Multimed Tools Appl (2013) 62:209–231
DOI 10.1007/s11042-011-0980-1

Adapting web pages using graph partitioning algorithms for user-centric multi-device web browsing

Jochen Huber · Yun Ding

Published online: 10 January 2012

Abstract In the era of ubiquitous computing, applications are emerging to benefit from using devices of different users and different capabilities together. This paper focuses on user-centric web browsing using multiple devices, where content of a web page is partitioned, adapted and allocated to devices in the vicinity. We contribute two novel web page partitioning algorithms. They differ from existing approaches by allowing for both, automatic and semi-automatic partitioning. On the one hand, this provides good automatic, web page independent results by utilizing sophisticated structural pre- and postprocessing of the web page. On the other hand, these results can be improved by considering additional semantic information provided through user-generated web page annotations. We further present a performance evaluation of our algorithms. Moreover, we contribute the results of a user study. These clearly show that (1) our algorithms provide good automatic results and (2) the application of user-centric, annotation-based semantic information leads to a significantly higher user satisfaction.

Keywords Multi-device web browsing · Web page partitioning · Partitioning algorithms · Web page annotation · Mobile and ubiquitous multimedia

1 Introduction

During the last decades an increasing diverse range of mobile devices, personal computers, intelligent home and office appliances, as well as shared public devices

J. Huber (✉)
Technische Universität Darmstadt, Hochschulstraße 10 64289 Darmstadt, Germany
e-mail: jochen.huber@acm.org

Y. Ding
RIB Software AG, 9 Temasek Boulevard, 31st Floor Suntec Tower Two,
Singapore 038989, Singapore
e-mail: yun.ding@ribitwo.com

have been reaching the mass market. Each device is specialized for one or multiple usage scenarios, and they differ significantly from each other in its computing, networking and I/O capabilities. Borrowing survival mechanisms from biology, symbiotic environments suggest using devices of different specializations and capabilities together to overcome the limitations of single ones [25, 26]. Apart from complementing the functionalities of multiple devices, using them together facilitates the collaboration within groups and communities or with the public. In mobile and ubiquitous environments, users move and change their location, and interact with different people and a changing set of devices. In addition to the diversity of devices, user interfaces of these applications have to take privacy, psychological and social aspects into account. The following scenario exemplifies these challenges.

Mike arranged to go out with his friends after work. On his way, he uses his smartphone to browse the urban information portal for parties and cinemas. Standing in front of an internet-enabled public information terminal, Mike views the map showing the events' locations on the large screen of the terminal. Simultaneously, he interacts with his smartphone to navigate through the information. Mike's friends Alice and Tom have joined him. In order to evaluate the events in parallel, Mike moves the event list to the smartphones of Alice and Tom. After a while, the friends move the description of their favored pubs and cinemas to the screen of the terminal. They jointly make a decision, and Mike volunteers to book movie tickets for all. The login page of his bank is displayed on Mike's smartphone, since his private, sensitive information is involved.

In this paper, we contribute two novel web page partitioning algorithms which support user-centric web browsing of existing web applications as described in our scenario above. Our algorithms differ from existing approaches by allowing for both, automatic web page independent partitioning and semi-automatic partitioning. To the best of our knowledge, this is the first approach which combines both techniques. On the one hand, our algorithms exploit structural information of the web page contained in its Document Object Model (DOM). Mathematically, the problem of splitting a web page into different, not necessarily complementary parts for the different devices can be reduced to the problem of graph partitioning (i.e. partitioning the DOM tree). On the other hand, with the emphasis on user-centric multi-browsing, our algorithms utilize additional semantic information provided by user-generated annotations. This is motivated by our previous user study [11] that confirms the significant difference between annotations from different users even for the same scenario.

The remainder of this paper is structured as follows. At first, we define requirements for architectures supporting multi-browsing and describe related work. Then, we briefly introduce our architecture and our annotation vocabulary for the self-containedness of this paper. An extensive description thereof can be found in our previous publication. Afterwards, we present our main contribution of this paper, two novel graph partitioning algorithms for multi-browsing and show how annotations can be incorporated into the underlying mathematical framework. Moreover, we contribute the evaluation results. Finally, we sum up and point out potential future work.

2 Requirements & related work

There is a considerable body of research on collaborative, multi-device (web) applications [18, 24, 25] and usage [20]. Most of the proposed frameworks demonstrate

their feasibility with one or several specially designed scenarios. Either significant redesign of the framework is necessary to apply it to another scenario, or it is difficult to evaluate whether a framework is appropriate to support another scenario.

In order to overcome this limitation, this paper focuses on (semi-)automatic partitioning algorithms for multi-browsing applications. The main difference between our approach and existing work is motivated by our observation that different people prefer different ways of displaying information in symbiotic environments. Our previous user study [11] has confirmed how opinions about semantically meaningful and visually appealing web page distributions differ among users. Hence, we set the following requirements for our architecture for multi-browsing applications:

- *User-centric experience:* It is important to provide a flexible architecture which allows both developers and end users to express their different preferences, for example by annotating existing web pages. In ubiquitous environments in which the set of devices in the vicinity and situations dynamically change, the architecture must allow end-users to dynamically change their preferences.
- *Underlying partitioning algorithm:* It is vital that the partitioning algorithms take the annotated information into account to enable user-centric experiences. Nevertheless, it can not be assumed that every web page is annotated. Consequently, the partitioning algorithm shall be able to utilize heuristics to produce reasonable results even without the annotations–*fully automatically*.

Probably closest to our approach is the one by Maekawa et al. [22, 23]. They propose a system for collaborative web browsing, supporting multiple mobile users. They have developed a partitioning algorithm which is based on the prominent FM algorithm [13]. However, their algorithm relies on a pre-defined grouping of web page elements. The grouping has to be generated *manually* in advance for every web page which is to be browsed collaboratively. This system requires expert knowledge about the web page and does not support a user-centric collaboration, which is an important requirement for our approach. Moreover, this system focuses only on the interaction between mobile devices. Neither adaptation for larger displays nor multimodal adaptation is considered.

Another approach focusing on distributing web application user interfaces across multiple devices is WebSplitter [16]. It splits a web page among multiple devices of multiple users. Additionally, it enables presenting different partial views of the web page to different users. A separately required, manually generated annotation file defines the rules for the splitting. This file also consists of information (e.g. passwords) to perform user authentication. In our opinion, the intermingling of different concerns (e.g. access control and adaptation) makes the annotation file complex, and inflexible. Moreover, it seems that WebSplitter is specially designed for one particular scenario in which a teacher presents a web-based lecture to a group of students as opposed to our approach which is scenario independent.

Multibrowsing [19] is a system which allows moving web pages among multiple displays. Its main drawback is the lack of distinction between private and public artifacts, although both public and private displays have been explicitly mentioned. Additionally, there is no concept to adapt the web pages in order to take the different capabilities and constraints of the devices into account. Web pages can only be completely moved from one device to another.

Alapetite [2] explored a different approach for moving web pages across multiple devices. He proposed a system for web session migration, relying on visual 2D barcodes. Once a user wishes to continue browsing on a different device, a 2D bar code containing the session information is generated and displayed on screen. The code can be directly captured using e.g. a mobile phone's camera. The web browser of the new device then resumes the browsing session and a user can continue browsing. While this is perfectly suited for a sequential "device handover", the system does not focus on using multiple devices together.

A more recent work by Wiltse and Nichols [28] adopts a rather task-centered perspective on collaborative web browsing. Their PlayByPlay system records a user's browsing activity and communicates this in natural language over an instant messaging channel to other collaborators; e.g. "Bob clicked on the 'search' button" [28]. The collaborating users can thence replay actions on their own device. Moreover, users can exchange clipped web page snippets over the very same messaging channel. The overall system however does not focus on using multiple devices together, leveraging for instance the capabilities of large public displays.

Similar to PlayByPlay, UsaProxy [4] supports synchronous remote web collaboration between two users in different locations. Here, users share a browsing session and both users have full control over their browsers. The focus of UsaProxy is the synchronization of the views seen by the users.

CoSearch [3] is a system supporting co-located collaborative web search. The system primarily targets desktop computers. Here, multiple users can concurrently interact with the system. Each user has her own mouse. Additionally, users can pair their mobile phones with the desktop computer, allowing them to use the mobile phones as a mouse replacement. Users can also continue with the web search on their own using their mobile devices. For this purpose, the system allows users to move just the search results to their mobile phones.

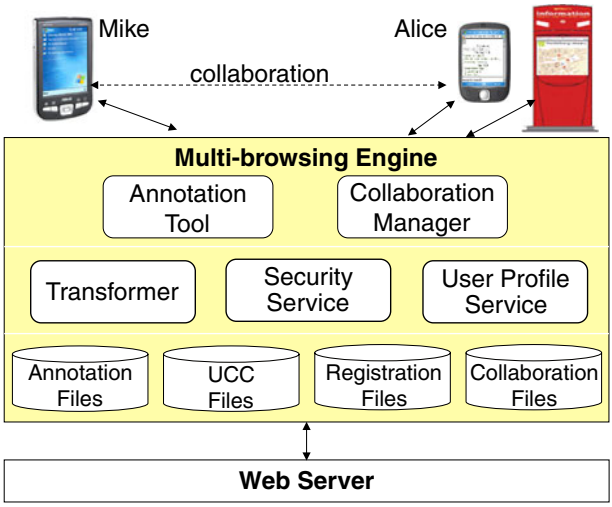
Co-located collaborative web search was also investigated in WebSurface [27]. Tuddenham et al. proposed using a tabletop interface for collaborative interaction. Their exploratory study revealed potential for a better information layout than when two collaborators each have their own laptop or even share the same. They did not investigate using other devices together with a tabletop for multi-device web browsing.

Hattori et al. [17] propose a hybrid segmentation method for a single device. Their algorithm utilizes structural information (e.g. content-distance) of a web page, and page layout information. The aim is to generate visually appealing and usable web layouts for mobile browsers. Supported by our previous user study, we doubt that the generated layouts could be appreciated by a majority of users without considering additional semantic information as well as user-specific preferences.

3 Architecture

Our architecture for multi-browsing applications is illustrated in Fig. 1. The multi-browsing engine acts as a proxy lying between the web browsers of multiple devices and the web server. It forwards a browser request to the web server, intercepts the fetched response, and returns a multi-browsing enabled web page to the user.

Fig. 1 Architecture



Consequently, users have to configure their browser to use the proxy in order to be able to use the multi-browsing application.

In the current implementation, the proxy dynamically injects JavaScript files into the web-page fetched from the web server. On the one hand, the JavaScript files insert a new toolbar (see Fig. 2) to support annotation, collaboration and registration of users and their devices. On the other hand they interact with the components of the multi-browsing engine. To allow for asynchronous interaction, AJAX [29] is used on the client side, whereas Reverse AJAX [12] is employed on the server side. Four databases are used to persistently store a) annotation files of the annotated web pages, b) user created content, c) registered users and devices, and d) registered collaborations respectively.

3.1 Core components

The *Annotation Tool* allows users to add, modify or remove annotations for the currently browsed web page (see Fig. 3). Furthermore, it is able to generate a preview which simulates the splitting of a web page using the added annotations. Hence, users are able to study the impact of their annotations on a potential web page distribution. Section 4 provides an overview over the available annotations.

Collaborations among users and their devices can be created or terminated using the *Collaboration Manager* (see Fig. 4). The menu in Fig. 4 shows an active, ongoing collaboration between John and Michael. There are two plus buttons beside Tom. With the left plus button, Michael can create a new collaboration together with Tom. With the right button, Michael can ask Tom to join the currently active collaboration with John. Whenever interaction spaces are to be composed, members in the ongoing collaboration need to permit the inclusion of new interaction spaces and decide on the relationship between the new and existing interaction spaces. Whenever an interaction space has been invited to a collaboration, the *collaboration manager* component of the architecture also asks its owner if she wants to join.

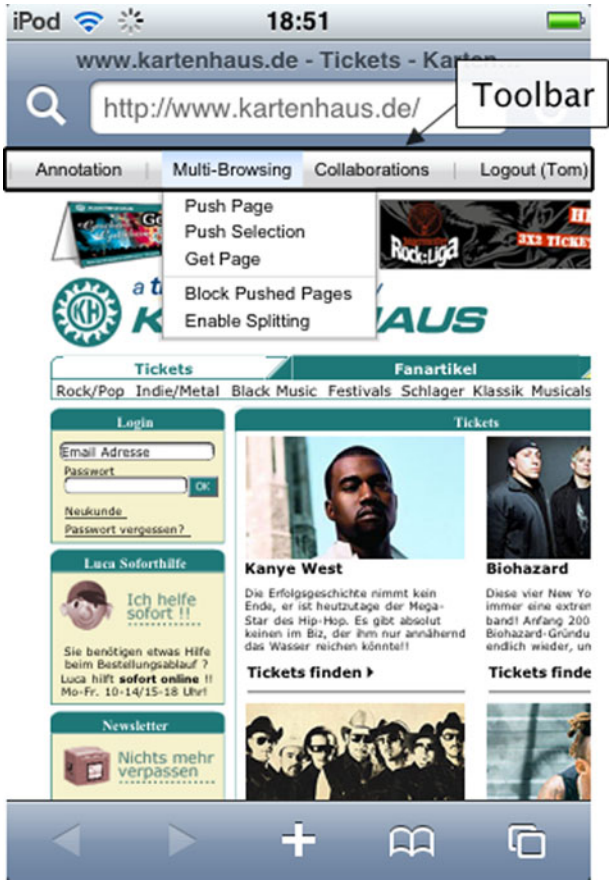


Fig. 2 Multi-browsing menu on the iPhone OS

The multi-browsing menu facilitates *collaboration requests* (see Fig. 2). Using a collaboration request, Michael can, for example, push the whole web page he is viewing or only a selected part of it to John's device. Alternatively, John is able to actively pull the web page on Mike's device to his own device.

The Collaboration Manager utilizes the service components to process collaboration requests.

3.2 Service components

The *Transformer* component is responsible for adapting and splitting the web page. It applies the graph partitioning algorithms which are the focus of this paper. When splitting a web page, the transformer decides *where* to place a certain content. For example, it avoids placing private annotated content to a public interaction space. If a part of the information is of private category, its movement to any target device not owned by the requesting user will potentially reveal a secret. The transformer warns by initiating the dialog "This may disclose your private information. Do you want

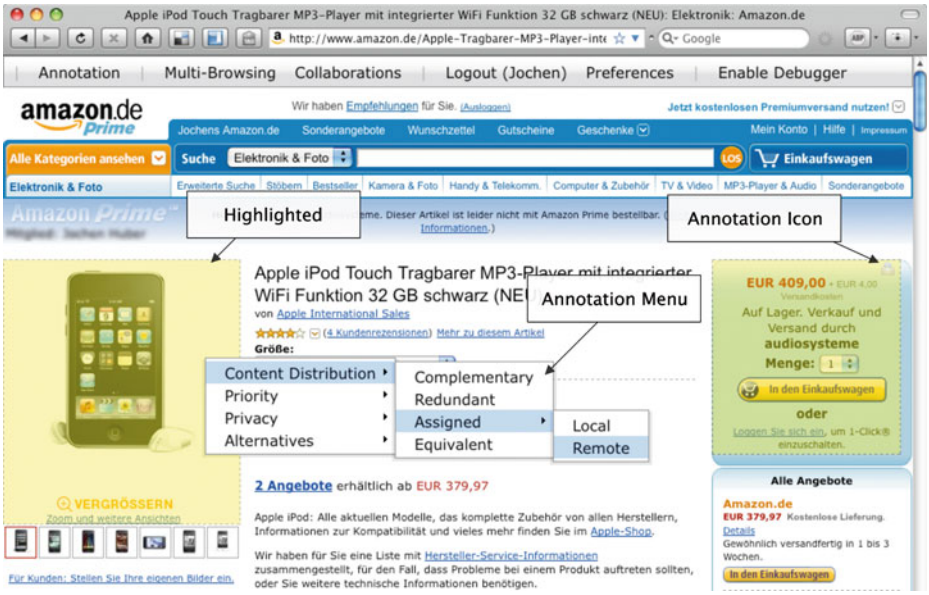


Fig. 3 Screenshot of the multi-browsing environment: the dynamically injected toolbar at the top allows users to use both, annotation and multi-browsing tools. Moreover, available annotations are highlighted

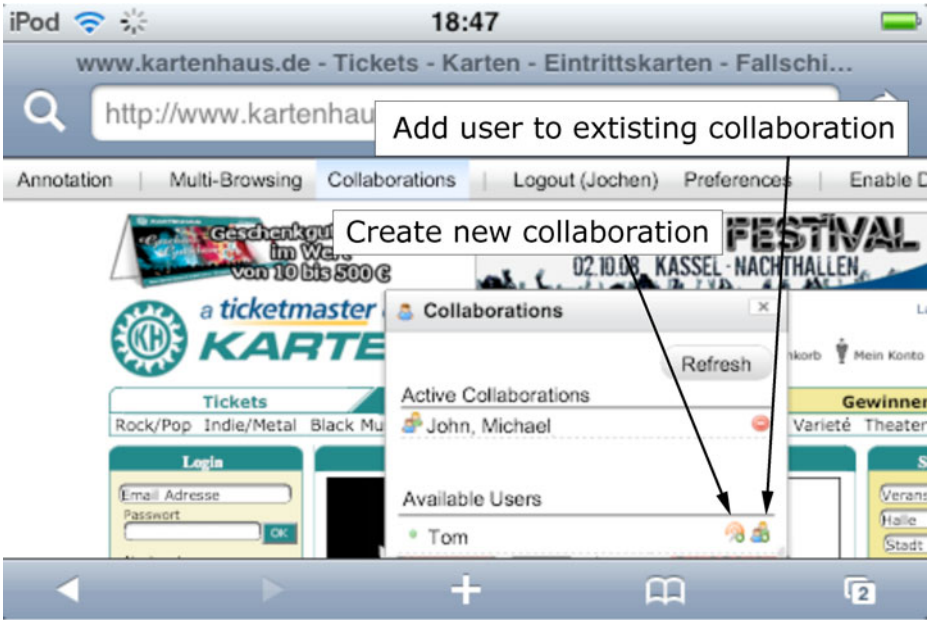


Fig. 4 Collaboration Manager on the iPhone OS

to proceed?". In this way the user has got the opportunity to dynamically adapt the privacy level of her information.

The *Security Service* component is responsible for identifying and warning the users of potential privacy violations. Whenever people use social or public devices to access private information, or use public devices to access social information, privacy will be violated. Displaying information on a device consumes its I/O resources. Hence, when moving web content to a non-public device, its owner must grant the access. The security service explicitly asks the user for permission.

The *User Profile Service* maintains user preferences. For example, a users prefers the audio instead of the visual modality. Then the service determines whether a piece of information exists in alternative representations and their locations. For instance, the desired information in an audio format could be in the database of the user created content.

4 Annotation

Web pages are usually rendered using pure (X)HTML by a web browser. Hence, heuristics relying solely on a web site as-is can only benefit from information encoded either explicitly or implicitly in the DOM. Further semantic information can be extracted from the structure of a web page by applying rule-based heuristics. Systems based on this technique typically rely on an unmanageable set of rules [7, 14]. Whether the result of a web page partitioning, adaptation and distribution is visually appealing lies in the eye of the beholder and is therefore highly subjective. Heuristics are for instance unable to reliably decide whether content elements of a web page shall be displayed exclusively on a particular device or redundantly on every participating device. This problem can be addressed by manually adding further semantic information through web page annotations which can then be utilized by the corresponding heuristics (see the next section). Our previous user study [11] shows the significant difference between annotations from different users for the same scenario.

In addition to information about the semantic structure of a web page such as the atomicity, grouping and ordering of information, we exploit semantic information which is concerned with the mapping between the information embedded in a web page and the collaborating devices rendering the information. The latter category concerning the mapping is particularly important for multi-browsing. Table 1 lists our annotation vocabulary for the second category which are now described in more detail.

The CARE (**C**omplementarity, **A**ssignment, **R**edundancy, and **E**quivalence) properties were originally introduced to map a piece of information to a set of

Table 1 Annotation vocabulary

Type	Vocabulary
Content	Complementary, redundant,
Distribution	Assigned (local, remote), equal
Priority	Optional, recommended, mandatory
Privacy	Public, social, private
Alternatives	Add, modify and delete content

modalities [5]. We use them to describe how content is being distributed among the participating devices.

- Redundancy and complementarity: When a piece of information is to be displayed repeatedly on various devices, it can either be displayed in the same way on every device (*Redundancy*) or adaptively, depending on the devices' characteristics (*Complementarity*). For example, a static map can be provided for mobiles in parallel to JavaScript-enabled maps suitable for capable devices, or voice output serves as an alternative to textual output for multi-modal web pages.
- Assignment and equivalence: When selecting a device in an ensemble to display a piece of information, the hardware and software requirements for the devices to properly render the information must be taken into account. A piece of information must be assigned to a certain device (*Assignment*), since others in the ensemble do not fulfill these requirements, or all the devices in the ensemble offer equivalent choices for the rendering (*Equivalence*).

Information can be prioritized using three different importance categories: *optional*, *recommended* and *mandatory*. Content categorized as mandatory will be displayed under any circumstances, even on tiny devices like mobiles. Content of the category recommended will preferably be displayed. However, content marked as optional will only be displayed, if there is sufficient space, e.g. when desktop computers or larger devices are available.

Our annotation vocabulary also provides means to describe the privacy levels of web page elements and of participating devices. The spectrum of publicness is subdivided into three regions [9]: *public*, *social* and *private*. In real life, the boundaries between private, social and public are neither rigid nor fixed. People fluidly shift their artifacts from personal to public and the many shades in between [15]. We allow users to individually annotate the privacy levels. Moreover, users can dynamically adapt the privacy levels by interacting with the security service of our architecture.

Alternatives (e.g. the same content presented in different modalities) can be created both by the developers and users. In the latter case, the alternative information belongs to "user created content", emphasizing content created by non-expert users at runtime.

5 Page partitioning

This section presents our main contribution, two novel web page partitioning algorithms. These algorithms allow for automatic, web page independent partitioning. Moreover, our algorithms are able to utilize user-generated web page annotations as introduced in the previous section to further improve the automatically generated partitionings. The rest of this section is structured as follows. We first show how the problem of partitioning a web page can be formulated mathematically. Afterwards, we introduce important preprocessing steps and show how annotations are taken into consideration. Next, the algorithms are presented. Last, we outline the postprocessing steps which also benefit from user-generated annotations.

5.1 Problem formulation

Each web page is represented by a graph, its DOM tree. Graph partitioning algorithms build the theoretical foundation for partitioning a web page for collaborative browsing. The partitioning of objects of a particular set (e.g. the vertices of a graph) can be seen as a discrete classification problem. The objective is to classify these objects under certain constraints, therefore assigning at least one class (or label) to each object.

The metric labeling problem, originally formulated in [21], defines an optimization problem which asks for a discrete labeling of minimum total cost. A precise definition is given in the following.

Definition 1 (Metric Labeling Problem) Let V be a set of $n \in \mathbb{N}$ objects which are to be classified. Let L be a set of k possible labels. Then $f : V \to L$ is called a *labeling* of V over L. The cost of assigning a certain label l to an object u is denoted by $c : V \times L \to \mathbb{R}_0^+$. The pairwise relationship of the objects can be expressed in terms of a graph $G = (V, E)$. An edge $e = (u, v)$ indicates that u and v are related. The similarity of u and v is measured by the distance $d : L \times L \to \mathbb{R}_0^+$. $d(\cdot, \cdot)$ is a metric over L. This leads to the total cost $Q(f)$ of the labeling:

$$Q(f) = \sum_{u \in V} c(u,\ f(u)) + \sum_{e=(u,v) \in E} d(f(u),\ f(v)). \tag{1}$$

The goal is to find a discrete labeling of minimum total cost.

For web page partitioning, the labeling problem can be illustrated as follows. Let $G = (V, E)$ be the graph induced by the DOM of an arbitrary web page. Let N be a subset of V and $u, v \in N$. The original web page shall be viewed on, for example, two devices. Thus, let L be a set of labels with $|L| = 2$ and $\alpha, \beta \in L$, representing the two devices respectively. The functions in the above definition can be interpreted as follows:

- $c(u, l)$ for all $u \in N$ and $l \in L$ defines the non-negative cost of displaying a vertex $u \in N$ on device $l \in L$. If $c(u, l)$ is small, only low cost has to be paid to display u on the device assigned to label l. However, if $c(u, l)$ is high, then one will have to pay high cost by labeling u with l.
- $d(f(u),\ f(v))$ defines the edge weight representing the relatedness of two connected vertices u and v. If u and v are related, they shall be displayed on the same device. Consequently, high cost shall be paid when related vertices are labeled differently.

The metric labeling problem is MAX SNP-hard [10] and therefore unlikely to have a polynomial-time approximation scheme. Hence, we have enhanced two existing approximation algorithms to support the particular demands of collaborative web browsing. The first algorithm is based on the correlation clustering problem, which is a specialization of the metric labeling problem. The second algorithm is a graph-theoretic approach which is based on energy minimizing graph cuts. Moreover, the challenging task is to determine the weights $c(\cdot, \cdot)$ and $d(\cdot, \cdot)$ in (1). We apply three steps to transform a web page into sub-pages for collaborating devices, namely, the preprocessing, the actual partitioning and the postprocessing step.

5.2 Preprocessing

Using all vertices of a DOM as input for the graph partitioning algorithms would lead to a combinatorial explosion, especially for larger web sites such as Amazon.com. We decided to use only the leaf nodes of a DOM. Leaf nodes mostly contain links, images and text elements which are the most important content elements. Leaf nodes which only contain style information (e.g. typeface) are left out.

Preprocessing depends on the utilized algorithm. Our first algorithm based on correlation clustering requires a complete graph as input. Hence the leaf nodes have to be transformed into a complete graph. The second algorithm uses a set of neighboring vertices. The original tree is traversed in postorder, leafs are being extracted and put in a sequential list.

The cost functions $c(\cdot,\cdot)$ and $d(\cdot,\cdot)$ in (1) are mapped to edge weights of the corresponding graph which is used by our partitioning algorithms. In order to determine these weights, we introduce the function φ in the following definition. The actual computation of the edge weights depends on the utilized algorithm and is therefore described in the subsequent sections.

φ utilizes structural information by considering the vertices' tree distances. Moreover, additional semantic information provided by annotations is taken into account by mapping them to a series of real numbers in $\mathbb{R}|_{[0,1]}$.

Definition 2 Let $d : L \times L \to \mathbb{R}_0^+$ be a metric over a label set L. Let $G = (V, E)$ be a graph. Let further $u, v \in V$, $\delta \in \mathbb{N}$ and $(s_i)_{i\in\{1,\ldots,n\}}, s_i \in \mathbb{R}|_{[0,1]}$ and $n \in \mathbb{N}$. Then let $\varphi : V \times V \to \mathbb{R}|_{[0,1]}$ be defined as

$$\varphi(u, v) = \begin{cases} \min(1, d_{uv} + coh_{uv} / 10) & \text{if } d_{uv} \neq 0, \\ 0 & \text{else,} \end{cases} \tag{2}$$

with $coh_{uv} := \left(1 + \sum_{i=1,\ldots,n} s_i\right)(\delta - \text{dist}(u, v))$.

φ relates the initial metrical value $d_{uv} := d(f(u), f(v))$ (i.e. the initial edge weight) for two vertices u and v in V to their cohesion coh_{uv}. It consists of the term $\delta - \text{dist}(u, v)$ which limits the tree distance of two vertices in the DOM by δ. Under the constraint that $\text{dist}(u, w) + \text{dist}(w, v) < \delta$ for every $u, v, w \in V$, φ does not alter the metric imposed by $d(\cdot,\cdot)$ [1]. When two vertices are too far away from each other (i.e. $\text{dist}(u, v) \geq \delta$), the corresponding edge weight is not increased. Each s_i defines a piece of semantic information (e.g. a local assignment annotation with value 1.0) for either u or v. This allows to further control the actual edge weight.

φ is self-regulating due to the fact that highly cohesive pairs of vertices (i.e. vertices with small tree distance) are evaluated to higher values. Less cohesive pairs of vertices are mapped to lower values contrarily. An example illustrating the application of φ for a specific algorithm is shown in Section 5.3.

5.3 Algorithm based on correlation clustering

An approximation algorithm based on correlation clustering has been proposed by Ailon et al. [1]. It treats the set of labels as indistinguishable ($c(u, \alpha) \equiv 0$) for all

[1] Proof omitted in this version due to space limitations.

vertices u and labels α. Hence, it only matters whether a pair of vertices is labeled using the same label or different labels. The metric $d(\cdot, \cdot)$ can then be formulated as follows for any vertices $u, v \in V$ and edge weight $\varphi(u, v) \in \mathbb{R}|_{[0,1]}$:

$$d(f(u), f(v)) = \begin{cases} \varphi(u, v) & \text{if } f(u) \neq f(v), \\ 1 - \varphi(u, v) & \text{if } f(u) = f(v). \end{cases} \tag{3}$$

This leads to the modification of (1) for $u, v \in V$ as

$$Q(f) = \sum_{f(u) \neq f(v)} \varphi(u, v) + \sum_{f(u) = f(v)} 1 - \varphi(u, v). \tag{4}$$

Our approximation algorithm for this problem is based upon those presented in [1, 8]. Our approach differs first by utilizing $\varphi(\cdot, \cdot)$ (see Definition 2) to compute the edge weights and therefore considering additional semantic information. Second, the number of Clusters n is known a-priori.

Figure 5 exemplifies the computation of the edge weights for a small tree $G = (V, E)$ with leaf nodes u, v and w. These leaves are being used to generate a complete graph which serves as the input for Algorithm 1. The edge weights of the complete graph are initialized with 0.5 (step 1). Let $\delta = 4$ in (2) and assume that the graph has neither been annotated (i.e. $s_i = 0$ for all $i = 1, \ldots, n$), nor labeled before. Then the edge weight for (u, v) is $d(f(u), f(v)) = \varphi(u, v) = \min(1, 0.5 + 0.2) = 0.7$; analogously $d(f(u), f(w)) = d(f(v), f(w)) = \min(1, 0.5) = 0.5$ due to $\text{dist}(u, w) = \text{dist}(v, w) = 4$ in G. Consequently, the relationship between u and v is evaluated to a higher value because they are siblings in G.

The algorithm starts with a weighted, complete graph $K = (V, E)$ and a cluster threshold $t \in \mathbb{R}|_{[0,1]}$ (e.g. 0.6) as input values (see Algorithm 1). It outputs a clustering $(C_i)_{i=1,\ldots,n}$, with n being the total number of labels. A clustering $(C_i)_{i=1,\ldots,n}$ corresponds to a labeling f in a natural way. A cluster C_i is just a set of vertices with the same label. Consequently, the terms labeling and clustering can be used interchangeably.

The clusters are generated as follows (cf. Algorithm 1). The algorithm uses a temporary set W, being a copy of V, for the computational process (step 1). This way, the original set of vertices V is not altered. While W is non-empty and $i < n$, a vertex $v \in W$ is selected uniformly at random, removed from W and added to a new cluster set C_i (step 3 to 5). Every vertex $u \in W$ with $e = (v, u) \in E$ and $d(f(u), f(v)) \geq t$ is added to C_i and removed from W as well (step 6 to 11). In other words, each

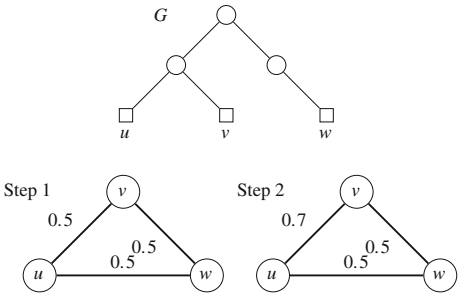


Fig. 5 Example computation of edge weights

Algorithm 1 CorrelationClustering

Input: Weighted, complete graph $K = (V, E)$; threshold t.
Output: Clustering $(C_i)_{i=1\ldots n}$.

1: $W \leftarrow V$
2: **while** $W \neq \emptyset$ and $i < n$ **do**
3: Choose $v \in W$ uniformally at random.
4: $W \leftarrow W \backslash \{v\}$
5: Create new cluster $C_i = \{v\}$
6: **for all** $u \in W$ such that $\exists e = (v, u) \in E$ **do**
7: **if** $d(f(v), f(u)) \geq t$ **then**
8: $W \leftarrow W \backslash \{u\}$
9: $C_i \leftarrow C_i \cup \{u\}$
10: **end if**
11: **end for**
12: $i \leftarrow i + 1$
13: **end while**
14: $C_n \leftarrow W$
15: **return** $(C_i)_{i=1,\ldots,n}$

vertex u adjacent to v will be added to the same cluster C_i, if u and v are related to a certain extent (i.e. $d(f(u), f(v)) \geq t$). When the creation of C_i is completed, the algorithm continues for $i = i + 1$ and with another randomly selected vertex in W (step 12 and step 2 onwards). Finally, remaining vertices in W are being copied to the last cluster C_n. Since this algorithm is randomized (see step 3), several independent runs of Algorithm 1 are performed and the clustering with the least objective value $Q(f)$ is chosen.

5.4 Algorithm based on graph cuts

The correlation clustering algorithm permits only one label assignment per vertex at a time. In contrast to this, the following approximation algorithm for (1) allows to assign labels to various vertices simultaneously by pairwisely exchanging labels. This process is a so-called α-β-swap [6], where α and β are labels, corresponding to a particular device. These swap moves are performed using graph cuts in a flow network which consists of the labels as terminals and the node set V containing the neighboring vertices. Internally, the terminals are also leaf nodes which have been preliminary assigned a label. The edge weights $w_{u,v} \in \mathbb{R}|_{[0,1]}$ for $u, v \in V$ are defined as follows:

$$w_{u,v} = \begin{cases} \frac{1}{2} d(f(u), f(v)) & \text{if } u, v \in V \backslash \{\alpha, \beta\}, \\ c(u, f(u)) & \text{if } v \in \{\alpha, \beta\}, \\ c(v, f(v)) & \text{if } u \in \{\alpha, \beta\}, \end{cases} \tag{5}$$

whereas $d(f(u), f(v)) := \varphi(u, v)$ and for $s \in \mathbb{R}|_{[0,1]}$

$$c(u, f(u)) = \begin{cases} s & \text{if } u \text{ is annotated,} \\ \varphi(u, v) & \text{else.} \end{cases} \tag{6}$$

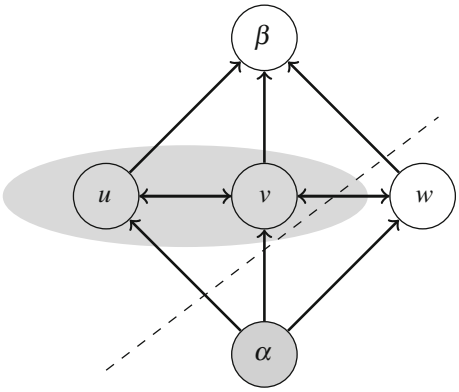


Fig. 6 Example flow network

In case a neighbor vertex is annotated, the annotation is again mapped to a real number s and assigned as an edge weight. This technique allows for instance to enforce an assignment of a vertex to a particular label. Figure 6 illustrates a swap move for a flow network with source α, sink β and a set of neighboring vertices $\{u, v, w\}$ which were originally labeled β.

An α-β-swap is performed by computing the minimum cut over the flow network (dashed line in Fig. 6). Each cut edge to either source or sink indicates an assignment to the corresponding label. In case of Fig. 6 the edges (α, u), (α, v) and (w, β) are cut. Hence, both u and v are assigned to α and w is labeled with β. By applying these swap moves for every pair of labels, an approximation for (1) is being calculated.[2]

Algorithm 2 is based upon [6] and uses α-β-swaps to find a local minimum for the total cost of a discrete labeling. In contrast to [6], our algorithm utilizes a set of neighboring vertices N (i.e. $\{u, v\} \in N \Rightarrow u$ and v are related) and a set of labels L with $|L| = m \in \mathbb{N}$ as input. Moreover, annotations are encoded in the edge weights (see (5) and (6)). The algorithm outputs a labeling f for which $Q(f)$ is a local minimum. First, each vertex in N is labeled with the same label $\alpha \in L$; α can be chosen arbitrarily (step 1 to 3). Next, for each pair of labels $\{\alpha, \beta\} \subset L$, α-β-swaps are performed over N on the basis of the current labeling f (step 5.1).

If the labeling f^*, yielding the lowest cost over all possible α-β-swaps from f, results in a lower total cost $Q(f^*) \leq Q(f)$, f^* is set as the new labeling of N (step 5.2). This process is repeated iteratively until the total cost $Q(f)$ can not be further reduced.

5.5 Postprocessing

In the postprocessing step, the labeling of the leaf nodes is used to label the complete DOM. The DOM is traversed from the leafs to the root in level-order. The most common label in each father node's subtree is assigned to the father itself. Figure 7 (step 1) shows an example DOM tree whose leaf nodes have been labeled by applying a graph partitioning algorithm. Next, a label is assigned to the father node of a, c and

[2]Proof omitted in this version due to space limitations.

Algorithm 2 GraphCuts

Input: Set of vertices N, set of labels L
Output: Labeling f.

1: **for all** $\{u\} \cap N \neq \emptyset$ **do**
2: $f(u) \leftarrow \alpha$, for any fixed $\alpha \in L$
3: **end for**
4: SUCCESS $\leftarrow$ *false*
5: **for each** pair of labels $\{\alpha, \beta\} \subset L$
5.1: $f^* = \arg\min Q(\hat{f})$ among $\hat{f}$ within one α-β-swap of f
5.2: **if** $Q(f^*) < Q(f)$ **then**
$f \leftarrow f^*$
SUCCESS $\leftarrow$ *true*
end if
6: **end for each**
7: **if** SUCCESS **then**
8: goto *4*
9: **end if**
10: **return** f

e. Both c and e are labeled grey and only a is labeled with a square. Grey, being the dominant label in this subtree, is assigned to the father node of a, c and e (step 2). The same procedure is repeated for the father node of n and o which is therefore labeled with a square. Note, that the root node is not labeled, since it groups the content elements of a web page and must be present in every cluster.

A further traversal of the DOM is required to subdivide the tree into different, not necessarily disjunctive partitions. When performed naively, this may result in an improper removal of content elements as shown in Fig. 8. The subdivision starts with the labeled DOM tree as shown in Fig. 7 (step 2). The tree is traversed in preorder, once per label. First, a cluster C_1 consisting of grey nodes is being generated. Each time, when the label of a child node differs from its father's label (grey), the child is

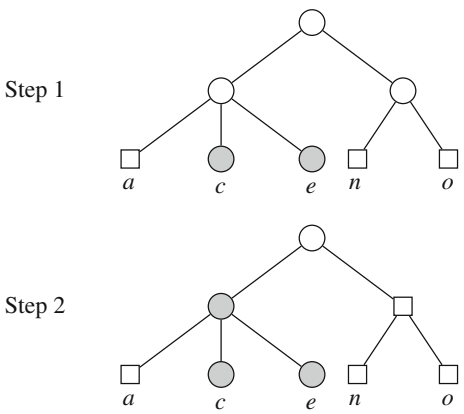


Fig. 7 Applying a labeling to the original DOM

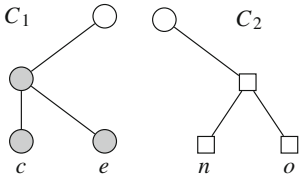


Fig. 8 Example, content accidently removed

being removed. C_1 is the result of removing node a and the right subtree of the root vertex. This procedure is repeated to construct C_2. Since a is neither contained in C_1, nor in C_2 it is not displayed on any device. Simply adding a to C_2 would actually solve this problem. However, the presentation of a additionally depends on that of its father. Hence, its father must not be removed from the tree to ensure a proper display (see Fig. 9).

When partitioning the labeled DOM, annotated CARE properties are considered. For example, if node o were to be displayed redundantly, it would have to be added to C_1. Moreover, its father would have to be added to C_1 as well (see Fig. 10).

6 Evaluation

We have implemented the two partitioning algorithms with the additional pre- and postprocessing steps, and integrated them into our architecture for multi-browsing applications. A controlled experiment has been conducted to evaluate and compare the algorithms both in terms of user satisfaction for the generated web page distribution and concerning their runtime performance. In particular, the impact of the annotations on results of the algorithms was investigated.

6.1 Method

We recruited 14 participants (10 male, 4 female) on their willingness to participate. We gave them a short introduction to multi-browsing in general, as well as our multi-browsing engine. The storyline for their tasks was based on the scenario introduced at the beginning of this paper: Mike and Alice want to browse the web together. Mike uses a smartphone, whereas Alice is in front of a large touchscreen. Mike wants to use his smartphone to present certain information contained in a web site to Alice by exploiting the large touchscreen. The participants were asked to take over Mike's role.

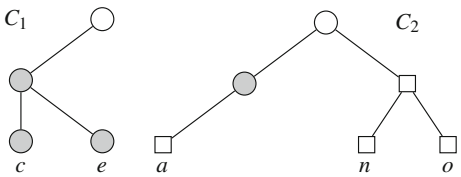


Fig. 9 Example, content duplicated

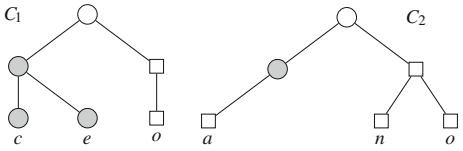


Fig. 10 Example, content *o* displayed redundantly

The procedure consisted of two parts which are described in the following. In the first part, they got printed versions of the following three web pages:

- *Google Maps*:[3] A map application like Google is a typical example of a web application which benefits from using multiple devices. The presentation of the map can be adapted to the rendering devices. Furthermore, a remote control pattern can be easily applied to this application.
- *101cookbooks.com*:[4] This web site has a complex table-based layout. It provides lots of different information (e.g. recipes and images) in different modalities. The web page provides recipes, images and content created with Adobe Flash.
- *Travel information portal of the German Railways*:[5] The portal features a large form and hence provides various input possibilities. Consequently, it is interesting to see how people deal with such a complex application in symbiotic environments.

The assignment to the participants was to subdivide each print of the web pages into different, not necessarily disjunctive segments. The participants had to decide where the segments should be displayed; either on Mike's smartphone, on the large touchscreen or on both. They sketched their decisions on the prints. An example annotation of one of our participants for the travel information portal of the German Railways is shown Fig. 11. At this stage, the participants were not influenced by the way the multi-browsing system performs the web page distribution.

In the second part, the multi-browsing system was utilized to perform the distribution of the web-pages among both devices. One after another, both algorithms were used, utilizing no annotations. The evaluation order was fully counterbalanced for each participant, regarding both, algorithms used and web pages partitioned. The participants had to compare these results with their own paper-based segmentations. Thus, they decided whether the computed results met their expectations and rated these results with grades from 1 (meaning "*deficient*") to 10 (meaning "*very good*").

Next, the web pages have been annotated to improve the results. When three annotations had been added to the web page, we paused the annotation process and used our multi-browsing system to compute the results for the annotated web pages. The participants had to rate these results again. This allowed us to evaluate the improvement compared to the previous result. The process ended when at most 12 annotations were added to the web page or when a result entirely fulfilled a user's expectations and was therefore graded with ten.

[3] http://maps.google.de/maps?output=html&q=heidelberg

[4] http://www.101cookbooks.com/pies_and_tarts

[5] http://reiseauskunft.bahn.de

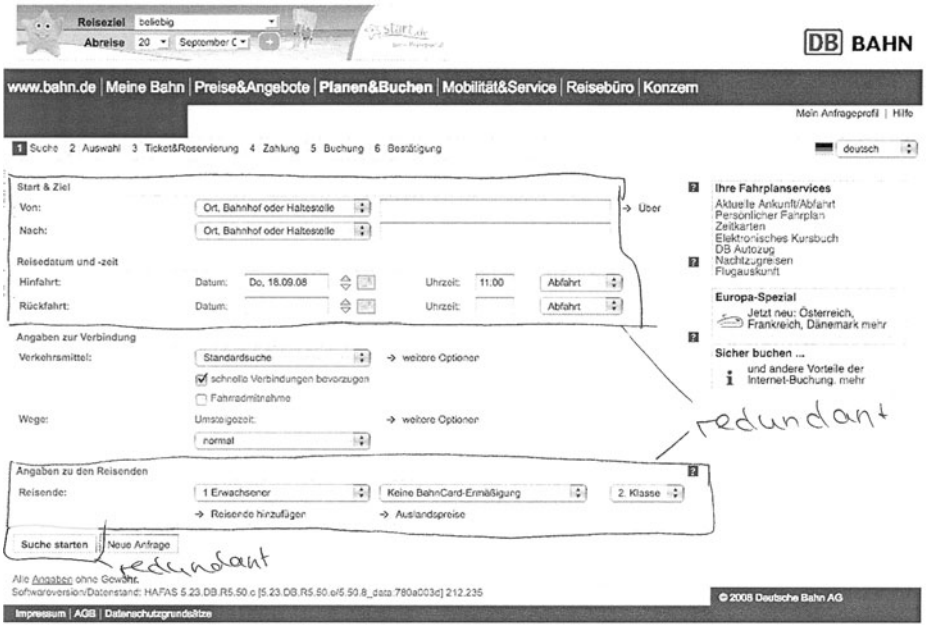


Fig. 11 User annotated print, two areas of the input form are marked as *redundant*. The participant noted that the rest of the web page should only be displayed on the mobile device

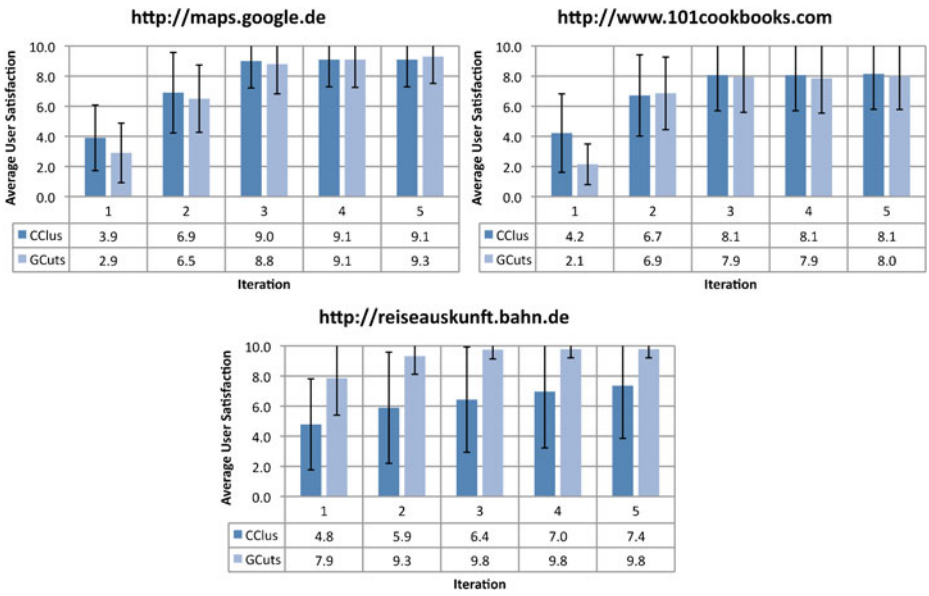


Fig. 12 Average user satisfaction per iteration

After the user study, each participant filled out a questionnaire with nine statements. Each statement was graded with a five point Likert scale (one meaning "I totally agree" and five meaning "I totally disagree").

6.2 User acceptance and annotation impact

Figure 12 shows the average user satisfaction for the three web sites respectively (*CClus* designates correlation clustering and *GCuts* designates our graph cuts algorithm). The term *iteration* designates the pauses of the whole process. No annotations were used in the first iteration. From the second iteration onwards, at most three user-generated annotations were added per iteration as described above. Each figure compares the utilized algorithms to each other. In the first iteration without any annotations, correlation clustering has been rated with grades from 3.9 to 4.8. Hence, the generated web page distributions were acceptable for the participants.

In all but one case, correlation clustering provided much better initial results than the graph cuts algorithm. Interestingly, correlation clustering was not able to generate satisfying results for the travel information portal of the German Railways. The information portal consists of various form elements which are semantically related and need to be displayed on the same device. For example, if the input fields for the starting location and the destination are displayed on different devices, continuity is being lost and people get confused. Such elements were assigned to different devices using correlation clustering.

Annotations were added in the second iteration. Their impact on the user satisfaction was significant for both algorithms ($p < 0.01$). Only the difference in user satisfaction using correlation clustering for the travel information portal was not significant. This is because the correlation clustering algorithm ignored several annotations, although annotations are strongly binding and must not be ignored. The cause for this indeterministic behavior remains to be investigated.

6.3 General user feedback: questionnaire

The questionnaire was subdivided into four different main categories. In the category regarding multi-device browsing as a feature, 11 out of 14 participants think that browsing the web collaboratively is useful in general. All of the participants think that the presented system is useful. Nevertheless, only ten would like to use the system frequently.

Regarding the implemented algorithms, a total of nine participants think that the web page distributions generated by our algorithms without any annotations were satisfying. Twelve participants were content with the time the algorithms took to calculate a result. However, correlation clustering is not suitable for larger web pages, due to its complexity. For instance, two participants remarked that correlation clustering took too long to generate a web page distribution for 101cookbooks.com.

The participants also had to state whether additional annotations improved the results of our algorithms. Nine out of 14 are of the opinion that adding annotations improved the results significantly. Additionally, fourthink that annotations improved the generated results to a certain extent. However, only six participants would take the time to annotate web pages. Eight participants think that annotating a web page is too time consuming and the results provided solely by the algorithms without any

Table 2 Comparison of algorithm runtime for two devices

	Nodes	Algorithm 1	Algorithm 2
apple.com	40	83 ms	36 ms
maps.google.de	42	110 ms	18 ms
reiseauskunft.bahn.de	106	2731 ms	83 ms
101cookbooks.com	207	18607 ms	145 ms
microsoft.com	213	20472 ms	127 ms
ebay.com	254	39099 ms	196 ms
tagesschau.de	476	>60000 ms	254 ms
amazon.com	483	>60000 ms	299 ms

annotations satisfied their needs. But Fig. 12 shows that after adding about three to six annotations, even for larger web pages like 101cookbooks.com, users were nearly satisfied with the results. Hence, only little effort is required to improve the results.

6.4 Runtime performance

We have measured the runtime performance of our algorithms for partitioning various web pages of different sizes for two devices. Table 2 shows an excerpt of the evaluated web pages, where Algorithm 1 is correlation clustering and Algorithm 2 is graph cuts.

The number of nodes designates the relevant leaf nodes of the corresponding DOM tree. The partitioning was computed on the server-side using a 2,4 GHz dual core processor. For *Google Maps* with 42 leaf nodes, correlation clustering and graph cuts required 110 ms and 18 ms respectively. For the travel information portal with 106 leaf nodes, correlation clustering and graph cuts required 2731 ms and 83 ms respectively. For 101cookbooks.com with 207 leaf nodes, correlation clustering and graph cuts required 18607 ms and 145 ms respectively. We can conclude that graph cuts clearly outperforms the correlation clustering algorithm, in particular when the size of a web page increases (e.g. number of leaf nodes ≥ 100). Since web pages shall be partitioned dynamically to take the prevailing context (e.g. the number of devices) into account, graph cuts shall be preferred over correlation clustering for large web pages to avoid long response time.

7 Conclusion

In this paper, we contribute two novel web page partitioning algorithms which differ from existing partitioning algorithms by allowing for both, automatic and semi-automatic partitioning. These algorithms provide good automatic, web page independent results, as shown by our user study. The results can be improved significantly by taking user-generated annotations into account.

The first algorithm is based on the correlation clustering problem, while the second one is based on energy minimizing graph cuts. Both algorithms utilize weighted graphs. We have defined a function which determines the edge weights using both structural information contained in a web page's DOM tree, and additional semantic

information provided by annotations. Applying the algorithms naively may result in the deletion of content elements of a web page. Hence, the postprocessing of the generated partitionings is a crucial step for the eventual presentation on different devices.

A user study has been conducted to evaluate the user satisfaction for web page distributions generated by the two algorithms. The correlation clustering algorithm provided better initial results for web pages which have not been annotated before. However, the quality of results generated by the graph cuts algorithm improved significantly by adding annotations. Correlation clustering has a much higher complexity than the graph cut algorithm. Moreover, it requires several runs to find a satisfactory solution due to using randomization. Consequently, concerning the runtime performance, it is obviously outperformed by the graph cuts algorithm, when the number of leaf nodes of a DOM exceeds a certain threshold.

We have presented our architecture supporting user-centric multi-browsing. Its annotation environment allows both developers and end users to individually annotate existing web-pages. In contrast to existing approaches, our architecture does not require web pages with static, pre-defined annotations. As an adaptation to the changing context, end users can dynamically change their annotations to trigger a new partitioning of the web pages.

In summary, the implemented architecture and the partitioning algorithms fulfill the requirements we set before. As future work, we will consider adjusting the behavior of our partitioning algorithms (e.g. by using different δ values in (2)), as well as adaptively selecting the partitioning algorithm based on, e.g. the number of leaf nodes of a concrete web page.

References

1. Ailon N, Charikar M, Newman A (2005) Aggregating inconsistent information: ranking and clustering. In: STOC '05: proceedings of the thirty-seventh annual ACM symposium on theory of computing. ACM, New York, NY, USA, pp 684–693. doi:10.1145/1060590.1060692
2. Alapetite A (2010) Dynamic 2d-barcodes for multi-device web session migration including mobile phones. PUC 14:45–52. doi:10.1007/s00779-009-0228-5
3. Amershi S, Morris MR (2008) Cosearch: a system for co-located collaborative web search. In: Proceeding of the twenty-sixth annual SIGCHI conference on human factors in computing systems, CHI '08. ACM, New York, NY, USA, pp 1647–1656. doi:10.1145/1357054.1357311
4. Atterer R, Schmidt A, Wnuk M (2007) A proxy-based infrastructure for web application sharing and remote collaboration on web pages. In: Proc. of the 11th International Conference on Human Computer Interaction (INTERACT)
5. Bouchet J, Nigay L, Ganille T (2004) ICARE software components for rapidly developing multimodal interfaces. In: Proc. of the International Conference on Multimodal Interfaces (ICMI 2004)
6. Boykov Y, Veksler O, Zabih R (2001) Fast approximate energy minimization via graph cuts. IEEE Trans Pattern Anal Mach Intell 23(11):1222–1239. doi:10.1109/34.969114
7. Cai D, Yu S, Wen J, Ma W (2003) Extracting content structure for web pages based on visual representation. In: Proc.5 th Asia pacific web conference, pp 406–417
8. Chakrabarti D, Kumar R, Punera K (2008) A graph-theoretic approach to webpage segmentation. In: WWW '08: proceeding of the 17th international conference on World Wide Web. ACM, New York, NY, USA, pp 377–386. doi:10.1145/1367497.1367549

9. Coles A, Deliot E, Melamed T, Lansard K (2003) A framework for coordinated multi-modal browsing with multiple clients. In: Proc. of the 12th international conference on World Wide Web (WWW '03). ACM Press, New York, NY, USA, pp 718–726
10. Dahlhaus E, Johnson DS, Papadimitriou CH, Seymour PD, Yannakakis M (1992) The complexity of multiway cuts (extended abstract). In: STOC '92: proceedings of the twenty-fourth annual ACM symposium on theory of computing. ACM, New York, NY, USA, pp 241–251. doi:10.1145/129712.129736
11. Ding Y, Huber J (2008) Designing multi-user multi-device systems—an architecture for multi-browsing applications. In: Proc. of the 7th international ACM conference on mobile and ubiquitous multimedia, pp 8–14
12. Direct Web Remoting (2011) http://www.directwebremoting.org. Accessed 2 Jan 2012
13. Fiduccia CM, Mattheyses RM (1988) A linear-time heuristic for improving network partitions. In: 25 years of DAC: papers on twenty-five years of electronic design automation. ACM, New York, NY, USA, pp 241–247. doi:10.1145/62882.62910
14. Florins M, Vanderdonckt J (2004) Graceful degradation of user interfaces as a design method for multiplatform systems. In: Proc. 9th International Conference on Intelligent User Interfaces (IUI 2004)
15. Greenberg S, Boyle M, Laberge J (1999) PDAs and shared public displays: making personal information public, and public information personal. PUC 3(1/2):54–64
16. Han R, Perret V, Naghshineh M (2000) WebSplitter: a unified XML framework for multi-device collaborative web browsing. In: CSCW '00: proceedings of the 2000 ACM conference on computer supported cooperative work. ACM Press, New York, NY, USA, pp 221–230
17. Hattori G, Hoashi K, Matsumoto K, Sugaya F (2007) Robust web page segmentation for mobile terminal using content-distances and page layout information. In: WWW '07: proceedings of the 16th international conference on World Wide Web. ACM, New York, NY, USA, pp 361–370. doi:10.1145/1242572.1242622
18. Johanson B, Fox A, Winograd T (2002) The interactive workspaces project: experiences with ubiquitous computing rooms. IEEE Pervasive Computing 1(2):67–74
19. Johanson B, Ponnekanti S, Sengupta C, Fox A (2001) Multibrowsing: moving web content across multiple displays. In: UbiComp '01: proceedings of the 3rd international conference on Ubiquitous Computing. Springer, London, UK, pp 346–353
20. Kane S, Karlson A, Meyers B, Johns P, Jacobs A, Smith G (2009) Exploring cross-device web use on pcs and mobile devices. In: INTERACT 2009. Springer, pp 722–735
21. Kleinberg J, Tardos É (2002) Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. J ACM 49(5):616–639. doi:10.1145/585265.585268. http://portal.acm.org/ft_gateway.cfm?id=585268&type=pdf&coll=GUIDE&dl=GUIDE&CFID=2588277&CFTOKEN=23748779
22. Maekawa T, Hara T, Nishio S (2006) A Collaborative web browsing system for multiple mobile users. In: PERCOM '06: proceedings of the fourth annual IEEE international conference on Pervasive Computing and Communications (PERCOM'06). IEEE Computer Society, Washington, DC, USA, pp 22–35
23. Maekawa T, Uemukai T, Hara T, Nishio S (2005) Content description and partitioning methods for collaborative browsing by multiple mobile users. In: Proc. of the 16th international workshop on Database and Expert Systems Applications (DEXA'05)
24. Magerkurth C, Tandler P (2002) Interactive walls and handheld devices—applications for a smart environment. In: Proc. of the workshop collaboration with interactive walls and tables, held in conjunction with UbiComp'02
25. Myers B (2001) Using handhelds and PCs together. Commun ACM 44(11):34–41
26. Raghunath M, Narayanaswami C, Pinhanez C (2003) Fostering a symbiotic handheld environment. Comput 36(9):56–65
27. Tuddenham P, Davies I, Robinson P (2009) Websurface: an interface for co-located collaborative information gathering. In: Proceedings of the ACM international conference on Interactive Tabletops and Surfaces, ITS '09. ACM, New York, NY, USA, pp 181–188. doi:10.1145/1731903.1731938
28. Wiltse H, Nichols J (2009) Playbyplay: collaborative web browsing for desktop and mobile devices. In: Proceedings of the 27th international conference on human factors in computing systems, CHI '09. ACM, New York, NY, USA, pp 1781–1790. doi:10.1145/1518701.1518975
29. Yahoo User Interface Library (2011) http://developer.yahoo.com/yui/. Accessed 2 Jan 2012

Jochen Huber is a doctoral researcher at Technische Universität Darmstadt, Germany. He is a scholar in the research training group on Feedback-based Quality Improvement in E-learning funded by the German Research Foundation (DFG). Jochen holds degrees in both computer science and mathematics from Technische Universität Darmstadt. He teaches Interaction Design at Fachhochschule Mainz, Mayence, Germany. Jochen is a member of ACM, ACM SIGCHI, ACM SIGMM and the German Informatics Society (GI e.V.).

Yun Ding is senior consultant at RIB Software AG. The major work of this article has been conducted when she was project manager and researcher at European Media Laboratory GmbH, Heidelberg, Germany. She received her PhD and MS in Computer Science from University of Stuttgart, Germany and Technische Hochschule Karlsruhe, Germany, respectively.