Exact Inference Algorithms and Their Optimization in Bayesian Clustering

Jukka Kohonen

Department of Mathematics and Statistics
Faculty of Science
University of Helsinki

ACADEMIC DISSERTATION

To be presented, with the permission of the Faculty of Science of the University of Helsinki, for public examination in auditorium CK112, Exactum (Gustav Hällströmin katu 2B), on 20th of March 2015, at 12 o'clock noon.

Helsinki 2015

Supervisor
Professor Jukka Corander, University of Helsinki, Finland

Pre-examiners
Professor Camilla Hollanti, Aalto University, Finland
Professor Lasse Holmström, University of Oulu, Finland

Opponent
Associate Professor Jose M. Peña, Linköping University, Sweden

Custos
Professor Jukka Corander, University of Helsinki, Finland

ISBN 978-951-51-0822-7 (paperback)
ISBN 978-951-51-0823-4 (PDF)
http://ethesis.helsinki.fi/

Printed at Unigrafia

Helsinki 2015

Abstract

Clustering is a central task in computational statistics. Its aim is to divide observed data into groups of items, based on the similarity of their features. Among various approaches to clustering, Bayesian model-based clustering has recently gained popularity. Many existing works are based on stochastic sampling methods.

This work is concerned with exact, exponential-time algorithms for the Bayesian model-based clustering task. In particular, we consider the exact computation of two summary statistics: the number of clusters, and pairwise incidence of items in the same cluster. We present an implemented algorithm for computing these statistics substantially faster than would be achieved by direct enumeration of the possible partitions. The method is practically applicable to data sets of up to approximately 25 items.

We apply a variant of the exact inference method into graphical models where a given variable may have up to four parent variables. The parent variables can then have up to 16 value combinations, and the task is to cluster them and find combinations that lead to similar conditional probability tables.

Further contributions of this work are related to number theory. We show that a novel combination of addition chains and additive bases provides the optimal arrangement of multiplications, when the task is to use repeated multiplication starting from a given number or entity, but only a certain kind of function of the successive powers is required. This arrangement speeds up the computation of the posterior distribution for the number of clusters. The same arrangement method can be applied to other multiplicative tasks, for example, in matrix multiplication.

We also present new algorithmic results related to finding extremal additive bases. Before this work, the extremal additive bases were known up to length 23. We have computed them up to length 24 in the unrestricted case, and up to length 41 in the restricted case.

Acknowledgements

During my doctoral studies I have had two great supervisors: professors Elja Arjas and Jukka Corander. I want to thank both of them for scientific guidance, for support, and for apparently an infinite supply of enthusiasm.

I thank my friends and colleagues at the Department of Mathematics and Statistics for creating an intellectually stimulating working atmosphere. These friends include Paul Blomstedt, Väinö Jääskinen, Elina Numminen, Jukka Sirén, and numerous others. Special thanks go to Mikhail Shubin for an improvement idea for one of the illustrations in this work.

Finally I want to thank my family. My father, professor Teuvo Kohonen, has always been for me the model of a true scientist. To my wife Laura I am deeply indebted for her love and her patience, and for our discussions about clustering of apples and oranges. My daughter Vilma I must thank for a genuinely fresh look into the addition of integers, reminding me of what a fascinating concept it is.

Helsinki, November 2014

Jukka Kohonen

Contents

List of articles

This thesis consists of the summary part and the following four articles, referred to in the text by Roman numerals I–IV. The articles are reproduced with the permission of their respective copyright holders.

[I] J. Kohonen and J. Corander. Computing exact clustering posteriors with subset convolution. *Communications in Statistics: Theory and Methods*. To appear.

[II] J. Kohonen and J. Corander. Addition chains meet postage stamps: reducing the number of multiplications. *Journal of Integer Sequences* 17(3), article 14.3.4, 2014.

[III] J. Kohonen. A meet-in-the-middle algorithm for finding extremal restricted additive 2-bases. *Journal of Integer Sequences* 17(6), article 14.6.8, 2014.

[IV] J. Kohonen, J. Pensar and J. Corander. Exact Bayesian learning of partition directed acyclic graphs. Submitted.

Author's contributions

[I] J.K. had the primary responsibility in all parts of this article.

[II] J.K. had the primary responsibility in all parts of this article.

[III] J.K. was the sole author.

[IV] All authors took part in designing the models and the methods. J.K. performed the numerical experiments.

The purpose of computing is insight, not numbers.

R. W. Hamming: *Numerical Methods for Scientists and Engineers*

1 Introduction

The main themes of this work are efficient methods for certain computationally intensive problems: relating to statistical clustering on one hand, and to number theory on the other.

While statistics and number theory are rather distant fields of study, we will encounter interesting connections. In particular, an application of number theory will provide a solution to arranging the clustering computations in an optimal way. Number theory was one of G. H. Hardy's examples of "useless mathematics", which was to be studied solely for the purpose of mathematical beauty, yet it has later become an extremely useful tool in several applied fields [6].

This work is structured as follows. Chapter 2 starts with the basics of clustering as a statistical technique in general, and then goes on to Bayesian model-based clustering in particular. In Chapter 3 we proceed to implementation and application: a computational method of exact posterior inference in clustering is presented (relating to Article I), and exact clustering is then applied to graphical probabilistic models (Article IV).

In Chapter 4 we take an excursion to additive number theory. Two previously established notions are discussed: addition chains and the postage stamp problem. We see how these two can be combined in a novel way to solve an optimization problem relating to exact clustering (Article II). Finally we turn to efficient computation of the postage stamp problem itself (Article III).

> "I checked it very thoroughly," said the computer, "and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is."
>
> Douglas Adams: *The Hitchhiker's Guide to the Galaxy*

2

Principles of Bayesian clustering

Clustering is the task of dividing a given set of objects into groups, such that objects within a group are similar or near to each other, while objects in different groups are dissimilar or far apart. The groups are called *clusters*.

While the opening paragraph states the general idea, numerous details must be specified before "clustering" is even a well-defined problem. How are the objects to be represented? One choice is to define them as points in the d-dimensional space $\mathbb{R}^d$, but this is not universally adequate, and there are numerous other alternatives. How is similarity or nearness measured? Euclidean distance is but one possibility. Finally, what additional constraints or objectives do we desire? Do we set the number of clusters *a priori*, or leave it to be selected by the clustering method? Different choices of specification lead to a myriad of different "clustering tasks" and methods. We will not even attempt an overview of them here; for that, the reader is directed to a comprehensive survey by Jain *et al.* [20].

In this chapter we set the scene for a particular kind of clustering, namely Bayesian, model-based clustering, and make some introductory observations.

2.1 Definitions, notation and terminology

Given that clustering is a large and multidisciplinary field of study, it is not surprising that its basic concepts have a plethora of alternate names. For example, the objects to be clustered may be called *items*, *patterns*,

observations or *feature vectors* [20] (and probably by other names as well). These various synonyms do not usually cause any real difficulty, but for ease of reference we begin by defining our notation and terminology.

We start with a *universe* $U = \{1, 2, \ldots, n\}$, containing n *items* indexed with integers. A *cluster* is a nonempty subset of U, and a *partition* is a collection of disjoint clusters whose union is U. A partition consisting of k clusters is a k-partition. A *trivial partition* has only one cluster, and a *singleton partition* has n clusters that are singletons.

Each item i is associated with some features y_i, which will form the basis of clustering. We do not here specify how the features are to be represented, but examples are d-dimensional real vectors or binary vectors; in that case we may identify items with *points* in a vector space.

We make a distinction between *ordered* and *unordered partitions*. An ordered k-partition is a tuple of clusters $S = (S_1, S_2, \ldots, S_k)$. An unordered partition is a *set* of clusters with no intrinsic ordering; each cluster is identified solely by its constituent items. For example, while

$$(\{1, 2\}, \{3, 4\}) \quad \text{and} \quad (\{3, 4\}, \{1, 2\}) \tag{2.1}$$

are different ordered 2-partitions of 4 items, they correspond to the same unordered partition

$$\{\{1, 2\}, \{3, 4\}\}. \tag{2.2}$$

Note that in both cases the clusters themselves are sets. Clearly, the clusters of an unordered k-partition could be ordered in $k!$ ways, giving $k!$ different ordered partitions. The distinction might seem moot, but it is crucial when counting how many partitions there are, and more importantly, when performing a search within the space of all partitions, or defining a probability distribution over it.

2.2 The space of all partitions

In a universe of n items, there are $2^n - 1$ nonempty subsets, or possible clusters. The number of all unordered k-partitions is the Stirling number of

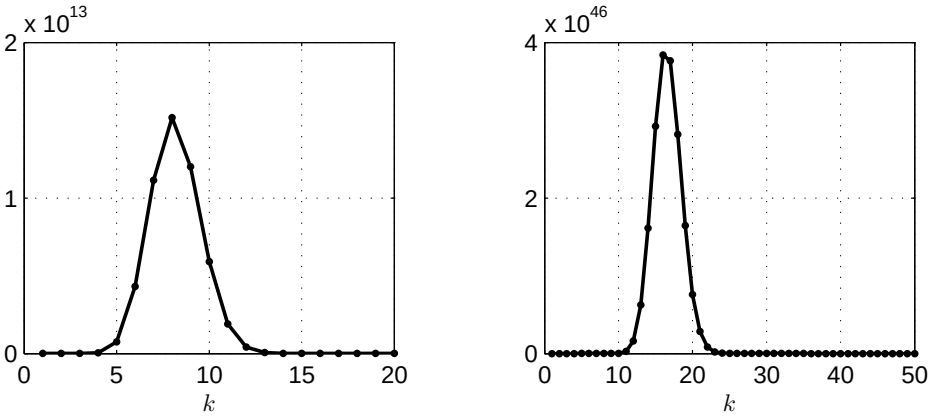

Figure 2.1: Numbers of unordered partitions of different sizes (on the left for $n = 20$ items; on the right for $n = 50$).

the second kind, denoted by $\left\{ {n \atop k} \right\}$. Note that $\left\{ {n \atop 1} \right\} = \left\{ {n \atop n} \right\} = 1$, corresponding to the trivial and singleton partitions. For the general case $1 < k < n$, consider two possibilities of forming a k-partition: (1) if the first $n-1$ items are in $k-1$ clusters, the nth item must form the kth cluster on its own; (2) if the first $n-1$ items are in k clusters, then the nth item must be in one of them. This gives the recurrence relation [43]

$$\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \left\{ {n-1 \atop k} \right\}, \tag{2.3}$$

by which $\left\{ {n \atop k} \right\}$ can be computed. Figure 2.1 illustrates it for $n = 20$ and $n = 50$.

The number of unordered partitions of n items is the nth Bell number, $B_n = \sum_{k=1}^{n} \left\{ {n \atop k} \right\}$. For example, $B_{20} \approx 5.2 \times 10^{13}$, and $B_{50} \approx 1.9 \times 10^{47}$.

We make two observations here. First, as B_n grows very rapidly with n, direct enumeration of all partitions (whether for searching for the optimum partition, or for a summation of posterior probabilities) is possible only for very small values of n. However, in Chapter 3 we will consider a technique that pushes the limit a little higher.

Secondly, $\left\{ {n \atop k} \right\}$ varies strongly with k. For a given n, most partitions have a fairly large number of clusters. In fact, the maximum of $\left\{ {n \atop k} \right\}$ occurs roughly at the root of the equation $k = e^{n/k}$ [26]. The implications of this to our probability models will be discussed in Section 2.6.

2.3 Clustering: optimization or inference?

Traditionally, clustering is a non-probabilistic optimization problem: among all possible partitions of given items, find the the best partition, in the sense of some defined *criterion*, or objective function J.

If the items have features in a metric space, naturally occurring criteria are aggregates of the distance function, such as: maximum (or average) pairwise distance (or squared distance) between two points within the same cluster (or between a point and its "cluster center"). The criterion is usually easy enough to compute for a given partition. But finding the optimum, or a near-optimum partition is generally difficult.

For example, Garey and Johnson define CLUSTERING as the following problem (in our notation): Given a finite set U, a distance $d(x, y) \in \mathbb{Z}_0^+$ for each pair $x, y \in U$, and two positive integers k and B, determine whether there exists a partition of U into $S_1, S_2, \ldots, S_k$ such that $d(x, y) \leq B$ for every pair x, y within each cluster. This problem is NP-complete, along with some variations, for example, where the upper bound B is for the *sum* of intra-cluster pairwise distances [16].

Practical approaches include iterative methods such as the K-MEANS [3, pp. 424–430]. Its underlying idea is that if a cluster's items are points in $\mathbb{R}^d$, they should be near to their centroid. The clustering criterion is defined as

$$J = \sum_{j=1}^{k} \sum_{i \in S_j} d(y_i - \mu_j)^2, \tag{2.4}$$

where μ_j is the centroid of the points in cluster S_j, and d is Euclidean distance. Now, given a particular partition S, the cluster centroids are easy to compute. But it may happen that some points are nearer to some other

cluster's centroid than their own. Then J is decreased (improved) if those points are re-assigned to those better clusters. Repeating this, a sequence of improving partitions is obtained, leading to a local optimum of J.

Now probability may enter the picture in at least two forms. First, it is natural to think that a cluster is associated with a probability distribution, or a process that has *generated* the features of its items. A partition then defines a probability model, and it is standard statistical practice to seek the *maximum-likelihood model*, that is, the partition under which the probability (density) for the observed data is maximized. Defined this way, clustering is still an optimization problem, but in a probabilistic setting.

For example, the distribution may be multivariate Gaussian, with an underlying mean vector (centroid) and a covariance matrix. As these cluster parameters are usually unknown, they may be estimated from the cluster's items, or more generally treated as latent variables. The distribution for the whole observed data is then a *mixture of Gaussians*, with mixture components arising from the different clusters. A practical method for seeking the maximum-likelihood partition is the EXPECTATION-MAXIMIZATION (EM) algorithm, which has strong resemblance to K-MEANS, but is motivated from the probabilistic model [3, pp. 430–455].

The second probabilistic aspect is related to the first one. If the data were generated according to a probability model implied by a particular partition (the *generating* or *true partition*), it quite possible that many other partitions *could* have produced exactly the same data. This is typically the case when the clusters overlap in the feature space. In such cases there is no particular reason to believe that the maximum-likelihood partition is exactly correct, though it may be *similar* to the true partition (see Figure 2.2). One might argue that based on the observed data, the maximum-likelihood partition is our best estimate of the true partition, but how much confidence do we have on it, and how should we express the uncertainty?

At the simplest, perhaps two clusters are mostly separate, but a few points lie near their boundary, so the classification of those few points is uncertain. Fuzzy clustering [20, pp. 281–282] addresses this problem by making the clusters fuzzy sets, so that an item may belong to several clusters in varying degrees (expressed as a membership function).

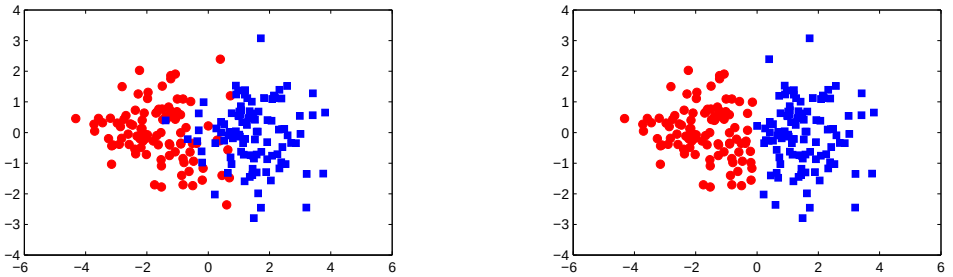

Figure 2.2: Overlapping clusters. Left: generating model (two bivariate Gaussians). Right: the partition that divides the points along the y axis provides much greater likelihood.

More generally, larger-scale properties of the true partition can be uncertain. Based on the data, we may not even be sure how many clusters there are; this is quite typical in real world situations. Such uncertainty cannot be represented by fuzzy clustering. Instead, we would like to express uncertainty within the whole partition space.

If we can somehow obtain a posterior distribution for partitions, in principle this opens up all kinds of possibilities for inference, in a manner that has sound basis in probability calculus. For example, suppose that k is uncertain. Adding up the posterior probabilities for all 5-partitions, we obtain the posterior probability for the event that $k = 5$. Repeating for all $k = 1, \ldots, n$, we get a genuine posterior distribution for k.

For another example, considering a particular pair of items x, y, we could infer the posterior probability that they are in the same cluster – whatever the partition for all other items may be. A related idea is to seek the partition that minimizes a particular *loss function*, such as the expected number of pairs of items that have been wrongly clustered either together or apart [24]. Such an optimum partition is usually different from the maximum-likelihood partition.

In general, if we are interested in a particular aspect of the true partition, we could marginalize out all other, uninteresting aspects.

2.4 Bayesian clustering model

The Bayesian approach provides a natural framework for expressing uncertainty within the space of all partitions [5]. Two ingredients are needed: a prior $p(S)$ over the partitions, and a likelihood $p(Y \mid S)$ indicating which data $Y = (y_1, \ldots, y_n)$ are likely to be generated from each partition. Note that we use $p(\cdot)$ as a generic notation for probability mass or density.

Bayes' theorem then states simply that our posterior probability for partition S, based on the data Y, is

$$p(S \mid Y) = p(S)\, p(Y \mid S) \,/\, Z, \tag{2.5}$$

where Z is a normalizing constant, equal to

$$Z = p(Y) = \sum_S p(S)\, p(Y \mid S). \tag{2.6}$$

Of particular interest are *product partition models*, where both the prior and the likelihood in (2.5) are defined so that they factorize into products over the clusters [34]:

$$\begin{aligned} p(S) &= C \prod_{j=1}^{k} c(S_j), \\ p(Y \mid S) &= \prod_{j=1}^{k} p(Y_j \mid S_j), \end{aligned} \tag{2.7}$$

where C is again a normalizing constant; c is a "cohesion function"; and Y_j represents the data in cluster S_j. The factorization of $p(Y \mid S)$ entails the assumption that data in different clusters are independent (conditional on the partition).

For more freedom in the partition prior, we extend the product partition model with constant factors that may vary with k, thus we take

$$p(S) = C\, w_k \prod_{j=1}^{k} c(S_j). \tag{2.8}$$

Stochastic methods can then be applied either to search for the partition that has maximum *a posteriori* probability (MAP) [12], or to estimate the posterior distribution of S. However, a stochastic search will necessarily visit only a tiny portion of the vast space of possible partitions, and the posterior probabilities of partitions will be known only up to the unknown normalizing constant Z. We may thus gain a good understanding of the *relative* merits of those partitions *that were visited*, but their true posterior probabilities remain unclear, unless we can ascertain that the search has covered a large part of the posterior distribution.

To complete the probability model, we must specify both the prior model for partitions $p(S)$, and the likelihood model for the data $p(Y \mid S)$. We will discuss these in the following two sections, starting with the likelihood.

2.5 Marginal likelihood of a cluster

The likelihood of the data in a particular cluster, $p(Y_j \mid S_j)$, could be defined in various ways. If a parametric model is employed, each cluster S_j is equipped with some parameters θ_j.

For example, if the data within a cluster are assumed to be multivariate Gaussian, θ_j will represent the cluster location and shape in terms of a mean vector μ_j and a covariance matrix C_j. If further the features in different dimensions are assumed (conditionally) independent, the model takes the form

$$(y_{im} \mid S_j, \theta_j) \sim N(\mu_{jm}, \sigma^2_{jm}) \tag{2.9}$$

where θ_j consists of the means μ_{jm} and variances σ^2_{jm} for cluster j and dimensions $m = 1, \ldots, d$.

For another example, if the data within a cluster are binary (Bernoulli variables), and dimensions are conditionally independent (the "Naive Bayes" assumption), we may model

$$(y_{im} \mid S_j, \theta_j) \sim \text{Bernoulli}(\mu_{jm}), \tag{2.10}$$

where $\theta_j = (\mu_{j1}, \ldots, \mu_{jd})$ is a vector of the probability parameters for cluster j and dimensions $m = 1, \ldots, d$.

Now these probability models are conditional on the unknown cluster parameters θ_j. For (2.7) we need the *marginal* likelihood of the data observed in a cluster, that is, we need to integrate over θ_j. There are two approches. In a stochastic search the parameters θ_j may be subject to sampling, along with the partition S itself. Another possibility is to assume a prior probability for θ_j that is conjugate with the likelihood model, such that $p(Y \mid S)$ can be readily computed analytically.

For the Gaussian likelihood, this is achieved if the mean parameter has a Gaussian distribution and the variance has an inverse-gamma distribution. For the Bernoulli likelihood, the probability parameter is taken to have a beta distribution. Detailed formulae for the marginal likelihoods can be found in Article I.

2.6 Prior on partitions

For a complete probability model, we need a prior distribution within the space of possible partitions. Recall from Section 2.2 that the space is rather large: B_n partitions, if we work with unordered partitions.

Uniform distribution may be seen as an innocent-looking "ignorance prior". If we take the uniform prior over unordered partitions,

$$p(S) = \frac{1}{B_n} \tag{2.11}$$

then the MAP (maximum *a posteriori* probability) and ML (maximum marginal likelihood) partitions coincide, and we may think we have evaded the whole issue of prior choice. This may be defensible if we do not seek full posterior inference, but wish to find one reasonably good partition.

However, if we wish to perform posterior inference about k or some other *property* of the partition, the choice of the prior may have a large impact. Uniform prior over partitions implies that the prior probability for k is proportional to $\left\{ {n \atop k} \right\}$, which is extremely peaked at fairly large values of k, as we saw in Figure 2.1.

From a subjective Bayesian viewpoint this may be quite unsatisfactory. Consider for example $n = 50$. The prior probabilities for $k = 2, 3, 4$ are

approximately 3.0×10^{-33}, 6.4×10^{-25} and 2.8×10^{-19}, respectively. On the other hand, the prior probability for $k = 16$ is 0.21. It seems far-fetched to claim that these probabilities represent our genuine *a priori* belief of the number of clusters in some data of 50 items. For larger n the effect is even more pronounced.

From a more pragmatic viewpoint, these uneven priors may be seen as an unwanted bias towards a large number of clusters in the posterior inference. Article I has some brief experimental results on this.

Another choice, which may feel more defensible from a subjective viewpoint, is to assume that the *number of clusters* has uniform distribution over whatever values we deem plausible; unless we have reason to think otherwise, we may let k have the discrete uniform distribution over the integers $\{1, \ldots, n\}$. If we have no preference between partitions of the same size, our prior is then

$$p(S) = \frac{1}{n \left\{ {n \atop k} \right\}} \tag{2.12}$$

since the probability $1/n$ is spread uniformly over the $\left\{ {n \atop k} \right\}$ partitions of size k.

Perhaps slightly unexpectedly, this simple prior cannot be represented in the form (2.7) required by the product partition model, since it does not factorize as a product over clusters, as proven by Quintana and Iglesias [34]. However, *conditional* on the value of k, the prior (2.12) is constant, so we can use our extended prior model (2.8).

Yet another, and a very common choice of partition priors is based on a Dirichlet process. Conceptually, it is based on a potentially infinite number of classes, out of which a finite number k realizes in a finite sample of items. This process implies a particular prior on k, as discussed by Quintana and Iglesias [34]. Dirichlet process has a concentration parameter α, which affects its tendency to produce more or less clusters. It should also be noted that the Dirichlet-implied prior distribution divides the prior probability unevenly between partitions of the same size k; in this it differs from both versions of uniform distribution described above.

> If you don't know where the strongly peaked regions are, you might as well ... quit: It is hopeless to expect an integration routine to search out unknown pockets of large contribution in a huge N-dimensional space.
>
> Press *et al.*: *Numerical Recipes in C*

3

Exact Bayesian clustering

In the previous chapter we formulated a probability model for Bayesian model-based clustering. Given a likelihood model and a prior for partitions, we could in principle compute (2.5) for all B_n unordered partitions, and from the resulting distribution make any posterior inference we wish.

Since B_n is huge (unless n is small), a common solution is to employ a stochastic algorithm that *samples* a large number of partitions of relatively large posterior probability, and either searches for an optimal model (*stochastic optimization*), or collects summary statistics such as posterior probability estimates for some parameters of interest (*Monte Carlo integration*). Although these two approaches have different goals, they may employ similar sampling techniques.

Monte Carlo integration is widely used in Bayesian statistics. In clustering, however, stochastic methods usually aim at optimization: searching for a partition that has high likelihood or posterior probability [12], or minimizes some loss function [24]. As an example of the summary statistic approach, posterior probabilities for k (the number of clusters) can be estimated from the stochastic sample [11]. But producing proper summary statistics is thought to be difficult, as the set of visited partitions may be very irregular [24].

Here we take a different approach. We shift the viewpoint from individual partitions into *classes* of partitions, defined by some particular statistic. We focus here on two statistics: (1) the number of clusters, and (2) *pairwise incidence* of two particular items, that is, the event that they are assigned

into the same cluster. The latter can of course be computed for all item pairs (i, j), producing a matrix of pairwise incidences (adjacency matrix).

Having chosen a particular statistic, we wish to infer its exact posterior distribution. For very small n this can be done by enumerating all partitions. For somewhat larger n it can still be done exactly without resorting to stochastic methods. The underlying idea is that the posterior probability is defined as a large sum of products, and this sum-of-products is then *decomposed* in a manner reminiscent of computing the distribution for a sum of random variables through convolution [38, pp. 56–58].

The decomposition method applicable in this context is *subset convolution*. Computing the sum still takes time exponential in n, but considerably less than the full enumeration of B_n partitions.

Subset convolution methods have been studied before by various authors [4, 15]. Directly relevant to our purposes, Koivisto has described how the exact posterior distribution for k can be computed via inclusion–exclusion [23], a method related to subset convolution.

In Article I, we describe an implementation of the computation for the posterior of k using subset convolution; extend it to computing the posterior probability for pairwise incidence; and present some empirical results.

3.1 Decomposition of the posterior sum

Let us fix k for the moment, and let us compute the total posterior probability for all k-partitions – that is, the posterior probability that the number of clusters is k.

For computational and notational convenience, we take the sum over *ordered* k-partitions, whose space we denote by $\mathcal{S}^k$. If we do not want to change the prior distribution of partitions, we basically have to include an extra factor of $1/k!$ in the prior $p(S)$. We write $c(S_j)p(Y \mid S_j) = f(S_j)$ for brevity, and for completeness define $f(\varnothing) = 0$, which ensures that empty sets do not contribute to the sum.

We assume that $f(X)$ is first computed for all 2^n subsets $X \subseteq U$, and the results stored as a table of 2^n entries. Note that while this is exponential

in n, in practice 2^n is small compared to the number of partitions.

The marginal posterior probability for k clusters is provided by

$$p(k \mid Y) = C\, w_k \sum_{S \in \mathcal{S}^k} \prod_{j=1}^{k} f(S_j). \tag{3.1}$$

The sum over partitions can be written as a $(k-1)$-fold sum over clusters

$$\sum_{S_1 \subseteq U} \sum_{S_2 \subseteq R_2} \cdots \sum_{S_{k-1} \subseteq R_{k-1}} \prod_{j=1}^{k} f(S_j), \tag{3.2}$$

where $R_j = U \setminus (S_1 \cup \cdots \cup S_{j-1})$ is the set of items available for the jth cluster, after the first $j-1$ clusters have already been selected; for the last cluster there is no choice left, so $S_k = R_k$. This corresponds to an algorithm that enumerates the unordered partitions via $k-1$ nested loops. The sum has $k!\left\{{n \atop k}\right\}$ terms since we work with ordered partitions.

Many terms in (3.2) have common factors. For a concrete illustration, let $k = 4$. Taking common f factors out of inner sums, the sum (3.2) becomes

$$\sum_{S_1 \subseteq U} f(S_1) \Big(\sum_{S_2 \subseteq R_2} f(S_2) \underbrace{\Big(\sum_{S_3 \subseteq R_3} f(S_3) f(\overbrace{R_3 \setminus S_3}^{S_4}) \Big)}_{(*)} \Big). \tag{3.3}$$

Observe that the innermost sum $(*)$ depends on R_3, that is, on what items are left for clusters S_3 and S_4, but it has absolutely no dependence on how the other items were divided between clusters S_1 and S_2.

For even more concreteness, let $n = 12$. If now $S_1 \cup S_2 = \{1, 2, 3, 4, 5\}$, then $R_3 = \{6, 7, 8, 9, 10, 11, 12\}$, and the innermost sum represents the possibilities of dividing R_3 into two clusters S_3 and S_4; it does not depend on how the items $\{1, 2, 3, 4, 5\}$ are divided between S_1 and S_2. Since the inner sum has the same value for every S_1, S_2 that have the same union, it needs to be computed only once (for each choice of that union).

Let us now define a set function that formalizes the above observations. Let $X \subseteq U$, and let g, h be two real-valued functions defined over the subsets of U. Then *subset convolution* $g * h$ is defined as

$$(g * h)(X) = \sum_{A \subseteq X} g(A)\, h(X \setminus A), \tag{3.4}$$

or equivalently, in a more symmetric fashion

$$(g * h)(X) = \sum_{\substack{A,B \subseteq X \\ A+B=X}} g(A)\, h(B), \tag{3.5}$$

where $A + B$ represents disjoint union. In essence, subset convolution represents the enumeration of all divisions of X into two subsets. For any given $X \subseteq U$, this sum has $2^{|X|}$ terms, and can be computed in $O(2^{|X|})$ time by direct summation, if the values of $g(A)$ and $h(B)$ are already available.

With this notation, the inner sum $(*)$ is simply $(f * f)(R_3)$, and the whole sum (3.3) can be rearranged as

$$\begin{aligned} &\sum_{A+B=U} \sum_{S_1+S_2=A} \big(f(S_1) f(S_2)\big)\ (f * f)(B) \\ &= \sum_{A+B=U} (f * f)(A)\ (f * f)(B) \\ &= \big((f * f) * (f * f)\big)\,(U). \end{aligned} \tag{3.6}$$

Having precomputed $f(X)$ for all $X \subseteq U$, we can compute (3.6) as follows: First, compute a table of $(f*f)(A)$ for all $A \subseteq U$. Then, take $g = h = (f*f)$, and compute $(g * h)(U)$. This provides the exact value for (3.2).

For a particular subset X, computing $(f * f)(X)$ by (3.4) involves the summation of $2^{|X|}$ terms. Since U has $\binom{n}{s}$ subsets of size s, computing the full table of $f * f$ for all $X \subseteq U$ involves

$$\sum_{s=0}^{n} \binom{n}{s} 2^s = 3^n \tag{3.7}$$

terms by the binomial theorem, and can be done in $O(3^n)$ time.

3.2 Posterior number of clusters

The method illustrated for $k = 4$ in the previous section can be generalized to arbitrary k as follows. From the definition (3.5), it follows that subset convolution is an associative operation over set functions, that is, $(f*g)*h = f*(g*h)$. We may thus define *k-fold subset convolution* as

$$\begin{aligned} f^1(X) &= f(X), \\ f^k(X) &= (f^{k-1} * f)(X) = (\underbrace{f * f * \cdots * f}_{k \text{ copies}})(X). \end{aligned} \tag{3.8}$$

It also follows that the sum (3.2) over ordered k-partitions of U can be expressed as

$$f^k(U). \tag{3.9}$$

By associativity, $f^k(X)$ can be computed as $(f^i * f^j)(X)$ for any positive integers $i + j = k$. The underlying idea is as follows. When computing a sum over ordered k-partitions of X, we can first consider all 2^n ways of dividing X into *two* disjoint subsets A and B. Then, for any given division $A + B = X$, we consider *separately* the possibilities of dividing A into i subsets, and the possibilities of dividing B into j subsets.

The exact posterior distribution for k can now be computed as follows.

Step 1. Precompute $f(X)$ for all subsets $X \subseteq U$. As a special case, set $f(\varnothing) = 0$ since a cluster cannot be empty.

Step 2. For $k = 2, \ldots, n$, compute the full table $f^k(X) = (f^{k-1} * f)(X)$ for all $X \subseteq U$.

Step 3. Compute $p(k \mid Y) = C\, w_k\, f^k(X)$ for $k = 1, \ldots, n$. The normalizing constant C is determined by $\sum_{k=1}^{n} p(k \mid Y) = 1$.

Since each subset convolution in Step 2 takes $O(3^n)$ time, and there are $n - 1$ convolutions to perform, the whole step takes $O(n\, 3^n)$ time. The computations are illustrated in Figure 3.1.

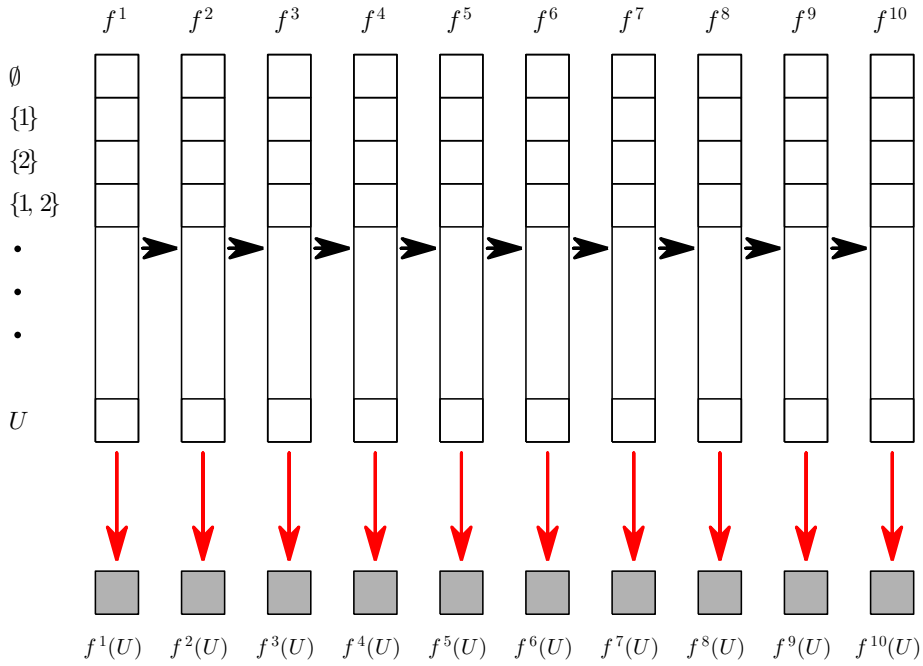

Figure 3.1: Computing the posterior distribution for k via subset convolutions. Each vertical column represents the table for $f^k(X)$ for all subsets $X \subseteq U$. Black horizontal arrows represent subset convolution $f^k = f^{k-1} * f$. Red vertical arrows represent the table lookup for $f^k(U)$.

The author has implemented this algorithm, nicknamed Xpost for *exact posterior* clustering inference, in MATLAB and partly in C (for enhanced speed in some innermost loops). Some illustrations and experimental results are shown in Article I.

We compared the exact posterior distribution for k against the distribution estimated with an MCMC method, and observed that MCMC did slightly underestimate the number of clusters. In future research, it is worth studying whether such deviation is systematic, what causes this behavior in

MCMC, and whether it can be removed. This perhaps illustrates the usefulness of having an exact posterior distribution available as a "gold standard", for evaluating the accuracy of other clustering methods.

Concerning computational speed, the current implementation of XPOST takes approximately 2.5 hours of processor time to compute the exact posteriors for a data set of $n = 20$ items. In comparison, we estimate that a straightforward computation by full enumeration of the $B_{20} \approx 5.2 \times 10^{13}$ unordered partitions would take about 200 days.

Since the time requirement of XPOST is $O(n\,3^n)$, we can further estimate that the exact clustering posteriors for $n = 25$ items could be computed in about 30 days of processor time, which is quite feasible with parallel computing. For larger numbers of items the current method is obviously not practical.

We note that there exists a method called *fast subset convolution* [4], which computes the subset convolution $(g*h)(X)$ simultaneously for all $X \subseteq U$ with the help of a cleverly constructed transformation of the set functions g and h. The method has better asymptotic performance than the direct method: instead of $O(3^n)$, it has a time requirement of $O(n^2\,2^n)$. The author implemented also this method, but despite the large *asymptotic* difference, in practice fast subset convolution did not offer much improvement in speed, considering that the feasible values of n are relatively small.

3.3 Posterior pairwise incidence

Aside from the number of clusters, we can define other summary statistics to characterize a partition of U. A particularly simple statistic is *pairwise incidence* of two given items i and j, or the event I_{ij} that the two items are in the same cluster. This statistic is also known as "pairwise co-occurrence" or "coincidence". We could also think of it as a binary random variable (the indicator variable for the incidence event).

As I_{ij} is a property of a partition, a posterior distribution for partitions implies a posterior probability for I_{ij}. The probability $p(I_{ij} \mid Y)$ indicates our *a posteriori* belief that items i and j belong in the same cluster.

We may also compute $p(I_{ij} \mid Y)$ for each pair $i, j \in U$. This gives us a *matrix of pairwise incidence probabilities*, easy to visualize as a grayscale image with, say, black indicating high posterior incidence. From such an image one may obtain a summary view of which clusters are particularly well discernible from the data, and which are more uncertain. Note that such a matrix represents the *marginal* posterior probabilities of incidences, and does not reflect their mutual dependence.

In Article I we show that $p(I_{ij} \mid Y)$ can be computed from the subset convolution tables $f^1, f^2, \ldots, f^n$. To arrive at this solution we need a few observations. First, we break I_{ij} into n disjoint events (k, I_{ij}), meaning that "there are k clusters, and items i, j are both in one of them". We compute the probabilities of these events separately for $k = 1, \ldots, n$.

The second observation is that the event (k, I_{ij}) may be further broken into k equiprobable events: the items i, j could be together in cluster S_1, or any of the other $k - 1$ clusters. By symmetry of the prior and likelihood models, these are equiprobable, so it suffices to compute the probability for $i, j \in S_1$, and then multiply by k.

The third, and crucial observation is that the probability for $i, j \in S_1$, with k clusters, can be written as a sum over ordered k-partitions such that for S_1, we sum over the 2^{n-2} clusters that do contain both i and j; then, if the first cluster is S_1, for the remaining items $U \setminus S_1$ we allow any $(k-1)$-partition, so in effect we simply look up $f^{k-1}(U \setminus S_1)$.

If the n full subset convolution tables $f^1, f^2, \ldots, f^n$ are already available, then (as shown in Article I) the posterior incidence probability for two given items i, j can be computed in $O(n\, 2^n)$ time. Repeating this for all pairs i, j gives the matrix of posterior incidence probabilities in $O(n^3\, 2^n)$ time. Detailed formulae, and two visual examples of posterior incidence matrices are shown in Article I.

We note here that the same method could be applied to any particular group of items, not just pairs. For example, a straighforward modification would give the posterior probability for a given triple of items to be incident in the same cluster.

3.4 An application to graphical models

Probabilistic graphical models are widely used in Bayesian statistics to represent the joint distribution of a collection of variables. They are based on a *graph* where the variables are represented by nodes (vertices). Each node may be connected to some other nodes with arcs or edges, and has a specific probability distribution *conditional* on the values of those neighbouring nodes [3, 30].

We focus here in the variant called *Bayesian networks* or *directed graphical models*, where the graphical structure is a directed acyclic graph (DAG); further we assume that the variables are binary. If a node X has m incoming arcs, then the variable X has a *conditional probability table* (CPT) that contains 2^m conditional probabilities

$$p(X = 1 \mid Y_1 = y_1, \ldots, Y_m = y_m) \tag{3.10}$$

where $Y_1, \ldots, Y_m$ are the *parent* variables of X, and $(y_1, \ldots, y_m)$ take all the 2^m possible values of the parents. Various methods exist for estimating these parameters and the structure of the graph from observed data [30].

For example, if X has 4 parents A, B, C, D, the model incorporates $2^4 = 16$ probability parameters, which we will denote by $p_{0000}, p_{0001}, \ldots, p_{1111}$. In general these 16 parameters are free, or independent parameters, so such a model accommodates a great amount of freedom.

If one wishes a simpler model (in the sense of number of parameters), one could simply reduce the number of parents. If X has only one parent A, this entails that if A is known, the values of B, C, D are irrelevant for predicting X.

However, one may wish to leave the possibility that for *some* values of A, the values of B, C, D are still relevant for X. Such models have been proposed under the name of *labeled DAGs* (LDAGs) [33]. In effect, a labeled DAG imposes a restriction that some of the conditional probabilities must have equal values. For example, the model may specify that X has four parents A, B, C, D such that if $A = 0$, then the probability of X does not depend on the other parents, but if $A = 1$ then it can depend on

them. This means that within the CPT of 16 entries, the first eight entries $p_{0000}, p_{0001}, \ldots, p_{0111}$ are constrained to have the same value, while the remaining eight entries can be all different.

In Article IV we propose a generalization of LDAGs. Given a node that has m parents (thus 2^m entries in the CPT), we allow an arbitrary *partition* of the CPT entries such that within a cluster, the entries have the same value. For at most 4 parents, this implies a clustering problem of at most 16 items, which can be solved relatively efficiently with a variant of the XPOST algorithm.

We applied this proposed model (called *partition DAG* or PDAG) to a data set containing 6 binary variables indicating different risk factors for coronary heart disease, and found an interesting, nontrivial association within the CPT for the binary variable indicating high systolic blood pressure. According to our findings, this variable could be linked to a specific combination of three other variables. Of course, such findings should not be taken too seriously before further studies perhaps support them, but this example illustrates that partition DAGs may reveal potential associations that are not visible with ordinary DAG models.

> If the theory of numbers could be employed for any practical and obviously honourable purpose ... surely neither Gauss nor any other mathematician would have been so foolish as to decry or regret such applications.
>
> G. H. Hardy: *A Mathematician's Apology*

4

An excursion to number theory

The Springer Encyclopedia of Mathematics defines number theory as "the science of integers" [31]. For most people this probably conjures up ideas like primality, factorization and modular arithmetic: concepts involving multiplication and division. But now we take an excursion into *additive* number theory, which studies the addition of (sets of) integers.

Our interest in additive number theory stems from the well-known algorithm of evaluating powers via repeated multiplication, such as computing x^8 from x by three successive squarings yielding x^2, x^4 and x^8. When powers are multiplied, exponents are added, so the task is essentially additive. The optimal choice of multiplications to perform is provided by *addition chains* [21, p. 402].

Similarly we can reduce the number of laborious subset convolutions performed by the Xpost algorithm. Recall that it computes the posterior distribution of the number of clusters via repeated subset convolutions; and that subset convolution is associative. The same convolution could be computed in different ways – for example, $f^{10} = f^5 * f^5$, or $f^7 * f^3$. This leads to the problem of choosing which convolutions to perform, so as to minimize the computational work.

Our actual computational problem is slightly more involved, and its solution, published in Article II, is based on a novel combination of addition chains and the *postage stamp problem*. For ease of exposition we will first revisit the basic task of repeated multiplication.

4.1 Addition chains and repeated multiplication

Suppose that given a number x, we must compute x^n with as few multiplications as possible. At each step we may multiply any two numbers (powers of x) so far obtained. More generally, x may be an element of any semigroup, *i.e.*, a set with an associative operation: for example, real matrices with matrix multiplication.

If $n = 2^r$, then the optimal way is to perform r squarings:

$$x \to x^2 \to x^4 \to \ldots \to x^n. \tag{4.1}$$

To see that this is indeed optimal, observe that each multiplication can *at most* double the highest exponent attained so far, thus fewer than r multiplications cannot reach the exponent 2^r.

If n is not a power of two, the optimal solution may not be obvious. Consider x^{15}. The *binary method* [21, p. 399] computes it in six multiplications

$$x \to x^2 \to x^3 \to x^6 \to x^7 \to x^{14} \to x^{15}, \tag{4.2}$$

where each step takes the highest power computed so far, and either squares it or multiplies it by x. But the *factor method* [21, p. 401] computes $y = x^3$ in two multiplications, then y^5 in three more, for a total of five:

$$x \to x^2 \to (x^3 = y) \to y^2 \to y^4 \to (y^5 = x^{15}). \tag{4.3}$$

When powers of x are multiplied, the exponents are added ($x^a x^b = x^{a+b}$), so the problem reduces into an additive one: starting from 1, add successively any two integers so far computed, finally reaching the *target* n. A sequence of integers thus obtained is an *addition chain*, formally,

$$(1 = a_0,\ a_1,\ \ldots,\ a_r = n) \tag{4.4}$$

such that for any $k > 0$, we have $a_k = a_i + a_j$ for some $i, j < k$. Note that i and j need not be distinct.

The alternative computations (4.2) and (4.3) are represented by the addition chains $(1, 2, 3, 6, 7, 14, 15)$ and $(1, 2, 3, 6, 12, 15)$. The trivial method,

which multiplies by x successively, is represented by $(1, 2, 3, \ldots, 13, 14, 15)$. Evidently, for a given target n there can exist many addition chains. Usually one wishes to find the shortest ones.

For a bibliography on addition chains see, *e.g.*, Guy [18, pp. 169–171]. The Online Encyclopedia of Integer Sequences has several entries about them, including A003313, "Length of shortest addition chain for n" [32].

The term *addition chain* is somewhat of a misnomer, since an element need not be computed from the immediately previous element (as in a "chain") but from any two earlier elements. In fact the additions form a directed, acyclic multigraph. Clift has used graph-theoretic methods to determine the shortest addition chains for targets $n \leq 2^{32}$ [10].

By minimizing the number of multiplications we are in essence assuming that each multiplication has unit cost. For many applications this may be true, or a reasonable approximation, but as a counterexample note *matrix-chain multiplication* of matrices of different dimensions [13, pp. 302–309].

4.2 Computing several powers

Let us expand the previous section's task by requiring *all* consecutive powers $x^2, \ldots, x^n$ by multiplication from x. Is there now any room for optimization?

It only takes a moment of thought to realize that $n - 1$ multiplications are both necessary and sufficient. Since we need $n-1$ different results, surely we must perform at least that many operations. A straightforward solution is to multiply by x repeatedly.

Although optimizing the computation of all consecutive powers is trivial (assuming unit cost of multiplication), there is an interesting generalization where one requires the computation of a given *set* of powers such as $\{x^4, x^{10}, x^{20}\}$. This leads to a nontrivial optimization problem [14].

4.3 The postage stamp problem

Before continuing to our task of optimizing convolutions, let us embrace another problem concerning addition of integers.

Suppose that an envelope may carry at most two stamps. Stamps are available in k different positive integer values $A = \{a_1 < a_2 < \ldots < a_k\}$. The *local postage stamp problem* is: Find maximal integer n such that any integer postage fare between 1 and n can be paid, as a sum of one or two stamps. Possible sums are said to be *generated* by A. The integer n is called the *range* of the stamps, and denoted by $n_2(A)$.

The *global postage stamp problem* is: Given k, choose k stamp values $A = \{a_1, \ldots, a_k\}$ so as to maximize $n_2(A)$. The set A is then *extremal*, and the range thus attained is $n_2(k)$. The extremal set need not be unique. For example, $n_2(7) = 26$ with three solutions:

$$\begin{aligned} A_1 &= \{1, 2, 5, 8, 11, 12, 13\}, \\ A_2 &= \{1, 3, 4, 9, 10, 12, 13\}, \\ A_3 &= \{1, 3, 5, 7, 8, 17, 18\}. \end{aligned} \tag{4.5}$$

Note that $n_2(A)$ is not necessarily the largest sum generated. It is the largest of the *consecutive* generated sums. Although A_3 generates $18 + 18 = 36$, its range is only 26, since the consecutive integers $1, \ldots, 26$ are generated, but 27 is not.

Instead of *at most* two stamps, we can quite equivalently require that the postage is paid with *exactly* two stamps, if we include a zero stamp $a_0 = 0$; then with any stamp a_i we can pay $a_i = 0 + a_i$. The zero stamp is usually not counted in the length of the stamp set.

The history of the postage stamp problem is difficult to trace, as it seems to have been reinvented many times. Alter and Barnett [2] credit its initial mathematical statement to Rohrbach's 1937 article [37]. The allusion to postage stamps probably started in recreational mathematics. In number theory, the set of postage stamps is usually called an *additive 2-basis*, or *eine Basis zweiter Ordnung* [37, 40]. For bibliography see Selmer [39] and Guy [18, pp. 123–127].

Extremal solutions are known only for $k \leq 24$, the largest ones giving $n_2(24) = 212$. They have been found by several authors with variations of

k	year	author	method, other notes
5	1955	Stöhr [40, p. 48]	"Durch systematisches Probieren"
7	1960	Riddell [35]	"A few days of hand labor" [36]
11	—	Victor; Varol	Unpublished, cited in [42, p. 307]
12	1969	Lunnon [25]	Backtracking algorithm
13	1977	Alter & Barnett [1]	} Priority unclear
13	1978	Riddell & Chan [36]	
16	1981	Mossige [27]	
19	1993	Challis [7]	Combinatorial pruning
22	2010	Challis & Robinson [8]	
23	2013	Challis & Robinson [9]	
24	2014	Kohonen & Corander [22]	

Table 4.1: History of $n_2(k)$ by year of publication.

exhaustive search, as summarized in Table 4.1.[1] We will discuss some of these computational methods in Section 4.5.

Currently known solutions are listed at the end of Article II. For up-to-date results see the OEIS sequence A001212, "Solution to the postage stamp problem with n denominations and 2 stamps" [32].

Since $n_2(k)$ is difficult to compute exactly, much attention has been paid to asymptotic lower and upper bounds of the form

$$Lk^2 + O(k) \;\le\; n_2(k) \;\le\; Uk^2 + O(k). \tag{4.6}$$

The tightest known bounds are given by

$$L = \frac{2}{7} \approx 0.28571, \qquad U = 0.46906. \tag{4.7}$$

The lower bound is due to Mrose [28] and has not been improved since. The upper bound has seen a series of incremental improvements, the latest by Habsieger [19], who also gives an account of earlier results.

[1] It is unclear whether Alter & Barnett or Riddell & Chan were the first to compute $n_2(13)$. The references to Riddell's master's thesis in 1960, and Victor's and Varol's unpublished results are from Riddell & Chan [36] and Wagstaff [42].

Here we describe only the most elementary bounds. A lower bound on $n_2(k)$ can be proven by constructing an additive 2-basis, such as

$$A = \{1, 2, 3, \ldots, m, 2m, 3m, \ldots, m^2\}, \tag{4.8}$$

where m is a positive integer, and $k = |A| = 2m-1$. Then $n_2(A) = m^2+m$, and we have

$$n_2(k) \geq n_2(A) = \frac{k^2}{4} + O(k). \tag{4.9}$$

An easy upper bound is obtained from a counting argument. From $k+1$ stamp values (including the zero stamp), one can choose $\binom{k+2}{2}$ unordered pairs (note that using the same stamp value twice is allowed); thus at most that many different sums can be formed, and we get

$$n_2(k) \leq \frac{k^2}{2} + O(k). \tag{4.10}$$

It is interesting that these elementary arguments already provide bounds with coefficients 0.25 and 0.5, which are remarkably close to the state-of-the-art coefficients (4.7).

We will encounter the counting argument again in Section 4.5 as an algorithmic technique, and yet again in Section 4.6.

4.4 Postage stamps and multiplication

Let us now return to our clustering task: in particular, to determining the posterior distribution for the number of clusters. We denote it here by c, so as not to be confused with k, the number of stamps.

For this we need $f^c(U)$ for all $c = 1, \ldots, n$. In light of Section 4.2 it would seem that the best we can do is to compute sequentially $f^2, f^3, \ldots, f^n$, requiring $n-1$ subset convolutions.

Not so! We do not need the full convolution tables f^c. From each table we only need the single value $f^c(U)$. Computing this one value is substantially less work than computing $f^c(X)$ for all sets $X \subseteq U$. If the full subset

convolution tables are available for some suitable smaller integers a and b, such that $a + b = c$, then

$$f^c(U) = \sum_{X \subseteq U} f^a(X)\, f^b(U \setminus X), \tag{4.11}$$

where the sum contains $2^{|U|}$ terms. Each term requires just two table lookups and one scalar multiplication.

Direct evaluation of this sum takes $O(2^{|U|})$ time. Since it gives the convolution only for a single set U, we will call it *single subset convolution*. Although this is still exponential, in practice it can be vastly faster than full subset convolution, which takes $O(3^{|U|})$ or $O(2^{|U|}\ |U|^2)$ depending on the method used.

For a concrete example, suppose that $n = 10$, we are given f as input, and we need the values $f^c(U)$ for $c = 1, \ldots, n$. Only three full subset convolutions are required if we proceed as follows (see Figure 4.1).

Step 1. Compute $f * f = f^2$, then $f^2 * f^2 = f^4$, and then $f^4 * f = f^5$.

Step 2. Extract $f(U)$, $f^2(U)$, $f^4(U)$ and $f^5(U)$ from the tables.

Step 3. Compute the remaining values of $f^c(U)$ according to (4.11).

For more generality, let n be arbitrary, and again require $f^c(U)$ for $c = 1, \ldots, n$. For simplicity assume unit cost for a full convolution, and zero cost for a single convolution. We aim to perform as few full convolutions as possible. How many, and which ones should we perform?

Let A be the set of indices a for which f^a is computed in full. It is convenient to include the value 1, corresponding to the input f^1. In the previous example we thus had $A = \{1, 2, 4, 5\}$.

There are two crucial requirements that A must satisfy. First, when computing the full convolution f^c, where $c \in A \setminus \{1\}$, we must already have the full convolutions f^a and f^b for some smaller indices a, b such that $a + b = c$. In other words, A must be an addition chain.

Secondly, after the full convolutions have been computed, we must be able to obtain $f^c(U)$ for all indices $c = 1, \ldots, n$. For every such index c, we

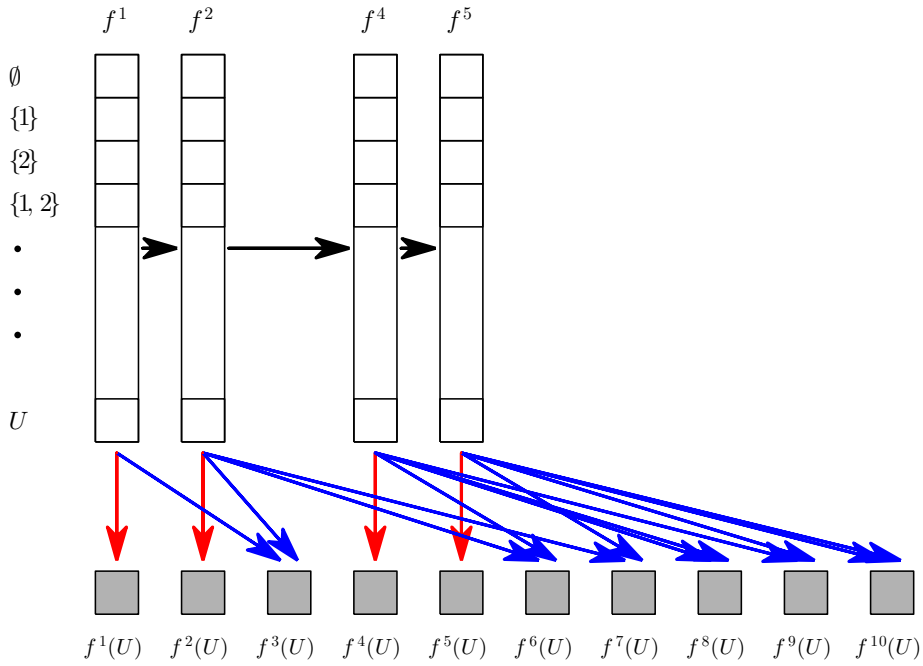

Figure 4.1: Reducing the number of subset convolutions. Only f^2, f^4 and f^5 are computed in full (black horizontal arrows), and $f^c(U)$ is then extracted (red vertical arrows). For other indices $f^c(U)$ is obtained by single subset convolution (blue slanted arrows). Compare to Figure 3.1 on page 18.

must either have the full f^c, so we can simply extract $f^c(U)$; or there must exist smaller indices a, b such that $a+b=c$, and that we have f^a and f^b, so we can use a single subset convolution. In short, A must be a set of postage stamps with range at least n.

Thus A must be *both* an addition chain *and* a postage stamp solution. In Article II, where we introduce this problem, we call such sets *stamp chains*. It is not immediately clear how one might find such sets.

Fortunately there is a direct connection between stamp chains and or-

dinary postage stamp solutions. If A is an extremal postage stamp solution with range n, then

$$\{1\} \cup (A + 1) \tag{4.12}$$

is an extremal stamp chain with range $n + 2$ (Theorem 21 in Article II). Simply take a known postage stamp solution, increase each stamp value by 1, and adjoin a new stamp of value 1. The result is not just another postage stamp solution: it is an addition chain as well.

For example, the unique extremal stamp set of length 5 is $\{1, 3, 4, 7, 8\}$, with range 16. It is not an addition chain. From (4.12) we get $\{1, 2, 4, 5, 8, 9\}$. This is a stamp chain with range 18.

Furthermore the construction (4.12) produces *all* extremal stamp chains (Theorem 23 in Article II). In other words, the problem of finding stamp chains completely reduces to the well-known postage stamp problem, for which extremal solutions are known up to length 24 (Article II), and good solutions are known for much larger lengths [8].

Although our task in this section was stated in terms of subset convolution, it can be generalized to other multiplicative tasks. Suppose for example that X is a square matrix of large dimensions $m \times m$; and that from consecutive powers $X^1, X^2, \ldots, X^n$ we need just a particular element, for example the lower right-hand corner $(X^i)_{mm}$. Now, if for two exponents a, b we have computed the full matrix powers X^a and X^b, then for exponent $a + b$ the desired element is easily obtained as

$$(X^{a+b})_{mm} = \sum_{j=1}^{m} (X^a)_{mj} (X^b)_{jm} \tag{4.13}$$

without needing to compute the matrix X^{a+b} in full. The minimal number of full matrix multiplications to perform is provided by a stamp chain.

4.5 Searching for extremal additive bases

We now depart from multiplicative tasks, and study the postage stamp problem in its own right: in particular, the search for extremal solutions.

4.5.1 Lunnon's algorithm

Computational approaches to the postage stamp problem start in 1969, with Lunnon describing a backtracking search algorithm [25]. We examine it in some detail as it forms a foundation for more sophisticated algorithms.

An additive basis $A_k = \{a_1 < \ldots < a_k\}$ is *admissible* if $n_2(A_k) \geq a_k$. Any basis that is *not* admissible cannot be extremal. When seeking extremal bases, we may as well confine our search within admissible bases.

Lunnon's algorithm constructs admissible bases incrementally. We must have $a_1 = 1$, otherwise the sum 1 would not be generated. Suppose we have selected the first j stamps, or the *j-prefix* $A_j = \{a_1 < \ldots < a_j\}$. The next stamp cannot be greater than $n_2(A_j) + 1$, otherwise we would leave a "gap" and the resulting basis would not be admissible.

Thus a_{j+1} must be chosen within $\{a_j + 1, a_j + 2, \ldots, n_2(A_j) + 1\}$. These values are tried in turn; for each $(j+1)$-prefix thus formed, continue recursively. This depth-first search enumerates all admissible bases of length k, as illustrated in Table 4.2. Their ranges are computed, and the record holders are the extremal bases.

The range distribution among admissible bases is striking (see Figure 4.2). Most bases visited by the search are far short of the maximum range. The search time is proportional to the number of admissible bases, which grows roughly exponentially with k. There are 305 226 admissible bases of length 10; over two million bases of length 11; almost 16 million bases of length 12; for more, see the OEIS sequence A167809 [32].

4.5.2 Challis's algorithm

Much effort is saved if parts of the search tree can be pruned by proving that those parts cannot contain extremal bases. Challis's algorithm does so through a combinatorial counting argument [7].

Recall from Section 4.3 that from a basis of k stamps, and the zero stamp, we get $\binom{k+2}{2}$ unordered pairs with repetition allowed. By the same token, from a j-prefix we get $\binom{j+2}{2}$ pairs. Thus when a j-prefix is extended

a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	range	
1	2	3	4	5	6	7	8	9	10	11	22	
1	2	3	4	5	6	7	8	9	10	12	22	
1	2	3	4	5	6	7	8	9	10	13	23	
1	2	3	4	5	6	7	8	9	10	14	24	
⋮					⋮					⋮	⋮	
1	2	3	7	11	15	19	23	25	26	27	30	
1	2	3	7	11	15	19	23	25	26	28	54	← extremal
1	2	3	7	11	15	19	23	25	26	29	34	
⋮					⋮					⋮	⋮	
1	3	5	7	9	11	13	15	17	18	37	38	
1	3	5	7	9	11	13	15	17	19	20	40	
1	3	5	7	9	11	13	15	17	19	21	22	

Table 4.2: Lunnon's algorithm searches through all admissible bases in lexicographic order (shown: partial listing of the 2 122 983 bases for $k = 11$).

with $k - j$ more stamps into a basis of length k, we get

$$p_{jk} = \binom{k+2}{2} - \binom{j+2}{2} \tag{4.14}$$

more pairs, and consequently at most that many new sums.

Suppose we know that $n_2(k) \geq T$, perhaps because during the search we have already found a basis with range T. Then any extremal basis must generate at least the integers $\{1, 2, \ldots, T\}$, which we call the *target set*.

Consider the search at a phase when a j-prefix has been constructed, such that it generates v different sums within the target set. We still have $g = T - v$ values not generated, or "gaps". If now $g > p_{jk}$, it is not possible for the remaining $k-j$ stamps to generate enough new sums to fill the gaps, and we can skip this branch of the search tree.

The computational savings are substantial. Consider the search for extremal bases of length 16, if we already know that $n_2(16) \geq 104$. The unpruned search tree contains about 9.3×10^{10} nodes. With pruning the search visits only about 2.9×10^8 of them, or 0.3%. For a concrete example,

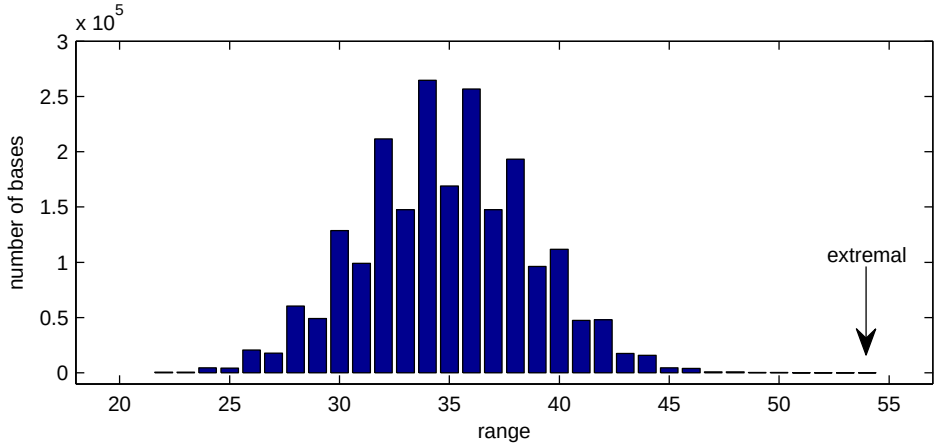

Figure 4.2: Distribution of the ranges of admissible bases of length 11. Very few bases come even close to the extremal range $n_2(11) = 54$.

the search visits among others the 14-prefix

$$\{1, 2, 3, 6, 9, 10, 11, 12, 13, 21, 28, 34, 42, 46\}, \tag{4.15}$$

which generates only 69 sums within the target set $\{1, 2, \ldots, 104\}$, with 35 still missing. By equation (4.14) the remaining two stamps can generate at most 33 new sums, so the prefix cannot lead to an extremal basis.

The counting argument is appealing in its simplicity. Rohrbach used it to prove an upper bound on $n_2(k)$ in general [37]. Here *within* a search, it provides an upper bound on $n_2(A_k)$ for bases A_k with a given j-prefix. Perhaps the more sophisticated upper bounds might be used similarly.

Our result $n_2(24) = 212$, published in Article II, was computed with a streamlined implementation of this algorithm. As a note of implementation, the search is straightforward to parallelize. We enumerated all 47 098 admissible bases of length 9, then started the search separately from each of these prefixes. The search was conducted on 200 processors over a weekend, taking 606 days of processor time.

4.5.3 Restricted and symmetric bases

Certain families of additive 2-bases have proven interesting. We have already observed that extremal bases are necessarily *admissible*. We now turn our attention to *restricted* and *symmetric* bases.

For convenience we now explicitly include $a_0 = 0$ in a basis, so we write $A_k = \{0{=}a_0 < a_1 < \ldots < a_k\}$. We also use the following number-theoretic notation when A is a set and a, b are integers.

Interval of integers	$[a, b]$	$= \{a, a+1, \ldots, b\}$
Sumset	$2A$	$= \{a + a' \ : \ a, a' \in A\}$
Mirror image	$b - A$	$= \{b - a \ : \ a \in A\}$

A *symmetric* basis is its own mirror image from the largest element:

$$A_k = a_k - A_k. \tag{4.16}$$

In a *restricted* basis, every element (and the largest one in particular) is at most half the range: $a_k \leq \frac{1}{2} n_2(A_k)$. Then in fact we have exactly

$$n_2(A_k) = 2a_k, \tag{4.17}$$

since for any basis $n_2(A_k) \leq 2a_k$.

The sumset $2A$ corresponds to our earlier notion of "sums generated from postage stamps A". Sumsets have interesting affine-like properties. In particular, if A_k is mirrored, the sumset is similarly mirrored:

$$2\,(a_k - A_k) = 2a_k - 2A_k. \tag{4.18}$$

Mirroring leads to the following observation [37, p. 4]. If A_k is symmetric and admissible, then by admissibility $2A_k \supseteq [0, a_k]$, and by (4.18) $2A_k \supseteq 2a_k - [0, a_k] = [a_k, 2a_k]$. Combining these we see that A_k is restricted.

This is a strong connection between symmetric and restricted bases. If a symmetric basis is at least admissible ($n_2(A_k) \geq a_k$), then it is automatically restricted as well ($n_2(A_k) = 2a_k$). Because of this "efficiency", it is not surprising that many extremal 2-bases are symmetric as noted by several authors [36, 42, 7].

The converse is not true. A restricted basis need not be symmetric. But if A_k is a restricted basis, then by (4.18) its mirror image $B_k = a_k - A_k$ is also a restricted basis (possibly different). For example, since the following basis A is restricted, with range $2 \times 22 = 44$, then so is its mirror image $B = 22 - A$ (shown in reverse order):

$$\begin{array}{lccccccccccc} A = & 0 & 1 & 2 & 3 & 7 & 11 & 15 & 17 & 20 & 21 & 22 & \\ & 22 & 21 & 20 & 19 & 15 & 11 & 7 & 5 & 2 & 1 & 0 & = B \end{array}$$

Let A_k be a restricted basis with range n (thus $a_k = n/2$). Since any prefix must be admissible, we have upper bounds for basis elements:

$$a_j \le n_2(A_{j-1}) + 1 \le n_2(j-1) + 1. \tag{4.19}$$

But since the mirror image $B_k = a_k - A_k$ is also a restricted basis, and thus admissible, we have likewise $b_j \le n_2(j-1) + 1$. Combining with the mirroring identity $b_{k-j} = a_k - a_j$ we obtain lower bounds

$$a_j \ge (n/2) - n_2(k-j-1) - 1. \tag{4.20}$$

These bounds are illustrated in Figure 4.3. They are rather tight near the middle of the basis ($j \approx k/2$). When searching for restricted bases of length k and range n, these bounds reduce the search space considerably.

A restricted basis can be composed of a prefix $A_j = \{a_0 < \ldots < a_j\}$ and a suffix $R = \{a_{j+1} < \ldots < a_k\}$. The prefix must be an admissible basis. Possible prefixes can be enumerated with Challis's algorithm, but most of them can be rejected because of the lower bound (4.20). The suffix is a mirror image of a prefix of the mirrored basis $B_k = a_k - A_k$, so we can likewise enumerate the possible suffixes.

We can then try each possible prefix-suffix pair in turn, checking whether their union is indeed a restricted basis. With this search algorithm, described in detail in Article III, we enumerated all restricted bases of maximum range for lengths $k = 25, \ldots, 41$. Previously they were known only for $k \le 24$. In 1981 Mossige listed all *symmetric* bases of maximum range for lengths $k \le 30$ [27], but it was not known whether there are asymmetric restricted

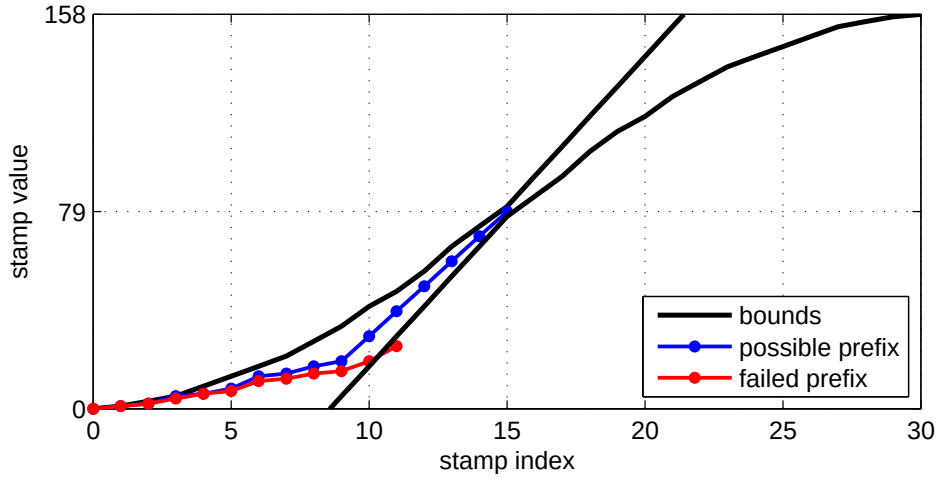

Figure 4.3: Search for restricted bases of length $k = 30$, range $n = 316$. Most admissible prefixes fail early by crossing the lower bound.

solutions as well. In fact our search discovered two such asymmetric solutions for $k = 30$. For length $k = 41$ our search took five days of processor time; compare to the 606 processor days we used for nonrestricted bases of the much smaller length $k = 24$.

Finally we must note that in Article III we used the lower bound only at the midpoint of the basis. As seen in Figure 4.3, a prefix may be rejected even earlier, if we check for the lower bound at each stamp. With this (unpublished) enhancement the search for $k = 41$ takes less than five *minutes* of processor time. Combinatorial searches like this can be tremendously affected by small changes in the pruning technique.

4.6 Large and small sumsets

We close this chapter with a brief glimpse into additive number theory in general, trying to see where postage stamps fit in that picture.

A central question in additive number theory is when the number of elements in the sumset $2A$ is large and when it is small, related to A itself. The upper bound is

$$|2A| \leq \binom{|A|+1}{2} = \frac{|A|^2 + |A|}{2}, \tag{4.21}$$

due to the counting argument already familiar to us: from $|A|$ elements one can form $\binom{|A|+1}{2}$ unordered pairs, allowing pairs of the form $\{a, a\}$. Equality is reached when all pairs have distinct sums, and A is then a *Sidon set*. An example is the geometric progression $\{1, p, \ldots, p^k\}$ [41, p. 74].

The lower bound is

$$|2A| \geq 2|A| - 1, \tag{4.22}$$

with equality when A is an arithmetic progression [29, p. 6]. In particular, if $A = [a, b]$, then $2A = [2a, 2b]$, and $|2A| = 2b - 2a + 1 = 2|A| - 1$.

Much of current additive number theory concerns situations where the sumset is small, typically $|2A| < c|A|$ with some small constant c. Sets with large sumsets are less well understood: it has even been written that "the study of sets with close to maximal doubling appears to be hopeless at present" [41, p. 58]. "Doubling" refers here to the quotient $\frac{|2A|}{|A|}$.

The postage stamp problem is concerned with sets whose sumset *contains* a large consecutive part $\{1, 2, \ldots, n\}$. Recalling (4.6) and (4.7), extremal bases have ranges roughly between $\frac{1}{4}|A|^2$ and $\frac{1}{2}|A|^2$, and their sumsets are at least this large, remarkably close to the maximum (4.21).

Perhaps it is not too facetious to hope that studies of the postage stamp problem will, for their part, contribute to the understanding of large sumsets. Towards that end, research on various generalizations may be needed, such as allowing negative stamp values. This and other generalizations have been suggested [17, pp. 246–247], but little is known about them.

Bibliography

[1] Ronald Alter and Jeffrey A. Barnett. Remarks on the postage stamp problem with applications to computers. In *Congressus Numerantium 19: Proceedings of the 8th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, pages 43–59, 1977.

[2] Ronald Alter and Jeffrey A. Barnett. A postage stamp problem. *The American Mathematical Monthly*, 87(3):206–210, 1980.

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] Andreas Björklund, Thore Husfeldt, Petteri Kaski, and Mikko Koivisto. Fourier meets Möbius: fast subset convolution. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing (STOC 2007)*, pages 67–74, 2007.

[5] Hans H. Bock. Probabilistic models in cluster analysis. *Computational Statistics & Data Analysis*, 23(1):5–28, 1996.

[6] Jonathan Borwein and David Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A. K. Peters, 2004.

[7] Michael F. Challis. Two new techniques for computing extremal h-bases a_k. *The Computer Journal*, 36(2):117–126, 1993.

[8] Michael F. Challis and John P. Robinson. Some extremal postage stamp bases. *Journal of Integer Sequences*, 13(2):Article 10.2.3, 2010.

[9] Michael F. Challis and John P. Robinson. Addendum to [8], July 2013.

[10] Neill E. Clift. Calculating optimal addition chains. *Computing*, 3(91): 265–284, 2011.

[11] Jukka Corander, Patrik Waldmann, Pekka Marttinen, and Mikko J. Sillanpää. BAPS 2: enhanced possibilities for the analysis of genetic population structure. *Bioinformatics*, 20(15):2363–2369, 2004.

[12] Jukka Corander, Mats Gyllenberg, and Timo Koski. Bayesian unsupervised classification framework based on stochastic partitions of data and a parallel search strategy. *Advances in Data Analysis and Classification*, 3(1):3–24, 2009.

[13] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press and McGraw–Hill, 1999.

[14] Peter Downey, Benton Leong, and Ravi Sethi. Computing sequences with addition chains. *SIAM Journal on Computing*, 10(3):638–646, 1981.

[15] Fedor V. Fomin and Dieter Kratsch. *Exact exponential algorithms*. Springer-Verlag, 2010.

[16] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[17] C. Sinan Güntürk and Melvyn B. Nathanson. A new upper bound for finite additive bases. *Acta Arithmetica*, 124(3):235–255, 2006.

[18] Richard K. Guy. *Unsolved Problems in Number Theory*. Springer, second edition, 2004.

[19] Laurent Habsieger. On finite additive 2-bases. *Transactions of the American Mathematical Society*, 2014. Published electronically.

[20] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[21] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison–Wesley, 1969.

[22] Jukka Kohonen and Jukka Corander. Addition chains meet postage stamps: Reducing the number of multiplications. *Journal of Integer Sequences*, 17(3):Article 14.3.4, 2014.

[23] Mikko Koivisto. An $O^*(2^n)$ algorithm for graph coloring and other partitioning problems via inclusion-exclusion. In *Proceedings of the*

47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), pages 583–590, 2006.

[24] John W. Lau and Peter J. Green. Bayesian model-based clustering procedures. *Journal of Computational and Graphical Statistics*, 16: 526–558, 2007.

[25] W. F. Lunnon. A postage stamp problem. *The Computer Journal*, 12: 377–380, 1969.

[26] V. V. Menon. On the maximum of Stirling numbers of the second kind. *Journal of Combinatorial Theory, Series A*, 15(1):11–24, 1973.

[27] Svein Mossige. Algorithms for computing the h-range of the postage stamp problem. *Mathematics of Computation*, 36(154):575–582, 1981.

[28] Arnulf Mrose. Untere Schranken für Extremalbasen fester Ordnung. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 48(1):118–124, 1979.

[29] Melvyn B. Nathanson. *Additive Number Theory: Inverse Problems and the Geometry of Sumsets*. Springer, 1994.

[30] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2003.

[31] Number theory. *Encyclopedia of Mathematics*. Springer, 2012. http://www.encyclopediaofmath.org/index.php?title=Number_theory. Cited January 30, 2015.

[32] OEIS. The On-line Encyclopedia of Integer Sequences. http://oeis.org. Cited November 23, 2014.

[33] Johan Pensar, Henrik Nyman, Timo Koski, and Jukka Corander. Labeled directed acyclic graphs: a generalization of context-specific independence in directed graphical models. *Data Mining and Knowledge Discovery*, DOI:10.1007/s10618-014-0355-0, 2014.

[34] Fernando A. Quintana and Pilar L. Iglesias. Bayesian clustering and product partition models. *Journal of the Royal Statistical Society B*, 65(2):557–574, 2003.

[35] James Riddell. On bases for sets of integers. Master's thesis, University of Alberta, 1960.

[36] James Riddell and Clara Chan. Some extremal 2-bases. *Mathematics of Computation*, 32(142):630–634, 1978.

[37] Hans Rohrbach. Ein Beitrag zur additiven Zahlentheorie. *Mathematische Zeitschrift*, 42:1–30, 1937.

[38] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, seventh edition, 2000.

[39] Ernst S. Selmer. The local postage stamp problem. part 1: general theory. Technical Report 42, Department of Pure Mathematics, University of Bergen, 1986.

[40] Alfred Stöhr. Gelöste und ungelöste Fragen über Basen der natürlichen Zahlenreihe. I. *Journal für die reine und angewandte Mathematik*, 194: 40–65, 1955.

[41] Terence Tao and Van Vu. *Additive Combinatorics*. Cambridge University Press, 2006.

[42] Samuel S. Wagstaff, Jr. Additive h-bases for n. In *Number Theory Carbondale 1979: Proceedings of the Southern Illinois Number Theory Conference Carbondale, March 30 and 31, 1979*, pages 302–327, 1979.

[43] Eric W. Weisstein. Stirling number of the second kind. From MathWorld, a Wolfram Web Resource. http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html. Cited November 23, 2014.