Probabilistic Document Model for Automated Document Composition

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

Eamonn O'Brien-Strain
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
eob@hp.com

ABSTRACT

We present a new paradigm for automated document composition based on a generative, unified probabilistic document model (PDM) that models document composition. The model formally incorporates key design variables such as content pagination, relative arrangement possibilities for page elements and possible page edits. These design choices are modeled jointly as coupled random variables (a Bayesian Network) with uncertainty modeled by their probability distributions. The overall joint probability distribution for the network assigns higher probability to good design choices. Given this model, we show that the general document layout problem can be reduced to probabilistic inference over the Bayesian network. We show that the inference task may be accomplished efficiently, scaling linearly with the content in the best case. We provide a useful specialization of the general model and use it to illustrate the advantages of soft probabilistic encodings over hard one-way constraints in specifying design aesthetics.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design

Keywords

automated publishing, layout synthesis, variable templates

1. INTRODUCTION

In order to compose documents, one must make aesthetic decisions on how to paginate content, how to arrange document 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. 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 formatted for a plethora of device form factors.

Automated document composition attempts to transform raw content automatically into documents with high aesthetic value. This has been a topic of much research [10] [6] and is the focus of this paper. In this paper we propose a new way to think about automated document composition based on a generative probabilistic document model (PDM) that incorporates the key design choices described above as random variables with associated probability distributions. The model is generative in the sense that documents may be produced as samples from an underlying probability distribution. We model the coupling between the design choices explicitly using a Bayesian network [12]. At the core of our model is the idea of a probabilistic encoding of aesthetic design judgment with uncertainty encoded as *prior* probability distributions. For example a designer may specify that the whitespace between two page elements have a mean (desired value), and a variance (flexibility). If the variance is large then a larger range of values is tolerated for the whitespace. A small variance implies a tighter range of values. As a consequence of the PDM model we show that automated document composition may be efficiently realized as an inference problem in the network where the inference task is to simultaneously find parameter estimates that maximize the joint probability distribution of all network variables. Probabilistic specification of design intent allows the inference procedure to make appropriate design tradeoffs while laying out content. While the resulting layout may require deviation from the designer's most desired parameter settings, parameters with tighter variances deviate less than parameters with larger variances. Such soft encodings are in contrast with state of the art template selection methods that use hard one-way constraints to encode layout design preferences [7]:

1. We explicitly model the editing process using template parameters that are automatically and *actively* adjusted. This allows our templates to *continuously* (and optimally) morph to fit content. A continuous distribution of the whitespace possibilities between even two items on a page would require an infinite number of distinct one-way constraint templates to encode. With one-way constraints, this results in a template explosion due to the need for (1) templates (and/or alternate versions of content) to cover for variability *within* a relative arrangement of elements (2) templates to cover variability *between* relative arrangements. The PDM model in this paper shows how (1) can be addressed effectively. New template parameterizations (that are beyond the scope of this

©ACM, (2011). This is the author's version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. The definitive version was published in the Proceedings of the 11th ACM symposium on Document engineering (DocEng '11), ISBN: 978-1-4503-0863-2, (Sep 19–22, 2011). http://doi.acm.org/10.1145/2034691.2034695. All photographs used have Creative Commons licenses and are copyright their owners: Mike Baird (boat), D Sharon Pruitt (punk girl and veiled girl), and Steve Brace (owl).

paper, but within the scope of the general PDM framework) may be needed to address the latter case.

2. We capture local aesthetic design judgments of designers explicitly using prior probability distributions of template parameters. This allows designers to capture flexibility in parameter settings allowing greater freedom in choosing the parameters to fit content. One-way constraints are a much more rigid design specification.

Our work is greatly influenced by similar work on document content modeling and analysis. Probabilistic graphical models (especially Bayesian network models) are very popular in the text mining, document retrieval and document content understanding communities. For example the probabilistic models for text content in documents proposed by Hoffman [5] and Blei et. al. [3] model words in a document by first sampling a topic from a topic distribution and then sampling words from specific topic distributions. While these models are also referred to as probabilistic document models, they model the actual text content and not document composition. Our aim was to bring the flexibility of such an approach to bear on the document composition problem.

2. RELATED WORK

The survey papers by Lok et al. [10] and Hurst et al. [6] provide a comprehensive background on automated document composition technologies. We only discuss closely related work in this section.

At the page composition level, constraint solvers are often used in variable data printing (VDP) to accommodate content block size variations via *active* templates [9]. In VDP, templates are designed for specific content by a professional graphic designer. The template containers can be *nudged* to accommodate content size variations. The adjustment of the width/height of containers is based on a set of constraints constraining relative positions of content blocks [2]. The set of constraints are solved using a constraint solver [1] to determine final block positions. Since all page layouts fitting constraints are considered equally good, such constraints are often called *hard* constraints. Layout synthesis is reduced to the problem of generating a solution consistent with all the constraints/rules. The document description framework (DDF) [11] is a format (XML) innovation that extends the notion of variable templates to allow variations in number of content blocks (flows) within fixed template regions. DDF is not an optimization framework but rather a format that allows a designer to program rules encoded within XML that may be used to determine how to flow content into local template regions.

While most of the automated layout handled by variable data printing (VDP) is often page based (i.e. content is specified a page at a time) an important component of general document quality is pagination which determines what content to place on each page. This influences how close the referenced figures, sidebars etc. are to the text blocks that referenced them. LaTeX uses a first fit sequential approach for floating objects. It places a floating object on the first page on which it fits, after it has been cited. Bruggeman-Klein et. al. [4] extended the pioneering work of Plass [13] to solve for the optimal allocation of content to pages using dynamic programming to minimize the overall number of page turns (in looking for referenced items). However, these pagination methods do not consider the impact content allocation has on document composition quality.

Jacobs et. al. [7] use dynamic programming to select a template to compose each page of the document from a template library using dynamic programming. Content is flowed into these templates using one-way constraints [15]. One-way constraints are simply mathematical expressions in terms of other constraint variables. Constraint dependencies may be expressed as a directed acyclic graph. Greedy traversal of this graph computes a layout of all elements. The highest level constraint variables are read only (ex: page.width and page.height). As an example, to flow a title and body text into a template, the constraint system may evaluate[1]:

$$\begin{aligned} title.top &= page.height/10 \\ title.bottom &= title.top + < rendered\ Title\ height > \\ body.top &= title.bottom + 10 \end{aligned}$$

Since the absolute position on the page of different blocks change when page dimensions or content changes the authors refer to their layouts as *adaptive* layouts [7]. Templates are scored based on how well content fills the containers, and how many widowed and orphaned text there are. Schrier et al. [14] develop a template description language that automatically generates several templates from a small set of flexible template descriptions. Such template generation methods are complementary to the models described in this paper.

Our goal in this paper is to formally unify under one framework the design choices explored in the prior work, including page level adjustments [9], pagination [4] and choice of relative content arrangements [7]. As discussed in the introduction, probabilistic modeling is a natural choice for encoding soft aesthetic judgments which provides added flexibility.

3. PROBABILISTIC DOCUMENT MODEL

This section introduces the basic concept of PDM and its representation as a Bayesian network. We pose automated document composition as an inference problem over the Bayesian network and derive an algorithm for optimal document composition.

3.1 Notation

We use the following general mathematical notation: random variables (uppercase, ex: T), sample realizations of a random variable (lowercase of corresponding random variable, ex: t), matrices (bold uppercase, ex: $\mathbf{X}$), vectors (bold lowercase, ex: $\mathbf{h}$), random sets (uppercase script, ex: $\mathcal{A}$), sample realizations of a random set (lowercase script of corresponding random set, ex: a). X^* denotes the *optimal* sample value of random variable X. $|\mathcal{A}|$ is a count of the number of elements in the set $\mathcal{A}$. We use $\sim$ to indicate the distribution of a random variable. Thus, $X \sim \mathrm{N}(\bar{X}, \sigma^2)$ indicates that random variable X is normally distributed with mean $\bar{X}$ and variance σ^2). A normal probability distribution of X may also be written compactly as $\mathbb{P}(X) = \mathrm{N}(X|\bar{X}, \sigma^2)$. We use $\approx$ to indicate normal variation around a desired equality. Thus $X \approx x$ is equivalent to $X - x \sim \mathrm{N}(0, \sigma^2)$.

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 c that is a particular sample of content from a random set $\mathcal{C}$ with sample space comprising sets of all possible content input sets. Text units could be words, sentences, lines of text or whole paragraphs.

We denote by c' a set comprising all sets of discrete content allocation possibilities over one or more pages starting with and including the first page. Content subsets that do not form valid allocations (e.g. allocations of non-contiguous lines of text) do not exist in c'. If there are 3 lines of text and 1 floating figure to be composed, $c = \{l_1, l_2, l_3, f_1\}$ while $c' = \{\{l_1\},\{l_1, l_2\},\{l_1, l_2, l_3\},\{f_1\},$ $\{l_1, f_1\},\{l_1, l_2, f_1\},\{l_1, l_2, l_3, f_1\}\} \cup \{\emptyset\}$. Note that the specific

[1]heights are measured from the top of the page

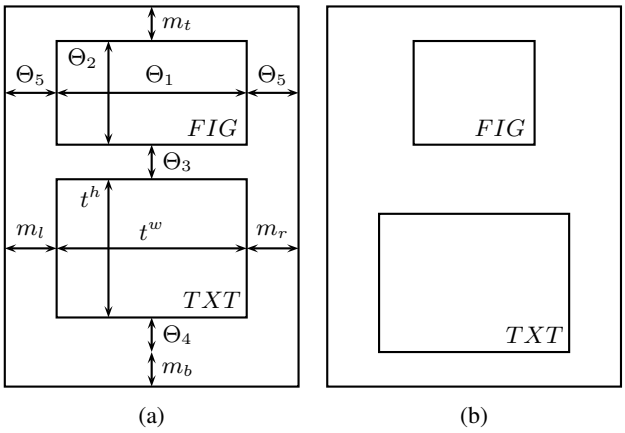


Figure 1: (a) Example Probabilistic page template containing two elements (a figure and a text stream) with random parameters $\mathbf{\Theta} = [\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5]^T$, and (b) A *static* template derived by sampling random variable $\mathbf{\Theta}$. In this example, values m_t, m_b, m_l, m_r are constants describing the margins, and t^w and t^h are constants describing the width and height of a block of text.

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 $\{l_1, l_3, f_1\} \notin c'$ since lines 1 and 3 cannot be in the same allocation without including line 2. c' includes the empty set to allow the possibility of a null allocation. In general, if there are $|c_l|$ lines of text and $|c_f|$ figures in c, we have $|c'| = (|c_l| + 1)\ (|c_f| + 1)$, if figures are allowed to float freely. Thus, $|c'| \gg |c|$ even when c has a moderate number of figures.

We represent the index of a page by $i \geq 0$. $\mathcal{C}_i$ is a random set representing the content allocated to page i. $\mathcal{C}_{\leq i} \in c'$ is a random set of content allocated to pages with index 0 through i. Hence we have $\mathcal{C}_{\leq i} = \cup_{j=0}^{i} \mathcal{C}_j$. If $\mathcal{C}_{\leq i} = \mathcal{C}_{\leq i-1}$ then $\mathcal{C}_i = \emptyset$ (page i has no content allocated to it). We define $\mathcal{C}_{\leq -1} = \emptyset$ for convenience, so that all pages $i \geq 0$ have valid content allocations to the previous $i - 1$ pages.

3.2 Representation

Traditionally, a page template is an abstract representation of page composition, consisting of *copy holes* for different page elements (such as figures, text streams, sidebars etc.). A template encodes the absolute positions of all elements to be placed on the page. A page composition using a particular template is produced by simply pasting images (either by cropping or scaling) into the image holes and flowing text into the text holes. A template thus determines the absolute positions of all page elements. We refer to such templates as *static* templates.

In this paper we expand the classical notion of a *static* template to what we call a *probabilistic* template. A *probabilistic* template is parameterized by parameters (ex: whitespace between elements, image dimensions) that are themselves random variables. A random sampling of the random variables representing the template gives rise to a particular *static* template. Thus a probabilistic template represents only relative positions of page elements. A page layout obtained from a *probabilistic* template is allowed to continuously morph (as template parameters are sampled). Fig 1 illustrates the concept of a *probabilistic* page template.

According to PDM the i^{th} page of a *probabilistic* document may be composed by first sampling random variable T_i representing choice from a library of page templates representing different relative arrangements of content, sampling a random vector $\mathbf{\Theta}_i$ of template parameters representing possible edits to the chosen template, and sampling a random set $\mathcal{C}_i$ of content representing content allocation to that page (a.k.a. pagination).

A template t_i for page i is sampled from a probability distribution $\mathbb{P}_i(T_i)$ over a set Ω_i of template indices with $|\Omega_i|$ possible template choices. This formulation allows for example, first, last, even and odd page templates to be sampled from different sublibraries of the overall template library. Once a template is sampled we may sample its parameter vector $\boldsymbol{\theta}_i$ from the conditional multi-variate probability distribution $\mathbb{P}(\mathbf{\Theta}_i | t_i)$. This distribution may be regarded as a *prior* probability distribution that determines the *prior* uncertainty (before seeing content) of template parameters. Graphic design knowledge regarding parameter preferences may be directly encoded into this distribution. Thus sampling from this distribution makes *aesthetic* parameter settings more likely. Finally, the allocation for the current and previous pages $c_{\leq i}$ is sampled from a probability distribution $\mathbb{P}(\mathcal{C}_{\leq i} | c_{\leq i-1}, \boldsymbol{\theta}_i, t_i)$. This distribution reflects how well the content allocated to the current page fits when $T_i = t_i$ and template parameters $\mathbf{\Theta}_i = \boldsymbol{\theta}_i$. The allocation to the previous pages affects the probability of an allocation for the current page via the logical relationship content in previous pages has to content on the current page. For example, previous page allocation allows us to determine if a figure or sidebar appearing on the current page is referenced in a prior page (a dangling reference). The current page allocation can be obtained as $c_i = c_{\leq i} - c_{\leq i-1}$. In summary a random document can be generated from the probabilistic document model by using the following sampling process for page $i \geq 0$ with $c_{\leq -1} = \emptyset$:

$$\text{sample template } t_i \quad \text{from} \quad \mathbb{P}_i(T_i) \tag{1}$$

$$\text{sample parameters } \boldsymbol{\theta}_i \quad \text{from} \quad \mathbb{P}(\mathbf{\Theta}_i | t_i) \tag{2}$$

$$\text{sample content } c_{\leq i} \quad \text{from} \quad \mathbb{P}(\mathcal{C}_{\leq i} | c_{\leq i-1}, \boldsymbol{\theta}_i, t_i) \tag{3}$$

$$c_i \quad = \quad c_{\leq i} - c_{\leq i-1} \tag{4}$$

The sampling process naturally terminates when the content runs out. Since this may occur at different random page counts each time the process is initiated, the document page count I is itself a random variable defined by the minimal page number at which $\mathcal{C}_{\leq i} = c$. Formally, $I = 1 + \underset{\{i : \mathcal{C}_{\leq i} = c\}}{\operatorname{argmin}}\ i < \infty$. A document $\mathcal{D}$ in PDM is thus 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 equations (1)-(4).

For a specific content c, the probability of producing document $\mathcal{D}$ of I pages via the sampling process described in this section is simply the product of the probabilities of all design (conditional) choices made during the sampling process. Thus,

$$\mathbb{P}(\mathcal{D}; I) \quad = \quad \prod_{i=0}^{I-1} \mathbb{P}(\mathcal{C}_{\leq i} | \mathcal{C}_{\leq i-1}, \mathbf{\Theta}_i, T_i) \mathbb{P}(\mathbf{\Theta}_i | T_i) \mathbb{P}_i(T_i) \tag{5}$$

The probability distribution of equation (5) may be represented as a directed graph representing the relationship between the design choices as shown in Fig. 2. The distribution can be generated from the graph by simply multiplying the conditional probability distributions of each node conditioned only on its parents. Such a model is called a Bayesian network model for the underlying probability distribution [12]. The model is generative, in the sense that a sequential process (as described above) can be used to generate documents from the model. The documents generated by this process are simply samples drawn from the probability distribution described above. Further, in a Bayesian network every node is

conditionally independent of its non-descendants given its parents [12]. Thus our PDM model implicitly assumes that the allocation to page i is independent of the templates and parameter selections of all previous pages, given the content allocations to previous pages and the template and template parameters for the current page.

Although the sampling procedure described in this section generates documents with various probabilities (higher probabilities translate to higher quality) we are interested only in finding the document that has the highest probability. Our goal is to compute the optimizing sequences of templates, template parameters, and content allocations that maximize overall document probability. It is clear that a naive approach that generates and scores all possible documents and picks the one with the maximum probability is infeasible since there are an infinite number of possible documents. In fact the structure of the graph (representing conditional independence in the joint probability distribution) may be used to derive an efficient Bayesian inference procedure (essentially equivalent to dynamic programming) that computes the optimal solution. We turn to model inference next.

3.3 Model Inference

We refer to the task of computing the optimal page count and the optimizing sequences of templates, template parameters, content allocations that maximize overall document probability as the model inference task.

$$(\mathcal{D}^*, I^*) \quad = \quad \operatorname*{argmax}_{\mathcal{D}, I \geq 1} \mathbb{P}(\mathcal{D}; I) \tag{6}$$

The key to efficient inference in Bayesian networks is the fact that although all the variables influence each other in general, a variable is only directly influenced by a few neighbors at most. This allows us to break up the desired maximization sequentially into several sub-problems. We start by gathering terms in (5) involving $\mathbf{\Theta}_i$ and maximizing over $\mathbf{\Theta}_i$.

$$\Psi(\mathcal{C}_{\leq i}, \mathcal{C}_{\leq i-1}, T_i) = \max_{\mathbf{\Theta}_i} \mathbb{P}(\mathcal{C}_{\leq i} | \mathcal{C}_{\leq i-1}, \mathbf{\Theta}_i, T_i) \mathbb{P}(\mathbf{\Theta}_i | T_i) \tag{7}$$

The maximization over $\mathbf{\Theta}_i$ effectively eliminates $\mathbf{\Theta}_i$ from further consideration, resulting in a function Ψ of the remaining variables in the RHS of (7). Now grouping the remaining terms in (5) involving T_i with $\Psi(\mathcal{C}_{\leq i}, \mathcal{C}_{\leq i-1}, T_i)$ and maximizing over T_i we have

$$\Phi_i(\mathcal{C}_{\leq i}, \mathcal{C}_{\leq i-1}) = \max_{T_i \in \Omega_i} \Psi(\mathcal{C}_{\leq i}, \mathcal{C}_{\leq i-1}, T_i) \mathbb{P}_i(T_i) \tag{8}$$

Note that the function Φ_i depends on i since the maximization over allowed templates for each page occurs over distinct sub-libraries Ω_i that depend on i. We can now rewrite the maximization of (5) purely in terms of functions Φ_i of content allocations as

$$\max_{\mathcal{D}, I \geq 1} \mathbb{P}(\mathcal{D}; I) = \max_{I \geq 1} \max_{\{\mathcal{C}_{\leq i}\}_{i=0}^{I-1}} \prod_{i=0}^{I-1} \Phi_i(\mathcal{C}_{\leq i}, \mathcal{C}_{\leq i-1}) \tag{9}$$

We may now sequentially maximize over content allocations $C_{\leq i}$ for $i = 0, 1, \cdots, I-1$. First, grouping terms involving $C_{\leq 0}$ in (9) and maximizing over $C_{\leq 0}$ we have

$$\tau_1(\mathcal{C}_{\leq 1}) \quad = \quad \max_{\mathcal{C}_{\leq 0}} \Phi_1(\mathcal{C}_{\leq 1}, \mathcal{C}_{\leq 0}) \Phi_0(\mathcal{C}_{\leq 0}, \mathcal{C}_{\leq -1}) \tag{10}$$

$$\quad = \quad \max_{\mathcal{C}_{\leq 0}} \Phi_1(\mathcal{C}_{\leq 1}, \mathcal{C}_{\leq 0}) \Phi_0(\mathcal{C}_{\leq 0}, \emptyset) \tag{11}$$

Then maximizing over content allocations $C_{\leq 1}$ we have

$$\tau_2(\mathcal{C}_{\leq 2}) = \max_{\mathcal{C}_{\leq 1}} \Phi_2(\mathcal{C}_{\leq 2}, \mathcal{C}_{\leq 1}) \tau_1(\mathcal{C}_{\leq 1}) \tag{12}$$

We can easily see that this process is governed by the following general recursion for $i \geq 1$

$$\tau_i(\mathcal{C}_{\leq i}) = \max_{\mathcal{C}_{\leq i-1}} \Phi_i(\mathcal{C}_{\leq i}, \mathcal{C}_{\leq i-1}) \tau_{i-1}(\mathcal{C}_{\leq i-1}) \tag{13}$$

with the added definition $\tau_0(\mathcal{C}_{\leq 0}) = \Phi_0(\mathcal{C}_{\leq 0}, \emptyset)$. Note that at the end of the recursive computation of the functions τ_i, since $C_{\leq I-1} = c$, the entire content to be composed, $\tau_{I-1}(c)$ is only dependent on I. Therefore we can finally write down the maximum of (5) as

$$\max_{I \geq 1} \max_{\mathcal{D}} \mathbb{P}(\mathcal{D}; I) \quad = \quad \max_{I-1 \geq 0} \tau_{I-1}(c), \tag{14}$$

The optimal page count can now be determined as the corresponding maximizer over all last page possibilities (where all content $\mathcal{C}$ has been allocated). Thus,

$$I^* \quad = \quad 1 + \operatorname*{argmax}_{i \geq 0} \tau_i(c) \tag{15}$$

We now discuss how the optimal allocations $\mathcal{C}^*_{\leq i}$ can be inferred. First note that for the final page (page number $I^* - 1$) we must have $\mathcal{C}^*_{\leq I^*-1} = c$. From $\mathcal{C}^*_{\leq I^*-1}$ we can compute the optimal allocations to the previous pages, $\mathcal{C}^*_{\leq I^*-2}$ by substituting the known $\mathcal{C}^*_{\leq I^*-1}$ in the recursion (13) for τ_{I^*-1} and finding the value of $\mathcal{C}_{\leq I^*-2}$ that maximizes τ_{I^*-1}. Specifically,

$$\mathcal{C}^*_{\leq I^*-2} = \operatorname*{argmax}_{\mathcal{C}_{\leq I^*-2}} \Phi_{I^*-1}(\mathcal{C}^*_{\leq I^*-1}, \mathcal{C}_{\leq I^*-2}) \tau_{I^*-2}(\mathcal{C}_{\leq I^*-2}) \tag{16}$$

In general we can set up a recursion that allows us to solve for optimal allocations for all I^* pages. Once the allocations for each page are determined, we may look up the optimal template for each page by finding the template T_i^* that corresponds to $\Phi(\mathcal{C}^*_{\leq i}, \mathcal{C}^*_{\leq i-1})$. Once the template and the allocation are known, optimal template parameters $\mathbf{\Theta}_i^*$ may be computed. This procedure is given below for $i = I^* - 1, \cdots, 0$:

$$\mathcal{C}^*_{\leq i-1} \quad = \quad \operatorname*{argmax}_{\mathcal{C}_{\leq i-1}} \Phi_i(\mathcal{C}^*_{\leq i}, \mathcal{C}_{\leq i-1}) \tau_{i-1}(\mathcal{C}_{\leq i-1}) \tag{17}$$

$$T_i^* \quad = \quad \operatorname*{argmax}_{T_i} \Psi(\mathcal{C}^*_{\leq i}, \mathcal{C}^*_{\leq i-1}, T_i) \mathbb{P}_i(T_i) \tag{18}$$

$$\mathbf{\Theta}_i^* \quad = \quad \operatorname*{argmax}_{\mathbf{\Theta}_i} \mathbb{P}(\mathcal{C}_{\leq i^*} | \mathcal{C}_{\leq i^*-1}, \mathbf{\Theta}_i, T_i^*) \mathbb{P}(\mathbf{\Theta}_i | T_i^*) \tag{19}$$

$$\mathcal{C}_i^* \quad = \quad \mathcal{C}^*_{\leq i} - \mathcal{C}^*_{\leq i-1} \tag{20}$$

Once the optimal page count, content allocations to pages, templates and template parameters are determined the inference task is complete. The solution found by this Bayesian network inference approach is a globally optimal maximizer of (5). The order of variable elimination affects efficiency but not optimality [12].

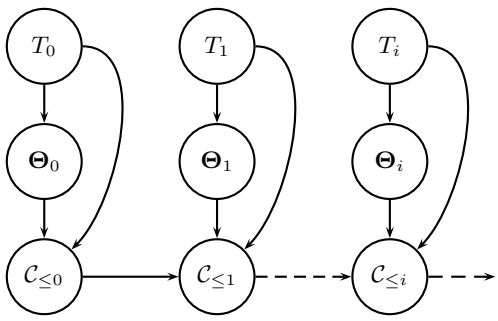


Figure 2: PDM as a graphical model.

It is important to note that while a document composed using the PDM inference algorithm described here has its page count optimally selected, the user may want to constrain the page count to a specified number of pages. To force a page count of I_f we only need to compute (17)-(20) with $I^* \leftarrow I_f$. This simply corresponds to a maximization of the conditional distribution $\mathbb{P}(\mathcal{D}|I)$.

The optimal document composition algorithm derived above may be succinctly summarized as a two pass process. In the forward pass we *recursively* compute, for all valid content allocation sets $\mathcal{A}, \mathcal{B} \in c'$ with $\mathcal{A} \supseteq \mathcal{B}$ the following coefficient tables:

$$\Psi(\mathcal{A}, \mathcal{B}, T) = \max_{\mathbf{\Theta}} \mathbb{P}(\mathcal{A}|\mathcal{B}, \mathbf{\Theta}, T)\mathbb{P}(\mathbf{\Theta}|T) \quad (21)$$

$$\Phi_i(\mathcal{A}, \mathcal{B}) = \max_{T \in \Omega_i} \Psi(\mathcal{A}, \mathcal{B}, T)\mathbb{P}_i(T), i \geq 0, \quad (22)$$

$$\tau_i(\mathcal{A}) = \max_{\mathcal{B}} \Phi_i(\mathcal{A}, \mathcal{B})\tau_{i-1}(\mathcal{B}), i \geq 1 \quad (23)$$

with $\tau_0(\mathcal{A}) = \Phi_0(\mathcal{A}, \emptyset)$. Computation of (23) depends on (22) which in turn depends on (21). In the backward pass we use these coefficients used to infer the optimal document using (17)-(20).

The innermost function $\Psi(\mathcal{A}, \mathcal{B}, T)$ is essentially a score of how well content in the set $\mathcal{A}-\mathcal{B}$ is suited for template T. It is the maximum of a product of two terms. The first term $\mathbb{P}(\mathcal{A}|\mathcal{B}, \mathbf{\Theta}, T)$ represents how well content fills the page and respects figure references while the second term $\mathbb{P}(\mathbf{\Theta}|T)$ assesses how close, the parameters of a template are to the designer's *aesthetic* preference. Thus the overall probability (score) is a tradeoff between page fill and a designer's aesthetic intent. When there are multiple parameters settings that fill the page equally well, the parameters that maximize the prior (and hence are closest to the template designer's desired values) will be favored.

Note also, that $\Psi(\mathcal{A}, \mathcal{B}, T) \propto \max_{\mathbf{\Theta}} \mathbb{P}(\mathbf{\Theta}|\mathcal{A}, \mathcal{B}, T)$ where the proportionality follows from Bayes rule. Thus $\Psi(\mathcal{A}, \mathcal{B}, T)$ is proportional to the maximum *a posteriori* (MAP) probability of the template parameters given content and a template. The maximizer $\mathbf{\Theta}^*$ is the corresponding MAP estimate. $\mathbb{P}(\mathcal{A}|\mathcal{B}, \mathbf{\Theta}, T)$ represents the likelihood function of a particular allocation that refines our *prior* belief regarding the template parameters, $\mathbb{P}(\mathbf{\Theta}|T)$ upon seeing $\mathcal{A}$ and $\mathcal{B}$. Efficiency in computing $\Psi(\mathcal{A}, \mathcal{B}, T)$ may be obtained by a) screening allocations $\mathcal{A}$ and $\mathcal{B}$ to avoid computation altogether for invalid allocations (ex: figure or sidebar occurring before its reference, widowed or orphaned text chunks etc.) b) screening templates for compatibility (ex: content with two figures cannot be allocated to a template with only one figure) c) screening for too much and too little content. These screening approaches significantly reduce the number of cases for which the expensive optimization of (21) needs to be performed.

The function $\Phi_i(\mathcal{A}, \mathcal{B})$ scores how well content $\mathcal{A} - \mathcal{B}$ can be composed onto the i^{th} page, considering all possible relative arrangements of content (templates) allowed for that page. $\mathbb{P}_i(T)$ allows us to boost the score of certain templates, increasing the chance that they will be used in the final document composition.

Finally functions $\tau_i(\mathcal{A})$ is a pure pagination score of the allocation $\mathcal{A}$ to the first i pages. The recursion (23) basically says that the pagination score for an allocation $\mathcal{A}$ to the first i pages, $\tau_i(\mathcal{A})$ is equal to the product of the best pagination score over all possible previous allocations $\mathcal{B}$ to the previous $(i-1)$ pages with the score of the current allocation $\mathcal{A} - \mathcal{B}$ to the i^{th} page, $\Phi_i(\mathcal{A}, \mathcal{B})$. Note that the pure pagination score $\tau_i(\mathcal{A})$ encapsulates dependency on relative arrangements of content (represented by templates T) and possible template edits ($\mathbf{\Theta}$) via recursive dependency on $\Phi_i(\mathcal{A}, \mathcal{B})$ and $\Psi(\mathcal{A}, \mathcal{B}, T)$.

The PDM inference framework may easily be extended to handle alternate and optional content. Alternate images may be handled by simply evaluating the RHS of (21) for all possible alternatives and choosing the maximum value for $\Psi(\mathcal{A}, \mathcal{B}, T)$. Optional images may be handled by allowing content allocations with optional images to match templates with or without available template slots to render them. Templates without available slots for any images may simply ignore optional images. However the optional images are still deemed to have been allocated.

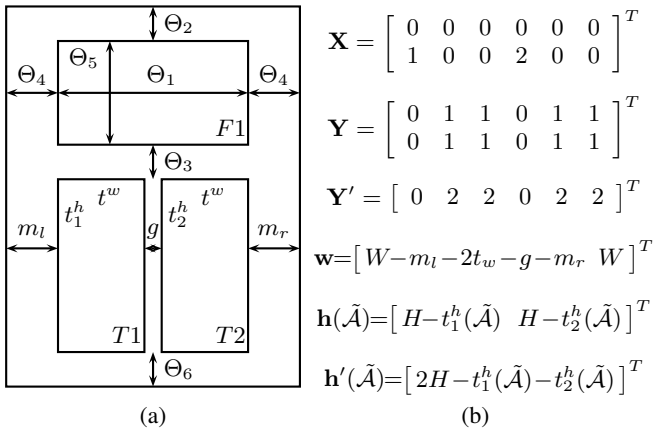


Figure 3: (a) Example Probabilistic page template containing one figure and two linked text stream blocks with random parameters $\mathbf{\Theta} = [\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6]^T$. (b) Matrix representation

The entire inference process is essentially dynamic programming with the coefficients $\Psi(\mathcal{A}, \mathcal{B}, T)$, $\Phi_i(\mathcal{A}, \mathcal{B})$, and $\tau_i(\mathcal{A})$ playing the role of dynamic programming tables. The optimization itself may also be considered analogous to the minimization of an energy function proportional to the negative logarithm of (5). In this case we simply replace products with sums, throughout but the essential math remains the same. Inference complexity is analyzed in the Appendix.

4. MODEL PARAMETERIZATION

The results of Section 3 are valid for arbitrary template parameterizations and general probability distributions. This section focuses one specific efficient parameterization of the general PDM model to illustrate the value of probabilistic modeling with aesthetic priors. This specialization can express arbitrary *soft* linear parameter relationships probabilistically. It can support multi-column layouts (with sidebars) but without element overlaps. This is still an important class of documents. This is not a limitation of the general model. This formulation is convenient since it makes the the innermost sub-problem of model inference, given by equation (21) particularly efficient.

4.1 Template Representation

Content fit to a template is assessed along all *paths* that go from top to bottom and left to right on a page. X-paths and Y-paths are computed for each template. For the example in Figure 3(a) we define two Y-paths ($\mathcal{Y}_1 = \Theta_2 \rightarrow \Theta_5 \rightarrow \Theta_3 \rightarrow t_1^h \rightarrow \Theta_6$, $\mathcal{Y}_2 = \Theta_2 \rightarrow \Theta_5 \rightarrow \Theta_3 \rightarrow t_2^h \rightarrow \Theta_6$) and two X-paths ($\mathcal{X}_1 = m_l \rightarrow t_w \rightarrow g \rightarrow t_w \rightarrow m_r$, $\mathcal{X}_2 = \Theta_4 \rightarrow \Theta_1 \rightarrow \Theta_4$).

Paths have constant and variable parameter components. In the PDM model described in Section 3 each template had fixed text elements and potentially variable image and whitespace parameters. We sampled the content for a page after the template parameters were sampled. In our model inference procedure however the situ-

ation is reversed. When we compute $\Psi(\mathcal{A}, \mathcal{B}, T)$ we are given the allocation to the current page $\tilde{\mathcal{A}} = \mathcal{A} - \mathcal{B}$ and then solve for the maximizing template parameters. Thus, text heights on a template can vary with the content allocation $\tilde{\mathcal{A}}$ to the template. However, we still regard text heights as constants since, while they vary with $\tilde{\mathcal{A}}$ they do not vary as the variable parameters change.

Variable path components of the N X-paths and M Y-paths comprising V random parameters may be described by matrices $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \cdots\ \mathbf{x}_N]$ and $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \cdots\ \mathbf{y}_M]$ consisting of column vectors of dimension $V \times 1$ for each path. The element (v, n) in matrix $\mathbf{X}$ is the number of times random parameter Θ_v appears in the n^{th} X-path. Similarly, the element (v, m) in matrix $\mathbf{Y}$ is the number of times random parameter Θ_v appears in the m^{th} Y-path. $\mathbf{X}$ and $\mathbf{Y}$ matrices corresponding to the example template in Figure 3(a) are shown in Figure 3(b).

Residual path heights and widths represent the *adjustable* portion of each of the paths. These are the path heights and widths after subtracting constant path elements. Residual path sizes are represented by the vectors $\mathbf{h}(\tilde{\mathcal{A}})$ and $\mathbf{w}$ in Figure 3(b) where H and W represent the height and width of the page respectively. Note the dependence of the residual path heights on $\tilde{\mathcal{A}}$ due to the dependency of the text heights on $\tilde{\mathcal{A}}$ as discussed earlier.

In many cases we have *multi-column flows* where text flows from one column to the next. In this case, the Y-paths with linked text flows are be grouped into a single path by simply summing the corresponding columns of $\mathbf{Y}$ and $\mathbf{h}$ giving rise to new matrices $\mathbf{Y}'$ and $\mathbf{h}'$ respectively.

Thus, we represent a probabilistic template t by matrices $\mathbf{X}_t$, $\mathbf{Y}_t$, $\mathbf{Y'}_t$ and vectors $\mathbf{h}_t(\tilde{\mathcal{A}})$, $\mathbf{w}_t$ and $\mathbf{h'}_t(\tilde{\mathcal{A}})$. Figure 3(b) gives the complete matrix representation of the example template in Figure 3(a). This representation will prove useful when we discuss the parameterization of the probability distributions in PDM, next.

4.2 Parameterized probability distributions

In this section we parameterize probability distributions used in the PDM model of (1)-(3) to make various conditional design choices more or less probable.

4.2.1 Aesthetic priors

We use a multinomial distribution for the template probabilities. This gives us the flexibility of making certain templates more or less likely to be used in creating a document. Thus $\mathbb{P}_i(T_i = t_i) = p_{t_i} = 1/|\Omega_i|,\ \forall t_i$, if there is no *a priori* preference for a template.

We model the *prior* distribution of template parameters $\mathbb{P}(\mathbf{\Theta}|T)$ in equation (2) to be a multi-variate Normal distribution:

$$\mathbb{P}(\mathbf{\Theta}|T = t) \quad = \quad \mathrm{N}(\mathbf{\Theta}|\bar{\mathbf{\Theta}}_t, \mathbf{\Lambda}_t^{-1}) \tag{24}$$

where $\bar{\mathbf{\Theta}}_t$ is the mean of $\mathbf{\Theta}|T = t$. $\mathbf{\Lambda}_t$ represents the precision (inverse of covariance) matrix. For the rest of the paper we will use the shorthand $\mathbf{\Theta}_t$ for $\mathbf{\Theta}|T = t$ (the parameters of a particular template t [2]).

While the full multivariate distribution of equation (24) is quite general, it is hard for a designer to specify a full covariance matrix. It is easier to specify the means, variances (hence precisions), minimum and maximum values of each random parameter. This leads to a diagonal precision matrix $\mathbf{\Lambda}_{t_d} = diag[\lambda_{t1}, \lambda_{t2}, \cdots, \lambda_{tV}]$.

It is also desirable to consider linear relationships among parameters (ex: ratios) whose aesthetics should be encoded. Thus we consider general *soft* linear relationships expressed by

$$\mathbf{C}_t\mathbf{\Theta}_t - \mathbf{d}_t \quad \sim \quad \mathrm{N}(\mathbf{0}, \mathbf{\Delta}_{t_d}^{-1}) \tag{25}$$

[2] not to be confused with $\mathbf{\Theta}_i$ used in Section 3.2 which refers to the parameters sampled for the i^{th} page

for general $\mathbf{C}_t$ and $\mathbf{d}_t$ or equivalently, $\mathbf{\Theta}_t \sim \mathrm{N}(\bar{\mathbf{\Theta}}_t, \mathbf{\Lambda}_t^{-1})$, with

$$\mathbf{\Lambda}_t \quad = \quad \mathbf{C}_t^T\mathbf{\Delta}_{t_d}\mathbf{C}_t \tag{26}$$

$$\bar{\mathbf{\Theta}}_t \quad = \quad \mathbf{\Lambda}_t^{-1}\mathbf{C}_t^T\mathbf{\Delta}_{t_d}\mathbf{d}_t \tag{27}$$

Note that in the case $\mathbf{C}_t = \mathbf{I}$, we have $\bar{\mathbf{\Theta}}_t = \mathbf{d}_t$ and $\mathbf{\Lambda}_t = \mathbf{\Delta}_{t_d} = \mathbf{\Lambda}_{t_d}$, so this formulation is a more general than simple mean and diagonal precision specification. This representation is particularly useful in developing priors for image scaling and re-targeting as discussed in Section 4.2.2.

4.2.2 Incorporating image scaling and re-targeting

A template may have parameters Θ_w and Θ_h that describe the width and height of an image. However, these cannot vary independently. In this case we have

$$\Theta_w \approx a\ \Theta_h \quad , \quad \text{with precision } \rho \tag{28}$$

This is simply a linear relationship that may be handled by encoding it into the prior as described in Section 4.2.1. If we do not want to allow the aspect ratio of an image to change (in the case of pure image scaling) we let $\rho \to \infty$. On the other hand, we may use image re-targeting algorithms to allow the image aspect ratio to change. In this case the value of ρ will determine if we allow small (by setting ρ to a large value) or large (by setting ρ to a small value) changes in aspect ratio. We use the re-targeting algorithm described in [8] to generate the results presented in this paper.

4.2.3 Content allocation likelihood

The probability distribution that determines the likelihood of an allocation as described by equation (3) is represented by the following distribution

$$\begin{aligned}\mathbb{P}(\mathcal{A}|\mathcal{B}, \mathbf{\Theta}, T) \quad \propto \quad & \exp(-\gamma|R(\mathcal{A}, \mathcal{B})|) \times \\ & \mathrm{N}(\mathbf{w}_t|\mathbf{X}_t^T\mathbf{\Theta}_t, \alpha^{-1}\mathbf{I}) \times \\ & \times\ \mathrm{N}(\mathbf{h'}_t(\mathcal{A} - \mathcal{B})|\mathbf{Y'}_t^T\mathbf{\Theta}_t, \beta^{-1}\mathbf{I})\end{aligned} \tag{29}$$

In the above equation, $|R(\mathcal{A}, \mathcal{B})|$ represents the number of dangling references due to the allocation $\mathcal{A} - \mathcal{B}$ to the current page and $\mathcal{B}$ to the previous pages. The constant γ represents an exponential weighting factor that represents how much to penalize mismatched references in the probability distribution. For good page fill in the Y and X directions the heights and widths of all Y-path groups must satisfy $\mathbf{h'}_t(\mathcal{A} - \mathcal{B}) - \mathbf{Y'}_t^T\mathbf{\Theta}_t \approx \mathbf{0}$ and X-paths must satisfy $\mathbf{w}_t - \mathbf{X}_t^T\mathbf{\Theta}_t \approx \mathbf{0}$. The normal distributions above simply assign high probability to these events and lower probability for deviations from ideal. The constants α and β are precision parameters of the normal distribution with diagonal precision matrices ($\mathbf{I}$ is the identity matrix) that control the degree to which we want to produce full pages.

4.3 MAP estimation of template parameters

The particular parameterization of the prior and likelihood given above makes the computation of $\Psi(\mathcal{A}, \mathcal{B}, T)$ particularly efficient since the posterior distribution formed from the product of $\mathbb{P}(\mathbf{\Theta}|T)$ (24) and $\mathbb{P}(\mathcal{A}|\mathcal{B}, \mathbf{\Theta}, T)$ (29) is a multi-variate Normal distribution in $\mathbf{\Theta}_t$. With some algebraic manipulation (not shown here due to space constraints) we can compute the optimal MAP estimate in closed form by simply calculating the mean of this product posterior distribution. This results in the following closed form solution for $\mathbf{\Theta}_t^* = \mathbf{A}_t^{-1}\mathbf{b}_t$, where:

$$\mathbf{A}_t \quad = \quad \mathbf{\Lambda}_t + \alpha\mathbf{X}_t\mathbf{X}_t^T + \beta\mathbf{Y'}_t\mathbf{Y'}_t^T \tag{30}$$

$$\mathbf{b}_t \quad = \quad \mathbf{\Lambda}_t\bar{\mathbf{\Theta}}_\mathbf{t} + \alpha\mathbf{X}_t\mathbf{w}_t + \beta\mathbf{Y'}_t\mathbf{h'}_t(\mathcal{A} - \mathcal{B}) \tag{31}$$

In general, however, there may be bound constraints on some of the components of $\mathbf{\Theta}_t$ (ex: figure dimensions cannot be negative). To incorporate these constraints we solve the following bound-constrained least-squares quadratic program.

$$\mathbf{\Theta}_t^* = \underset{\{\mathbf{\Theta}_t : \mathbf{l} \leq \mathbf{\Theta}_t \leq \mathbf{u}\}}{\operatorname{argmax}} \left(\mathbf{A}_t\mathbf{\Theta}_t - \mathbf{b}_t\right)^T \left(\mathbf{A}_t\mathbf{\Theta}_t - \mathbf{b}_t\right) \quad (32)$$

where $\mathbf{l}$ and $\mathbf{u}$ are lower and upper bound vectors constraining $\mathbf{\Theta}_t$.

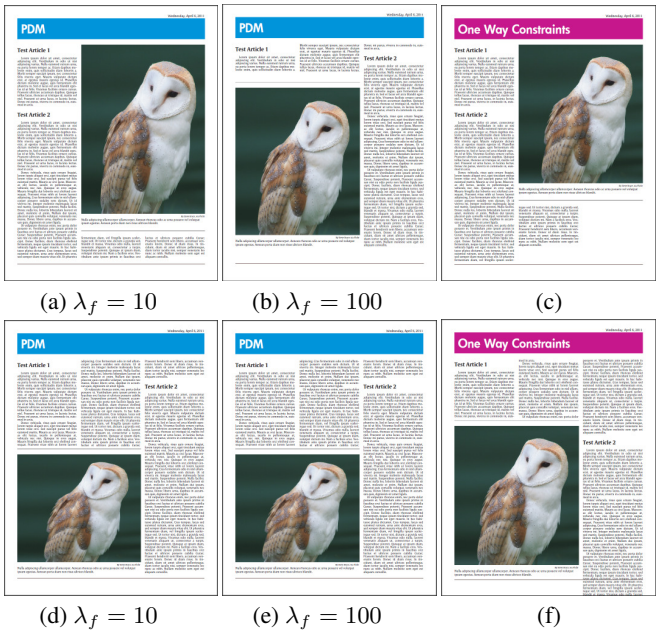
Note that our likelihood formulation of equation (29) attempts to ensure that the content fits the X-paths and the aggregate Y-path flows represented by ${\mathbf{Y'}}_t$. In an ideal world if text flow is continuous this would be all that is required. Y-paths $\mathbf{Y}_t$ were left out of this optimization because text flow across columns is discrete and it is unclear how text allocated to a page is to be distributed across columns. Using ${\mathbf{Y'}}_t$ instead of $\mathbf{Y}_t$ allows us to get around this issue by considering how the whole flow (aggregating across columns) fits the page. This approach of course implicitly assumes that the flow is continuous. This is not true for text, but is usually a good approximation.

Fortunately, in practice, we may refine this approximation since $\mathbf{\Theta}_t^*$ computed using equation (32) effectively converts the probabilistic template into a static template. This means that the text block sizes are now known, so we can distribute the text across the blocks. This may be done without rendering the text if we knew the number of lines to be allocated and the font size. This procedure will produce actual text block height estimates (taking into account discrete line breaks across columns). We can therefore re-compute $\mathbf{h}_t$ using the correct text heights for text blocks in every path and re-solve for $\mathbf{\Theta}_t$ using equation (32) with $\mathbf{Y}_t$ substituted for ${\mathbf{Y'}}_t$ and $\mathbf{h}_t(\mathcal{A} - \mathcal{B})$ substituted for ${\mathbf{h'}}_t(\mathcal{A} - \mathcal{B})$.

5. PRACTICAL CONSIDERATIONS

The input *raw* content to the document composition engine is an XML file. The XML elements match the element types that are allowed on a page template. In our examples we allow three element types including text blocks, figures and sidebars (a grouping of figures and/or text blocks that must appear together on a page). The XML file also encodes figure/sidebar references via an XML attribute indicating the id of the text block that referenced it. Each content XML file is coupled with a style sheet. Content blocks within the content XML have attributes that denote their type. For example, text blocks may be tagged as head, subhead, list, para, caption etc. The document style sheet defines the type definitions and the formatting for these types. Thus the style sheet may require a heading to use Arial bold font with a specified font size, linespacing etc. The style sheet also defines overall document characteristics such as, margins, page dimensions etc.

We use a GUI based authoring tool to author templates and style sheets. Note that the template library design task is fixed overhead cost. The design process a graphic designer must go through to create a template includes a) content block layout, b) specification of linked text streams and c) specification of prior probability distributions (mean-precision and min-max). For images the designers set min-max of height/width and the precision of the aspect ratio (we use the native aspect ratio as the mean). Note that it is the relative values of variances (precisions) that matter. So a designer could use order of magnitude precision changes to indicate preference. For example if a whitespace precision is set to 100, setting figure aspect precision to 10 would allow figure aspect ratio to change much more freely than whitespace. We automatically compute overall prior of (24) using equations (26) and (27). All other distributions and probabilities are calculated computationally without designer involvement.



(a) $\lambda_f = 10$ (b) $\lambda_f = 100$ (c)

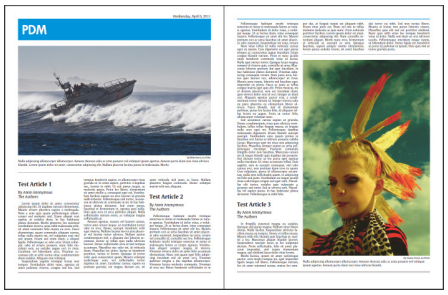
(d) $\lambda_f = 10$ (e) $\lambda_f = 100$ (f)

Figure 4: Soft probabilistic encodings vs. one way constraints. ***Owl: ©Steve Brace (http://www.flickr.com/photos/steve_brace/)***

To run PDM inference we need to generate several discrete content sets $\mathcal{A}$ and $\mathcal{B}$ from the input content. Since figures and sidebars cannot break across pages, it is straightforward to allocate them to sets. A single text block in the content stream may be chunked as a whole if it cannot flow across columns or pages (ex: headings, text within sidebars etc.). However if the text block attribute indicates that it is allowed to flow (paragraphs, lists etc.), it must be decomposed into smaller atomic chunks for set allocation. We use a line based text chunk decomposition in our experiments. We assume that text width is selected from a discrete set of widths allowed for a template library. The height of a chunk is determined by rendering the text chunk at all possible text widths using the specified style sheet style in a preprocess rendering pass. We use the open source desktop publishing software Scribus as the rendering engine and are able to query the number of rendered lines via an API. We use the number of lines and information regarding the font style, line spacing etc. to calculate the rendered height of a chunk. Thus when allocating a text chunk to a text element of a template, we may simply look up its height using the chunk index and template text element width. We can do this if text column widths are known and text is not allowed to wrap around figures. The choice of lines as the unit of text allocation restricts the class of layouts to multi-column layouts with no text wrapping around figures, since when wrap-around is allowed a paragraph may break into different number of lines depending on the extent of wrap-around and so would not be a reliable unit of allocation. Since our specific template parameterization also does not allow text wrap-around, line based allocation is an efficient choice. This is however not a restriction on the generality of the PDM model.

6. EXPERIMENTAL RESULTS

In this section we illustrate the performance PDM based document composition with example 1 and 2-page document compositions (due to space restrictions) shown in Figures 4 and 5. Our template library is designed for 2-page News content with multiple articles in a 3-column format. Each article has at most one figure associated with it and must appear naturally adjacent to article text.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5: PDM compositions with variations in content and form factor. Each row has exactly the same content with variation only in form factor. (a) and (b) show three articles composed onto two pages. (c) and (d) have a new article inserted between the original articles 2 and 3. (e) and (f) have a new figure with a different aspect ratio substituted for the original figure in article 3. Our algorithm actively paginates content, selects content arrangements (templates), re-targets images, re-flows text and manages whitespace to fit the content in each case. The same template library is used for both form factors. Since we only encode relative position in templates we only need to set H and W to the appropriate values for the new form factor. *Punk girl and Veiled girl: ©D Sharon Pruitt (http://www.flickr.com/photos/pinksherbet/) , Boat: ©Mike Baird (http://www.flickr.com/photos/mikebaird/)*

Figure 6: First six pages of the HP Labs 2010 annual report generated using PDM inference. The entire report can be downloaded from http://www.hpl.hp.com/news/2011/jan-mar/annual_report.html

Our template library consists of a total of 133 probabilistic templates including 83 first page and 50 back page templates encoding various relative arrangements of figures, article titles, and text streams. Since the design intent was to have images span columns, only the height of the images was parameterized with a mean equal to the height that maintained aspect ratio. Most whitespace was parameterized with a desired mean of 8 points between text and margins and 16 points between text and figures. Whitespace precision was set to 100 points^{-2}. The precision parameters for the likelihood function in equation (29) are set as follows, $\alpha = \beta = 1000$ and $\gamma = 10000$. These settings reflect our desire to produce full pages and force figures to appear on the same page as their corresponding articles (since the corresponding probability distributions have a tight spread).

Fig. 4 illustrates the value of soft probabilistic encodings vs. hard one way constraint based specification (ex: [7]). To simulate a constraint-based template evaluation system, we set the parameter values of all the templates in our library to their mean values (most aesthetically desired values). This gives us a set of *static* templates that can be interpreted using one-way constraints. When a figure is flowed into an image copy hole, the height of the hole is adjusted to maintain the aspect ratio, and the text below it is constrained as a result. We evaluate the posterior probability of the mean parameter settings and use this to score templates.

Each row of Fig. 4 shows compositions of the same content. The second row has more content than the first row. Each PDM template has precision matrix of (24) set to $\mathbf{\Lambda}_t = diag[\lambda_f, \lambda_w, \cdots, \lambda_w]$. The precision of whitespace, $\lambda_w = 100$ for all the PDM compositions. Along each row for PDM, the precision of the figure height λ_f is varied from a small value $\lambda_f = 10$ to a value equal to the whitespace precision $\lambda_f = 100$. Thus the PDM compositions (a) and (d) use mainly figure height (hence, aspect ratio) changes to fit content since it has a low precision relative to whitespace variation. The compositions (b) and (e) allow both figure aspect ratio and whitespace to vary from nominal desired values to fit content since their precisions are the same [3]. While changes in image aspect ratio and whitespace lower the contribution of prior parameter probability to the overall document probability (since they deviate from desired mean values), this impact is offset by relatively higher contributions for pagination or content fit. Of course, if possible, the algorithm will attempt to preserve the mean values of all quantities. In contrast, One one way constraints are much more rigid, not allowing figure aspect ratio and whitespace to change from desired value to fit content. This often results in voids (c) or overflows (d). One approach to add more flexibility to one way constraint methods is to allow several alternate versions of images and/or filler images [7]. Even if alternates were allowed, since each image has a discrete aspect ratio, it is very easy to make content by adding/deleting lines where voids and overflows occur. In general, one way constraint based composition requires huge template libraries so that a good sequence of templates can be found in the library [14]. For PDM compositions, since the templates are actively adjusted to fit content, consistently good results can be achieved even with modest template libraries.

Fig. 5 illustrates 2 page document compositions with variations in content along a column and form factor along a row. Note how we are able to actively paginate content, choose appropriate arrangements and actively re-target images to fit the content. In general our parameter prior favors aspect ratio changes to changes in whitespace since $\lambda_f = 10$ in this case.

As a real world example, we also used the PDM inference algorithm to create the 64 page HP Labs 2010 annual report using a template library of 37 templates. Figure 6 shows the first six pages of the report. The report includes richer formatting including sidebars and profile pictures with text overlay. Changes in image aspect ratio were not allowed in this case, so we had to parameterize whitespace precision around images to allow more flexible image resizing. A high emphasis was placed on referenced ($\gamma = 10000$) figures and sidebars occurring on the same spread as their text reference. Note how the algorithm automatically resizes the images to allow the figures to appear close to their references. While the same 1-figure template was chosen for pages 4 and 6, the algorithm automatically scaled down the image (allowing modest whitespace on its left and right) on page 4 to allow it to be close to the *HP Labs Singapore* heading.

Our code for these experiments was written in MATLAB (unoptimized for efficiency and for-loops). The longest 2-page content consisting of 171 lines of text and two figures took around 45 seconds while the report (1719 lines of text, 14 floating figures and 38 floating sidebars) took around 1 hour. Experiments were performed on a Linux machine with single core 3GHz Intel Xeon processor with 3GB of RAM. We expect that an order of magnitude performance improvement can be achieved with optimized C code.

[3](d) and (e) produced the same composition since the whitespace in (e) could not be further reduced since its min value was reached

7. CONCLUSIONS

This paper presented a probabilistic framework for adaptive document layout that supports automated generation of paginated documents for variable content. We attempted to address one of the main weaknesses of template based automated document composition, the fact that one-way constraint-based templates can be overly rigid. Our approach uniquely encodes soft constraints (aesthetic priors) on properties like whitespace, exact image dimensions and image rescaling preferences. Our main contribution is a probabilistic document model (PDM) that combines all of these preferences (along with probabilistic formulations of content allocation and template choice) into a unified model. In addition, we describe a model inference algorithm for computing the optimal document layout for a given set of inputs. PDM opens the door to leveraging probabilistic machinery and machine learning algorithms for both inference and learning of parameterizations directly from examples. This may reduce if not eliminate the need for designer involvement in creating templates and specifying prior distributions.

8. REFERENCES

[1] G. J. Badros, A. Borning, and P. J. Stuckey. The cassowary linear arithmetic constraint solving algorithm. *ACM Trans. Comput.-Hum. Interact.*, 8(4):267–306, 2001.

[2] G. J. Badros, J. J. Tirtowidjojo, K. Marriott, B. Meyer, W. Portnoy, and A. Borning. A constraint extension to scalable vector graphics. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 489–498, New York, NY, USA, 2001. ACM.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[4] A. Brüggemann-Klein, R. Klein, and S. Wohlfeil. On the pagination of complex documents. In *Computer Science in Perspective: Essays Dedicated to Thomas Ottmann*, pages 49–68, New York, NY, USA, 2003. Springer-Verlag New York, Inc.

[5] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in*

information retrieval, pages 50–57, New York, NY, USA, 1999. 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] C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin. Adaptive grid-based document layout. *ACM Transactions on Graphics*, 22(3):838–847, Jul. 2003.

[8] Z. Karni, D. Freedman, and C. Gotsman. Energy-based image deformation. In *Proceedings of the Symposium on Geometry Processing*, SGP '09, pages 1257–1268, Aire-la-Ville, Switzerland, Switzerland, 2009. Eurographics Association.

[9] X. Lin. Active layout engine: Algorithms and applications in variable data printing. *Comput. Aided Des.*, 38(5):444–456, 2006.

[10] 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.

[11] J. Lumley, R. Gimson, and O. Rees. A framework for structure, layout & function in documents. In *DocEng '05: Proceedings of the 2005 ACM symposium on Document engineering*, pages 32–41, New York, NY, USA, 2005. ACM.

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

[13] M. F. Plass. *Optimal pagination techniques for automatic typesetting systems*. PhD thesis, Stanford University, Stanford, CA, USA, 1981.

[14] E. Schrier, M. Dontcheva, C. Jacobs, G. Wade, and D. Salesin. Adaptive layout for dynamically aggregated documents. In *IUI '08: Proceedings of the 13th international conference on Intelligent user interfaces*, pages 99–108, New York, NY, USA, 2008. ACM.

[15] B. T. Vander Zanden, R. Halterman, B. A. Myers, R. McDaniel, R. Miller, P. Szekely, D. A. Giuse, and D. Kosbie. Lessons learned about one-way, dataflow constraints in the garnet and amulet graphical toolkits. *ACM Trans. Program. Lang. Syst.*, 23(6):776–796, 2001.

APPENDIX

A. INFERENCE COMPLEXITY

The forward pass given by equations (21)-(23) dominates the asymptotic complexity of PDM inference. A naive computation of the table $\Psi(\mathcal{A},\mathcal{B},T)$ has asymptotic complexity $O(|\Omega||c'|^2)$ (where $|\Omega| = |\bigcup_i \Omega_i|$) since we must loop over all sets $\mathcal{A},\mathcal{B} \in c'$ with $\mathcal{A} \supseteq \mathcal{B}$ and over all $|\Omega|$ distinct templates in the library. Recall that c' is the set of all legal content allocations (Section 3.1).

We observe that we can effectively bound the set $\mathcal{A}-\mathcal{B}$ that represents the content allocated to a page. This assumption implies that we do not need to loop over all legal subsets $\mathcal{A}$ and $\mathcal{B}$ in building $\Psi(\mathcal{A},\mathcal{B},T)$, but only those that are *close enough* so that the content $\mathcal{A}-\mathcal{B}$ can reasonably be expected to fit on a page. In general, for each $\mathcal{A}$ the allowed $\mathcal{B}$'s are in a neighborhood $\mathcal{N}_{\mathbf{f}}(\mathcal{A}) = \{\mathcal{B} : \boldsymbol{\delta}(\mathcal{A}-\mathcal{B}) \leq \mathbf{f}\}$. The function $\boldsymbol{\delta}(\mathcal{A}-\mathcal{B})$ returns a vector of the counts of various page elements in the set $\mathcal{A}-\mathcal{B}$. $\mathbf{f}$ is a vector that expresses what it means to be *close* by bounding the numbers of various page elements allowed on a page. For example we may set $\mathbf{f} = [100\,(\text{lines}), 2\,(\text{figures}), 1\,(\text{sidebar})]^{\mathrm{T}}$. This will eliminate an allocation comprising 110 lines of text and 1 sidebar. This page capacity bound improves the complexity of computing $\Psi(\mathcal{A},\mathcal{B},T)$ to $O(|\Omega||c'|)$.

Once the table $\Psi(\mathcal{A},\mathcal{B},T)$ has been computed, the computation of $\Phi_i(\mathcal{A},\mathcal{B})$ for each i is $O(|\Omega_i||c'|)$ since the maximization over templates only occurs over a sub-library containing $|\Omega_i|$ templates. The computation of $\Phi_i(\mathcal{A},\mathcal{B})$ for all i is thus $O(\max_i(|\Omega_i|)|c'|\hat{I})$, where $\hat{I}$ is the estimated page count for the forward pass. Since $\hat{I} \propto |c|$ this complexity may also be expressed as $O(\max_i(|\Omega_i|)|c'||c|)$.

Once all the tables $\Phi_i(\mathcal{A},\mathcal{B})$ have been computed, we recursively compute $\tau_i(\mathcal{A})$ for *all* $\mathcal{A}, i$ with $\tau_0(\mathcal{A}) = \Phi_0(\mathcal{A},\emptyset)$. This computation has complexity $O(|c'||c|)$ since we loop over every $\mathcal{A}$ and all pages. Thus, overall asymptotic algorithm complexity of PDM inference (under the mild assumption that page capacity is bounded) is $O(|\Omega||c'||c|)$ or $O(|\Omega||c|^3)$, since[4] $|c|^2 > |c'|$.

If we make a further assumption that $\Phi_i(\mathcal{A},\mathcal{B})$ is independent of i (i.e. templates for all pages are drawn from the same template library) then $|\Omega_i| = |\Omega|$ and $\Phi_i(\mathcal{A},\mathcal{B}) = \Phi(\mathcal{A},\mathcal{B})$. Thus the computation of $\Phi(\mathcal{A},\mathcal{B})$ now has complexity $O(|\Omega||c'|)$. This does not change overall asymptotic complexity since the computation of $\tau_i(\mathcal{A})$ for *all* $\mathcal{A}, i$ is still $O(|c'||c|)$ or $O(|c|^3)$.

However, if we are seeking to automatically determine optimal page count we only need to compute and store $\tau(\mathcal{A}) = \max_{i\geq 0}\tau_i(\mathcal{A})$ since, from (15), we have

$$\begin{aligned}\max_{i\geq 0}\tau_i(c) &= \max_{i\geq 1}\left\{\tau_0(c), \max_{\mathcal{B}}\Phi(c,\mathcal{B})\tau_{i-1}(\mathcal{B})\right\} \\ &= \max\left\{\tau_0(c), \max_{\mathcal{B}}\Phi(c,\mathcal{B})\underbrace{\max_{i\geq 0}\tau_i(\mathcal{B})}_{\tau(\mathcal{B})}\right\}\end{aligned}$$

If we allow $\mathcal{B} = \emptyset$ above (so that $\tau_0(c) = \Phi(c,\emptyset)$), define $\tau(\emptyset) = 1$ and substitute $\mathcal{A}$ for c we have the much more succinct general recursion for the computation of $\tau(\mathcal{A})$ and page count $Pg(\mathcal{A})$

$$\tau(\mathcal{A}) = \max_{\mathcal{B}}\Phi(\mathcal{A},\mathcal{B})\tau(\mathcal{B}) \quad (33)$$

$$\mathcal{B}^* = \operatorname*{argmax}_{\mathcal{B}}\Phi(\mathcal{A},\mathcal{B})\tau(\mathcal{B}) \quad (34)$$

$$Pg(\mathcal{A}) = 1 + Pg(\mathcal{B}^*) \quad (35)$$

with $Pg(\emptyset) = 0$. Thus the complexity of computing $\tau(\mathcal{A})$ for *all* $\mathcal{A}$ is $O(|c'|)$. Overall algorithm complexity now becomes $O(|\Omega||c'|)$ or $O(|\Omega||c|^2)$. In fact, we can generalize these asymptotic results even more, to support the common case that templates for even and odd pages are drawn from distinct libraries. In this case we need to compute and store $\Phi_{odd}(\mathcal{A},\mathcal{B})$, $\Phi_{even}(\mathcal{A},\mathcal{B})$, $\tau_{odd}(\mathcal{A})$ and $\tau_{even}(\mathcal{A})$. The additional computational complexity does not grow with content and so has no impact on asymptotic inference complexity.

Finally, if we make the assumption that content follows a linear ordering (i.e. text and figures are organized and allocated within a single flow order), then $|c'| = |c| + 1$ (the +1 is to include the empty set). This assumption implicitly means that a figure that appears after a text block must be allocated immediately after it. In practice this forces figures to appear on the same page or on the next page instead of allowing them to float freely. This extra assumption means that the best case complexity of PDM inference is $O(|\Omega||c|)$. Thus in the case of linear content ordering, finite sub-libraries and bounded page capacity, the task of infering the optimal document is linear in content and the size of the template library.

[4] $|c|^2 = (|c_l| + |c_f|)^2 > |c'| = (|c_l| + 1)(|c_f| + 1)$ as $|c| \to \infty$