p

CERIAS Tech Report 2004-19

SUCCINCT SPECIFCATIONS OF PORTABLE DOCUMENT ACCESS POLICIES

by Marina Bykova, Mikhail Atallah

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Succinct Specifications of Portable Document Access Policies*

Marina Bykova, Mikhail Atallah
CERIAS and Computer Sciences Department
Purdue University
{mbykova,mja}@cs.purdue.edu

ABSTRACT

When customers need to each be given portable access rights to a subset of documents from a large universe of n available documents, it is often the case that the space available for representing each customer's access rights is limited to much less than n, say it is no more than m bits. This is the case when, e.g., limited-capacity inexpensive cards are used to store the access rights to huge multimedia document databases. How does one represent subsets of a huge set of n elements, when only m bits are available and m is much smaller than n? We use an approach reminiscent of Bloom filters, by assigning to each document a subset of the m bits: If that document is in a customer's subset then we set the corresponding bits to 1 on the customer's card. This guarantees that each customer gets the documents he paid for, but it also gives him access to documents he did not pay for ("false positives"). We want to do so in a manner that minimizes the expected total false positives under various deterministic and probabilistic models: In the former model we assume k customers whose respective subsets are known a priori, whereas in the latter we assume (more realistically) that each document has a probability of being included in a customer's subset. We cannot use randomly assigned bits for each document (in the way Bloom filters do), rather we need to consider the a priori knowledge (deterministic or probabilistic) we are given in each model in order to better assign a subset of the m available bits to each of the n documents. We analyze and give efficient schemes for this problem.

*Portions of this work were supported by Grants IIS-0325345, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

SACMAT'04, June 2–4, 2004, Yorktown Heights, New York, USA.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection; E.4 [**Data**]: Coding and Information Theory—*data compaction and compression*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems.

General Terms

Security, Design, Algorithms.

Keywords

Portable access rights, compact policy representation, access control, access control enforcement, algorithm design, computational complexity.

1. INTRODUCTION

We explore a model of a very large collection of objects where the customer is not limited to a predefined set of objects to which he may obtain access. With the rapid recent increase in the number and the level of maturity of on-line document collections (digital libraries and the like) which provide payment-based access to the documents, this model has emerged as an appropriate way of dealing with this explosive growth. In order to make access as flexible and convenient to the customer as possible, such systems might let each subscriber specify a set of documents to which to have access, and charge a fee according to the (subscription) request.

With such a scheme in place, each customer receives an access policy configuration unique to her subscription request. Access control enforcement should be performed at the object level, which requires each object in the data collection to carry its own access policy. From Bertino et al. [5], we adopt binary strings as the method of policy representation, which combines flexibility with light policy storage requirements. These are especially important in contexts where the access rights are portable and stored on space and/or battery-limited devices like cards and sensors.

As the number of documents in the data collection grows, it becomes infeasible to store very long access policy strings with each document or, alternatively, the user might be constrained in space with e.g., card-based access. Therefore we propose a scheme for reducing the length of policy strings, thus saving space but with the unavoidable side effect of losing precision of the customer requests' representation: A customer still has access to her whole subscription, but also

has access to some extra documents (we refer to such documents as "false positives"). More specifically, the customer has to be given an access-rights bitstring (called *access policy configuration* in the terminology of [5]) that is the logical OR of all her subscription documents' access policy bitstrings, because (as in [5]) access is granted to a document as long as the customer's access-rights bitstring is bitwise larger than that document's own access policy bitstring. This guarantees access to all subscribed documents, but also to some extraneous ones as well (the false positives). The goal of this work is to provide a scheme for document policy assignment such that the expected number of false positives is minimized.

We identify two flavors of the problem: deterministic, in which all customer subscription requests are known prior to policy assignment, and probabilistic, in which every document in the data repository has a number associated with it that indicates the probability of that document being included in a single customer request; the latter model has the advantage of not requiring prior knowledge of customer subscription requests (all it needs is a subscription probability for each document), so customer subscriptions can be processed as they arrive (well after policy assignment).

In the following summary, we provide a brief description of our results:

- For the probabilistic model, we give a scheme for minimizing the total additional cost, i.e. the total number of false positives over all documents in the repository, analyze its complexity, and give two policy assignment algorithms that run in $O(mn^2)$, where n denotes the number of documents in the data repository and m is the length of each policy string. We also give a scheme for minimizing the *maximum* number of false positives for any one subscription, and give a policy assignment algorithm that runs in $O(n \log n)$ time.
- For the deterministic model, we also analyze schemes for minimizing the total (cumulative over all customers) as well as the maximum (per customer) false positives. We do not provide algorithms for this model but instead show that this flavor of the problem can be solved using the probabilistic model.

After presenting our schemes, we discuss the issue of "drift" away from optimality as the system evolves over time through the addition/removal of customers, documents, or simply changes in the access probabilities.

The layout of this paper is as follows: In section 2, we start by presenting an overview of the work performed in this area. Section 3 provides a detailed description of the problem, followed by sections 4 and 5 which express the two flavors of the problem. These sections give efficient algorithms for computing a set of document policies that minimize expected false positives, and also prove the intractability of some more general versions of the problem. Section 6 discusses implementation and deployment issues. Finally, section 7 presents possible extensions and directions for future work. To avoid breaking the flow of the exposition, the highly technical proofs are given in the appendices.

2. RELATED WORK

Research in the area of access control for on-line document repositories mostly focuses on digital libraries and access to XML documents. Digital libraries usually employ access control enforcement at the system level, but work conducted by Payette and Lagoze [16] uses digital objects that carry and enforce policies. Payette and Lagoze [16], however, provide only a general framework for such a system and do not explore the problem of policy assignment.

Many documents on the web use XML, which provides a flexible way of representing documents of various types. Recently, there has been extensive work on securing access to XML documents and using XML as a tool for specifying security policies [6, 7, 8, 11, 12]. Security enhanced XML documents, among other security fields, include a policy field which lets the publisher enforce access control mechanisms. The most space efficient method of policy representation, binary strings, was used in Bertino et al. [5]. Bertino et al. [5], however, allocate one bit per policy and do not provide more insight on the policy assignment process. As was state earlier, this approach becomes expensive as the data collection grows in size. In this work we concentrate on finding an efficient solution to the policy assignment problem with reduced policy strings.

The idea of achieving space efficiency at the cost of a small probability of false positives was introduced in Bloom [9]. Bloom filter is a ingenious randomized data-structure that supports approximate membership queries with a very low rate of false positives and has been adopted in a wide range of applications ([10, 13, 15, 17] to name a few). Bloom filters, however, cannot apply directly to solve our problem. To keep the rate of false positives small, Bloom filters imply that the length of the filter (i.e., m) is larger than the number of items n it represents, which is not the case in our approach. Also, the original design of Bloom filters does not support the notion of probability of an item being selected, thus producing a suboptimal solution in our case.

Recent work that addresses the problem of software license management through portable card-based access rights was done by Aura and Gollmann [4] and more recently by Atallah and Li [3]. These provide ways of managing software licenses with tamper-resistant smart cards, where each smart card can store many software licenses and allow partial or complete transfers of licenses between cards, while using only $O(1)$ storage space. Constant storage space is achieved through the use of an untrusted user workstation (for which the software licenses were issued) that stores a list of software licenses bound to a smart card. This scheme works exceptionally well for managing software licenses but can hardly apply to our problem, in which there is no single logical place where additional information about requested documents can be stored, including the data provider's server itself.

Work in the direction of Digital Rights Management (DRM) is also concerned with the problem of unauthorized access to similar (copyrighted) material (for an overview of existing DRM challenges and solutions see [2]). Even though in our work we attempt to protect data objects in a context different from that of DRM (i.e., in our model we assume that the data provider has full control over what items a customer is allowed to have access to, while in the DRM settings an item must be protected when the customer controls the object itself, as well as the environment in which the object is used), there is a potential use of the techniques that we are developing for DRM due to portability of the access rights.

3. DESCRIPTION OF THE PROBLEM

As was mentioned above, we use binary strings to represent policies — both the policies of the documents in the data repository and the customer policy configurations. A customer is said to have access to a document if his configuration policy is bitwise greater than or equal to the policy of the document; i.e., every bit of the configuration policy is greater than or equal to the bit at the same position in the document policy. If a subscription request contains multiple documents, the resulting policy configuration, which is then given to the subscriber, is formed using a bitwise OR of the policies of all of the documents that compose the request. This guarantees that the subscriber gets access to all of the documents requested.

Assume that we are given n documents $1, 2, \ldots, n$, and each document i has a policy number r_i associated with it and can be accessed for the price c_i. We desire to have policies of less than n bits long (i.e., less than one bit per document), and bound the length of the policies by some number $m < n$, which unavoidably leads to the fact that some documents or subsets of the documents will have to share the same policy, and therefore some configuration policy might result in false positives. Below we address the problem of minimizing the false positives by assigning optimal policies to each document through the use of probabilistic and deterministic models.

4. PROBABILISTIC MODEL

Assume that each document i has an access probability p_i associated with it and all probabilities $p_1, \ldots, p_n$ are not correlated. Each document is viewed as an independent event with a probability $0 < p_i \leq 1$ of being included in a single subscriber request, and p_i with the value of 1 means that every customer subscribes to the document i. We view this model as more likely to exist in real life than the deterministic model since it is not tied to a specific set of customer requests and can be used in situations even when subscribers' requests are not known in advance. The deterministic variation of the problem can be easily converted to the probabilistic model by computing probabilities for each document based on the customers' requests.

The expected number of false positives, or "cost" of a particular policy assignment, is now the sum of the probabilities of all of the possible subsets of the documents weighted by the costs of documents not in the subset (not requested). Similar to the previous model, here we look at the problem of minimizing the total (cumulative) and the maximum (per customer) false positives.

4.1 Minimizing the Total False Positives

In the general case, the number of false positives of a policy assignment is computed by looking at every policy configuration possible, computing the probability that it occurs based on the documents that might result in that configuration and weighting them by the document costs.

To simplify this scheme, one might consider only certain policies for individual documents rather than allowing any random configuration. One approach that performs rather well in an average case is to assign to each document a policy number with only one out of m policy bits set to 1 and all others set to 0. Further simplification might allow only one document be requested by a single customer. These schemes are described in detail in the next subsections. In addition to the algorithms, we prove the intractability of the more general problem.

We henceforth assume, without loss of generality, that the documents are sorted by their probabilities in a non-increasing order, i.e., $p_1 \geq p_2 \geq \ldots \geq p_n$.

4.1.1 One Bit per Document

To minimize the total expected false positives using policies with one bit set, we divide of the documents into m groups: All documents within one group are assigned the same policy with only one bit set to 1 (the first group's documents have the first bit set to 1, the second group's documents have the second bit set to 1, etc.). Since a cost of an assignment is formed by adding the costs of all possible subsets, the cost of a group is computed as the sum of all possible requests made of the documents within the group weighted by the total cost of documents that were not in the request. There is an exponential number of subsets, but the next theorem shows that the sum can be computed efficiently. Specifically, for a group i of size s_i, the next theorem implies that we can compute the cost C_i as the following:

$$C_i = \sum_{j=1}^{s_i} c_{i_j}(1 - p_{i_j}) - \Big(\sum_{j=1}^{s_i} c_{i_j}\Big)\Big(\prod_{j=1}^{s_i}(1 - p_{i_j})\Big) \quad (1)$$

THEOREM 1. *For any set S of elements $i_1, i_2, \ldots, i_s$ with respective probabilities $p_1, p_2, \ldots, p_s$ and costs $c_1, c_2, \ldots, c_s$, the value of $\sum_{i=1}^{s} c_i(1-p_i) - \Big(\sum_{i=1}^{s} c_i\Big)\Big(\prod_{i=1}^{s}(1-p_i)\Big)$ is equal to the sum of probabilities of all possible subsets of S, weighted by the cost of elements not in the subset.*

PROOF. See Appendix A.

COROLLARY 1. *For any given set of elements 1, 2, ..., s, with probabilities p_i and costs c_i for $1 \leq i \leq s$, the cost of the group containing them all can be calculated in linear time.*

The total cost of grouping n documents into m groups can be calculated as the sum of the costs over all groups:

$$C = \sum_{i=1}^{m} C_i = \sum_{i=1}^{m} \left(\sum_{j=1}^{s_i} c_{i_j}(1-p_{i_j}) - \Big(\sum_{j=1}^{s_i} c_{i_j}\Big)\Big(\prod_{j=1}^{s_i}(1-p_{i_j})\Big) \right)$$

Now to solve the problem of minimizing the expected false positives count, we need to solve the problem of minimizing the value of C over all possible partitionings of n documents into m groups. As will be shown later in this paper, allowing arbitrary and distinct document costs makes the problem intractable. We therefore consider next the case where all c_i's are equal (e.g., to 1).

The Algorithm

Suppose we have n sorted documents with probabilities p_i and costs $c_i = 1$ for $1 \leq i \leq n$. The goal is to partition the documents into m groups, so that the total of false positives is minimized. Let s_i denote the number of documents in group i. Then the "cost" of group i, according to equation (1), is now equal to the sum of all possible requests made of the documents within the group, weighted by the number of documents that were not in the request, and is calculated as follows:

$$C_i = \sum_{j=1}^{s_i}(1 - p_{i_j}) - s_i \prod_{j=1}^{s_i}(1 - p_{i_j}) \quad (2)$$

We build a dynamic programming bottom-up implementation with running time of $O(mn^2)$ for finding an optimal partitioning of n documents into m groups, such that the value of $C = \sum_{i=1}^{m} C_i$ is minimized over all possible partitionings of the documents. The algorithm relies on the following lemma.

LEMMA 1. *Given n items with probabilities $0 < p_i \leq 1$ sorted in a non-increasing order and m, such that $m < n$, there exists an optimal solution to the problem of grouping the n items into m groups, in which the items are partitioned contiguously.*

PROOF. See Appendix B.

The above lemma tells us that we can find an optimal solution with no "gaps", i.e., if documents i and j $(i < j)$ are assigned to a certain group then so are all the documents $i+1, i+2, \ldots, j-1$.

DEFINITION 1. *Let $C_{i,j,t}$ be the minimum expected number of false positives if the only documents were $i, i+1, \ldots, j$ $(i \leq j)$ and only t bits were available for them $(t \leq m)$. Let $\mathcal{P}_{i,j,t}$ be the partition of the documents $i, i+1, \ldots, j$ into t groups that achieves $C_{i,j,t}$.*

The $C_{i,j,1}$'s are not hard to compute in $O(n^2)$ time for all i and j, $i \leq j$, using precomputed values for the sum and product:

$$C_{i,j,1} = \sum_{k=i}^{j}(1-p_k) - (j-i+1)\prod_{k=i}^{j}(1-p_k).$$

The corresponding $\mathcal{P}_{i,j,1}$ is a single group: $\{i, i+1, \ldots, j\}$.

THEOREM 2. *If $C_{1,j,t'}$ and $\mathcal{P}_{1,j,t'}$ are known for all $t' \leq j \leq n$ and $t' \leq t$, then $C_{1,j,t+1}$ and $\mathcal{P}_{1,j,t+1}$ for fixed j can be computed in linear time.*

PROOF. To find the best value for $C_{1,j,t+1}$, we must try to divide the set $1, \ldots, j$ into two sets of t and 1 groups, using all positions suitable for the split. For the partitioning to be optimal, the total cost of both sets should be minimal. The formula for finding $C_{1,j,t+1}$ then looks like the following:

$$\begin{aligned} C_{1,j,t+1} = \min_{k=1\,to\,j-t} \Bigg\{ & C_{1,j-k,t} + \sum_{l=j-k+1}^{j}(1-p_l) - \\ & -k\sum_{l=j-k+1}^{j}(1-p_l)\Bigg\} = \\ & = \min_{k=1\,to\,j-1}\{C_{1,j-k,t} + C_{j-k+1,j,1}\} \end{aligned}$$

From the formula we can see that the cost of any single split is computed in constant time, therefore computation of the optimal partitioning for $C_{1,j,t+1}$ takes linear time. □

COROLLARY 2. *For a fixed t, there are $O(n)$ $C_{1,j,t}$'s over all possible values of j, therefore the one bit per document case can be solved in $O(mn^2)$ time.*

This algorithm is naturally expandable to policies of arbitrary length.

4.1.2 One Bit per Document, One Document at a Time

We consider the situation here where a document is assigned only one bit, and a request is for only one document. This implicitly introduces an upper bound on the aggregate probability over all documents $\sum_{i=1}^{n} p_i \leq 1$. In this case, after partitioning all of the documents into groups $1, \ldots, m$ of size $s_1, s_2, \ldots, s_m$ respectively, the expected false positives for group i becomes:

$$C_i = \sum_{j=1}^{s_i} p_{i_j} \sum_{\substack{k=1 \\ k\neq j}}^{s_i} c_{i_k} \tag{3}$$

And the total cost respectively is:

$$C = \sum_{i=1}^{m} C_i = \sum_{i=1}^{m}\left(\sum_{j=1}^{s_i} p_{i_j} \sum_{\substack{k=1 \\ k\neq j}}^{s_i} c_{i_k}\right)$$

What if the "loss" due to the unintended granting of access to document i is proportional to its probability, i.e. $c_i \sim p_i$? The rationale for this model is that p_i is a good measure of the "desirability" of a document, and a very undesirable document should not contribute the same amount (to the total loss) as a very desirable document. The problem of optimizing allocation in this model turns out to be NP-complete, even for the simple case where a document is assigned to only one bit, and a request is for only one document.

As before, we let m be the number of bits available, n be the number of documents, and $p_1, \ldots, p_n$ be their respective probabilities of occurrence. For any given solution, for each document j, $f(j)$ denotes the position of the bit that corresponds to that document, $1 \leq f(j) \leq m$. For each bit position i, $1 \leq i \leq m$, let

$$f^{-1}(i) = \{j : f(j) = i\}$$

that is, $f^{-1}(i)$ is the set of documents corresponding to the ith bit. We use P_i to denote the sum of the probabilities of the documents in $f^{-1}(i)$: $P_i = \sum_{k\in f^{-1}(i)} p_k$. Using this notation, the expected loss of a given solution f is

$$\sum_{i=1}^{m}\sum_{k\in f^{-1}(i)} p_k * (P_i - p_k) = \sum_{i=1}^{m} P_i^2 - \sum_{j=1}^{n} p_j^2.$$

The second summation in the above is fixed (independent of f). So the optimization consists of minimizing the first summation, that is $\sum_{i=1}^{m} P_i^2$. Given n quantities $p_1, \ldots, p_n$ and a value U, partitioning these n quantities into m buckets such that the sum of the squares of bucket weights is $\leq U$ is known to be NP-Complete [14, 18].

The Algorithm

Here we consider the case when only one bit is set in a single document policy, only one document is requested by each customer, and the cost of each document i is $c_i = 1$. In this case, the formula for computing a cost of grouping s_i documents together into group i, according to equation (3), becomes:

$$C_i = (s_i - 1)\sum_{j=1}^{s_i} p_{i_j}$$

We give an $O(mn^2)$ time algorithm for policy assignment for that case.

For each document j, let $f(j)$ be the position of the bit that corresponds to that document, $1 \leq f(j) \leq m$. For each bit position i, $1 \leq i \leq m$, let $f^{-1}(i) = \{j : f(j) = i\}$, that is, $f^{-1}(i)$ is the set of documents corresponding to the ith bit. We henceforth use s_i to denote the number of documents in $f^{-1}(i)$.

THEOREM 3. *If documents x, y are such that $x \in f^{-1}(i)$ and $y \in f^{-1}(j)$, then in any optimal solution $p_x > p_y$ implies $s_i \leq s_j$.*

PROOF. Suppose not; that is, assume $p_x > p_y$ and $s_i > s_j$. Then swapping x and y between $f^{-1}(i)$ and (respectively) $f^{-1}(j)$ changes the expected number of false positives by:

$$p_x * s_j + p_y * s_i - p_x * s_i - p_y * s_j = (p_x - p_y) * (s_j - s_i)$$

which is negative. This contradicts the assumed optimality of the solution considered. □

COROLLARY 3. *Given $s_1, s_2, \ldots, s_m$ for an optimal solution, $s_1 \leq s_2 \leq \ldots \leq s_m$, then that optimal solution can be realized by assigning the first s_1 documents to bit 1 (these are the documents with the highest probabilities), the next s_2 documents to bit 2, ..., the last s_m documents to bit m.*

The above lemma reduces the problem to that of computing the s_i's, that is, of computing an optimal partition of the integers $1, 2, \ldots, n$ into m groups. Trying all possible partitions (subject to the increasing sorted order requirement) gives an algorithm that is exponential in m. Here we give an $O(mn^2)$ dynamic programming algorithm for the case of one document per access request.

DEFINITION 2. *Let $C_{j,t}$ be the minimum expected number of false positives if the only documents were $1, 2, \ldots, j$ and only t bits were available for them ($t \leq m$). Let $\mathcal{P}_{j,t}$ be the partition of the integers $1, 2, \ldots, j$ into t groups that achieve $C_{j,t}$. We use $s_{j,t,1}, \ldots, s_{j,t,t}$ to denote the sizes of the groups of partition $\mathcal{P}_{j,t}$, $s_{j,t,1} \leq \ldots \leq s_{j,t,t}$.*

The $C_{j,1}$'s are trivial to compute in linear time for all j: $C_{j,1} = (j-1)\sum_{k=1}^{j} p_k$. The corresponding $\mathcal{P}_{j,1}$ is a single group: $\{1, \ldots, j\}$.

THEOREM 4. *If $C_{j,t'}$ and $\mathcal{P}_{j,t'}$ are known for all $1 \leq j \leq n$ and $t' \leq t$, then $C_{j,t+1}$ and $\mathcal{P}_{j,t+1}$ for all $1 \leq j \leq n$ can be computed in $O(n^2)$ time.*

PROOF. Here is how we compute $C_{j,t+1}$ and $\mathcal{P}_{j,t+1}$. First, observe that if we knew $s_{j,t+1,t+1}$ then we could easily compute $C_{j,t+1}$ as:

$$C_{j,t+1} = C_{j-s_{j,t+1,t+1},t} + (s_{j,t+1,t+1} - 1)\times$$
$$\times(p_{j-s_{j,t+1,t+1}} + \ldots + p_j)$$

and the corresponding $\mathcal{P}_{j,t+1}$ would simply be $\mathcal{P}_{j-s_{j,t+1,t+1},t+1}$ augmented with the group $\{j - s_{j,t+1,t+1} + 1, \ldots, j\}$. If we pre-compute all sums of probabilities from p_1 on, then we could do the above in constant time. But we do not know $s_{j,t+1,t+1}$, and must try the $O(j)$ possible values for it and choose the best among them. We must of course carry this out for all j values, which is why the theorem claims the $O(n^2)$ time bound. □

COROLLARY 4. *The one bit per document, one document per request case can be solved in $O(mn^2)$ time.*

4.2 Minimizing the Maximum Per-Customer False Positives

Here instead of minimizing the cumulative total of false positives, we minimize the maximum per-customer false positives. In the probabilistic model each combination of documents has a non-zero probability of being requested, therefore the problem of minimizing the maximum per-customer false positives reduces to the problem of minimizing the false positives introduced by a single document.

If subscription to any document costs the customer the same amount of money as any other document ($c_i = c_j$ for $1 \leq i \leq j \leq n$), then we solve the minimization problem by dividing all documents into m (the number of bits available for policy numbers) groups in such a way that each group contains the same number of documents. In other words, we have m groups of size n/m and assign the same policy bit to all documents within the same group.

If the subscription fee varies from document to document, then we should take into account the varying costs. In this case, the sum of costs of all documents within the same group should stay about the same value across all groups, i.e., the sum of costs of all documents within a group is about $(\sum_{i=1}^{n} c_i)/m$.

Assume that we calculated m optimal sets of documents and they are S_1, S_2, ..., S_m. Then the maximum per-customer false positives is computed as:

$$C = \sum_{i=1}^{n} c_i - \sum_{j=1}^{m} \min_{i=1 \text{ to } n} \{c_i \,|\, i \in S_j\} \tag{4}$$

i.e., the cost of all documents decreased by the minimal document cost within each group. As we can see, the maximum per-customer false positives stays high even after its minimization.

The Algorithm

The purpose of the algorithm is to come up with a partitioning (i.e., $S_1, \ldots, S_m$) that minimizes the value in the above Equation 4. The first term of that formula is constant and cannot be changed, therefore the problem of computing an optimal solution reduces to maximizing the second term of the equation. In other words, in the optimal partitioning of n documents into m groups the sum of minimal elements of the groups is maximal. The following theorem states that, given n sorted elements, such a partitioning can be found in linear time.

THEOREM 5. *Given n documents 1, 2, ..., n sorted according to their costs c_i for $1 \leq i \leq n$, so that $c_1 \geq c_2 \geq \ldots \geq c_n$, a solution to the problem of computing a document policy assignment that minimizes the maximal (over all individual customers) individual customer false positives, can be computed in $O(n)$ time.*

PROOF. The algorithm for constructing an optimal partitioning works as follows. We first create $m-1$ groups of size 1 by placing the $m-1$ documents with the highest costs in a group by themselves. The rest of the $n-m+1$ documents compose the last mth group. With this approach, we have:

$$\sum_{j=1}^{m} \min_{i=1 \text{ to } n} \{c_i \,|\, i \in S_j\} = c_1 + c_2 + \ldots + c_{m-1} + c_n$$

where S_j refers to the jth group in the partitioning of n documents into m groups.

It is not hard to see that this algorithm maximizes the sum of minimal elements over all groups because the element with the smallest cost c_n is unavoidably included in the sum and all other costs in the summation are maximal. Therefore moving any document from one group to another decreases the value of the sum, thus making it suboptimal. □

COROLLARY 5. *If the documents are not originally sorted by their costs, the optimal solution to the problem of minimization of the maximal per-customer false positives can be computed in $O(n \log n)$ time.*

Note that document probabilities are not included in the solution because the goal here is to minimize the cost of a request in the worst case scenario, regardless of its probability of occurring (which is always non-zero).

5. DETERMINISTIC MODEL

Suppose there are k subscribers 1, ..., k who request s_1, ..., s_k documents, respectively. Then for optimal policy assigning all documents should be grouped according to the subscribers' requests. The problem of minimization can target either the total (cumulative) false positives or the maximum per-customer false positives. We describe each of the models in turn.

Although the deterministic model is less realistic, it is still useful to characterize its inherent difficulty. As was pointed out earlier, in practice the probabilistic framework can be used to approximately solve the deterministic one, and as this section shows the latter one is essentially intractable even in a simplified special case.

5.1 Minimizing the Total False Positives

Let subscriber i request documents $i_1, \ldots, i_{s_i}$, then the false positives count for that customer is:

$$C_i = f^{-1}(\bigvee_{j=1}^{s_i} r_{i_j}) - \sum_{j=1}^{s_i} c_{i_j} \tag{5}$$

where the cost of policy r is calculated as:

$$f^{-1}(r) = \sum_{i=1}^{n} \{c_i \,|\, r \geq_b r_i\}^*$$

Then the problem is to minimize the total false positives (cumulative over all customers):

$$C = \min\left\{\sum_{i=1}^{k} C_i\right\} = \min\left\{\sum_{i=1}^{k}\left(f^{-1}(\bigvee_{j=1}^{s_i} r_{i_j}) - \sum_{j=1}^{s_i} c_{i_j}\right)\right\}$$

DEFINITION 3. *A policy assignment to documents $1, \ldots, n$ is said to be optimal if the value of C is minimal over all possible policy assignments to the documents.*

5.2 Minimizing the Maximum Per-Customer False Positives

In the problem of minimizing the maximum per-customer false positives, the extra cost per customer i is the same as in equation (1), and the maximum per-customer false positives count is

$$C = \min\left\{\max_{i=1 \text{ to } k} C_i\right\}$$
$$= \min\left\{\max_{i=1 \text{ to } k}\left(f^{-1}(\bigvee_{j=1}^{s_i} r_{i_j}) - \sum_{j=1}^{s_i} c_{i_j}\right)\right\}$$

5.3 Complexity of the Deterministic Version of the Problem

Here we refer to the special case of the problem when there is only one bit set to 1 in a policy of a single document and all other bits are set to 0, each customer requests 2 documents and the length of policy numbers $m = 2$. We show that the deterministic case, even its special case, is NP-hard.

The reduction is from the known NP-hard problem of graph bisection: Given an n-vertex undirected graph (n is even), partition its vertices into two sets V_1 and V_2, of size $n/2$ each, so as to minimize the number of edges between V_1 and V_2.

Given an instance $G = (V, E)$ of the graph bisection problem, the instance of our problem is created as follows. Create an $(n+2)$-document version of our problem where n of the documents correspond to vertices of G, and documents $n+1$ and $n+2$ are additional ones that do not correspond to any vertices of G.

For every edge (i, j) of G, create a customer who requests documents i and j. Then create n^5 additional customers each of whom requests a pair of documents: n^4 of the customers request the documents 1 and $n+1$, another n^4 of them request the documents 2 and $n+1$, ..., n^4 request documents n and $n+1$. Similarly create n^5 additional customers each of whom requests a pair of documents: n^4 of the customers request the documents 1 and $n+2$, another n^4 of them request the documents 2 and $n+2$, ..., n^4 request documents n and $n+2$. Let D be the deterministic problem so created.

Claim: Any solution that optimally solves D also solves the minimum graph bisection of G by simply putting in V_1 the vertices that correspond to documents that were assigned to the first bit, and in V_2 the vertices that correspond to documents that were assigned to the second bit.

The rest of this section proves the above claim. Consider any optimal solution S for D, and assume without loss of generality that document $n+1$ was assigned to bit 1.

Let n_1 be the number of documents from $\{1, \ldots, n\}$ assigned to bit 1, n_2 be the number of documents from $\{1, \ldots, n\}$ assigned to bit 2, $e_{1,1}$ be the number of edges of G both of whose corresponding documents are assigned to bit 1, $e_{2,2}$ be the number of edges of G both of whose corresponding documents are assigned to bit 2 (hence $n_2 = n - n_1$), and $e_{1,2}$ be the number of edges of G whose corresponding documents are assigned to different bits. If we use C_1 (resp., C_2) to denote the total loss in solution S if document $n+2$ happens to be assigned by S to bit 1 (resp., to bit 2), then we have:

$$C_1 = (e_{1,1} + 2n_1n^4)(n_1 + 2 - 2) + e_{2,2}(n_2 - 2) + \\ +(e_{1,2} + 2n_2n^4)(n + 2 - 2)$$

$$C_2 = (e_{1,1} + n_1n^4)(n_1 + 1 - 2) + (e_{2,2} + n_2n^4)(n_2 + 1 - 2) + \\ +(e_{1,2} + n_1n^4 + n_2n^4)(n + 2 - 2)$$

Now observe that, in the above, the terms not involving n^4 add up to less than n^4 (in fact less than $n^3 + 2n^2$), so that any

*Here we define $\geq_b$ to mean "bitwise greater than or equal to".

solution that minimizes C_1 (resp., C_2) must also minimize the following C_1' (resp., C_2') obtained from C_1 (resp., C_2) by considering only the terms that multiply n^4:

$$C_1' = 2n_1^2 + 2n_2 n$$

$$C_2' = n_1(n_1 - 1) + n_2(n_2 - 1) + n_1 n + n_2 n.$$

Using $n_1 + n_2 = n$ in the above gives:

$$C_1' = 2n_1^2 - 2n_1 n + 2n^2$$

$$C_2' = n_1^2 + (n - n_1)^2 - n + n^2 = 2n_1^2 - 2n_1 n - n + 2n^2.$$

Because $C_1' - C_2' = n > 0$, it follows that S must have assigned document $n + 2$ to bit 2. Therefore the problem is to minimize C_2'. The minimum for C_2' is easily seen to occur for $n_1 = n/2$, which implies that $n_2 = n/2$. This immediately implies that $|V_1| = |V_2| = n/2$, as required. We now must prove that S minimizes $e_{1,2}$. Re-writing L with each of n_1 and n_2 replaced by $n/2$ gives:

$$C_2 = n^5(n-2)4^{-1} + e_{1,1}(n-2)2^{-1} + n^5(n-2)4^{-1} + \\ + e_{2,2}(n-2)2^{-1} + n^6 + ne_{1,2}$$

which is minimized when the following L'' is minimized:

$$C'' = (e_{1,1} + e_{2,2})(n-2)2^{-1} + ne_{1,2}.$$

Because $e_{1,1} + e_{2,2} + e_{1,2} = |E|$, any solution that minimizes C'' must necessarily minimize $e_{1,2}$. □

This guarantees that the deterministic problem in general, for arbitrary document policies and variable values of m, is also NP-hard.

6. IMPLEMENTATION AND DEPLOYMENT ISSUES

The simplicity of the algorithms developed for the probabilistic model is balanced by the potential vulnerability of the scheme to information-sharing, among different users, about correlations between different documents' bit patterns, e.g., "if you pay for documents $i_1, i_2, \ldots, i_k$ then you get free access to document j". Schemes that do not have this vulnerability are considerably more computationally expensive [1], so the present scheme has a crucial virtue of being simple and inexpensive. The ideal deployment for the scheme of this paper is in dynamic situations where the document repository changes rapidly, the policies need to be re-generated on a periodic basis, or other situations where the card contents are refreshed periodically through interaction (say, once a month or more frequently) with a server. For example, access to web sites or portions thereof often satisfies this rapid-change condition. In such cases, the above-mentioned correlation attacks are harder to carry out because the existing correlations become obsolete by the time they are discovered by the attackers. Furthermore, the sharing of correlation information among attackers is often problematic for the attackers, because the internal (to the document server and the smartcard) representation of a document is not accessible to the attackers, who may not be able to share a coherent description of the document (content changes daily or hourly, as does the URL).

What happens in between two of the above-mentioned periodic re-generations? During that in-between time period, new customers join, some existing ones drop out, new documents are added and old ones removed, or access patterns (i.e., the probabilities used in the optimization) change as previously "hot" documents go out of fashion or vice-versa. Of course these events are processed in real-time as they occur (they cannot wait for the next re-generation), but as these things happen the system is slowly drifting away from optimality. This is in fact another reason (in addition to the above-mentioned necessity of making the system less vulnerable to information-sharing attacks) why optimization needs to be periodically re-done, i.e., our scheme used with the new data to compute the new optimal solution. We leave the design of a self-evolving system that does not need such periodic check-points for future research (in fact we already have partial solutions to this [1]).

In the rest of this section we discuss operational and implementation issues which might be useful for realization of this approach. In this discussion we follow the probabilistic mode of operation.

In the initial (or setup) stage, the data owner runs the policy assignment algorithm and determines an exact policy assignment for the data collection. In order to do this, the document probabilities should be known a priori. For existing repositories, such information can be estimated using the past history; for new collections of documents, such probabilities can be set based on expected access frequencies and refined over the course of operation. The probabilities of access are considered to be the data owner's private data and are not known to the general public (which complicates discovery of documents that "come for free" with an order).

Due to the large size of the data repository, it is expected that a (possibly significant) number of documents will not have unique probabilities. In this case, the data owner has the freedom to choose how the documents with the same probabilities are combined into different groups. A good strategy under these circumstances is to place the same topic documents into different groups, so that the likelihood of having documents of the attacker's interest among the false positives is minimized. In other words, if an attacker is interested in documents on, say, computer security, then such documents are distributed among a number of groups, and are mixed with a large variety of documents on different topics. Furthermore, the set of documents (and/or their topics) that are combined together in one group should vary from one policy assignment to another (note that such randomization only changes the ordering of documents in the entire set of documents, but does not require modifications to the dynamic programming algorithms themselves). Such a distribution is useful to the attacker only if he has a broad range of topics of interest.

Even more randomization can be done to thwart attackers if, for example, we modify the algorithm so it ignores probabilities' least-significant bits: This causes many more "ties" and choice points in the optimization, which when resolved randomly will result in a very different set of policies from one periodic optimization to the next even if very little change to the customers/documents/probabilities occurred in the in-between time period.

After generating the document policy assignment, the data repository is set to accept user requests. When a customer subscribes to the service and provides a list of documents to be included in the subscription, a policy configuration for that request is constructed in such a way that all doc-

uments are covered (using bitwise OR of all documents in the requests), and the customer is charged accordingly. The customer, along with the a smart card that contains the access policy configuration, is given a list of documents in the request. Now any request to access a document from the list of the purchased documents, which is made with the use of the card, will be authorized. Access to a number of other documents might also be granted due to existence of false positives.

In order to minimize the loss caused by the presence of false positives, the data owner might want to reduce document probing by limiting the number of requests to access documents that result in denial of access. One possibility here is to use a strategy similar to "t strikes and you are out", where the card erases itself when the maximum number of unauthorized requests (in this case t) is made. When such a strategy is adopted, each customer needs to be explicitly notified of this policy at the time of purchase and know what t for the card is (which, for example, can be a function of the number of documents in the subscription request). The list of documents included in the order will help the customer to decrease the probability of error due to mistyped references to the legitimate documents. This strategy can significantly decrease the amount of document probing because of the fear of losing the card, but the care should be exercised to decide what values of t are appropriate in order to not deny access to documents in cases when no probing is taking place.

7. CONCLUSIONS AND FUTURE DIRECTIONS

In this work we examined two basic models of the policy assigning optimization problem: deterministic and probabilistic. A model for the problem could be more complicated, with certain requirements added to the documents, their policies, or some other parameters, and its solution might differ significantly from results presented here. We envision the following ways of extending the current work.

In the probabilistic model we assumed independent probabilities, while in real life they are likely to be correlated. For example, if a customer selects a document from one section, he is more likely to select another document from the same section than from a new one. In this case, the expected false positives count should be smaller since we can assign similar policies to the documents from the same sections.

There could be more than one type of data item. For example, a rather simple approach can use elements and element attributes as two major categories of items. A more sophisticated approach might include documents, packages of documents, sections within documents, and single items inside sections. Alternatively, one could extend this work to hierarchically structured sets of documents (think of web pages).

8. ACKNOWLEDGMENTS

We would like to thank Ethan Blanton and anonymous reviewers for their useful comments.

9. REFERENCES

[1] M. Atallah and M. Bykova. "Portable and Flexible Document Access Control Mechanisms," Working paper, 2004.

[2] M. Atallah, K. Frikken, C. Black, S. Overstreet, and P. Bhatia. "Digital Rights Management," *Practical Handbook of Internet Computing*, Munindar Singh (Ed.), CRC Press, 2004.

[3] M. Atallah and J. Li. "Enhanced Smart-card based License Management," *IEEE International Conference on E-Commerce (CEC),* 24–27 June 2003, pp. 111–119.

[4] T. Aura and D. Gollmann. "Software License Management with Smart Cards," *USENIX Workshop on Smart Card Technology*, USENIX Association, May 1999.

[5] E. Bertino, B. Carminati, E. Ferrari, B. Thuraisingham and A. Gupta. "Selective and Authentic Third-party Distribution of XML Documents," *Working Paper,* Sloan School of Management, MIT, 2002, `http://papers.ssrn.com/sol3/papers.cfm?abstract_id=299935.`

[6] E. Bertino, S. Castano and E. Ferrari. "On Specifying Security Policies for Web Documents with an XML-based Language," *SACMAT'01,* Chantilly, USA, May 2001.

[7] E. Bertino, S. Castano and E. Ferrari. "Securing XML Documents with Author-$\mathcal{X}$," *IEEE Internet Computing,* Vol. 5, No. 3, pp. 21–31, 2001.

[8] E. Bertino and E. Ferrari. "Secure and Selective Dissemination of XML Documents," *ACM Transactions on Information and System Security,* Vol. 5, No. 3, August 2002, pp. 290–331.

[9] B. Bloom. "Space/time trade-offs in hash coding with allowable errors." *Communications of the ACM,* Vol. 13, No. 7, pp. 422–426, 1970.

[10] A. Broder and M. Mitzenmacher. "Network Applications of Bloom Filters: A Survey," *Allerton Conference*, 2002.

[11] D. Damiani, S. De Capitani Di Vimercati, S. Paraboschi and P. Samarati. "A Fine-Grained Access Control System for XML Documents," *ACM Transactions on Information and System Security,* Vol. 5, No. 2, May 2002, pp. 169–202.

[12] P. Devanbu, M. Gertz, A. Kwong, C. Martel and G. Nuckolls. "Flexible Authentication of XML Documents," *CCS'01,* Philadelphia, USA, November 2001.

[13] L. Fan, P. Cao, J. Almeida and A. Broder, "Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol," *IEEE/ACM Transactions on Networking,* Vol. 8, No. 3, pp. 281–293, 2000.

[14] M. Garey and D. Johnson. *Computers and intractability: A guide to the theory of NP-completeness.* Freeman, Oxford, UK, 1979.

[15] M. Mitzenmacher, "Compressed Bloom Filters," *ACM symposium on Principles of distributed computing*, Newport, Rhode Island, US, August 2001.

[16] S. Payette and C. Lagoze. "Policy-Carrying, Policy-Enforcing Digital Objects," *Research and Advanced Technology for Digital Libraries, 4th European Conference, ECDL 2002,* Vol. 1923, pp. 144–157, 2000.

[17] M. Stonebraker and L. Rowe, "The Design of POSTGRES," *ACM SIGMOD Conference on*

Management of Data, Washington D.C., May 1986.

[18] C. Wong and A. Yao. "A Combinatorial Optimization Problem Related to Data Set Allocation," *Revue Francaise D'Automatique, Informatique, Recherche Operationnelle,* Suppl. No. 5 (1976), pp. 83–96.

Appendix A: Proof of Theorem 1

PROOF. This is a proof by induction on a group's size s. For a set S of size j, we refer to the expression $\sum_{i=1}^{j} c_i(1-p_i) - \Big(\sum_{i=1}^{j} c_i\Big)\Big(\prod_{i=1}^{j}(1-p_i)\Big)$ as the "left part" or $\mathcal{L}^j$ and to the expression representing the sum of probabilities of all possible subsets of S weighted by the cost of elements not in the subset as the "right part" or $\mathcal{R}^j$.

Basic Step: For $s = 1$, we have:

$$\mathcal{L}^1 = c_1(1-p_1) - c_1(1-p_1) = 0,\ \mathcal{R}^1 = p_1 \cdot 0 = 0 \Rightarrow \mathcal{L}^1 = \mathcal{R}^1$$

i.e., the equality holds.

Induction Hypothesis: Assume that the equality holds for some $s = k$ and we have:

$$\begin{aligned}\mathcal{L}^k &= \sum_{i=1}^{k} c_i(1-p_i) - \Big(\sum_{i=1}^{k} c_i\Big)\Big(\prod_{i=1}^{k}(1-p_i)\Big)\\ &= \sum_{i=1}^{k} p_i\Big(\sum_{\substack{j=1\\ j\neq i}}^{k} c_j\Big)\Big(\prod_{\substack{j=1\\ j\neq i}}^{k}(1-p_j)\Big) +\\ &+ \sum_{i=1}^{k-1} p_i \sum_{j=i+1}^{k} p_j\Big(\sum_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k} c_l\Big)\Big(\prod_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k}(1-p_l)\Big) + \ldots +\\ &+ \sum_{i=1}^{k} c_i(1-p_i)\prod_{\substack{j=1\\ j\neq i}}^{k} p_j = \mathcal{R}^k\end{aligned}$$

Induction Step: Given that the statement hold for $s = k$, we show that it also holds for $s = k+1$.

When we add element $k+1$ to the set, it can be either in or not in a particular subset of $k+1$ elements. If the element is in a subset, it does not increase the false positives (compared to the k-element case), but when the element is not included in a subset, the false positives of the subset increases by c_{k+1}. In general, for $\mathcal{R}^{k+1}$ we have:

$$\begin{aligned}\mathcal{R}^{k+1} &= p_{k+1}\Bigg(\sum_{i=1}^{k} p_i\Big(\sum_{\substack{j=1\\ j\neq i}}^{k} c_j\Big)\Big(\prod_{\substack{j=1\\ j\neq i}}^{k}(1-p_j)\Big) +\\ &\sum_{i=1}^{k-1} p_i \sum_{j=i+1}^{k} p_j\Big(\sum_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k} c_l\Big)\Big(\prod_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k}(1-p_l)\Big) +\\ &+ \ldots + \sum_{i=1}^{k} c_i(1-p_i)\prod_{\substack{j=1\\ j\neq i}}^{k} p_j\Bigg) +\\ &+\big(1-p_{k+1}\big)\Bigg(\sum_{i=1}^{k} p_i\Big(c_{k+1} + \sum_{\substack{j=1\\ j\neq i}}^{k} c_j\Big)\Big(\prod_{\substack{j=1\\ j\neq i}}^{k}(1-p_j)\Big) +\\ &+ \sum_{i=1}^{k-1} p_i \sum_{j=i+1}^{k} p_j\Big(c_{k+1} + \sum_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k} c_l\Big)\Big(\prod_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k}(1-p_l)\Big) +\\ &+ \ldots + \sum_{i=1}^{k}(c_i + c_{k+1})(1-p_i)\prod_{\substack{j=1\\ j\neq i}}^{k} p_j + c_{k+1}\prod_{i=1}^{k} p_i\Bigg)\end{aligned}$$

$$\begin{aligned}&= p_{k+1}\cdot\mathcal{R}^k +\\ &+\big(1-p_{k+1}\big)\Bigg(\mathcal{R}^k + c_{k+1}\Bigg(\sum_{i=1}^{k} p_i\prod_{\substack{j=1\\ j\neq i}}^{k}(1-p_j) +\\ &+ \sum_{i=1}^{k-1} p_i \sum_{j=i+1}^{k} p_j \prod_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k}(1-p_l) + \ldots +\\ &+ \sum_{i=1}^{k}(1-p_i)\prod_{\substack{j=1\\ j\neq i}}^{k} p_j + \prod_{i=1}^{k} p_i\Bigg)\Bigg)\end{aligned}$$

Notice that the last summation in the equation represents the probability that at least one element out of k happens. In other words, we have:

$$\begin{aligned}&\sum_{i=1}^{k} p_i\prod_{\substack{j=1\\ j\neq i}}^{k}(1-p_j) + \sum_{i=1}^{k-1} p_i \sum_{j=i+1}^{k} p_j \prod_{\substack{l=1\\ l\neq i,\, l\neq j}}^{k}(1-p_l) + \ldots +\\ &+ \sum_{i=1}^{k}(1-p_i)\prod_{\substack{j=1\\ j\neq i}}^{k} p_j + \prod_{i=1}^{k} p_i = 1 - \prod_{i=1}^{k}(1-p_i)\end{aligned}$$

Thus, substituting the summation in the formula for $\mathcal{R}^{k+1}$ we obtain:

$$\begin{aligned}\mathcal{R}^{k+1} &= p_{k+1}\cdot\mathcal{R}^k + \big(1-p_{k+1}\big)\Bigg(\mathcal{R}^k + c_{k+1}\Big(1 - \prod_{i=1}^{k}(1-p_i)\Big)\Bigg)\\ &= \mathcal{R}^k + c_{k+1}(1-p_{k+1}) - c_{k+1}(1-p_{k+1})\Big(1 - \prod_{i=1}^{k}(1-p_i)\Big)\end{aligned}$$

After applying the formula for a k-element set from the induction hypothesis, we get:

$$\begin{aligned}\mathcal{R}^{k+1} &= \sum_{i=1}^{k} c_i(1-p_i) + c_{k+1}(1-p_{k+1}) -\\ &-\Big(\sum_{i=1}^{k} c_i\Big)\Big(\prod_{i=1}^{k}(1-p_i)\Big) -\\ &-c_{k+1}(1-p_{k+1})\Big(1 - \prod_{i=1}^{k}(1-p_i)\Big)\\ &= \sum_{i=1}^{k+1} c_i(1-p_i) - \Big(\sum_{i=1}^{k} c_i\Big)\Big(\prod_{i=1}^{k+1}(1-p_i)\Big) = \mathcal{L}^{k+1}\ \square\end{aligned}$$

Appendix B: Proof of Lemma 1

PROOF. We want to prove that, for unit costs (unweighted documents), if documents i and j ($i < j$) are assigned to a certain group then so are all the documents $i+1, i+2, \ldots, j-1$. For simplicity, we assume that $p_1 > p_2 > \cdots > p_n$ rather than $p_1 \geq p_2 \geq \cdots \geq p_n$ (the proof can be extended to the case of "$\geq$"). The proof is by contradiction: Assume to the contrary that there are three documents $i < v < j$ such that documents i and j are assigned to group x and document v is assigned to group y. Equation (2) implies that we can write the cost of group y as

$$C_y = -\alpha * (1-p_v) + \beta + (1-p_v)$$

where α and β are positive and depend on how many (and which) other documents than v are in group y. Equation (2) also implies that we can write C_x as

$$C_x = -\gamma * (1-p_i) * (1-p_j) + \theta + (1-p_i) + (1-p_j)$$

where γ and θ are positive and depend on how many (and which) other documents than i, j are in group x. Now, note that interchanging i and v from their groups (i.e., moving document i to group x and document v to group y) would result in a change in total cost of

$$\begin{aligned}\Delta_{i,v} &= \alpha * (p_i - p_v) - (p_i - p_v) + \gamma * (1 - p_j)(p_v - p_i) - (p_v - p_i) \\ &= (\alpha - \gamma * (1 - p_j)) * (p_i - p_v)\end{aligned}$$

and interchanging j and v similarly results in a change in total cost of

$$\Delta_{j,v} = (\alpha - \gamma * (1 - p_i)) * (p_j - p_v)$$

Optimality requires that both $\Delta_{i,v}$ and $\Delta_{j,v}$ are positive. This, and the fact that $p_i > p_v > p_j$, imply that

$$(\alpha - \gamma * (1 - p_j)) > 0$$

$$(\alpha - \gamma * (1 - p_i)) < 0$$

But this implies that $p_j > p_i$, a contradiction. □