Ontology-Based Management of Conflicting Products in Pixel Advertising

Ferry Boon, Sabri Bouzidi, Raymond Vermaas, Damir Vandic, Flavius Frasincar

Erasmus University Rotterdam
Erasmus School of Economics
PO Box 1738, NL-3000 DR, Rotterdam, the Netherlands
ferry.boon@gmail.com, sabribouzidi@outlook.com,
research@raymondvermaas.nl, vandic@ese.eur.nl,
frasincar@ese.eur.nl

Abstract. Pixel advertising represents the placement of multiple pixel blocks on a banner for the purpose of advertising companies and their products. In this paper, we investigate how one can avoid product conflicts in the placement of pixel advertisements on a Web banner, while maximizing the overall banner revenue. Our solution for this problem is based on a product ontology that defines products and their relationships. We evaluate three heuristic algorithms for generating allocation patterns, i.e., the left justified algorithm, the orthogonal algorithm, and the GRASP constructive algorithm. The results show that the left justified algorithm and the orthogonal algorithm are most effective in terms of profit per pixel, while the GRASP constructive algorithm is identified as most efficient in terms of computational time.

1 Introduction

Web advertising is a billion dollar business in which most large Web companies have found their main stream of revenue. The total advertising revenues from Google were 37.9 billion USD in 2011, 46.1 billion USD in 2012, and 55.5 billion USD in 2013 [5]. Companies such as Yahoo!, Facebook, Microsoft, and AOL also report advertising revenues in the billions, making online advertisement revenues larger than the revenues obtained in printed media [7].

Pixel advertising is a form of display advertising on the Web in which the cost of each advertisement is calculated based on the number of pixels it occupies. The general idea is to have a banner that consists of several small advertisements (i.e., a multi-picture banner), instead of just one advertisement occupying the whole banner. Pixel advertising was invented in 2005 by the English student Alex Tew, who created the "Million Dollar Homepage" [12]. This Web page holds a 1000 by 1000 pixel grid from which blocks of 10 by 10 pixels can be bought for 1 dollar per pixel. Buyers can place an image on their pixels and have the image link to their Website.

Although the "Million Dollar Homepage" has been incredibly successful, pixel advertising has not been yet widely adopted. There are a number of

issues that need to be addressed in order to make the concept more appealing to advertising companies. One of these issues is that there is no motivation for consumers to return to a pixel advertisement banner. Furthermore, when advertisements are placed on a banner, the content of the advertisement is often not taken into account, making it possible to place conflicting advertisements of competing products and/or brands in one Web banner. In this paper, we do take the content of an advertisement into account and tackle the problem of placing conflicting advertisements. Customers often have to choose between products from various domains. Marketing differentiated products, i.e., products that serve the same need of the customer and thus need to be pushed to the consumer via advertisements, frequently develop and compete on the basis of brands or labels. The Coca-Cola Company vs. Pepsi is a typical example of such a competition.

Placing advertisements on a banner is often done by employing heuristics, as the problem of finding an optimally constructed banner for a set of advertisements is NP-hard [15]. In the literature, there are studies that focus on finding heuristics for the purpose of optimally placing rectangular advertisements on a rectangular banner, in such a way that the revenue generated by the banner is maximized [3, 9, 15]. Unfortunately, none of these approaches takes the content of an advertisement into account.

In this paper, we propose an approach that avoids conflicts when placing pixel advertisements on a banner, while maximizing banner revenue. Using an ontology-based solution, our current research extends the heuristics discussed in [3, 9] by avoiding placing advertisements of conflicting brands on a banner. Conflicts can be identified by categorizing products using a domain ontology. In order to avoid product conflicts, we restrict the number of advertisements per category that is placed on the banner to one. We investigate which heuristic copes best with this new constraint and discuss the implementation of such an approach in a Web application.

The paper is structured as follows. In Section 2 we discuss related work. In Section 3 we give a formal problem definition, after which we present heuristics that deal with the conflict constraint banner placement problem. Section 4 explains the experimental setup and presents the obtained results. The concluding remarks and proposed future work are given in Section 5.

2 Related Work

Even though pixel advertising has not been successful so far, it is still interesting to study this topic, as it has great potential for new forms of advertising campaigns. In [14] the success of the "Million Dollar Homepage" and the failure of the many copycats that spawned from the original success is discussed. The authors argue that visitors do not return to the "Million Dollar Homepage", they only visit it once to check it out. The paper proposes some improvements to the concept of pixel advertising in Web pages. In [15], the authors extend the idea of pixel advertising by placing small advertisements in banners.

The work presented in [3] is the most related research to this paper and goes one step further than [15]. The authors propose a Web application that can automatically fill the banner with provided advertisements. Several heuristics are explored that optimize the building time and the revenue generated by the banner. More specific, heuristics for optimally placing rectangular advertisements on a rectangular banner in such a way that the revenue generated by the banner is maximized are discussed. The authors show that the orthogonal heuristic is the most suitable for a Web application. Furthermore, the authors experimented with the left justified algorithm, the GRASP constructive algorithm, and the greedy stripping algorithm. However, when placing the advertisements on a banner, the content is not considered, with the possible result of two competitive advertisements on the same banner. The pixel advertising and the multiple advertising allocation approach is related to other problems which are further discussed in [3], including the ad placement problem, knapsack problem, MINSPACE, and MAXSPACE problem, which are known NP-hard problems, hence the need for heuristics.

As previously mentioned, when placing advertisements on the banner, the advertisement content, and its associated semantics, is not taken into account. Especially with a small set of possible advertisements, it is highly probable that advertisements with conflicting messages or competing products are placed on the same banner. When we are looking at the nature of the conflicts that arise, they can be traced back to the same problem brands have in stores. Customers only have a limited amount of time and money to spend on products and services, and often have to choose between products. Marketing differentiated products frequently develop and compete on the basis of brands or labels. Several examples of this inter-brand competition would be Coca Cola vs. Pepsi-Cola, Levi vs. Pall Mall Jeans, and Pizza Hut vs. Dominos.

Another possible conflict would be the intra-manufacturer conflict, where a manufacturer owns, produces, and/or sells different brands that (in)directly compete with one another. A good example would be The Coca-Cola Company owns several soft drink brands such as Coca Cola, Fanta, and Sprite. These products obviously compete with each other as they are all carbonated soft drinks, but according to [11] we can disregard this form of competition as being a conflict. This form of brand extension is the so-called substitute brand extension, where a substitute product is branded differently than the original product. In the case of the Coca-Cola Company, all products are marketed separately with each brand having its own management and thus each brand has its own goals. Nevertheless, one can also argue that the intra-manufacturer conflicts are negligible as often in printed media brands from the same manufacturer are promoted near or even next to each other.

In order to avoid product conflicts, we extend the heuristics presented in [3]. Furthermore, we aim to identify which heuristic in the new setup is the most effective in terms of profit and which one is the most efficient in terms of speed. Speed is important because we plan to use the heuristics in a Web application, and according to a psychology research Web users are not willing to wait for

more than 15 seconds without feedback from the system that it is working [10]. Therefore our constraint on time is 15 seconds. Because the problem that we consider in this paper is NP-hard, heuristics are needed to obtain good solutions for large inputs in a timely manner. To identify conflicts we use an ontology-based approach. The type of conflict that this study uses is called Class Assertion Conflict. Such conflicts occur when constraints placed on classes are violated, as shown in [4].

3 Optimal Advertisement Allocation

In this section, we discuss in more detail the problem and solutions of optimally allocating advertisements, under the constraint that no conflicting advertisements should be placed and the revenue should be maximized. In Section 3.1 we give a formal problem definition that results in an integer programming model. Section 3.2 presents the considered heuristics to solve the presented problem. Last, in Section 3.3 we discuss a Web implementation of our approach.

3.1 Problem definition

The formal definition of the problem to be solved is similar and based on the problem definition given in [3]. We have a set $\mathcal{A}$ with a fixed number of advertisements $|\mathcal{A}|$ to allocate in a banner. We assume we have more advertisements in $\mathcal{A}$ than would fit on the banner, thus not every advertisement is placed. Each advertisement $a_i \in \mathcal{A}$ has a width w_i, height h_i, and a price per pixel pp_i, with $i \in \{1, ..., |\mathcal{A}|\}$. The banner has width $\mathcal{W}$ and height $\mathcal{H}$. The advertisements from $\mathcal{A}$ should be allocated on the banner such that the total value of the set of allocated advertisements $\mathcal{A}'$ (subset of $\mathcal{A}$) is maximized. Each advertisement a_i in $\mathcal{A}'$ has its top-left corner at position (x, y) on the banner, starting from $(0, 0)$ which represents the top left corner of the banner. The value of an allocated advertisements $\mathcal{A}'$ is defined by v_i, where $v_i = pp_i$ x w_i x h_i. Our objective is to maximize the total value of allocated advertisements in $\mathcal{A}'$.

We can formulate the problem as a 0-1 integer programming problem, which is a simplification of the problem formulation from [6], since our problem assumes that every advertisement can be allocated only once. In order to make sure that advertisements do not overlap on the banner we have reused a constraint from [2]. Let

$$\mathcal{X}_i = \{x | 0 \leq x \leq \mathcal{W} - w_i\}, \forall i \in \{1, ..., |\mathcal{A}|\}. \quad (1)$$

be the set of all possible points along the width of the banner such that an advertisement a_i from $\mathcal{A}$ can be placed on the banner with its top-left corner at these x-axis positions. Similarly we define

$$\mathcal{Y}_i = \{x | 0 \leq y \leq \mathcal{H} - h_i\}, \forall i \in \{1, ..., |\mathcal{A}|\}. \quad (2)$$

as the set of all possible allocation points along the height of the banner. We define

$$x_{ip} = \begin{cases} 1 & \text{if } a_i\text{is placed with top-left corner at x-position } p\text{, where } p \in \mathcal{X}_i \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

$$y_{ip} = \begin{cases} 1 & \text{if } a_i\text{is placed with top-left corner at y-position } q \text{ where } q \in \mathcal{Y}_i \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

and let

$$b_{ipqrs} = \begin{cases} 1 & \text{if advertisement } i\text{, placed with top-left corner at } (p,q), \\ & \text{cuts out point } (r,s) \text{ of the banner} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

which can be restated as

$$b_{ipqrs} = \begin{cases} 1 & \text{if } 0 \leq p \leq r \leq p + w_i - 1 \leq \mathcal{W} - 1 \\ & \text{and } 0 \leq q \leq s \leq q + h_i - 1 \leq \mathcal{H} - 1 \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Figure 1 visualizes x_{ip}, y_{iq}, and b_{ipqrs}. Now the integer programming formulation can be stated as follows:

$$max \sum_{i=1}^{|\mathcal{A}|} v_i \sum_{p \in \mathcal{X}_i} x_{ip} \tag{7}$$

subject to

$$\sum_{i=1}^{|\mathcal{A}|} \sum_{p \in \mathcal{X}_i} \sum_{q \in \mathcal{Y}_i} b_{ipqrs} \cdot x_{ip} \cdot y_{iq} \leq 1, \forall r \in \{0, ..., \mathcal{W} - 1\}, \forall s \in \{0, ..., \mathcal{H} - 1\} \tag{8}$$

$$\sum_{p \in \mathcal{X}_i} x_{ip} \leq 1, \forall i \in \{1, ..., |\mathcal{A}|\} \tag{9}$$

$$\sum_{p \in \mathcal{X}_i} x_{ip} = \sum_{q \in \mathcal{Y}_i} y_{iq}, \forall i \in \{1, ..., |\mathcal{A}|\} \tag{10}$$

$$x_{ip}, y_{ip} \in \{0, 1\}, \forall i \in \{1, ..., |\mathcal{A}|\}, \forall p \in \mathcal{X}_i, \forall q \in \mathcal{Y}_i \tag{11}$$

In order to avoid conflicting advertisements, a conflict matrix $\mathcal{C}$ is introduced where c_{ij} is the conflict between advertisement i and advertisement j and is defined as:

$$c_{ij} = \begin{cases} 1 & \text{if advertisement } i \text{ and } j \text{ are conflicting} \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

A new constraint can be added so that no conflicts are allowed between the allocated advertisements on the banner:

$$\sum_{p \in \mathcal{X}_i} x_{ip} = 1 - \sum_{j \neq i} c_{ij} \sum_{p \in \mathcal{X}_j} x_{jp}, \forall i \in \{1, ..., |\mathcal{A}|\} \tag{13}$$

In Eq. 7 the objective function maximizes the total value of the allocated advertisements. Constraint 8 ensures that any banner point is used by at most one advertisement. Constraints 9 and 10 ensure that any advertisement is allocated at most once on the whole banner. The ranges of p and q, i.e., $\mathcal{X}_i$ and $\mathcal{Y}_i$, respectively, ensure that advertisements are always placed inside the banner. Constraint 11 is the integrality constraint. Constraint 13 is added so that no conflicts are allowed in the allocated advertisements on the banner. The model can be linearized by replacing variables x_{ip} and y_{ip} with a variable that represents both x_{ip} and y_{ip} (e.g., z_{ipq}), as shown in [2].

3.2 Heuristics

Following the typology presented in [13], we can characterize the problem that we presented in the previous section as a *two-dimensional*, *single*, *orthogonal*, knapsack problem with conflict restrictions. The knapsack problem is a combinatorial optimization problem where strongly heterogeneous assortment of small items has to be allocated to one or more larger objects. The limitation here is that there is not enough space for all items, which means that a choice has to be made. The term *single* indicates that we only deal with one large object in which the smaller items (i.e., the advertisements) have to be placed. The term *orthogonal* refers to the fact that the edges of the smaller items are orthogonal to the edges of the larger object(s) and that rotation is not allowed.

In order to deal with the fact that the previously identified problem is NP-hard, one often employs optimized search algorithms to find solutions. These algorithms work by searching through the solution space in order to find the optimal solution. Such approaches can be classified as uninformed and informed algorithms. An uninformed algorithm tests all the possible solutions whereas

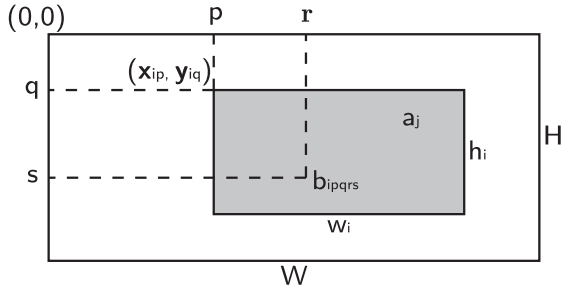

Fig. 1. Visualization of x_{ip}, y_{iq}, and b_{ipqrs}.

an informed algorithm uses knowledge of the search space to find the solution. The downside of finding the optimal solution is that it takes very long to find it using any of the current approaches. This is an important limitation because we aim to have a Web-service that allows users to create an optimal banner, which requires a relatively short execution time.

Because optimal solutions are hard to find, we focus on finding, in a relatively short amount of time, solutions that are close to the optimal one. For this purpose, we analyze and adapt three heuristic algorithms that can solve the problem presented in the previous section. For all three considered heuristics, which will be discussed shortly, we introduce an extra condition that is meant to avoid placing conflicting advertisements, i.e., we only place an advertisement on the banner if there is no other banner placed from the same product category (obtained from the product ontology).

Sorting. The heuristic approaches we use rely on the order the advertisements are processed by the algorithm. For this reason we apply a sorting procedure as an initial step. Besides considering the possibility of randomly sorting the advertisements, we define a set of properties of the advertisement on which we can sort. These are price per pixel (pp), width (w), height (h), total area ($w \times h$), flatness (w/h), and proportionality ($|\log w/h|$). The flatness property indicates whether an advertisement is flat ($w > h$) or tall ($h > w$). The proportionality property refers to what extent the advertisement dimensions resembles a square, where a value of 0 represents an exact square advertisement. Using these properties, we apply a two-way sort, i.e., a primary and a secondary sort. This means that we first apply the primary sort on a particular criterion and for advertisements that are ranked the same we apply the secondary sort (on one of the remaining criteria).

Left justified heuristic. The first heuristic that we consider iterates through the list of available advertisements. For each advertisement, it scans the columns of the banner from top to bottom. If the end of a column is reached the algorithm continues to the next column on the first row. This process is then repeated for each advertisement. When an empty location has been found and the advertisement size fits this location, the advertisement is placed on the banner with the top left corner at the current position. If the advertisement goes horizontally out of bounds, we are unable to get it allocated and continue with the next advertisement. In contrast, if the advertisement goes vertically out of bounds, we move the 'current location' to the first row of the next column. The algorithm stops once we iterated through all available advertisements.

Orthogonal heuristic. In the orthogonal algorithm we iterate through the list of available advertisements and place advertisements as close as possible to the top left corner. The algorithm looks for empty locations for the current advertisement by moving along the diagonal from the top left corner of the banner. At each step, the algorithm first searches, on the right and the bottom of the current location, for empty locations where the advertisement can be allocated. After that, we compare the two found free locations (one on the right of the current location and one below the current location) with respect to the sum

of the distances to the top and to the left. We allocate the currently considered advertisement on the position that yields the smallest sum. When there is a tie we choose the one on the vertical search path. After the advertisement is allocated, we start again in the top-left corner of the banner, trying to allocate the next advertisement.

If we fail to allocate an advertisement for a certain location, we continue to walk along the diagonal. When the final row is reached, and there are still columns left, we deviate from or walk on the diagonal and move to the next column. When the final column is reached, and there are still rows left, we move to the next row. This means that after we start walking diagonally, we will eventually switch to walking either right or down, except for the situation in which the banner is a square. When the final row and column are reached and we have failed to allocated the currently considered advertisement, we start again in the top left corner of the banner and try to allocate the next available advertisement. The algorithm stops once we iterated through all available advertisements.

GRASP constructive heuristic. The GRASP constructive algorithm is based on the greedy randomized adaptive search procedure (GRASP) for the constrained two-dimensional non-guillotine cutting problem [1]. It has a different approach than the algorithms discussed previously. Instead of searching the banner for free space to place the currently considered advertisement, it considers the rectangles of free space and finds matching advertisements that fit into these free spaces.

Initially, the full banner is considered as one large free space. First, we take the smallest rectangle of free space in which an advertisement that has not yet been placed can fit, after which we place the corresponding advertisement in this smallest rectangle of free space. Whenever an advertisement is placed in a rectangle, new free rectangles are formed and added to set of free spaces, while the original rectangle is marked as non-free. We always place the advertisement in a corner of the rectangle which is closest to a corner of the banner, and cut the free space left in such a way that it yields optimal new free rectangles. In order to obtain the optimal new free rectangles we choose to merge rectangles in such a way that the largest rectangle can accommodate the next advertisement. If there is a tie, we choose the merge which yields a new free rectangle with the largest area. We mark an empty location as used whenever we fail to allocate an advertisement to it, otherwise the algorithm could fall into an endless loop. After we have processed a rectangle we continue with the next smallest rectangle that has not been used before. When there are no free rectangles left (i.e., the whole banner is allocated) or no more available advertisements that fit any of the remaining rectangles, the algorithm stops.

We propose some modifications to the original GRASP approach that aim to decrease the total execution time. Differently from the original approach, we merge immediately adjacent free rectangles after we obtain new free rectangles. This is done in order to increase the probability of allocating advertisements to free rectangles.

3.3 Web Implementation

For the Web implementation we need a different approach than presented in the previous section, since we have to deal with an additional number of constraints. First, the Web implementation has to be fast and accurate, since users on the Web expect good results and do not want to wait too long. Second, it must be easy for users to submit their advertisement data and conflicting products so that this information can be used by the algorithm.

The Web application is available on `http://pixmax.damirvandic.com`. The software is implemented in Java. For the trade-off between the execution time and the quality of the solution, we set the maximum processing time to 15 seconds. This setting seems to give good results for a relatively low amount of processing time. Figure 2 shows the result for a 728×90 banner, with the orthogonal algorithm and sorting on price per pixel and proportionality. As we can see in this example, there are no conflicting items on the banner.

Users are also able to upload a zip file with images, a configuration file, and a domain ontology containing different product groups. The Web application provides the user with an example ontology that covers the beverages domain. The relationships between objects in the ontology specify how ontology individuals are related to each other. The relations used in the beverages ontology is the subsumption relation. By the use of the subsumption relation a taxonomy (a tree-like structure) of products is created. It is possible to create conflicts on all levels in the tree except the root.

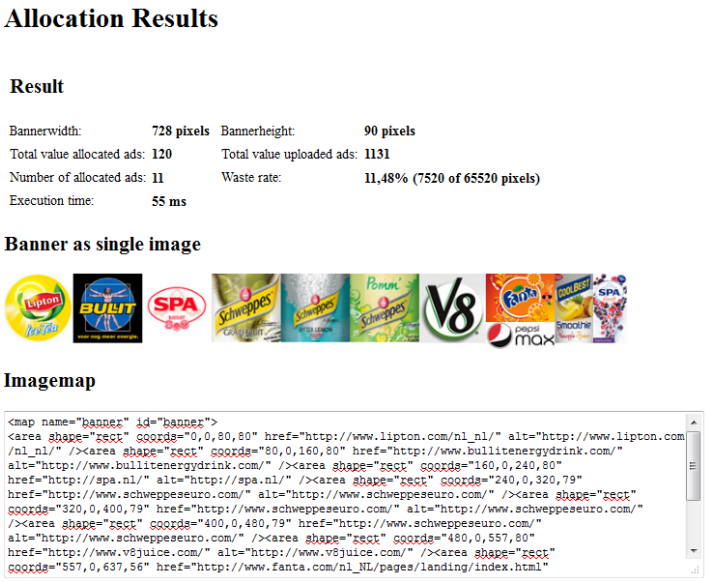

Fig. 2. A screenshot of our Web application, where users can generate optimally allocated banner advertisements.

4 Evaluation

In this section, the results of our experiments are discussed. First, we give more insight in the dataset we have used and our experimental setup, after which we discuss the performance of the considered heuristics.

4.1 Dataset

In order to test our implementation, we built a dataset that contains 113 advertisements of different sizes for non-alcoholic beverages. The price per pixel of each advertisement is drawn from a uniform distribution between 9 and 11. We chose for non-alcoholic beverages, because it offers a large variety of products. The ontology contains 34 different product groups, with 23 products groups as leaf nodes that contain product instances. An excerpt of the ontology and its structure is shown in Figure 3.

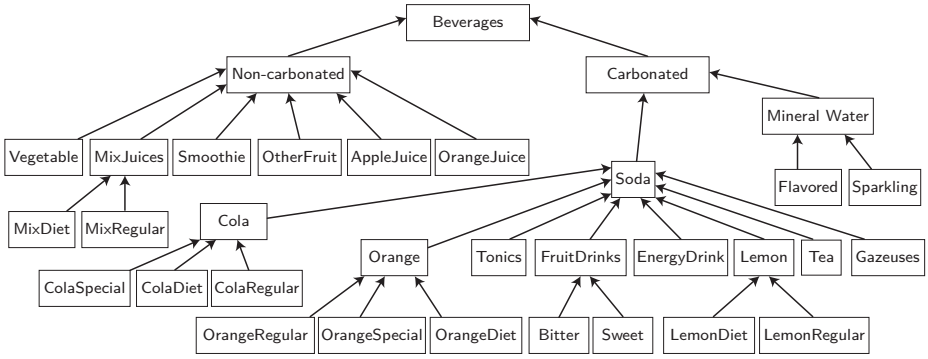

Fig. 3. An excerpt of the product ontology that is used in our approach. Every arrow represents an 'is-a' relationship between two concepts.

The individuals in our ontology, also known as instances, used in this study are concrete objects. For example, beverages from the Coca-Cola Company and PepsiCo Inc. are represented, because these companies have clear competitive products, which could cause conflicts in the advertisement, and that should be avoided. The classes, or concepts, represent the types of individuals. Each class has a number of individuals. In the advertisement banner, only one individual from each class can be represented. We consider here only the first two levels, i.e., the leaf classes and their parents. For example, the class Cola contains the individuals Coca-Cola and Pepsi, when Coca-Cola is on the advertisement it is not possible to have Pepsi on the advertisement banner as well. This is one example of conflicting advertisements, which is given here for illustrative purposes. Other types of conflicts can be defined and incorporated in the domain ontology and our application.

4.2 Experimental setup

For each of the three algorithms, we tested all combinations between the 121 sorting combinations and the 9 banner sizes. The 121 sorting combinations follow from the fact that we have 6 fields (price per pixel, width, height, total area, flatness, and proportionality), two directions (ascending and descending), and one random order, which yield $\frac{(6 \cdot 2)!}{10!} - 12 + 1 = 121$ combinations. This can be considered as choosing a pair from 12 options (6 fields $\times$ 2 directions) where the order matters and no duplicates are allowed. Our experimental setup resulted in 3267 simulation cycles (121 sorting combinations $\times$ 9 banner sizes $\times$ 3 algorithms), resulting in a total run time of 5 hours on a AMD Phenom II X4 810 with 4GB of memory.

We have chosen to use 5 banner sizes that are commonly used in Web advertising [8]. The five banner sizes (expressed in $w \times h$) are 728×90 (leader board), 234×60 (half banner), 125×125 (square button), 120×600 (skyscraper), and 336×280 (large rectangle). Because we also consider inverting every banner size, and one banner size is square, we have a total of $2 \times (5 - 1) + 1 = 9$ banner sizes.

4.3 Results

For overall performance, we consider the mean and five point statistics summary for the profit per pixel and the execution of all simulations, without taking the banner size or sorting into account. For this purpose, we take an average over all banner sizes and sorting combinations for each heuristic. Table 1 shows an overview of these results. As we can see, the orthogonal algorithm performs best on account of profit per pixel, closely followed by the left justified algorithm. The profit per pixel for the GRASP constructive algorithm is significantly lower than that of the first two.

Table 1. Mean and five point summary of the profit per pixel for each algorithm.

$Algorithm$	$Minimum$	Q_1	$Median$	$Mean$	Q_3	$Maximum$
Left justified	3.620	5.130	6.130	6.347	7.720	9.590
Orthogonal	3.620	5.200	6.250	6.439	7.720	9.590
GRASP con.	0.520	1.180	2.060	2.949	4.590	9.240

Table 2 shows the average execution time for each of the algorithms. The GRASP constructive algorithm performs best on execution time, since all the values of the five point summary and the mean are below 1 second. The execution time for the other two algorithms are significantly higher, where the left justified algorithm scores better in almost all of the cases. The orthogonal algorithm scores only better on the maximum value which is 35.7 seconds compared to 38.2 seconds with the left justified algorithm.

Table 2. Mean and five point summary of the execution time in seconds for each algorithm.

$Algorithm$	$Minimum$	Q_1	$Median$	$Mean$	Q_3	$Maximum$
Left justified	0.090	2.480	4.210	6.454	6.890	38.200
Orthogonal	1.891	4.384	7.188	9.602	12.620	35.700
GRASP con.	0.003866	0.009210	0.017000	0.019610	0.024560	0.125300

From Tables 1 and 2 we can conclude that the GRASP algorithm is fast but performs poorly. The left justified and orthogonal algorithms have the same performance with respect to profit per pixel, but the left justified algorithm is in general faster. Furthermore, the interquartile range (IQR), defined as $Q_3 - Q_1$, is 4.41 and 8.24 for the left justified and orthogonal algorithm, respectively. The lower IQR of the left justified algorithm indicates that the variance of the execution time is lower than the orthogonal algorithm, which is an advantage because it allows for more precise predictions of the execution time.

Sorting Criteria. The heuristics are influenced significantly by the sorting of the incoming advertisements. Table 3 shows the profit per pixel and the execution time, averaged over all banner sizes and secondary sorting combinations, for combinations between the 13 primary sorting combinations and the three algorithms. The table shows only combinations that have a price per pixel higher than 9. The results show again that the orthogonal algorithm and the left justified perform much better than the GRASP constructive algorithm. We observe that sorting descending on the price per pixel (PPP) property gives the best results for all three algorithms. We also observe that for most of the sorting combinations the descending order performs better than the ascending order. This can be explained by the fact that for most of the values we sort on, the highest values add the most value to the banner.

We also notice that dependency on the sorting criteria for the GRASP algorithm differs from the patterns we encounter for the left justified and orthogonal algorithms. The GRASP algorithm seems to be less dependent on the sorting criteria, as only sorting on price per pixel yields a total price per pixel of 9.24, while sorting on other properties yields the same price per pixel of 9.11. This might indicate that the GRASP algorithm, while having a lower overall average price per pixel, is more robust than the left justified and orthogonal algorithms.

Banner sizes. Table 4 shows for each considered banner size and for each algorithm, the price per pixel and the total execution time. For each banner size, the rows are sorted on the average price per pixel. The results indicate that for most of the banner sizes both the orthogonal and the left justified algorithms give the same results on profit per pixel, although

the left justified algorithm has a lower or equal execution time in all cases (which was also clear when we considered the Q_1 and Q_3 of the execution time).

Table 3. The evaluation results based on the primary sorting and considered algorithms.

Algorithm	*Primary sort*	*Exec. time (s)*	P_{pixel}
Orthogonal	PPP Desc.	11.143	9.59
Left-Justified	PPP Desc.	4.200	9.59
Orthogonal	Width Desc.	20.931	9.48
Left-Justified	Width Desc.	17.350	9.48
Orthogonal	Total Size Desc.	22.932	9.32
Left-Justified	Total Size Desc.	15.240	9.32
GRASP	PPP Desc.	0.007	9.240
Orthogonal	Height Desc.	17.084	9.11
Left-Justified	Proportional Asc.	11.00	9.11
Left-Justified	Height Desc.	11.450	9.11
GRASP	Proportional Asc.	0.006	9.11
GRASP	Total Size Desc.	0.006	9.11
GRASP	Height Desc.	0.006	9.11
GRASP	Width Desc.	0.006	9.11
Orthogonal	Proportional Asc.	16.626	9.11

The GRASP constructive algorithm is the fastest, while the price per pixel is lower than or equal to the price per pixel of the other two algorithms (for all banner sizes, except for the 60×234 banner).

In order to get a better sense of the difference in effectiveness (i.e., profit) between the left justified and the orthogonal algorithm, we compute the total profit if all the banners that are displayed in Table 4 would be sold to advertisers. As a result, the left justified algorithm would bring in €4,335,139 and the orthogonal algorithm €4,337,246. As we can see, the difference is negligible on such a high total amount. These results support our previous claim that with respect to profit, the left justified and orthogonal algorithms do not differ much. Because of this, and the fact that the left justified algorithm has a IQR lower execution time, we prefer the left justified algorithm over the orthogonal algorithm.

5 Conclusions

This paper presents an extension to the pixel advertising concept. We focused on avoiding conflicting advertisements on the same banner, using an ontology and heuristic algorithms adopted from [3], including the left justified algorithm, the orthogonal algorithm, and the GRASP constructive algorithm. The results of the experiments indicate that the left justified and orthogonal algorithm are most effective, which means that these algorithms give the highest price per pixel. The most efficient is the GRASP constructive algorithm, however the price per pixel is significantly lower for this algorithm. Furthermore, the left justified algorithm has on average a lower execution time than the orthogonal algorithm. Therefore, the left justified algorithm is considered to be the best choice when it

Table 4. The evaluation results for each of the banner sizes.

Size	*Algorithm*	*Sorting*	*Exec. time* (s)	P_{pixel}
728×90	Orthogonal	PPP Desc. & Proportional Asc.	14.106	9.59
	Left-Justified	PPP Desc. & Proportional Asc.	11.620	9.59
	GRASP	PPP Desc. & Width Desc.	0.034	5.26
600×120	Orthogonal	Width Desc. & PPP Desc.	12.341	9.02
	Left-Justified	Width Desc. & PPP Desc.	11.140	9.02
	GRASP	PPP Asc. & Height Asc.	0.030	3.47
336×280	Left-Justified	Total Size Desc. & Height Desc.	13.280	7.85
	Orthogonal	Total Size Desc. & Height Desc.	13.970	7.85
	GRASP	PPP Desc. & Proportional Asc.	0.038	7.02
280×336	Left-Justified	Total Size Desc. & Height Desc.	13.050	7.85
	Orthogonal	Total Size Desc. & Height Desc.	13.763	7.85
	GRASP	PPP Desc. & Proportional Asc.	0.041	7.17
234×60	Orthogonal	PPP Desc. & Height Asc.	7.112	7.07
	Left-Justified	PPP Desc. & Height Asc.	1.370	7.07
	GRASP	PPP Desc. & Width Asc.	0.009	7.02
125×125	Left-Justified	PPP Desc. & Proportional Asc.	10.860	9.24
	Orthogonal	PPP Desc. & Proportional Asc.	15.778	9.24
	GRASP	PPP Desc. & Proportional Asc.	0.007	9.24
120×600	Orthogonal	Width Desc. & PPP Desc.	21.718	8.51
	Left-Justified	Width Desc. & PPP Desc.	19.580	8.51
	GRASP	Width Desc. & PPP Asc.	0.0553	7.37
90×728	Left-Justified	PPP Desc. & Proportional Asc.	4.200	9.59
	Orthogonal	PPP Desc. & Proportional Asc.	11.143	9.59
	GRASP	PPP Asc. & Width Desc.	0.049	7.88
60×234	GRASP	Height Asc. & PPP Desc.	0.008	6.96
	Orthogonal	Height Asc. & PPP Desc.	6.573	6.81
	Left-Justified	Height Asc. & PPP Desc.	1.310	6.81

comes to avoiding conflicts in placing pixel advertisements on a banner while maximizing banner revenue.

The key contribution of this study is that we use a domain ontology to manage conflicts when placing advertisements in a banner. Furthermore, we incorporate this approach in three existing heuristics, which we then compare and evaluate. We can conclude that, from the currently available algorithms, the best performing are the left justified and orthogonal algorithms.

As future work, the use of pixel advertising can be made more attractive by coupling the content of the banner to the content of the Web page it is shown on. In this way, the price per pixel can be increased as the advertisements are more useful when they are custom tailored to specific target groups. Another technical approach would be the use of degrees of conflict, where products that are not direct substitutes of each other have a degree of conflict lower than one and are allowed on the same banner given that they are separated by some predefined minimum distance. This might allow for better coverage of the advertisement banner.

Acknowledgment

The authors are partially sponsored by the NWO Mosaic project 017.007.142: Semantic Web Enhanced Product Search (SWEPS).

References

[1] Alvarez-Valdes, R., Parreño, F., Tamarit, J.M.: A GRASP algorithm for constrained two-dimensional non-guillotine cutting problems. The Journal of the Operational Research Society 56(4), 414–425 (2005)

[2] Beasley, J.: An exact two-dimensional non-guillotine cutting tree search procedure. Operations Research 33(1), 49–64 (1985)

[3] Boskamp, V., Knoops, A., Frasincar, F., Gabor, A.: Maximizing revenue with allocation of multiple advertisements on a web banner. Computers & Operations Research 38, 1412–1424 (2011)

[4] Budak Arpinar I., Karthikeyan Giriloganathan, B.A.M.: Ontology quality by detection of conflicts in metadata. In: 4th International Workshop on Evaluation of Ontologies for the Web (EON 2006) (2006)

[5] Google: Financial tables 2014, http://investor.google.com/financial/tables.html

[6] Hadjiconstantinou, E., Christofides, N.: An exact algorithm for general, orthogonal, two-dimensional knapsack problems. European Journal of Operational Research 83(1), 39–56 (1995)

[7] Hof, R.: Online ad revenues to pass print in 2012. http://www.forbes.com/sites/roberthof/2012/01/19/online-ad-revenues-to-pass-print-in-2012/

[8] Interactive Advertising Bureau: Ad unit guidelines. http://www.iab.net/iab_products_and_industry_services/1421/1443/1452

[9] Knoops, A., Boskamp, V., Wojciechowski, A., Frasincar, F.: Single pattern generating heuristics for pixel advertisements. In: 10th International Conference on Web Information Systems Engineering (WISE 2009). pp. 415–428 (2009)

[10] Nah, F.F.H.: A study on tolerable waiting time: how long are web users willing to wait? Behaviour & Information Technology 23(3), 153–163 (2004)

[11] OECD: Competition: Economic issues. http://www.oecd.org/

[12] Tew, A.: Million dollar homepage. http://www.milliondollarhomepage.com/

[13] Wäscher, G., Haussner, H., Schumann, H.: An improved typology of cutting and packing problems. European Journal of Operational Research 183(3), 1109–1130 (December 2007)

[14] Wojciechowski, A.: An improved web system for pixel advertising. In: 6th Conference on E-Commerce and Web Technologies (EC-WEB 2006). pp. 232–241 (2006)

[15] Wojciechowski, A., Kapral, D.: Allocation of multiple advertisement on limited space: heuristic approach. In: Future Multimedia Networking, pp. 230–235. Springer (2009)