{\mathtt{C}}c_{\mathtt{R}}}{e^{c_{\mathtt{C}}c_{\mathtt{R}}}} - \dfrac{6+4c_{\mathtt{R}}-4c_{\mathtt{C}}c_{\mathtt{R}}+c_{\mathtt{R}}^2-2c_{\mathtt{C}}c_{\mathtt{R}}^2+c_{\mathtt{C}}^2c_{\mathtt{R}}^2}{e^{c_{\mathtt{R}}}}\right)\ln\left(1-\dfrac{c_{\mathtt{C}}c_{\mathtt{R}}}{2}\right) \end{array}\right\}\times t^2$$

For each fixed t and $c_{\mathtt{R}}$, we want to find the $c_{\mathtt{C}}$ that maximizes the value of this polynomial.

Let us write this polynomial as

$$A(c_{\mathtt{C}},c_{\mathtt{R}})t^2+B(c_{\mathtt{C}},c_{\mathtt{R}})t+C(c_{\mathtt{C}},c_{\mathtt{R}}).$$

Using any software for drawing graphs, one can quickly check that for parameters $0.01<c_{\mathtt{C}}<0.99$ and $0.01<c_{\mathtt{R}}<2$, we have

$$0<\frac{B(c_{\mathtt{C}},c_{\mathtt{R}})}{A(c_{\mathtt{C}},c_{\mathtt{R}})}<962 \quad\text{and}\quad -84.6<\frac{C(c_{\mathtt{C}},c_{\mathtt{R}})}{A(c_{\mathtt{C}},c_{\mathtt{R}})}<81.6.$$

In particular, this shows that, for t which is not too small, $A(c_{\mathtt{C}},c_{\mathtt{R}})t^2$ will dominate $B(c_{\mathtt{C}},c_{\mathtt{R}})t$ and $C(c_{\mathtt{C}},c_{\mathtt{R}})$ for typical $c_{\mathtt{C}}$ and $c_{\mathtt{R}}$. Also, since checkpoints are useless at both $c_{\mathtt{C}}=0$ and $c_{\mathtt{C}}=1$, the polynomial maximum will not be obtained near the boundary points. Thus, for $c_{\mathtt{C}}$ of interest and typical $c_{\mathtt{R}}$, the coefficients of this polynomial are such that the polynomial may be approximated simply by taking just the dominating highest term.

Finally, since for any fixed t, the single term approximation $A(c_{\mathtt{C}},c_{\mathtt{R}})t^2$ attains its maximum when the coefficient $A(c_{\mathtt{C}},c_{\mathtt{R}})$ is maximal, given any $c_{\mathtt{R}}$, it is now easy to numerically find $c_{\mathtt{C}}$ that maximizes the polynomial.