Ad Insertion in Automatically Composed Documents

Niranjan Damera-Venkata
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
damera@hpl.hp.com

José Bento
Stanford University
Dept. of Electrical Engineering
Stanford, CA 94305
jbento@stanford.edu

ABSTRACT

We consider the problem of automatically inserting advertisements (*ads*) into machine composed documents. We explicitly analyze the fundamental tradeoff between expected revenue due to *ad* insertion and the quality of the corresponding composed documents. We show that the optimal tradeoff a publisher can expect may be expressed as an *efficient-frontier* in the revenue-quality space. We develop algorithms to compose documents that lie on this optimal tradeoff frontier. These algorithms can automatically choose distributions of *ad* sizes and *ad* placement locations to optimize revenue for a given quality or optimize quality for given revenue. Such automation allows a market maker to accept highly personalized content from publishers who have no design or *ad* inventory management capability and distribute formatted documents to end users with aesthetic *ad* placement. The *ad* density/coverage may be controlled by the publisher or the end user on a per document basis by simply sliding along the tradeoff frontier. Business models where *ad* sales precede (*ad-pull*) or follow (*ad-push*) document composition are analyzed from a document engineering perspective.

Categories and Subject Descriptors

I.7.4 [**Computing Methodologies**]: Document and Text Processing:Electronic Publishing

General Terms

Algorithms, Design

Keywords

automated publishing, advertisement insertion, document composition, layout synthesis

DocEng'12, September 4–7, 2012, Paris, France.

1. INTRODUCTION

In traditional publishing every subscriber gets the same copy of a newspaper/magazine, and design costs are amortized over the subscriber base. While professional graphic design works well for the traditional publishing industry where a single high quality document may be distributed to an audience of millions, it is not economically viable (due to its high marginal cost) for the creation of highly personalized documents that change per subscriber and by device form factor. Publishers want to deliver targeted personalized content not only because it is much more engaging and relevant to their subscribers but also because of the potential advertisement (*ad*) revenue boost when content and advertisements are accurately targeted.

Automated document composition attempts to transform personalized content automatically (without per-copy manual graphic design) into documents with high aesthetic value. This has been a topic of much research [8] [6]. However, the problem of automated advertisement insertion into machine composed documents has not received attention from the document engineering community. In this paper we analyze the problem of automated advertisement insertion into machine composed documents from a document engineering perspective.

A key insight is that if content is to be formatted for a particular page count then the specific sizes of *ads* and their placement affect the overall aesthetics of a document composition. For example, consider the case where we are trying to place a large *ad* in a single page document with a lot of content. An automated document composition algorithm may deal with this by shrinking images (*ads* cannot be shrunk of course), reducing whitespace, reducing font sizes or line spacing. These changes may have undesirable effects on the quality of the resulting formatted document. On the other hand an *ad* insertion may also have a beneficial impact on document composition quality. Consider the case where the inserted *ad* fills an undesirable whitespace or void in the composition. The preceding examples illustrate that in general there is a fundamental tradeoff between revenue due to *ad* insertion and the quality of machine composed documents. The notion of quality could be expanded to include relevance of the *ads* to the target audience. For example, in internet advertising where pages are scrollable and *ads* are not integrated into the content (but instead appear in their own to one side in reserved area of the page) the tradeoff is between revenue and the relevance of an *ad* measured by its click-through-rate.

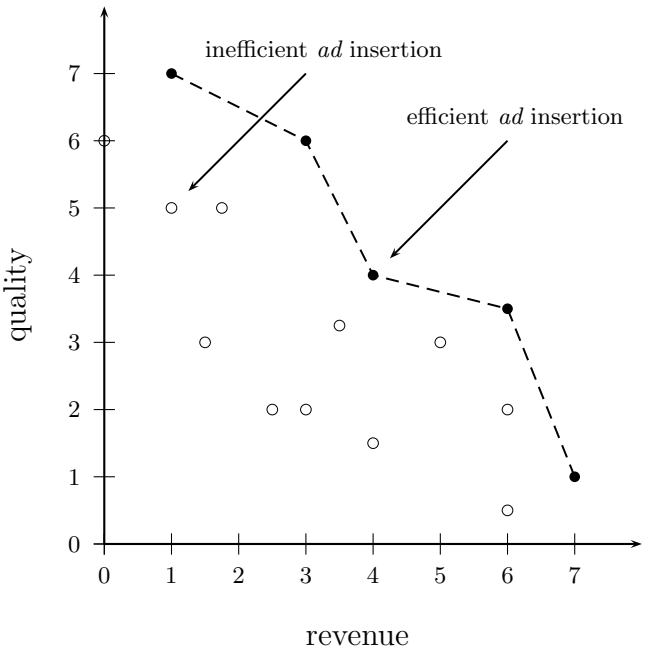


Figure 1: The *efficient-frontier*. Each dot represents a document composition of the given content with advertisements. The number of *ads*, *ad*-size distribution and the specific placement of *ads* is allowed to vary across compositions. Efficient documents are those for which we cannot find another document with at least the same revenue and better quality, or at least the same quality and better revenue. The discrete set of efficient points (represented here with the filled discs, connected with dashed lines) is called the *efficient-frontier*. This frontier represents the *optimal* tradeoff between revenue and quality when advertisements are inserted.

Fig. 1 illustrates the tradeoff between revenue and quality for a document with given content composed automatically with different *ad* size distributions and placements. Each point represents a document with a specific distribution of *ad* sizes with the *ads* placed at specific locations within the document. Note that many points (labeled *inefficient*) are not sensible document compositions since we can find other documents that have at least the same revenue and better quality or that have at least the same quality and better revenue. Note also that there are a set of points (labeled *efficient*) that represent a valid tradeoff between revenue and quality. There are no other documents that have at least the same revenue and better quality or have at least the same quality and better revenue. The set of *efficient* points is called the *efficient-frontier* and represents an optimal way to tradeoff revenue and document quality for *ad* insertion. The dashed connecting line in Fig. 1 simply serves as a visual aid to help judge the relative positions of points. The *efficient-frontier* is a discrete set.

In this paper we analyze algorithms that attempt to efficiently compute documents that are points on the *efficient-frontier*. The business model for *ad*-sales impacts problem formulation and hence the specific algorithms we develop. We consider two business models. In the *ad-pull* business model, *ad* sales precede document composition. Advertiser bidding creates a pool of *ads* (with associated bid-prices) for each publication. The goal is to select or *pull* specific *ads* from the pool and compose documents with *ad* placement that represents documents on the *efficient-frontier*. In the *ad-push* business model the *ad* sales come after the document composition. Here, the algorithm uses expected revenues of *ads* of particular sizes and automatically chooses *ad* size distributions and placement to generate documents with empty *ad* slots on an expected *efficient-frontier*. The *ad* slots (a.k.a. *ad* inventory) are then *pushed* to the market for sale. Note that with this model it is possible that an *ad* slot may be unsold and so may need to be filled with filler content. However, over time, the expected market value of *ads* of particular sizes will be more accurately estimated by the publisher. Adjusting his templates and/or *ad* reserve prices accordingly, over time, the publisher is less likely to have unsold *ads* slots in his documents.

The paper is organized as follows. Section 2 reviews related work. Section 3 reviews a structured probabilistic model of document quality used in developing the algorithms in this paper. Section 4 develops and analyzes algorithms for determining the discrete *efficient-frontier* for both the *ad-pull* and *ad-push* business models. Section 5 analyzes inter-subscriber tradeoffs that naturally gives rise to the concept of a continuous *stochastic efficient-frontier* that is the convex-envelope of the discrete *efficient-frontier*. Algorithms that allow us to produce documents on the *stochastic* frontier are presented. Finally Section 6 concludes the paper by summarizing the contributions and indicating future directions. We also provide an Appendix of mathematical complements.

2. RELATED WORK

The concept (and term) *efficient-frontier* (a.k.a. the *Pareto set*) was first introduced in finance by Harry Markowitz [9] in the context of selecting portfolios of securities that trade off risk (portfolio variance) versus expected return (portfolio mean). An investor seeks portfolios on the *efficient-frontier* risk-return space. Efficient portfolios are those where additional expected return cannot be gained without increasing the risk of the portfolio. This seminal work lead the 1990 Nobel prize in economics and forms the basis for modern portfolio theory. The *efficient-frontier* has also found several applications in pricing and revenue management [11] in the tradeoff analysis of competing goals like revenue and profit, short-term price discounting vs. lifetime customer value, optimization of airline booking classes etc. In the internet advertising space companies like Efficient Frontier[1] trade cost of keywords vs. expected return on investment (ROI) to manage their clients keyword marketing campaigns.

The computational task of determining the *efficient-frontier* efficiently from a discrete set of candidate points has also been studied in computer science [7, 4] where the problem is called the maximal vector computation or Skyline algorithm. However, in many cases (such as the problems discussed in this paper) the number of possible candidate points could be very large and each point itself corresponds to significant computational effort. So the naive method of generating all possible candidate points (documents in our case) and using the Skyline algorithm to compute the frontier is not viable.

[1]http://www.efrontier.com

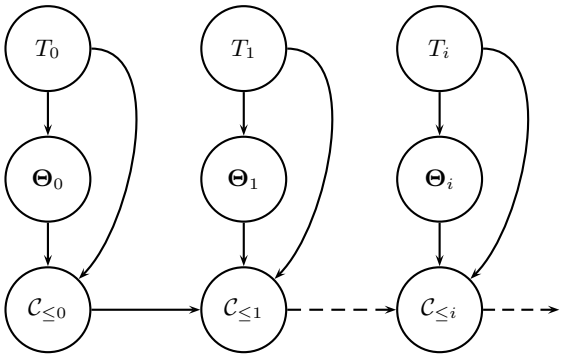


Figure 2: PDM as a graphical model.

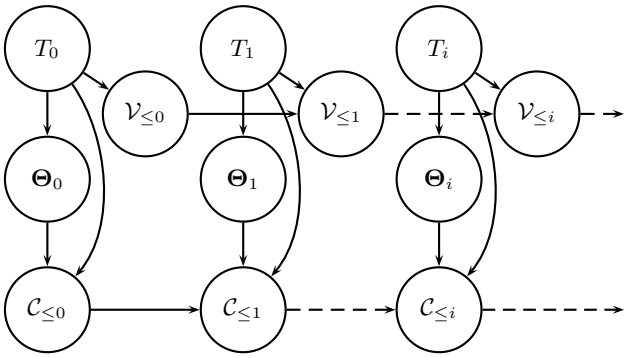


Figure 3: PDM with ordered Ad Insertion.

3. MODELING DOCUMENT QUALITY

In this section we review a structured model for document aesthetics based on a probabilistic modeling of designer choice in document design. This model will be used in developing efficient *ad* insertion algorithms. We represent the given set of all the units of content to be composed (ex: images, units of text, sidebars etc.) by a finite set $\mathcal{C}$. Text units could be words, sentences, lines of text or whole paragraphs. We denote by $\mathcal{C}'$ a set comprising all sets of discrete content allocation possibilities over one or more pages starting with and including the first page. For example, if there are 3 lines of text and 1 floating figure, $\mathcal{C} = \{l_1, l_2, l_3, f_1\}$ while $\mathcal{C}' = \{\{l_1\},\{l_1, l_2\},\{l_1, l_2, l_3\},\{f_1\},\{l_1, f_1\},\{l_1, l_2, f_1\},\{l_1, l_2, l_3, f_1\}\} \cup\{\emptyset\}$. Note that the specific order of elements within an allocation set is not important since $\{l_1, l_2, f_1\}$ and $\{l_1, f_1, l_2\}$ refer to an allocation of the same content. However allocation $\{l_1, l_3, f_1\} \notin \mathcal{C}'$ since lines 1 and 3 cannot be in the same allocation without including line 2. $\mathcal{C}'$ includes the empty set to allow the possibility of a null allocation.

In order to compose documents, one must make aesthetic decisions on how to paginate content, how to arrange page elements (text, images, graphics, sidebars etc.) on each page, how much to crop/scale images, how to manage whitespace etc. These decision variables are *not* mutually exclusive, making the aesthetic graphic design of documents a hard problem often requiring an expert design professional. The probabilistic document model (PDM) [2] explicitly models the dependency between key design choices including pagination, choice of relative arrangements for page elements, and page edits (including image re-targeting and whitespace adjustment). The coupling between these design variables is explicitly modeled as a Bayesian network shown in Fig. 2. A probability distribution can be associated with the network by multiplying the conditional probability distributions of each node conditioned only on its parents [10].

$$\mathbb{P}(D, I) \quad = \quad \prod_{i=0}^{I-1} \mathbb{P}(\mathcal{C}_{\leq i}|\mathcal{C}_{\leq i-1}, \mathbf{\Theta}_i, T_i)\mathbb{P}(\mathbf{\Theta}_i|T_i)\mathbb{P}(T_i) \quad (1)$$

Random variable T_i represents choice of a relative arrangement of page elements for the i^{th} page from a library of page templates representing different possible relative arrangements of content. Random vector $\mathbf{\Theta}_i$ encodes template parameters representing possible edits to the chosen template. Possible choices for variable template parameters Θ include figure dimensions, whitespace between page elements, margins etc. Random set $\mathcal{C}_{\leq i}$ represents choice of a content allocation to the first i pages from the set of possible allocations $\mathcal{C}'$. Note that content allocation to the i^{th} page (i.e. pagination) is computed as $\mathcal{C}_i = \mathcal{C}_{\leq i} - \mathcal{C}_{\leq i-1}$. In this paper we treat the page count I as a given constant. More generally we could treat page count as a random variable also. See [2] for details.

PDM is in fact a *micro* model for document quality that associates a probability (or quality score) with each *conditional* design choice made on each page. The overall probability of a document is the product of these *micro* probability scores. A document $\mathcal{D}$ of I pages is defined by a triplet $\mathcal{D} = \{\{\mathcal{C}_{\leq i}\}_{i=0}^{I-1}, \{\mathbf{\Theta}_i\}_{i=0}^{I-1}, \{T_i\}_{i=0}^{I-1}\}$ of random variables representing the various design choices made in the document creation process. The overall quality $\mathbb{P}(D, I)$ of a document $\mathcal{D}$ of I pages is the product of the conditional probabilities of all design choices made. PDM contrasts with *macro* models for document quality [1, 5] that attempt to quantify abstract aesthetic notions such as harmony, balance, regularity etc.

In equation (1) probability distribution $\mathbb{P}(T_i)$ governs relative preference of template T_i from a set Ω of possible templates. The conditional multi-variate probability distribution $\mathbb{P}(\mathbf{\Theta}_i|T_i)$ may be regarded as a *prior* probability distribution that determines the *prior* preference (before seeing content) for template parameters. Finally the probability distribution $\mathbb{P}(\mathcal{C}_{\leq i}|\mathcal{C}_{\leq i-1}, \mathbf{\Theta}_i, T_i)$ reflects how well the content allocated to the current page fits template T_i when template parameters are set to $\mathbf{\Theta}_i$.

Once the probability distributions in equation (1) are defined we can simply calculate $\mathbb{P}(\mathcal{D}, I)$ as a measure of document quality. We may also use $L(\mathcal{D}, I) = \log \mathbb{P}(\mathcal{D}, I)$ as the quality measure in practice, since taking the logarithm converts products to summations that are often easier to work with. A logarithmic transformation (in fact any monotonic function) of the quality axis does not change the *efficient-frontier* set.

In this paper, we choose $\mathbb{P}(T_i)$ to be the uniform over all templates in the template library, indicating no *a priori* preference for a particular template. Human design knowledge regarding parameter preferences may be used in specifying $\mathbb{P}(\mathbf{\Theta}_i|T_i)$ (ex: by designer input of mean, variance, min and max of the parameter values) [2]. $\mathbb{P}(\mathcal{C}_{\leq i}|\mathcal{C}_{\leq i-1}, \mathbf{\Theta}_i, T_i)$ is simply a goodness of fit function and is calculated based on

a specific parameterization of a template as a graph introduced in [2]. Fit is assessed by how well content fills the page along all paths in the graph from top to bottom and left to right. Please see [2] for more details on these probabilistic template parameterizations.

4. EFFICIENT-FRONTIER COMPUTATION

This section analyzes algorithms for the automated composition of documents with optimal *ad* insertion. Section 4.1 analyzes the case when *ad* sales precede document composition (*ad-pull*) while Section 4.2 analyzes the case when document composition precedes *ad* sales (*ad-push*).

4.1 Ad-Pull

In the *ad-pull* scenario advertisers bid on *ad* placements in documents based on several factors that may include demographics of the target audience, *ad*-size, *ad*-position etc. Business logic then creates an *ad*-pool for each document that selects candidate *ads* that are good matches for the documents target audience. Since more than one advertiser may bid on a particular size of *ad*, each bid is treated as a different *ad*-placement opportunity. The *ad*-pool thus has a specific distribution of *ad*-sizes and associated bid prices. The goal is to derive the *efficient-frontier* tradeoff over all subsets that meet certain constraints (ex: At least one *ad* per page must be inserted). The revenue of a subset is defined as the sum of all bid prices of *ads* in that subset.

For an *ad*-pool with M *ads*, there are $\sum_{k=0}^{M}\binom{M}{k}$ ways of choosing a subset of *ads* to be composed. For each of the subsets of size k there are $k!$ possible orderings of the *ads*. Thus, in general there are at least $\sum_{k=0}^{M}\binom{M}{M-k}$ possible *ad* insertions to be considered. Of course, a particular *ad* order may still correspond to several different possible *ad*-insertion points and hence *ad* positions within a document. So composing all possible documents and then using the skyline algorithm [7, 4] (see Appendix for pseudocode) to compute the *efficient-frontier* is not feasible.

While there is no optimal way (one may always resort to sub-optimal heuristics in practice) to simplify the subset selection and *ad* ordering problems, we may develop efficient algorithms to compose a document that is optimal given a particular ordering of *ads* to be inserted. To do this we extend the *ad*-free PDM model for document quality to include *ads*. The Bayesian network representation of this extended model is given in Figure 3. The resulting document probability becomes:

$$\mathbb{P}(D,I) = \prod_{i=0}^{I-1}\mathbb{P}(\mathcal{C}_{\leq i}|\mathcal{C}_{\leq i-1},\mathbf{\Theta}_i,T_i)\mathbb{P}(\mathbf{\Theta}_i|T_i)\cdots \mathbb{P}(\mathcal{V}_{\leq i}|\mathcal{V}_{\leq i-1},T_i)\mathbb{P}(T_i) \quad (2)$$

where the newly introduced random set $\mathcal{V}_{\leq i}$ refers to an allocation of *ads* to the first i pages of the document from the set $\mathcal{V}'$ of all possible *ad* allocations. The probability $\mathbb{P}(\mathcal{V}_{\leq i}|\mathcal{V}_{\leq i-1},T_i)$ is defined as

$$\mathbb{P}(\mathcal{V}_{\leq i}|\mathcal{V}_{\leq i-1},T_i) = \begin{cases} 1 & \mathcal{V}_{\leq i}-\mathcal{V}_{\leq i-1} \text{ matches } T_i \\ 0 & else \end{cases} \quad (3)$$

This filters cases where template T_i does not have *ad* slots that match the allocation $\mathcal{V}_{\leq i}-\mathcal{V}_{\leq i-1}$ of *ads* to page i.

The optimal document $\mathcal{D}^*$ is defined by the quadruple sequence $\mathcal{D}^* = \{\{\mathcal{C}^*_{\leq i}\}_{i=0}^{I-1},\{\{\mathcal{V}^*_{\leq i}\}_{i=0}^{I-1},\{\mathbf{\Theta}^*_i\}_{i=0}^{I-1},\{T^*_i\}_{i=0}^{I-1}\}$ that maximizes equation (2). The maximization (optimal document *inference*) algorithm is very similar to the dynamic programming algorithm derived in some detail in[2] for the maximization of equation (1) and is succinctly summarized in Algorithms 1 and 2 below.

Algorithm 1 Pull-based *ad* insertion: Forward pass

1: $\Psi(\mathcal{A}_c,\mathcal{B}_c,T) = \max_{\mathbf{\Theta}}\mathbb{P}(\mathcal{A}_c,|\mathcal{B}_c,\mathbf{\Theta},T)\mathbb{P}(\mathbf{\Theta}|T)$
2: $\Phi(\mathcal{A}_c,\mathcal{B}_c,\mathcal{A}_a,\mathcal{B}_a) = \max_{T\in\Omega}\Psi(\mathcal{A}_c,\mathcal{B}_c,T)\mathbb{P}(\mathcal{A}_a|\mathcal{B}_a,T)\mathbb{P}(T)$
3: $\tau_0(\mathcal{A}_c,\mathcal{A}_a) \leftarrow \Phi(\mathcal{A}_c,\emptyset,\mathcal{A}_a,\emptyset)$
4: $\tau_i(\mathcal{A}_c,\mathcal{A}_a) = \max_{\mathcal{B}_c,\mathcal{B}_a}\Phi(\mathcal{A}_c,\mathcal{B}_c,\mathcal{A}_a,\mathcal{B}_a)\tau_{i-1}(\mathcal{B}_c,\mathcal{B}_a), i\geq 1$

Algorithm 2 Pull-based *ad* insertion: Backward pass

$i \leftarrow I-1, \mathcal{A}_c \leftarrow \mathcal{C}, \mathcal{A}_a \leftarrow \mathcal{V}$
while $i \geq 0$ **do**
  $\mathcal{C}^*_{\leq i} \leftarrow \mathcal{A}_c, \mathcal{V}^*_{\leq i} \leftarrow \mathcal{A}_a$
  $\mathcal{B}^*_c, \mathcal{B}^*_a = \arg\max_{\mathcal{B}_c,\mathcal{B}_a}\Phi(\mathcal{A}_c,\mathcal{B}_c,\mathcal{A}_a,\mathcal{B}_a)\tau_{i-1}(\mathcal{B}_c,\mathcal{B}_a)$
  $T^*_i = \arg\max_{T\in\Omega}\Psi(\mathcal{C}^*_{\leq i},\mathcal{B}^*_c,T)\mathbb{P}(\mathcal{V}^*_{\leq i}|\mathcal{B}^*_a,T)\mathbb{P}(T)$
  $\mathbf{\Theta}^*_i = \arg\max_{\mathbf{\Theta}}\mathbb{P}(\mathcal{C}^*_{\leq i},|\mathcal{B}^*_c,\mathbf{\Theta},T^*_i)\mathbb{P}(\mathbf{\Theta}|T^*_i)$
  $\mathcal{A}_c \leftarrow \mathcal{B}^*_c, \mathcal{A}_a \leftarrow \mathcal{B}^*_a$
  $i \leftarrow i-1$
end while

The algorithm consists of two passes. In the *forward pass* we successively eliminate each variable by first grouping terms in equation (2) involving the variable and then finding the maxima of these terms with respect to that variable. This gives us functions of the remaining variables. These functions (dynamic programming tables) are then propagated to the next maximization step for further grouping and variable elimination. $\mathcal{A}_c$ refers to an allocation of content to all pages upto the current page. $\mathcal{A}_a$ refers to an allocation of *ads* to all pages upto the current page. $\mathcal{B}_c$ refers to an allocation of content to all previous pages. $\mathcal{B}_a$ refers to an allocation of *ads* to all previous pages. Table entries are computed for all valid content allocation sets $\mathcal{A}_c,\mathcal{B}_c \in \mathcal{C}'$ with $\mathcal{A}_c \supseteq \mathcal{B}_c$ and advertisement allocations $\mathcal{A}_a,\mathcal{B}_a \in \mathcal{V}'$ with $\mathcal{A}_a \supseteq \mathcal{B}_a$[2]. The continuous variable maximization in Step 1 of algorithm 1 may be performed efficiently as a quadratic programming problem with bound constraints for appropriate probability parameterization [2].

In the *backward pass* we traverse the tables backward starting from the term $\tau_{I-1}(\mathcal{C},\mathcal{V})$ that equals the global maximum of equation (2) over all the content and *ads*. The backward pass successively infers the allocations of content and *ads* to the previous pages along with templates and template parameters that are on the path to achieving the global maximum.

4.1.1 Example

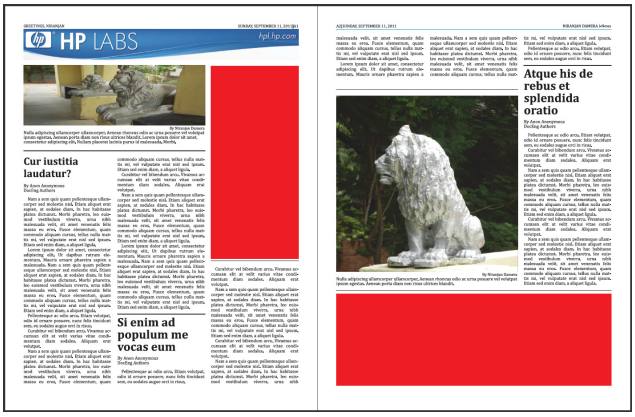
Table 1 illustrates an example *ad*-pool of four *ads* to be inserted into a given sample two-page Lorem Ipsum content with two figures.

[2]This embedding of previous page allocations inside the current allocation ensures that the *ad* order is never violated

(a) $(r, \log q) = (7.0, -0.20)$

(b) $(r, \log q) = (11.0, -0.27)$

(c) $(r, \log q) = (5.5, -0.33)$



(d) $(r, \log q) = (11.0, -1.47)$

Figure 4: *ad-pull* example results obtained using algorithms 1 and 2 on sample content with *ad*-pool given in Table 1. (a) and (b) are *efficient-frontier* document compositions representing a sensible tradeoff between revenue and quality. (c) and (d) show examples of inefficient document compositions since they are dominated by a frontier-composition which has at least the same revenue and better quality or at least the same quality and better revenue. The red areas indicate *ad* placement slots.

Sample *ad*-pool for *ad-pull* Insertion				
ads	V_1 (2,1/6)	V_2 (1, 1/6)	V_3 (1, 1/6)	V_4 (3,1/4)
Bids	2	3.5	5	6

Table 1: Example *ad*-pool. The first index in *ad* category $(\cdot, \cdot)$ refers to the column span while the second index refers to approximate fractional page area covered.

The template library used for document composition consisted of 16 templates including *ads* of various sizes placed at different positions. All templates had exactly one *ad*-slot. Each *ad* slot is characterized by columns spanned and *ad*-area. For example $(1, 1/6)$ refers to an *ad* that spans one column and has an area of roughly a sixth of the page. Note that the *ad*-pool has two *ads* of the same size with different bid prices (*ads* V_2 and V_3). We further require that all document compositions have exactly two *ads* and take up two pages. There are 12 possible *ad* orderings of 2 *ads* selected from the pool of 4 *ads*.

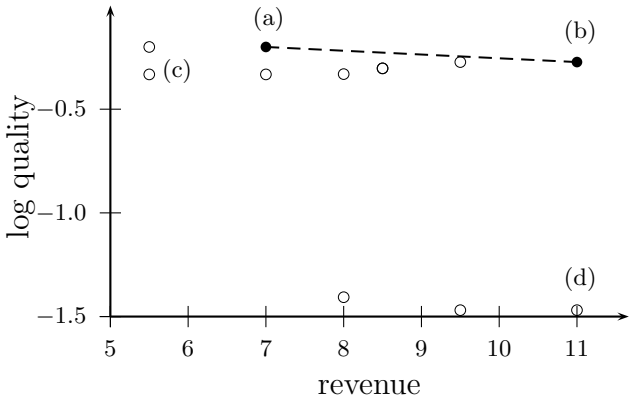


Figure 5: *efficient-frontier* for *ad-pull* example. The labeled points correspond to document compositions shown in Figure 4.

Figures 4(a)-(b) show the frontier compositions while Figures 4(c)-(d) show some examples of inefficient compositions. Figure 5 shows the revenue-quality points and the corresponding computed *efficient-frontier*. Note that as we move

along the frontier from left to right, revenue increases while quality drops. Figure 4(c) is inefficient because it has lower revenue and quality (notice column mis-alignment on page 2) than Figure 4(a). Figure 4(d) is inefficient because although it has the same revenue as Figure 4(b) it has lower quality (both images have more distortion).

Notice also that the points at $(5.5, -.20)$ and $(7.0, -.20)$ have the same quality but the one with higher revenue is on the frontier. In fact these are exactly the same composition with the same *ad* slot sizes. This is because there are 2 bids for the same size of *ad* in the *ad* pool (*ads* V_2 and V_3). The *ad* with the higher bid (V_3) is used in the frontier composition. There are two compositions at point $(8.5, -0.3)$. Both use *ads* V_2 and V_3 but in different order. This results in two compositions with the exact same revenue and quality.

4.1.2 Complexity

We may analyze asymptotic complexity of this algorithm in a similar manner to the analysis presented in [2]. For a given set of *ads* with a specific order the complexity of document composition with *ad*-insertion is thus $O(|\mathcal{C}||\mathcal{V}||\Omega|)$ assuming linear content ordering, bounded page capacity and bounded number of *ads* on a page. This is still significantly better than the naive approach of considering all possible document compositions (document compositions for all $\{\{\mathcal{C}_{\leq i}\}_{i=0}^{I-1}, \{\{\mathcal{V}_{\leq i}\}_{i=0}^{I-1}, \{\mathbf{\Theta}_i\}_{i=0}^{I-1}, \{T_i\}_{i=0}^{I-1}\})$ and choosing the best scoring document. The efficiency stems from the fact that possibilities are eliminated in the maximizations at a given step do not propagate to the successive steps in algorithm 1. This feature is a direct consequence of the structured probabilistic model in which direct dependencies between variables are limited (so conditional probability terms can be grouped without having to maximize over all variables at once). However, if we consider the overall problem of *efficient-frontier* computation, the complexity grows combinatorially in terms of the size of the *ad*-pool. Hence this is a viable method only for small *ad*-pools.

4.2 Ad-Push

In the *ad-push* scenario document composition takes place before *ad* sales and the inserted *ad*-slots in a document are *pushed* to the market for sale. Unsold *ad* inventory is often replaced by filler content. Since the *ads* are not yet bid on at the time of composition, the publisher must estimate the price of each *ad* size. Over time, the expected market value of *ads* of particular sizes will be more accurately estimated by the publisher. Adjusting his templates and/or *ad* reserve prices accordingly, over time, the publisher is less likely to have unsold *ads* slots in his documents. The publisher has an *ad*-matrix of *ad*-sizes and corresponding expected prices. Since there are no specific *ads* to be inserted the algorithm is free to determine the size, distribution and placement of advertising in the document.

The *ad*-free model of document quality introduced in section 3 of document quality can be used in this case without any explicit treatment of *ad* allocations to pages, with the additional understanding that the templates in the library already have designed *ad*-slots. The revenue of a template is simply the sum of the *ad* prices of all *ad*-slots in the template. Thus, when content is allocated to a template, each template may be associated with a revenue and a quality. The algorithm although similar in spirit to *ad*-free PDM inference algorithm developed in [2] operates directly on efficient-frontiers in the revenue-quality space. Algorithms 3 and 4 summarize the method for efficiently computing the *efficient-frontier* document compositions.

Algorithm 3 Push-based *ad* insertion: Forward pass

1: $\Psi(\mathcal{A}, \mathcal{B}, T) = \max_{\mathbf{\Theta}} \mathbb{P}(\mathcal{A}, |\mathcal{B}, \mathbf{\Theta}, T)\mathbb{P}(\mathbf{\Theta}|T)$

2: $\mathcal{F}(\mathcal{A}, \mathcal{B}) = \underset{T\in\Omega}{\text{eff}} \left\{ \left(\underbrace{r(T)}_{r(T)}, \underbrace{\log(\Psi(\mathcal{A}, \mathcal{B}, T)\mathbb{P}(T))}_{l(\mathcal{A},\mathcal{B},T)=\log q(\mathcal{A},\mathcal{B},T)} \right) \right\}$

3: $\mathcal{T}_0(\mathcal{A}) \leftarrow \mathcal{F}(\mathcal{A}, \emptyset)$

4: $\mathcal{T}_i(\mathcal{A}) = \underset{\mathcal{B}}{\text{eff}} \left[\text{eff} \left\{\mathcal{F}(\mathcal{A}, \mathcal{B}) \oplus \mathcal{T}_{i-1}(\mathcal{B})\right\}\right], i \geq 1$

Algorithm 4 Push-based *ad* insertion: Backward pass

1: $i \leftarrow I-1, \mathcal{A} \leftarrow \mathcal{C}$

2: select $(\mathrm{r}^*, \mathrm{l}^*) \in \mathcal{T}_{\mathrm{I}-1}(\mathcal{C})$

3: **while** $i \geq 0$ **do**

4: $\quad \mathcal{C}^*_{\leq i} \leftarrow \mathcal{A}$

5: $\quad T_i^*, \mathcal{B}^* = T, \mathcal{B} : (r^* - r(T), l^* - l(\mathcal{A}, \mathcal{B}, T)) \in \mathcal{T}_{i-1}(\mathcal{B})$

6: $\quad \mathbf{\Theta}_i^* = \arg\max_{\mathbf{\Theta}} \mathbb{P}(\mathcal{C}^*_{\leq i}, |\mathcal{B}^*, \mathbf{\Theta}, T_i^*)\mathbb{P}(\mathbf{\Theta}|T_i^*)$

7: $\quad r^* \leftarrow r^* - r(T_i^*), l^* \leftarrow l^* - l(\mathcal{C}^*_{\leq i}, \mathcal{B}^*, T_i^*))$

8: $\quad \mathcal{A} \leftarrow \mathcal{B}^*$

9: $\quad i \leftarrow i - 1$

10: **end while**

The *forward pass* algorithm 3 computes dynamic programming tables as before, but table entries are now sets instead of scalar values. $\mathcal{A}$ refers to an allocation of content to all pages upto the current page while $\mathcal{B}$ refers to an allocation of content to all previous pages. Table entries are computed for all valid content allocation sets $\mathcal{A}, \mathcal{B} \in \mathcal{C}'$ with $\mathcal{A} \supseteq \mathcal{B}$. Note that we have dropped the subscript in $\mathcal{A}_c$ that explicitly indicates a content allocation vs. an *ad* allocation, since we do not need to deal with *ad* allocations.

Step 1 is unchanged from PDM inference and essentially computes and stores a table of scores of how well content in the set $\mathcal{A} - \mathcal{B}$ is suited for template T. This step may be performed efficiently as a quadratic programming problem with bound constraints for appropriate template probability parameterization [2]. Table entries for the case when $\mathcal{A} - \mathcal{B}$ does not match template T may be set to zero without requiring any further computation.

Step 2 loops over templates in the library (only templates matching content in the set $\mathcal{A} - \mathcal{B}$ need be considered) and computes a revenue $r(T)$ and a log quality $l(\mathcal{A}, \mathcal{B}, T) = \log q(\mathcal{A}, \mathcal{B}, T)$ score for each template. Then an *efficient-frontier* $\mathcal{F}(\mathcal{A}, \mathcal{B})$ for the allocation $\mathcal{A}, \mathcal{B}$ with respect to the template library is computed using the skyline algorithm on all candidate revenue-quality points. This *efficient-frontier* is propagated for consideration in future steps.

Step 4 sets up a recursion to successively compute the efficient-frontiers $\mathcal{T}_i(\mathcal{A})$ for the allocation of content $\mathcal{A}$ to the first i pages of the document. First the intermediate *efficient-frontier* $\mathcal{F}(\mathcal{A}, \mathcal{B})$ from the previous step is *summed* with the *efficient-frontier* $\mathcal{T}_{i-1}(\mathcal{B})$ for an allocation $\mathcal{B}$ to the previous pages. By sum we mean the Minkowski sum [3] of two sets in Euclidean space represented by the operator $\oplus$ which is essentially the set formed by adding every element of the first set to every element of the second set. The pseudocode for this operation is given in algorithm 8 in the

(a) $(r, \log q) = (10.5, -0.20)$ (b) $(r, \log q) = (13.5, -0.57)$

(c) $(r, \log q) = (14.0, -1.68)$ (d) $(r, \log q) = (15.0, -10.82)$

Figure 6: *ad-push* frontier compositions obtained using algorithms 3 and 4 with *ad*-matrix given in Table 2. The red areas indicate *ad* placement slots. Note how quality degrades along the *efficient-frontier* as revenue is increased from (a)-(d). Fig. 7 shows the corresponding frontier.

Appendix. An *efficient-frontier* is computed over the points in the Minkowski sum and over all possible previous allocations $\mathcal{B}$ to obtain the *efficient-frontier* $\mathcal{T}_i(\mathcal{A})$. Step 3 seeds the recursion with initial values.

The recursion terminates when we have the final *efficient-frontier* $\mathcal{T}_{I-1}(\mathcal{C})$ which may be regarded as the *efficient-frontier* for the whole document when all content $\mathcal{C}$ has been allocated. A publisher must choose an operating point on this frontier set to actually produce a document with a certain revenue and log quality. Once a point (r^*, l^*) is chosen the *backward pass* algorithm 4 traverses the dynamic programming tables built during the forward pass to compute the optimal document that achieves the desired tradeoff. Note that instead of explicitly choosing a point the publisher may simply indicate a revenue or quality target. In the case of a quality target the maximal revenue frontier point whose quality is greater than or equal to the quality target is automatically chosen. In the case of a revenue target the maximal quality frontier point whose revenue is greater than or equal to the revenue target is automatically chosen.

Step 5 searches over previous page frontiers for a partial document with revenue $r^* - r(T)$ and log quality $l^* - l(\mathcal{A}, \mathcal{B}, T)$). The template T_i^* and previous allocation $\mathcal{B}^*$ that resulted in the desired partial document are identified as the optimal template for the current page and the best previous allocation of content to previous pages. From these optimal values the best template parameters $\mathbf{\Theta}_i^*$ to compose the current page are computed. This process recurses from the last page to the first to sequentially compose all pages of the optimal document.

4.2.1 Example

Expected Revenue Matrix for *ad-push* Insertion			
	1 column	2 column	3 column
Twelfth of a page	1	-	-
Sixth of a page	3.5	3.5	3
Quarter of a page	4.5	5	4

Table 2: *ad*-matrix for *ad-push* example

Table 2 shows an example *ad*-matrix for *ad-push* insertion. It lists *ads* of various sizes and their expected revenue potential. We also assume (to make the example more interesting) that first page *ads* are expected to receive twice the expected revenues shown. The goal is to compute the *efficient-frontier* document compositions when *ads* are inserted into the same two page content as in the *ad-pull* example in Section 4.1.1.

Figure 6 shows the *efficient-frontier* document compositions computed by our algorithm. Figure 7 shows the corre-

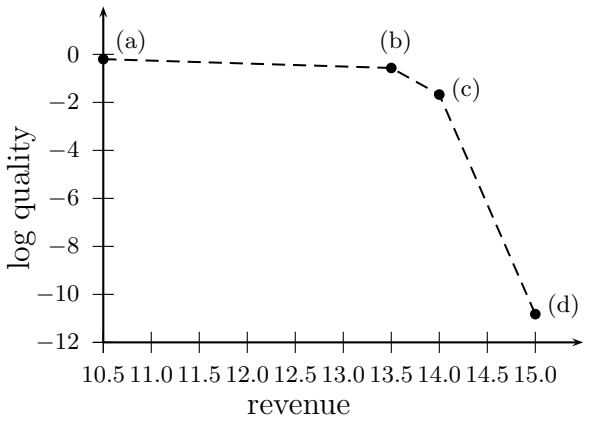


Figure 7: *efficient-frontier* for *ad-push* example. The labeled points correspond to document compositions shown in Figure 6.

sponding *efficient-frontier*. A publisher may choose to publish document 6(b) or 6(c) since quality only drops moderately for substantial increase in revenue. Note however that just because a document composition is on the *efficient-frontier* it does not imply that it is a good quality composition. For example the composition 6(d) corresponds to a significant drop in quality that is clearly visible and also reflected in the graph of Figure 7.

In general a publisher may set a quality threshold and seek to publish at a point on the frontier that maximizes revenue, subject to this threshold. Alternatively, a publisher (or end user) can have a slider that increases revenue (or *ad* coverage) by sliding along the *efficient-frontier*. This may also allow new business models where each user may substitute increased *ad* coverage (subject to publisher quality thresholds) for lower subscription cost according to his/her preference.

4.2.2 Complexity

Asymptotic complexity of *ad-push* algorithm for *efficient-frontier* computation is $O(|\mathcal{C}||\Omega|)$ assuming linear content ordering, bounded page capacity [2]. This is the same as asymptotic complexity of *ad*-free document composition using PDM inference [2]. It is important to note that unlike the *ad-pull* case, asymptotic complexity does not grow with the number of *ads* to be inserted. Although *efficient-frontier* computations involve more work than simple maxima computations, they do not grow with the number of *ads* inserted. The algorithms efficiency stems from the fact that possibilities that are eliminated in the *efficient-frontier* computations at a given step do not propagate to successive frontiers in algorithm 3.

5. INTER-SUBSCRIBER TRADEOFFS

The *efficient-frontier* is a discrete set representing tradeoffs for a single document for a single subscriber. This means that we must choose one of the frontier points to compose a single document. However, a publisher armed with automated composition technology may produce different document compositions for different subscribers using the exact same content (ex: by demographic group). The publisher may choose for example, to produce a first fraction of the documents for part of the subscriber base at one frontier point and a second fraction of documents for another part of the subscriber base at a different frontier point. We call this

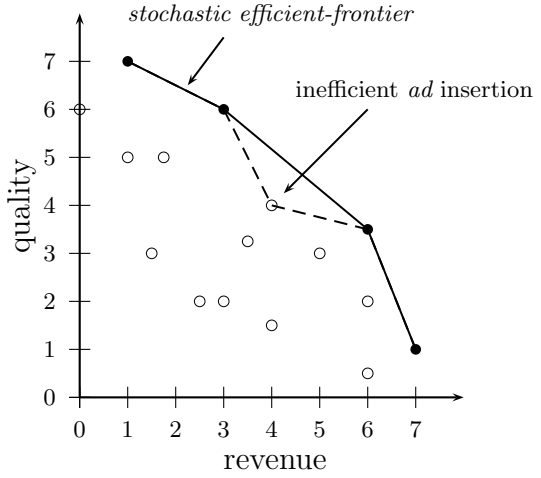


Figure 8: The *stochastic efficient-frontier*. This frontier allows the notion of a continuous tradeoff between revenue and quality by stochastically varying the documents over subscribers and considering average revenue and quality. Since any point between two frontier points is also on the frontier, this frontier is in fact the convex-envelope of the discrete *efficient-frontier* developed earlier.

stochastic document composition. If the notion of revenue and quality of a document could be extended to the notion of *average* revenue and *average* quality of a document collection, then all points on the dashed lines between frontier points now become possible *continuous* tradeoff possibilities.

We see from Figure 1 that the set of points connecting *adjacent* discrete *efficient-frontier* points may be non-convex. However, since we now consider the set of all points connecting any two discrete *efficient-frontier* points, we see that this set is convex. This is illustrated by the bold line in Figure 8 which shows the effect of considering *stochastic* compositions between any two discrete frontier compositions of Figure 1. In fact, the *stochastic efficient-frontier* is exactly the *convex-envelope* of the discrete *efficient-frontier*. Note from Figure 8 that there may now be points on the discrete frontier that are not on its convex-envelope and are hence inefficient from a *stochastic* document composition perspective. We call the points on both the discrete and continuous (*stochastic*) frontiers as *key points*. Any revenue-quality combination on the *stochastic* frontier may be realized by stochastically composing key point documents.

The notion of a *stochastic* frontier must be used with care. When two frontier points are far apart in quality, the *average* quality may not be a good indicator of quality since a publisher would definitely not want to send a bad quality document to a fraction of his subscriber base. This case may be mitigated by pruning discrete frontier points below a quality threshold before the convex envelope is calculated. Note also that a *stochastic* frontier exists strictly only in the revenue-quality space and not in the revenue-log(quality) space since averaging log(quality) is not meaningful. However, the revenue-log(quality) space may still be used to locate key points.

In the case of *ad-pull* insertion the computation of the key points on the *stochastic* frontier simply amounts to computing the discrete *efficient-frontier* and deleting vertices that are dominated by line segments connecting any two

other points. A point (r_0, q_0) is said to be dominated by a line segment between points (r_1, q_1) and (r_2, q_2) if any point $(\alpha r_1 + (1-\alpha) r_2, \alpha q_1 + (1-\alpha) q_2)$, $\alpha \in [0,1]$ on the line segment has a greater revenue for the same quality or a greater quality for the same revenue. This translates to the following two conditions respectively.

$$\frac{q_0 - q_2}{q_1 - q_2}\, r_1 + \frac{q_1 - q_0}{q_1 - q_2}\, r_2 \quad > \quad r_0 \ , \ \frac{q_0 - q_2}{q_1 - q_2} \in [0,1] \quad (4)$$

$$\frac{r_0 - r_2}{r_1 - r_2}\, q_1 + \frac{r_1 - r_0}{r_1 - r_2}\, q_2 \quad > \quad q_0 \ , \ \frac{r_0 - r_2}{r_1 - r_2} \in [0,1] \quad (5)$$

In the case of *ad-push* insertion one may derive an algorithm that directly optimizes the parametric convex envelope. A key-point document $\mathcal{D}_\alpha$ that is on the convex envelope must be a solution to the problem:

$$\mathcal{D}_\alpha = \arg\max_{\mathcal{D}} \alpha\ r(\mathcal{D}, I) + (1-\alpha)\ \log Q(D, I) \quad (6)$$

where $Q(D, I)$ is given by equation (1). We can easily modify the *ad*-free PDM algorithm [2] to the task of computing $\mathcal{D}_\alpha$ for a given $\alpha \in [0,1]$. Algorithms 5 and 6 summarize this approach. Note that other than Step 2 of the forward pass algorithm 5 and the taking of logarithms (converting products into sums), this approach is identical to the *ad*-free PDM algorithm[3].

Algorithm 5 Computing Document $\mathcal{D}_\alpha$: Forward pass

1: $\Psi(\mathcal{A}, \mathcal{B}, T) = \max_{\mathbf{\Theta}} \mathbb{P}(\mathcal{A}, |\mathcal{B}, \mathbf{\Theta}, T) \mathbb{P}(\mathbf{\Theta}|T)$
2: $\Psi_\alpha(\mathcal{A}, \mathcal{B}, T) = \alpha\ r(T) + (1-\alpha) \log \Psi(\mathcal{A}, \mathcal{B}, T)$
3: $\Phi(\mathcal{A}, \mathcal{B}) = \max_{T \in \Omega} \left(\Psi_\alpha(\mathcal{A}, \mathcal{B}, T) + \log \mathbb{P}(T)\right)$
4: $\tau_0(\mathcal{A}) \leftarrow \Phi(\mathcal{A}, \emptyset)$
5: $\tau_i(\mathcal{A})) = \max_{\mathcal{B}} \left(\Phi(\mathcal{A}, \mathcal{B}) + \tau_{i-1}(\mathcal{B})\right), i \geq 1$

Algorithm 6 Computing Document $\mathcal{D}_\alpha$: Backward pass

$i \leftarrow I - 1, \mathcal{A} \leftarrow \mathcal{C}$
while $i \geq 0$ **do**
  $\mathcal{C}^*_{\leq i} \leftarrow \mathcal{A}$
  $\mathcal{B}^* = \arg\max_{\mathcal{B}} \left(\Phi(\mathcal{A}, \mathcal{B}) + \tau_{i-1}(\mathcal{B})\right)$
  $T^*_i = \arg\max_{T \in \Omega} \left(\Psi_\alpha(\mathcal{C}^*_{\leq i}, \mathcal{B}^*, T) + \log \mathbb{P}(T)\right)$
  $\mathbf{\Theta}^*_i = \arg\max_{\mathbf{\Theta}} \mathbb{P}(\mathcal{C}^*_{\leq i}, |\mathcal{B}^*, \mathbf{\Theta}, T^*_i) \mathbb{P}(\mathbf{\Theta}|T^*_i)$
  $\mathcal{A} \leftarrow \mathcal{B}^*$
  $i \leftarrow i - 1$
end while

To compute the key-points of the *stochastic efficient-frontier* we sweep α through its range $[0,1]$ and store the resulting compositions (ignoring duplicate compositions). While this approach requires minimal modifications to the *ad*-free PDM algorithm and may be massively parallelized (documents for different α values may be computed independently) it is not an efficient way to compute the key points of the *stochastic* frontier. However, we have included it for completeness.

A much better approach for computing key points of the *stochastic efficient-frontier* is to modify the efficient algorithm for *ad*-push insertion given in Section 4.2 that operates directly on frontiers. In fact all we need to do is to use the key-points of the convex-envelope of the efficient frontiers computed in the intermediate steps 2 and 4 of the forward pass algorithm 3. The convex-envelope of an *efficient-frontier* is obtained as outlined earlier by applying conditions (4) and (5) to prune points that will be inefficient when we consider the *stochastic efficient-frontier*. Figure 9 illustrates the pruning criteria graphically.

[3]where templates are assumed to have pre-designed *ad*-slots

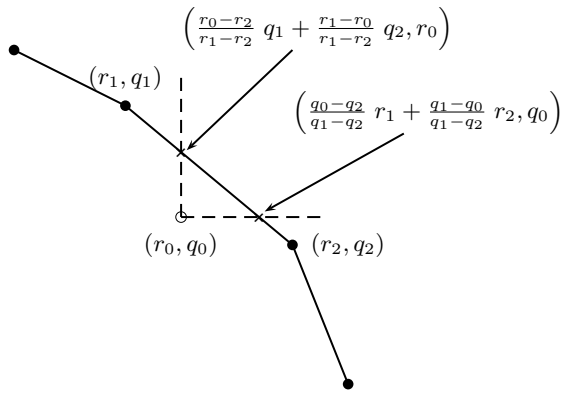


Figure 9: Constructing the convex envelope of a discrete *efficient-frontier*. We prune points on the discrete frontier that are dominated by line segments between any two other frontier points when one of the conditions in equations (4) or (5) is met.

In fact another aspect of algorithm 3 is benefited by the forced convexity of the efficient frontiers. The Minkowski sums $\oplus$ computed in step 4 of the forward pass become $O(m+n)$ instead of $O(mn)$ where m and n denote the number of points in the two summed frontiers respectively. This is because we may treat each frontier as a convex polygon (by linking the first and last points). It is well known that the Minkowski sum of two convex polygons of m and n vertices can be performed as an $O(m+n)$ operation [3]. See Appendix for pseudocode.

6. CONCLUSIONS

In this paper we showed that a fundamental tradeoff exists between revenue and document composition quality. The optimal tradeoff can be represented in terms of a set of *efficient* points called the *efficient-frontier*. We developed algorithms that can compute the frontier compositions, automatically determining how many *ads* of each size to place and how to best arrange them on the page. The user or the publisher may control *ad* density and the algorithms will produce compositions that slide along the frontier while avoiding *in-efficient* compositions. Our complexity analysis of *ad*-insertion business models in Section 4.1.2 and Section 4.2.2 reveals that both *ad-pull* and *ad-push* business models scale linearly with content size for a given ad-set size or ad-density (in the best case). However, for a given content size, as the ad-set size is increased, the optimal *ad-pull* algorithm grows combinatorially with ad-set size while the *ad-push* algorithm is not influenced by ad-density (the typical complexity of the skyline algorithm is dependent only on the expected number of *efficient* points [4]). Thus from a document composition perspective it is much more efficient to compose documents with *ad*-slots (*ad-push*) and then sell the resulting *ad* inventory than to first conduct *ad* sales and then compose documents that consider the bid prices of particular *ads* ((*ad-pull*). The downside of the *ad-push* model is

that the demand for *ad*-slots must be forecast in advance and there is the possibility of unsold inventory. Finally, we considered inter-subscriber tradeoffs and introduced the concept of a continuous and convex *stochastic efficient-frontier* and developed *ad*-insertion algorithms that exploited convexity.

While the complexity of optimal *ad-pull* insertion may be prohibitive, heuristic algorithms that work well in practice may exist. Also, we may expand the notion of quality to include keyword relevance enabling dynamic ad-insertion based on keyword proximity. This would allow advertisers to bid on keywords used in a document much like they do today for internet advertising. These are topics we intend to pursue in future work.

7. ACKNOWLEDGMENTS

The authors would like to thank Eamonn O'Brien-Strain, Jerry Liu and Qian Lin at HP Labs Palo Alto for their support of this research. Jose Bento was supported by the AFOSR grant FA9550-10-1-0360.

8. REFERENCES

[1] H. Y. Balinsky, A. J. Wiley, and M. C. Roberts. Aesthetic measure of alignment and regularity. In *DocEng '09: Proceedings of the 9th ACM symposium on Document engineering*, pages 56–65, New York, NY, USA, 2009. ACM.

[2] N. Damera-Venkata, J. Bento, and E. O'Brien-Strain. Probabilistic document model for automated document composition. In *Proceedings of the 11th ACM symposium on Document engineering*, DocEng '11, pages 3–12, New York, NY, USA, 2011. ACM.

[3] M. de Berg, O. Cheong, and M. van Kreveld, Marc. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, Berlin/Heidelberg, 2010.

[4] P. Godfrey, R. Shipley, and J. Gryz. Algorithms and analyses for maximal vector computation. *The VLDB Journal*, 16(1):5–28, jan 2007.

[5] S. J. Harrington, J. F. Naveda, R. P. Jones, P. Roetling, and N. Thakkar. Aesthetic measures for automated document layout. In *DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering*, pages 109–111, New York, NY, USA, 2004. ACM.

[6] N. Hurst, W. Li, and K. Marriott. Review of automatic document formatting. In *DocEng '09: Proceedings of the 9th ACM symposium on Document engineering*, pages 99–108, New York, NY, USA, 2009. ACM.

[7] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *J. ACM*, 22(4):469–476, oct 1975.

[8] S. Lok and S. Feiner. A survey of automated layout techniques for information presentations. In *SmartGraphics '01: Proceedings of SmartGraphics Symposium '01*, pages 61–68, New York, NY, USA, 2001. ACM.

[9] H. Markowitz. Portfolio selection. *Journal of Finance*, 7(1):77–91, March 1952.

[10] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[11] R. Phillips. Efficient frontiers in revenue management. *Journal of Revenue and Pricing Management*, pages 1–15, September 2011.

APPENDIX

Algorithm 7 Skyline algorithm [4] $\mathcal{F} = \text{eff}\,\mathcal{S}$.

$\mathcal{S} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_N\}$
function eff($\mathcal{S}$)
  $\mathcal{F} \leftarrow \emptyset$
  while ($\mathcal{S}$ is not empty) **do**
    $\mathbf{p} \leftarrow shift(\mathcal{S})$ // Get the first.
    for all $\mathbf{q} \in \mathcal{S}$ **do** // Find a max.
      if $(\mathbf{p} \succ \mathbf{q})$[4] **then**
        remove $\mathbf{q}$ from $\mathcal{S}$
      else if $\mathbf{q} \succ \mathbf{p}$ **then**
        remove $\mathbf{q}$ from $\mathcal{S}$
        $\mathbf{p} \leftarrow \mathbf{q}$
      end if
    end for
    $\mathcal{F} \leftarrow \mathcal{F} \bigcup \mathbf{p}$
    for all $\mathbf{q} \in \mathcal{S}$ **do** // Clean up.
      if $(\mathbf{p} \succ \mathbf{q})$ **then**
        remove $\mathbf{q}$ from $\mathcal{S}$
      end if
    end for
  end while
end function

Algorithm 8 General Minkowski sum $\mathcal{F}_3 = \mathcal{F}_1 \oplus \mathcal{F}_2$

$\mathcal{F}_1 = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M\}$
$\mathcal{F}_2 = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N\}$
function $\oplus(\mathcal{F}_1,\mathcal{F}_2)$
  $\mathcal{F}_3 \leftarrow \emptyset$
  for i = 1,M **do**
    for j = 1,N **do**
      $\mathbf{z} = \mathbf{p}_i + \mathbf{q}_j$
      $\mathcal{F}_3 \leftarrow \mathcal{F}_3 \bigcup \mathbf{z}$
    end for
  end for
end function

Algorithm 9 $\mathcal{F}_3 = \mathcal{F}_1 \oplus \mathcal{F}_2$, $\mathcal{F}_1$,$\mathcal{F}_2$ are convex polygons.

$\mathcal{F}_1 = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M\}$, ordered clockwise
$\mathcal{F}_2 = \{\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N\}$ ordered clockwise
function $\oplus(\mathcal{F}_1,\mathcal{F}_2)$
  $\mathcal{F}_3 \leftarrow \emptyset$
  Locate extreme $\mathbf{p}_i$ and $\mathbf{q}_j$ in -Y direction.
  $\mathbf{z}_1 = \mathbf{p}_i + \mathbf{q}_j$, $F_3 \leftarrow \mathcal{F}_3 \bigcup \mathbf{z}_1$
  Make parallel lines at $\mathbf{p}_i$,$\mathbf{q}_j$ so $\mathcal{F}_1$,$\mathcal{F}_2$ lie to the right.
  for k = 2, M+N **do**
    Determine angles θ_i and ϕ_j at $\mathbf{p}_i$ and $\mathbf{q}_j$

$$\mathbf{z}_k = \begin{cases} \mathbf{p}_{i+1} + \mathbf{q}_j & \theta_i < \phi_j \\ \mathbf{p}_i + \mathbf{q}_{j+1} & \theta_i > \phi_j \\ \mathbf{p}_{i+1} + \mathbf{q}_{j+1} & \theta_i = \phi_j \end{cases} \quad \text{// } i,j \text{ circular.}$$

    $\mathcal{F}_3 \leftarrow \mathcal{F}_3 \bigcup \mathbf{z}_k$
    Rotate parallel lines to get next $\mathbf{p}_i$ and $\mathbf{q}_j$.
  end for
end function

[4] $\mathbf{p} \succ \mathbf{q}$ means ($\mathbf{p}(1) \geq \mathbf{q}(1)$ and $\mathbf{p}(2) > \mathbf{q}(2)$) **or** ($\mathbf{p}(1) > \mathbf{q}(1)$ and $\mathbf{p}(2) \geq \mathbf{q}(2)$)