

# On the Probability Distribution of Superimposed Random Codes

Bernd Günther

*This paper has been published with copyright by IEEE  
IEEE Trans. Inf. Theory, 54(7):3206–3210, 2008; DOI 10.1109/TIT.2008.924658.*

## Index Terms

Database indexing, False drop estimates, Generating functions, Probability distribution, Superimposed coding

## Abstract

A systematic study of the probability distribution of superimposed random codes is presented through the use of generating functions. Special attention is paid to the cases of either uniformly distributed but not necessarily independent or non uniform but independent bit structures. Recommendations for optimal coding strategies are derived.

## I. INTRODUCTION

Chemical structure retrieval systems are frequently presented with the task to produce a list of all stored chemical graphs containing a prescribed subgraph[1], [2]. Due to the absence of a linear order among the stored data tree based search strategies fail, and a sequential search has to be performed. To accelerate this time consuming process, the actual graph theoretical substructure match is preceded by *prescreening*: the entire database is matched against a library of simple but common *descriptors*, and the validity of descriptors is recorded in a bitstring for each stored structure. Suitable choices for descriptors are small chemical subgraphs containing only few vertices, graph diameters, ring sizes or any other property that passes from subgraphs to supergraphs. When a query structure is submitted to the system, the descriptors are evaluated for this query structure resulting in a query bit string. Only those stored structures are candidates for a match where each bit is turned on in all those positions where the query bits are turned on and will be subjected to the expensive graph theoretical matching algorithm.

For example, let us consider the compounds in table I. A chemist might ask for a list of all structures in our database containing 2-(cyclohexylmethyl)naphthalene, which is too complex to be one of the index descriptors. However, any matching structure must necessarily contain cyclohexane and naphthalene, and these might be indexed. We will produce an intermediate result set that also contains 2-(2-cyclohexylethyl)naphthalene, which is not in accordance with the original query specification and must be singled out by graph matching.

The Beilstein database of organic compounds contains around 10 million structures, each a chemical graph of up to 255 vertices, and the number of descriptors will have the magnitude of one thousand. With such characteristics, the preevaluated bitstrings will consume a considerable amount of storage and hence of processing time. On the other hand, one may expect the 1-bits to be relatively scarce, whence it should be possible to compress the bitstrings without losing too much information. Thus, we are looking for a map  $\psi : \dot{I}^N \rightarrow \dot{I}^n$ ,  $\dot{I} = \{0, 1\}$  transforming bitstrings of length  $N$  into strings of length  $n$ . However,

Beilstein-Institut, Trakehner Straße 7-9, 60487 Frankfurt, Germany.

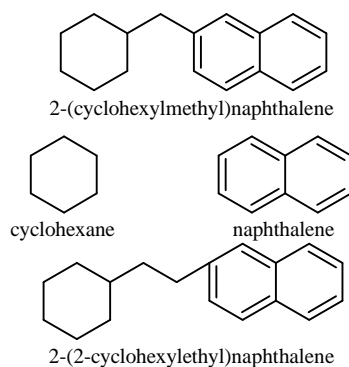


TABLE I

we have to observe the partial order relation  $\beta \leq \beta'$  between bitstrings defined such that the bits should satisfy  $\beta(i) \leq \beta'(i)$  at all positions  $i$ . Since we want to use the compressed strings in the same manner (by bitmasking) as the original ones the transformation must be monotone:  $\beta \leq \beta' \Rightarrow \psi(\beta) \leq \psi(\beta')$  and in particular  $\psi(\beta) \vee \psi(\beta') \leq \psi(\beta \vee \beta')$ , where the wedge denotes the bitwise inclusive or operation. If the latter inequality was strict information would be lost, hence we claim:

$$\psi(\beta) \vee \psi(\beta') = \psi(\beta \vee \beta') \quad (1)$$

for any two source strings  $\beta, \beta' \in \dot{I}^N$ . This is the defining condition of superimposed coding. If we denote by  $\beta_j \in \dot{I}^N$  the elementary string having bit 1 in position  $j$  and 0 elsewhere and set  $\psi_j := \psi(\beta_j) \in \dot{I}^n$  equal to the code word assigned to bit  $j$ , then we must have

$$\psi(\beta) = \bigvee_{j \in \beta^{-1}(1)} \psi_j, \quad (2)$$

explaining the term. Here  $\beta^{-1}(1)$  is the inverse image of 1, i.e. the set of all positions  $i$  where the  $i$ -th bit  $\beta(i)$  is turned *on*. As summary we emphasize: in our context superposition is not a matter of choice but a requirement.

We speak of *superimposed random coding* if the code words  $\psi_j$  are constructed using a random number generator. Non random approaches have been considered e.g. in [3], [4] and perform very well when the number of code words superposed in equation (2) is bounded, but are unpredictable beyond this barrier. The random approach was probably initiated by [5], but there an invalid probability analysis was given. A thorough study of the case where source bitstrings and code words are of fixed weight was given in [6]. Bloom filters (cf. [7] and [8, p.572]) constitute an application of this technique where the primary keys of a database are encoded in a bitstring; it should be noted that the approach favored by us produces a twodimensional bit array for the entire database, one bitstring for each record. Despite the continuing popularity of the subject in context of chemical substructure search a broader systematic approach seems to be lacking, a gap that is intended to be filled by the current paper. One question that has to be addressed is the optimal choice of codes. It is known at least since Roberts' paper [6] that the target bitstrings should contain a more or less even balance of 0 and 1-bits, but if we want to achieve this other than by try and error we must study the distribution of target bits. Of particular practical importance is the situation where the source bits are turned on with non uniform probability. A few statements hereabout are by Roberts in [6] but are based on intuition, not computation, and in fact we disagree with his conclusion. In [9] the author gives recommendations about the selection of descriptors, whereas we set ourselves the task of adapting the coding design to a given set of descriptors.

We must notice that the notion "random coding" has an inherent semantic difficulty. It must be understood clearly that coding and decoding of a bit pattern have to be performed within one and the same run of the code generation random experiment. After each coding-decoding cycle the code generation should be repeated independently. However, this procedure does not exactly fit when recording bit patterns in databases: here the codes must be fixed once and for all. Since a large set of  $N$  codes is needed, we might expect good statistical behavior in this context too.

The observant reader will have noticed that equation (2) allows for a target string  $\psi(\beta)$  to be covered by a comparison pattern,  $\psi(\beta) \leq \psi(\gamma)$ , even if the source patterns do not cover:  $\beta \not\leq \gamma$ . This will result in the inclusion of fake hits in our result sets. As long as their number is small this is acceptable, because the bit comparison will be followed by graph matching anyway. However, we want to predict and minimize this number.

In section II we will develop the probabilistic tools necessary to manage our random bit strings. In section III we demonstrate how to compute the distribution of the target bits from those of source and code bits, and binomially distributed and fixed weight code words are introduced as primary examples for code generation. This settles the case of uniform but not necessarily independent source bit distributions. The requirement of uniformity will be dropped in section IV but the assumption of independence will be added. The completely general case will be addressed in section V.

## II. ISOTROPIC BIT DISTRIBUTIONS

Let's consider a fixed source bit pattern  $\beta \in \dot{I}^N$  and a fixed target bit pattern  $\alpha \in \dot{I}^n$ ; then

$$P(\psi(\beta) \leq \alpha) = P(\forall j \in \beta^{-1}(1) : \psi_j \leq \alpha) \quad (3)$$

$$= \prod_{j \in \beta^{-1}(1)} P(\psi_j \leq \alpha), \quad (4)$$

where the probability is that of the code generation. Now varying the source pattern  $\beta$  independently we obtain an expression for the probability of target patterns:

$$P(\psi(-) \leq \alpha) = \sum_{\beta \in \dot{I}^N} P(\beta) \prod_{j \in \beta^{-1}(1)} P(\psi_j \leq \alpha). \quad (5)$$

*Definition 1:* We call a probability distribution on  $\dot{I}^n$  *isotropic*, if it depends only on the number of 1-bits but not on their position.

This means that the distribution is invariant under all coordinate permutations (In [10] the term homogeneous is used). As is evident from equation (5), the distribution of the target patterns is isotropic if the code generation is, even if the source patterns are non isotropically distributed. Since this observation leads to a considerable simplification of our analysis we now give a short exposition of isotropic distributions.

We are given numbers  $p_0, \dots, p_n \geq 0$  with  $\sum_{k=0}^n \binom{n}{k} p_k = 1$  such that the probability of a particular bit pattern  $\alpha \in \dot{I}^n$  is given by  $P(\alpha) = p_a$  with  $a := \#\alpha^{-1}(1)$ . Our main analytical tool will be the probability generating function

$$f(t) := \sum_{k=0}^n \binom{n}{k} p_k t^k, \quad (6)$$

this being the standard definition obtained by considering the number of 1-bits as random variable. For fixed  $\alpha \in \dot{I}^n$  with  $a := \#\alpha^{-1}(1)$  we define

$$F_a := P(\xi \leq \alpha) = \sum_{k=0}^a \binom{a}{k} p_k \quad (7)$$

$$G_a := P(\xi \geq \alpha) = \sum_{k=a}^n \binom{n-a}{k-a} p_k. \quad (8)$$

Of course the quantities  $F_a$  and  $G_{n-a}$  are dual to each other, in fact the one is transformed into the other by switching 0 and 1-bits. They play very different roles for us, though: notice the occurrence of  $F_a$  on the right hand side of equation (5).  $G_a$  is the relative number of candidates that will be selected by prescreening with bit pattern  $\alpha$ , because the condition  $\xi \geq \alpha$  singles out precisely those patterns  $\xi$  that have 1-bits in the same positions as  $\alpha$ , and probably some more. We define generating functions as follows:

$$F(t) := \sum_{m=0}^n \binom{n}{m} F_m t^m \quad (9)$$

$$G(t) := \sum_{k=0}^n \binom{n}{k} G_{n-k} t^k \quad (10)$$

The following relations are readily established:

$$F(t) = (1+t)^n f\left(\frac{t}{1+t}\right) \quad (11)$$

$$G(t) = (-1)^n F(-1-t) \quad (12)$$

$$G(t) = t^n f\left(\frac{1+t}{t}\right) \quad (13)$$

$$p_m = \sum_{k=0}^m (-1)^{m+k} \binom{m}{k} F_k \quad (14)$$

$$G_m = \sum_{k=0}^m (-1)^k \binom{m}{k} F_{n-k} \quad (15)$$

The three sets of coefficients  $p_m$ ,  $F_m$  and  $G_m$  carry the same information and can be readily converted into each other using the above relations. The reader will certainly notice the difference between (11) and the standard situation [11, Thm. XI.1], which arises from the fact that the values  $\alpha$  appearing in definition (7) are partially but not linearly ordered.

For later reference we need the derivatives of  $f$  at 1. By (13) we have  $f(1+\varepsilon) = \varepsilon^n G\left(\frac{1}{\varepsilon}\right) = \sum_{k=0}^n \binom{n}{k} G_k \varepsilon^k$ . This allows to read off the Taylor coefficients of the function  $f$  at the point 1 at once:

$$\frac{f^{(k)}(1)}{k!} = \binom{n}{k} G_k \quad (16)$$

We define the moments  $\mu_m = \sum_{k=0}^n k^m \binom{n}{k} p_k$  of our distribution as usual and remember their generating function[11, Exc. XI.7.24]:

$$f(e^t) = \sum_{m=0}^{\infty} \frac{\mu_m}{m!} t^m. \quad (17)$$

Equations (16) and (17) allow to express the first two moments in terms of the distribution coefficients  $F_m$ :

$$\mu_1 = n(1 - F_{n-1}) = nG_1 \quad (18)$$

$$\begin{aligned} \mu_2 &= n[n - (2n-1)F_{n-1} + (n-1)F_{n-2}] \\ &= n[(n-1)G_2 + G_1]. \end{aligned} \quad (19)$$

The variance evaluates to

$$\mu_2 - \mu_1^2 = n [F_{n-1} - nF_{n-1}^2 + (n-1)F_{n-2}], \quad (20)$$

whence in particular:

$$F_{n-2} \geq (n-1)^{-1} (nF_{n-1} - 1) F_{n-1}. \quad (21)$$

Two examples are of primary importance in our context:

**Example 1:** The binomial distribution  $p_m = (1-q)^m q^{n-m}$  with parameter  $1-q$ . Here we have:

$$f(t) = (q + (1-q)t)^n \quad (22)$$

$$F(t) = (q + t)^n \quad (23)$$

$$G(t) = (1-q + t)^n \quad (24)$$

$$F_m = q^{n-m} \quad (25)$$

$$G_m = (1-q)^m \quad (26)$$

$$\mu_1 = n(1-q) \quad (27)$$

$$\mu_2 = n(1-q) [n(1-q) + q] \quad (28)$$

$$\mu_2 - \mu_1^2 = nq(1-q). \quad (29)$$

**Example 2:** Fixed weight. Here only bit patterns with a fixed number  $w$  of 1-bits are permitted. We get:

$$p_m = \begin{cases} \binom{n}{w}^{-1} & m = w \\ 0 & m \neq w \end{cases} \quad (30)$$

$$f(t) = t^w \quad (31)$$

$$F(t) = (1+t)^{n-w} t^w \quad (32)$$

$$G(t) = (1+t)^w t^{n-w} \quad (33)$$

$$F_m = \begin{cases} \frac{\binom{n-w}{n-m}}{\binom{n}{m}} & m \geq w \\ 0 & m < w \end{cases} \quad (34)$$

$$G_m = \begin{cases} \frac{\binom{w}{m}}{\binom{n}{m}} & m \leq w \\ 0 & m > w \end{cases} \quad (35)$$

$$\mu_m = w^m \quad (36)$$

$$\mu_2 - \mu_1^2 = 0. \quad (37)$$

Hardly surprising, inequality (21) is sharp for this distribution.

### III. UNIFORM CODE WORD GENERATION

Denoting the target distribution's coefficients by  $\check{F}_a$  equation (5) can be written as:

$$\check{F}_a = \sum_{\beta \in \check{I}^N} P(\beta) \prod_{j \in \beta^{-1}(1)} F_a^{(j)}, \quad (38)$$

where the coefficients  $F_a^{(j)}$  describe the random experiment used for generating the  $j$ -th code word. If we take these independent of  $j$ , (38) simplifies considerably. Introducing the source bits probability generating function

$$\Pi(t) := \sum_{\beta \in \check{I}^N} P(\beta) t^{\#\beta^{-1}(1)}, \quad (39)$$

we can formulate a simple but significant theorem:

*Theorem 1:* If the code word generation is performed uniformly with coefficients  $F_a$  and  $\Pi$  is the generating function of the source pattern space, then the target bit distribution is given by  $\check{F}_m = \Pi(F_m)$ .

This theorem is the pivot enabling us to compute the target distribution when source and code distribution are known. The source bit distribution acts as transformation turning the code bit distribution into the target bit distribution. The target distribution depends linearly on the source distribution but in a complex way on the code distribution.

We can immediately derive the first two moments of the target bit distribution from equations (18) and (19):

$$\check{\mu}_1 = n [1 - \Pi(F_{n-1})] \quad (40)$$

$$\check{\mu}_2 = n [n - (2n-1)\Pi(F_{n-1}) + (n-1)\Pi(F_{n-2})] \quad (41)$$

$$\check{\mu}_2 - \check{\mu}_1^2 = n [\Pi(F_{n-1}) - n\Pi(F_{n-1})^2 + (n-1)\Pi(F_{n-2})] \quad (42)$$

**Example 3:** Let's consider fixed weight source words of weight  $r$  and binomially generated code words with parameter  $q$ . From (31) we can read off the transformation function  $\pi(t) = t^r$ , and by (25) the coefficients are given by  $F_m = q^{n-m}$ . Now our theorem implies  $\tilde{F}_m = q^{r(n-m)}$ , i.e., the target bits are binomially distributed with parameter  $q^r$ . Then (26) immediately implies  $\tilde{G}_m = (1 - q^r)^m$ , and by (29) the variance equals  $\tilde{\mu}_2 - \tilde{\mu}_1^2 = nq^r(1 - q^r)$ . This case is simple because the individual target bits are independently distributed, which is not true in general.

By definition  $\tilde{G}_m$  equals the expected relative number of candidates matching a test pattern of length  $m$ . They are determined by statistics without reference to the original content, hence they are considered “false drops”<sup>1</sup> and their number shall be minimized. Pursuing our example further, we assume that the test pattern is obtained by submitting a query source string of weight  $s$  to the same superimposed coding process. Any pattern of weight  $m$  in the target query space will occur with probability  $\tilde{p}_m = (1 - q^s)^m q^{s(n-m)}$  and we can expect a proportion of  $\vartheta = \sum_{m=0}^n \binom{n}{m} \tilde{G}_m \tilde{p}_m = [(1 - q^r)(1 - q^s) + q^s]^n$  random hits.  $\vartheta$  is minimal if we choose  $q^s = \frac{r}{r+s}$  with  $\vartheta = \left[1 - \left(\frac{r}{r+s}\right)^{\frac{r}{s}} \frac{s}{r+s}\right]^{\frac{r}{s}}$ . For  $r \gg s$  we have  $q^r = \left(\frac{r}{r+s}\right)^{\frac{r}{s}} = \left(1 - \frac{s}{r+s}\right)^{\frac{r}{s}} \approx e^{-\frac{r}{r+s}} \approx e^{-1}$  and  $\vartheta = \left[1 - \frac{1}{e}(1 - q^s)\right]^n = \left[1 - \frac{1}{e}(1 - e^{-\frac{s}{r}})\right]^n$ , hence  $\ln \vartheta \approx -\frac{ns}{er}$ . If we do not want the relative number of false drops to exceed a proportion of  $\vartheta_{\max}$  but have to allow for query words of length at least  $s_{\min}$ , then we must choose our target bit patterns of length  $n \gtrsim \frac{re}{s_{\min}} |\ln \vartheta_{\max}|$ . Notice that the original bitstring length  $N$  does not enter at all, just the number of 1-bits  $r$  is relevant.

**Example 4:** Our example above was chosen for its simplicity; we don't recommend using binomially distributed code words in practice. Fixed weight code words of weight  $w$  can be expected to exhibit a much more robust behavior. Because of equations (40) and (42) and the monotonicity of probability generating functions fixed weight code words will produce minimal variance in the target distribution if the expectation is prescribed. Both cases may be directly compared if the parameter of the binomial distribution is chosen as  $q = 1 - \frac{w}{n}$ , because by (34)  $F_{n-1} = \binom{n-w}{1} / \binom{n}{n-1} = \frac{n-w}{n} = q$ . Furthermore  $F_{n-2} = \binom{n-w}{2} / \binom{n}{n-2} = \frac{(n-w)(n-w-1)}{n(n-1)} = q \frac{nq-1}{n-1}$  and  $\tilde{F}_{n-2} = q^r \left(\frac{nq-1}{n-1}\right)^r$ . The variance

$$\text{var} = \tilde{\mu}_2 - \tilde{\mu}_1^2 = n \left\{ q^r - nq^{2r} + (n-1)q^r \left(\frac{nq-1}{n-1}\right)^r \right\} \quad (43)$$

$$= n \left\{ q^r - q^{2r} - (n-1) \left[ q^{2r} - q^r \left(q - \frac{1-q}{n-1}\right)^r \right] \right\} \quad (44)$$

can be computed asymptotically for large  $n$  by developing the rightmost bracket in a power series  $\left(q - \frac{1-q}{n-1}\right)^r = q^r - rq^{r-1} \frac{1-q}{n-1} \pm \dots$ , where all higher powers may be safely discarded:

$$\text{var} \approx nq^r(1 - q^r) \left[ 1 - r(1 - q) \frac{q^{r-1}}{1 - q^r} \right], \quad (45)$$

the first factor being equal to the value in the binomial case. With suitable parameters the factor in square brackets may be quite small, i.e., the fixed weight variance may be negligible while the binomial variance is not.

We reconsider example 3 with fixed weight code words. In [6] it is shown<sup>2</sup>  $\vartheta \approx (1 - q^r)^{n(1-q^s)}$ . The minimum value is attained for  $q^r \approx \frac{1}{2}$  with  $\ln \vartheta \approx -\frac{ns}{r} (\ln 2)^2$ . This value is slightly better than in example 3. We have  $\text{var} \approx \frac{ns^2}{r^2} (\ln 2)^2$ , which is by a factor  $\frac{s}{r} \ln 2$  smaller than the binomial case.

#### IV. ADAPTING TO NON UNIFORM BUT INDEPENDENT SOURCE BIT DISTRIBUTIONS

If the individual source bits are independent and the  $i$ -th bit is turned on with probability  $p_i$  the probability of a given source pattern  $\beta \in \tilde{I}^N$  is

$$P(\beta) = \prod_{i \in \beta^{-1}(1)} p_i \prod_{i \notin \beta^{-1}(1)} (1 - p_i) \quad (46)$$

$$\tilde{F}_a = \prod_{j=1}^N \left( p_j F_a^{(j)} + 1 - p_j \right) \quad (47)$$

(47) is obtained by inserting (46) into (38) and distributively collecting terms. In our selection of optimal code word distributions we let ourselves be guided by what we learned in section III: We set  $\tilde{F}_{n-1} = \frac{1}{2}$  and minimize  $\tilde{F}_{n-2}$ , observing that by equation (19) this is equivalent to minimization of the second moment and hence of the variance. By inequality (21)  $F_{n-2}^{(j)} \geq (n-1)^{-1} \left( nF_{n-1}^{(j)} - 1 \right) F_{n-1}^{(j)}$ , the lower bound being attained for fixed weight code words. Substituting

$$u_j := p_j F_{n-1}^{(j)} + 1 - p_j \quad (48)$$

<sup>1</sup>This idiomatic expression is historic and derives from the application to punched cards.

<sup>2</sup>As a matter of fact the equation is derived by using the binomial case as approximation and observing that target query weights have small variance.

we have to solve the minimization problem

$$\begin{aligned} \tilde{F}_{n-2} &= \frac{n^N}{(n-1)^N \prod_j p_j} \\ &\cdot \prod_{j=1}^N \left\{ (u_j - \nu_j (1 - p_j))^2 + \omega_j p_j (1 - p_j) \right\} = \min \end{aligned} \quad (49)$$

under the constraint

$$\frac{1}{2} = \tilde{F}_{n-1} = \prod_{j=1}^N u_j \quad (50)$$

in the domain

$$1 - \frac{n-1}{n} p_j \leq u_j \leq 1 \quad (51)$$

with

$$\nu_j := (1 - p_j)^{-1} \left[ 1 - \left( 1 - \frac{1}{2n} \right) p_j \right] \quad (52)$$

$$\begin{aligned} \omega_j &:= (1 - p_j)^{-1} \left[ 1 - \frac{1}{n} - \left( 1 - \frac{1}{2n} \right)^2 p_j \right] \\ &= p_j^{-1} [1 - \nu_j^2 (1 - p_j)]. \end{aligned} \quad (53)$$

Observe that the terms  $\nu_j$  and  $\omega_j$  introduced for abbreviation are very close to 1, in particular  $\omega_j > 0$ . Ignoring the restrictions (51) temporarily, we see that (49) tends to  $+\infty$  if at least one of the coordinates  $u_j$  does, therefore (49) must have an absolute minimum. We are going to locate it using a Lagrange multiplier and will then have to check conditions (51). Thus

$$0 = \frac{\partial}{\partial u_j} \left( \ln \tilde{F}_{n-2} - \lambda \sum_{j=1}^N \ln u_j \right) \quad (54)$$

$$2[u_j - \nu_j (1 - p_j)] u_j = \lambda [u_j^2 - 2\nu_j (1 - p_j) u_j + 1 - p_j] \quad (55)$$

$$\prod_{j=1}^N \frac{u_j - \nu_j (1 - p_j)}{p_j} = 2 \left[ \frac{\lambda}{2} \left( 1 - \frac{1}{n} \right) \right]^N \tilde{F}_{n-2}. \quad (56)$$

We see that  $\frac{\lambda}{2}$  will be approximately the geometric mean of the quantities  $F_{n-1}^{(j)}$  and hence close to 1. Expanding  $u_j = \alpha + \beta(\lambda - 2) + \gamma(\nu_j - 1) \pm \dots$  up to linear order in the small quantities  $\lambda - 2$  and  $\nu_j - 1$  and substituting into (55) leads to

$$\lambda - 2 \approx \frac{1}{n} - \frac{2 \ln 2}{\sum_{k=1}^N \frac{p_k}{1 - p_k}} \quad (57)$$

$$u_j \approx 1 - \frac{p_j}{1 - p_j} \frac{\ln 2}{\sum_{k=1}^N \frac{p_k}{1 - p_k}} \quad (58)$$

$$F_{n-1}^{(j)} \approx 1 - \frac{1}{1 - p_j} \frac{\ln 2}{\sum_{k=1}^N \frac{p_k}{1 - p_k}}. \quad (59)$$

Summarizing:

*Theorem 2:* An optimal target distribution is obtained by choosing fixed weight code words of weight  $\frac{n}{1 - p_j} \frac{\ln 2}{\sum_{k=1}^N \frac{p_k}{1 - p_k}}$  for encoding of bit  $j$ .

If the probabilities  $p_j$  are actually independent of  $j$  then this coincides with section III.

## V. THE GENERAL CASE

There remains the question what to do if the source bit distributions are neither uniform nor independent. We may take a clue from theorem 2: as long as the individual bit probabilities  $p_j$  are not too large, say  $< 1/2$ , then the code word lengths recommended there do not vary significantly. We may try one and the same code distribution for all source bits and thus place

ourselves in the situation of theorem 1. The generating function defined in equation (39) is the same that would be obtained from an isotropic bit distribution with

$$p_m = \binom{N}{m}^{-1} \sum_{\beta \in i^N, \# \beta^{-1}(1)=m} P(\beta). \quad (60)$$

Choosing fixed length code words of weight  $n(1 - q)$  for a suitable parameter  $q$  theorem 1 tells us that the target bits will have an expected weight of  $\Pi(q)$  and we want to arrange for  $\Pi(q) = 1 - \check{G}_1 = \frac{1}{2}$ . Substituting  $q = e^{-\varepsilon}$  with  $0 < \varepsilon \ll 1$  we derive from (17):

$$\check{G}_1 = 1 - \Pi(e^{-\varepsilon}) = \sum_{m=1}^{\infty} (-1)^{m+1} \frac{\mu_m}{m!} \varepsilon^m. \quad (61)$$

This equation is quite suitable for practical application, because the power series is fast converging and the lower moments of the source bit distribution are easily evaluated. We can solve for  $\varepsilon$ :

$$\varepsilon = \frac{1}{\mu_1} \check{G}_1 + \frac{\mu_2}{2\mu_1^3} \check{G}_1^2 + \frac{3\mu_2^2 - \mu_1\mu_3}{6\mu_1^5} \check{G}_1^3 \pm \dots \quad (62)$$

Convergence of this power series is again fast enough to use the partial sum above as practical estimate.

Notice that in case of source words of fixed weight  $r$  we have  $\mu_m = r^m$  and we recover section III exactly.

## REFERENCES

- [1] J. M. Barnard, "Substructure searching methods: Old and new," *J. Chem. Inf. Comput. Sci.*, vol. 33, pp. 532–538, 1993.
- [2] J. M. Barnard and D. Walkowiak, "Computer systems for substructure searching," in *The Beilstein System – Strategies for Effective Searching*, S. Heller, Ed. American Chemical Society, 1998, pp. 55–72.
- [3] W. H. Kautz and R. C. Singleton, "Nonrandom binary superimposed codes," *IEEE Trans. on Information Theory*, vol. 10, no. 4, pp. 363–377, October 1964.
- [4] A. M. Rashad, "Superimposed codes for the search model of a. renyi," *Intern. J. Computer Math.*, vol. 36, pp. 47–56, 1990.
- [5] C. N. Mooers, "Application of random codes to the gathering of statistical information," Master's thesis, Massachusetts Institute of Technology, 1948.
- [6] C. S. Roberts, "Partial-match retrieval via the method of superimposed codes," *Proceedings of the IEEE*, vol. 67, no. 12, pp. 1624–1642, December 1979.
- [7] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *CACM*, vol. 13, no. 7, pp. 422–426, July 1970.
- [8] D. E. Knuth, *The Art of Computer Programming*, 2nd ed. Addison-Wesley, 1998, vol. 3.
- [9] L. Hodes, "Selection of descriptors according to discrimination and redundancy. application to chemical structure searching," *J. Chem. Inf. Comput. Sci.*, vol. 16, pp. 88–93, 1976.
- [10] A. Renyi, "On the theory of random search," *Bull. Amer. Math. Soc.*, vol. 71, no. 6, pp. 809–928, 1965.
- [11] W. Feller, *An Introduction to Probability Theory and Its Applications*, 3rd ed. Princeton, 1970.