Single Pattern Generating Heuristics for Pixel Advertisements

Alex Knoops[1], Victor Boskamp[1], Adam Wojciechowski[2], and Flavius Frasincar[1]

[1] Econometric Institute
Erasmus University Rotterdam
PO Box 1738, NL-3000
Rotterdam, the Netherlands
{alex.knoops, victorboskamp}@gmail.com, frasincar@ese.eur.nl
[2] Institute of Computing Science
Poznan University of Technology
ul. Piotrowo 2, 60-965 Poznan, Poland
adam.wojciechowski@put.poznan.pl

Abstract. Pixel advertisement represents the presentation of small advertisements on a banner. With the Web becoming more important for marketing purposes, pixel advertisement is an interesting development. In this paper, we present a comparison of three heuristic algorithms for generating allocation patterns for pixel advertisements. The algorithms used are the orthogonal algorithm, the left justified algorithm, and the GRASP constructive algorithm. We present the results of an extensive simulation in which we have experimented with the sorting of advertisements and different banner and advertisement sizes. The purpose is to find a pattern generating algorithm that maximizes the revenue of the allocated pixel advertisements on a banner. Results show that the best algorithm for our goal is the orthogonal algorithm. We also present a Web application in which the most suitable algorithm is implemented. This Web application returns an allocation pattern for a set of advertisements provided by the user.

Key words: pixel advertisements, allocation patterns, heuristic algorithms

1 Introduction

With the Web usage still growing, Web advertising becomes a more dominant form of marketing every year. According to the Interactive Advertising Bureau, Web advertising revenues for 2008 are totaled $23.4 billion in the U.S. only [1]. Banner advertisements have a 22 percent share in these figures.

A special form of Web advertising is *pixel advertisement*. Pixel advertisement originated in 2005 from the English student Alex Tew's "Million Dollar Homepage" [2]. The homepage holds a 1000 by 1000 pixel grid from which blocks of 10 by 10 pixels could be bought for 1 dollar per pixel. Buyers could place an image on their pixels and let the image link to their website. The general idea of pixel advertising is to have a *banner* with several small advertisements in pixel blocks (i.e., multibanner), instead of just one advertisement occupying the banner.

In [3] the success of the "Million Dollar Homepage" and the failure of the many copycats that arose is analyzed. Since visitors do not return to the "Million Dollar Homepage" the paper proposes some improvements to the concept of pixel advertisement in Web pages. In [4] the authors extend the idea of pixel advertisement to placing small ads in banners. In this paper, we build upon their results, generalizing and thoroughly evaluating the proposed solutions.

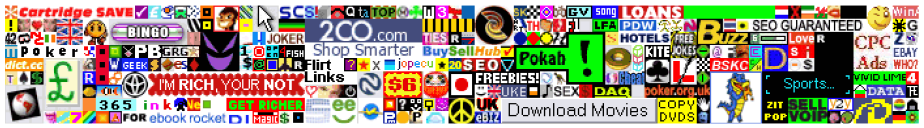


Fig. 1. Sample from "Million Dollar Homepage"

The research question tackled in this paper can be defined as follows: *how to arrange rectangular pictures of different sizes and different prices for advertisement on a banner, in order to maximize revenue?* An important assumption we make is that we have a predefined set of advertisements that can be placed on the banner. This differs from the "Million Dollar Homepage" approach, where buyers just select free pixel blocks they want to purchase. In that case, there is no arrangement necessary and the problem tackled in this paper is nonexistent. Another assumption is that the banner size is given and that the set of advertisements should contain more advertisements than would fit on the banner. Note that even if the ads fit on the banner, the placement is still a problem. Furthermore, the problem we face is a static allocation problem with no dynamic dimension like time-sharing of advertisements.

Finding the optimal allocation of advertisements in a banner may be defined as a *two-dimensional, single, orthogonal, knapsack problem* [5]. The problem is NP-hard [6], making it extremely time-consuming to find the optimal solution(s) using integer programming. In this paper, we focus on applying heuristics to find adequate solutions.

We use three heuristic algorithms to allocate advertisements. For this, we experiment with sorting of advertisements and use different banner sizes. Our main objective is to find a heuristic algorithm that generates advertisement allocation patterns that maximize profit. Our secondary objective is to create a Web application. Therefore, a good performance also requires that the execution time is acceptable (i.e., within 30 seconds) to users of the Web application.

Related work on the placement of Web advertisements has been focusing on the *ad placement problem*, introduced in [7] as a variant of the bin packing problem. Despite the name, the most important feature of the ad placement problem is time scheduling of advertisements on a banner in *time slots*. Furthermore, it is concerned only with the placement of one advertisement on a banner or some advertisements side-by-side, whereby the height of the advertisements is equal to the height of the banner.

In [7], a distinction is made between the *offline* and *online* scheduling of advertisements. In the offline problem, we have a predefined set of advertisements to be scheduled. In the online problem, requests for placement arrive sequentially and we have to decide whether to accept requests without knowledge of future ones.

Another distinction made concerns the MINSPACE and MAXSPACE problems. The MINSPACE problem minimizes the banner size required for allocating a given set of advertisements in a fixed amount of time slots. The MAXSPACE problem maximizes the total profit given a fixed banner size and a fixed amount of time slots, which provide not enough free space for allocating all advertisements. For both problems, several solutions are available using polynomial time approximation algorithms [8,9,10], Lagrangian decomposition [11,12], column generation [12], and a hybrid genetic algorithm [13]. The approach presented in this paper is different from the ad placement problem, since we do not take into account time scheduling and we allocate advertisements not only side-by-side but also two-dimensional. Based on the previous classifications we are dealing here with an offline and MAXSPACE problem.

The rest of the paper is organized as follows: In Sect. 2 and 3 the simulation variables and allocation algorithms are defined. An analysis of the results is presented in Sect. 4. Section 5 discusses the implementations of our approach in a Web application. Section 6 concludes the paper and identifies future research directions.

2 Simulation

In order to obtain unbiased results in finding the most suitable environment for our purposes, we tested the allocation algorithms in a simulation using different configuration parameters, which defined the properties of each simulation cycle. These parameters consisted of 9 different banner sizes, 120 different sortings of the set of advertisements, and 6 different maximum sizes of the advertisements for each of the 3 algorithms. Each combination of configuration parameters represents a single simulation cycle. Altogether this resulted in 19440 simulation cycles. The details of these configuration parameters are described in the next few paragraphs. During each simulation cycle, one set of advertisements was allocated to one banner. The complete simulation was implemented in Matlab and run as a single batch file. All simulations were done on a Intel Core 2 Duo CPU P8400 at 2.26 GHz.

Size of the banner. Five standard banner sizes [14], commonly used in Web advertising, have been selected to be used for each of the simulation cycles. The width W and height H of the banners are shown in Table 1. During the simulation the widths and the heights of the banners are also reverted to avoid bias towards particular sorting of the set of advertisements or banner dimensions. In total this amounts to 9 different banners (the square banner need not be reverted).

Table 1. Standard banner sizes

$W \times H$	Banner
728×90	Leader board
234×60	Half Banner
125×125	Square Button
120×600	Skyscraper
336×280	Large Rectangle

Price of the banner. In practice, an existing banner may already generate revenue. One of the attributes of the banner is its price. This price however, will be set to a single fixed price during the simulations. This is done in order to avoid ambiguous results, in which it may not be clear if profit comes from the original banner or the allocated advertisements. During the simulations the price per pixel for the banner has been set to 4 which is much lower than the price range per pixel of the advertisements. This is done in order to avoid that no advertisements are allocated, when the banner generates more revenue than any of the advertisements.

Size of the advertisements. For our simulation the advertisements where pseudo-randomly generated. The minimum width and height are 10 pixels, like the implementation on the "Million Dollar Homepage". In our experiment we allow the dimensions of the advertisement to vary between a minimum of 10 pixels and a variable maximum. The maximum width w_{max} and height h_{max} are defined as fraction of the banner width and the banner height. For this simulation the combinations of the maximal width and the maximal height are:

$$\{w_{max}, h_{max}\} \in \{\{1/5, 1/2\}, \{1/2, 1/2\}, \{1/3, 1/3\}, \{1/5, 1/5\}, \{1/2, 1/5\}, \{1, 1\}\}$$

Sorting of advertisements. The heuristic algorithms iterate through the set of advertisements sequentially. The sorting of the set influences the generated pattern and is part of the heuristics. The simulation uses the following attributes of the advertisements to sort the set: (1) price per advertisement pixel p, (2) width w, (3) height h, (4) total area $w \times h$, (5) flatness w/h, and (6) the proportionality $|\log(w/h)|$, the last attribute refers to how much the rectangle resembles a square. A value of 0 for this attribute means that the rectangle is a square. Any higher value signifies that the rectangle is flat or tall. The sorting can be done in either *ascending* or *descending* order.

Once the set has been sorted based on the values of the attributes, a secondary sort is executed using one of the remaining attributes. The secondary sort has a minor influence on the resulting ordered set. Altogether the set of advertisements is sorted in 120 different ways, ($\frac{12!}{10!} - 12 = 120$, since we want to exclude the situations were the primary sort equals or is opposite of the secondary sort).

The prices of the advertisements are proportional to their dimensions. The price per pixel of an advertisement is set to 10 with random value between -1 and 1 added to this value, resulting in a uniform distribution between 9 and 11. The price of the advertisement is calculated by multiplying this price per pixel with its area.

During each cycle of the simulation, the configuration parameters are registered. For each cycle the waste rate (ratio of unallocated space over the total space in the banner) and the total profit of the generated allocation pattern are calculated. The execution time and the number of advertisements placed are also registered.

3 Heuristic algorithms

We implemented three different heuristic algorithms: the *left justified*, the *orthogonal* algorithm, and the *GRASP constructive algorithm*.

The initialization step is identical for all algorithms. The algorithms assume a banner B with width w_B and height h_B. First the values of the primary and secondary sort, s_1 and s_2 are checked. Their values correspond to the attributes described in Sect. 2 and may be either positive or negative corresponding to an ascending and descending sorting order. There is a set A with n advertisements a_i where, $1 \leq i \leq n$. Sorting A according to s_1 and s_2 yields A_0. This is the ordered set of advertisements through which we iterate in each algorithm. Furthermore, the iterator i for the ordered set of advertisements A_0 is initialized at 1. The initiation step is given in Alg. 1.

Algorithm 1 Heuristic algorithm initialization

```
Ensure: $s_1 \neq s_2$ & $s_1 \neq -s_2$ {Avoid duplicate sorting in either direction}
    Sort all $a_i$ in $A$ first by $s_1$ and then by $s_2$
    $A_0$ {Ordered set $A$}
    $i := 1$ {Iterator for $A_0$}
```

Left justified algorithm. The *left justified algorithm* iterates through the ordered set of advertisements A_0. For each advertisement a_i it scans through the columns of the banner B from top to bottom. If the end of the column is reached, the iterator continues at the next column on the first row, and so on. When an available field is found and the advertisement fits on the empty location, it is placed in the banner. Advertisements are placed with the top left corner at the current field. When the end of ordered set A_0 is reached or when the banner is completely filled, the allocation pattern is returned. The details of this algorithm are shown in Alg. 2.

Algorithm 2 *Left justified* algorithm

```
for $i = 1$ to $n$ do
  Select $a_i$ from $A_0$
  $finished := false$
  $r := 1$ {Current row in $B$}
  $c := 1$ {Current column in $B$}
  while $finished = false$ do
    if $a_i$ fits on $B_{r,c}$ then
      {Allocate $a_i$ on $B_{r,c}$}
      for $p = c$ to $c + x_i$ do
        for $q = r$ to $r + y_i$ do
          $B_{p,q} := i$
        end for
      end for
      $finished := true$
    else if $r + y_i > h_B$ then
      if $c < w_B$ then
        $c := c + 1$
        $r := 1$
      else
        $finished := true$
      end if
    else if $r + x_i > w_B$ then
      $finished := true$
    else
      if $r < h_B$ then
        $r := r + 1$
      else
        if $c < w_B$ then
          $c := c + 1$
          $r := 1$
        else
          $finished := true$
        end if
      end if
    end if
  end while
end for
return $B$
```

Orthogonal algorithm. The *orthogonal algorithm* looks for new free locations for the current advertisement by moving diagonally from the top left corner $(r, c) = (1, 1)$ of banner B.

At each step, the algorithm searches for the next free space where the advertisement can be allocated at the location $(r, i)\,, i \in \{1 \ldots c\}$ and $(i, c)\,, i \in \{1 \ldots r\}$. At the first free location closest to the border of the banner the advertisement a_i is allocated. When there is a tie we choose the one on the vertical

Algorithm 3 *Orthogonal* algorithm

```
for $i = 1$ to $n$ do
  select $a_i$ from $A_0$
  $r := 1, c := 1$
  $verticalfound := false$, $horizontalfound := false$
  $verticalplace := (0,0)$, $horizontalplace := (0,0)$
  $colscomplete := false$, $rowscomplete := false$
  while ($colscompl$ && $rowscompl$) $= false$ do
    if $colscomplete = false$ then
      for $p = 1$ to $r$ do
        if $a_i$ fits on $B_{c,p}$ then
          store $(c,p)$ in $verticalplace$, $verticalfound := true$, break
        end if
      end for
    end if
    if $rowscomplete = false$ then
      for $q = 1$ to $c$ do
        if $a_i$ fits on $B_{q,r}$ then
          store $(q,r)$ in $horizontalplace$, $horizontalfound := true$, break
        end if
      end for
    end if
    if $horizontalfound = true$ or $verticalfound = true$ then
      {Select location closest to left or upper border}
      {Assume selected location is $B_{k,l}$: allocate $a_i$ on this location}
      for $p = k$ to $k + x_i$ do
        for $q = l$ to $l + y_i$ do
          $B_{p,q} := i$
        end for
      end for
    end if
    if $r < h_B$ then
      $r := r + 1$
    else
      $rowscomplete = true$
    end if
    if $c < w_B$ then
      $c := c + 1$
    else
      $colscomplete = true$
    end if
  end while
end for
return $B$
```

search path. When we fail to allocate an advertisement for a certain (r, c) we continue to walk diagonally down-right by increasing both r and c by one. When the final row is reached, but there are still columns left, we only increase the column. When the final column is reached, but there are still rows left, we only increase the row. This means that after we start walking diagonally, we will eventually switch to walking either right or down, except for the situation when the banner B is a square.

When a_i is allocated, we start again in the top left corner of the banner and try to allocate the next advertisement from A_0. The details of this algorithm are shown in Alg. 3.

GRASP constructive algorithm. The *GRASP constructive algorithm*, is based on the constructive phase of the greedy randomized adaptive search procedure (GRASP) for the constrained two-dimensional non-guillotine cutting problem [15]. Since the algorithm was produced for the cutting stock problem, it has a somewhat different approach. We have adapted the algorithm to fit our problem.

In the GRASP algorithm, besides an ordered set of advertisements A_0, a list of empty rectangles L is maintained. Empty rectangles are parts of the banner where no advertisement is allocated yet. Initially, list L contains only the full banner. To allocate advertisements, the following procedure is followed.

First, we take the smallest rectangle of L in which an advertisement from list A_0 can fit. Then, we place an advertisement a_i from ordered set A_0 that fits in the free rectangle. Whenever an advertisement is placed in a rectangle, new empty rectangles are formed and added to L, while the original rectangle is removed from L. We always place the advertisement in a corner of the rectangle which is closest to a corner of the banner, and cut the empty space left in such a way that it yields optimal new free rectangles. In Fig. 2 the empty rectangles 1, 2, and 3 are formed by placing an advertisement. In order to obtain the optimal new empty rectangles we merge either empty rectangles 1 and 2, or 2 and 3. We choose the combination which yields the largest area for the merged rectangle. When there are no empty rectangles left (L is empty, the full banner is allocated) or no advertisements from list A_0 fits the rectangles in L, the algorithm stops.

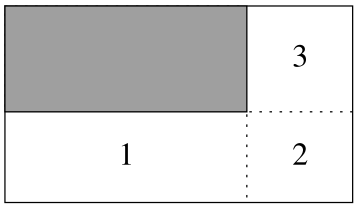


Fig. 2. Empty rectangles in GRASP algorithm

4 Analysis

The analysis of the simulation results was done with the tool R. This statistical software package allows all tasks to be automated in scripts. After the results are prepared, we normalized the profit and execution time by adding two extra columns with the profit per banner pixel $P_{pixel} = \frac{P_{total}}{BW \times BH}$ and the execution time per banner pixel $E_{pixel} = \frac{E_{total}}{BW \times BH}$. P_{total} is the total profit of the allocated pattern, E_{total} is the total execution time for the allocated pattern, BW is the banner width, BH is the banner height and w is the waste rate.

We are primarily interested in the profit per banner pixel of the allocation pattern P_{pixel}. The execution time is only relevant for the implementation of the algorithm. Since the same set of advertisements is used for all heuristic algorithms, we can evaluate their performance by comparing the normalized profits and execution times.

In Table 2 the distribution of the profit per banner pixel P_{pixel} is displayed for each of the algorithms. The *orthogonal* algorithm has resulted in a higher average profit per banner pixel and has been selected to be implemented in the Web application described in Sect. 5.

Table 2. Five point summary of the profit per banner pixel per algorithm

Algorithm	Minimum	1^{st} Quartile	Median	Mean	3^{rd} Quartile	Maximum
Orthogonal	6.079	8.585	9.082	8.887	9.427	10.620
Left justified	5.748	8.155	8.626	8.509	9.042	10.540
GRASP	4.730	6.978	8.044	7.962	9.083	10.600

As expected there is a strong correlation between the waste rate and the profit per banner pixel (P_{pixel}). The obtained value -0.9802 shows that a lower waste rate will result in a higher profit per banner pixel.

Banner size. From the dot chart in Fig. 3 the average profit per banner pixel is categorized by the dimensions of the banner. Besides the obviously better performance of the *orthogonal* algorithm, the graph shows that the banner size influences the performance of the algorithm. There is no solid evidence that a particular banner size benefits the performance of the algorithms.

Sorting. The preliminary sorting of the advertisements influences the final allocation pattern. From the dotchart in Fig. 4 it shows that sorting the advertisements based on their dimensions is of greater influence than sorting them based on their price. For each of the three algorithms, allocating the highest, the widest, or the advertisements with the largest total area first, yields the highest profit per banner pixel. This can be explained by the strong negative correlation between the waste rate and the profit per banner pixel. Furthermore, the figure shows that the *orthogonal* algorithm is less sensitive to the preliminary sorting of the advertisements, showing that the algorithm is more robust than the other ones.

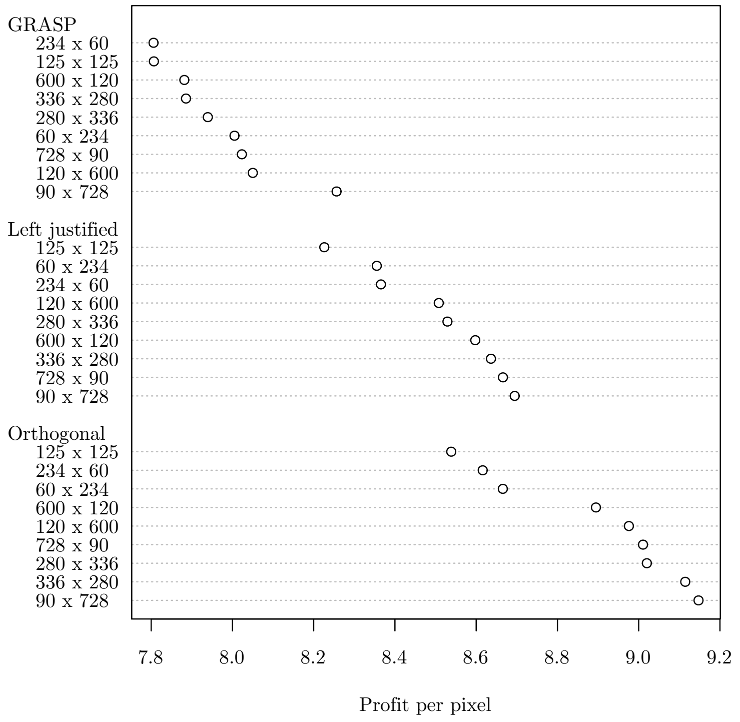


Fig. 3. Dotchart: profit per banner pixel for each banner size per algorithm

Execution times. The main interest in the execution time of the algorithm is a practical one. Though the *GRASP* algorithm shows the lowest execution times, the main issue is not to find the fastest algorithm, but merely one that is usable within the context of a Web application. Usable patterns of allocated advertisements should not come at the cost of waiting for more than 30 seconds for a result. In Table 3 the main characteristics of the execution times are displayed.

Though it is clear that the *orthogonal* algorithm has the highest execution times, its outliers are only around 20 seconds of execution time. During the migration to Java for the implementation of the Web application, the execution times decreased remarkably.

Table 3. Five point summary of the CPU time per algorithm in seconds

Algorithm	Minimum	1st Quartile	Median	Mean	3rd Quartile	Maximum
Orthogonal	0.016040	0.432000	2.632000	3.151000	4.745000	20.49000
Left justified	0.008244	0.156200	0.571000	0.956600	1.203000	17.25000
GRASP	0.003904	0.025610	0.072310	0.071180	0.099090	0.320000

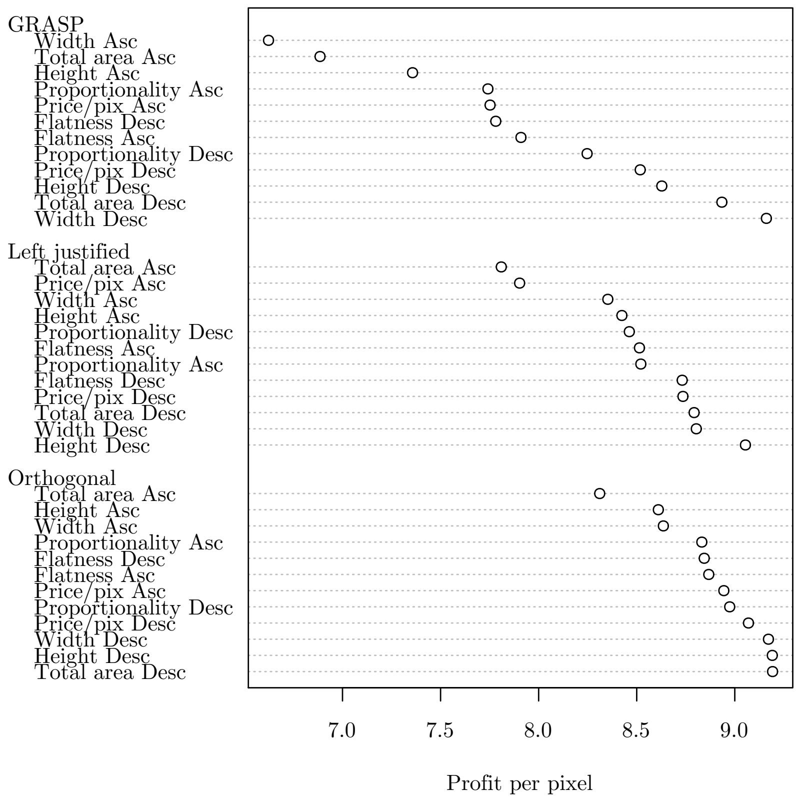
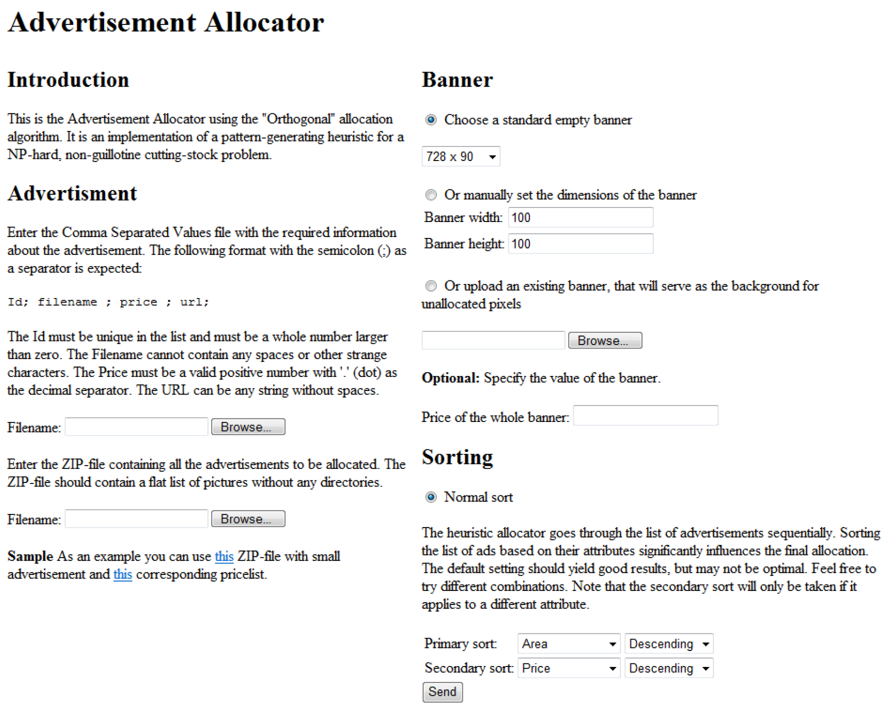


Fig. 4. Dotchart: profit per banner pixel for each primary sorting order per algorithm

5 Software

The implementation of the *orthogonal* heuristic algorithm as a Web application is available at `http://headshredder.homelinux.net:8080/java/`. It provides an allocation pattern for a set of advertisements using the orthogonal algorithm. A screenshot of the frontpage is displayed in Fig. 5.

Users can upload their own set of advertisements in the form of a comma separated values file and a zip file containing the pictures. The comma sepa-



Advertisement Allocator

Introduction

This is the Advertisement Allocator using the "Orthogonal" allocation algorithm. It is an implementation of a pattern-generating heuristic for a NP-hard, non-guillotine cutting-stock problem.

Advertisment

Enter the Comma Separated Values file with the required information about the advertisement. The following format with the semicolon (;) as a separator is expected:

```
Id; filename ; price ; url;
```

The Id must be unique in the list and must be a whole number larger than zero. The Filename cannot contain any spaces or other strange characters. The Price must be a valid positive number with '.' (dot) as the decimal separator. The URL can be any string without spaces.

Filename: Browse...

Enter the ZIP-file containing all the advertisements to be allocated. The ZIP-file should contain a flat list of pictures without any directories.

Filename: Browse...

Sample As an example you can use this ZIP-file with small advertisement and this corresponding pricelist.

Banner

● Choose a standard empty banner

728 x 90

○ Or manually set the dimensions of the banner

Banner width: 100

Banner height: 100

○ Or upload an existing banner, that will serve as the background for unallocated pixels

Browse...

Optional: Specify the value of the banner.

Price of the whole banner:

Sorting

● Normal sort

The heuristic allocator goes through the list of advertisements sequentially. Sorting the list of ads based on their attributes significantly influences the final allocation. The default setting should yield good results, but may not be optimal. Feel free to try different combinations. Note that the secondary sort will only be taken if it applies to a different attribute.

Primary sort: Area Descending

Secondary sort: Price Descending

Send

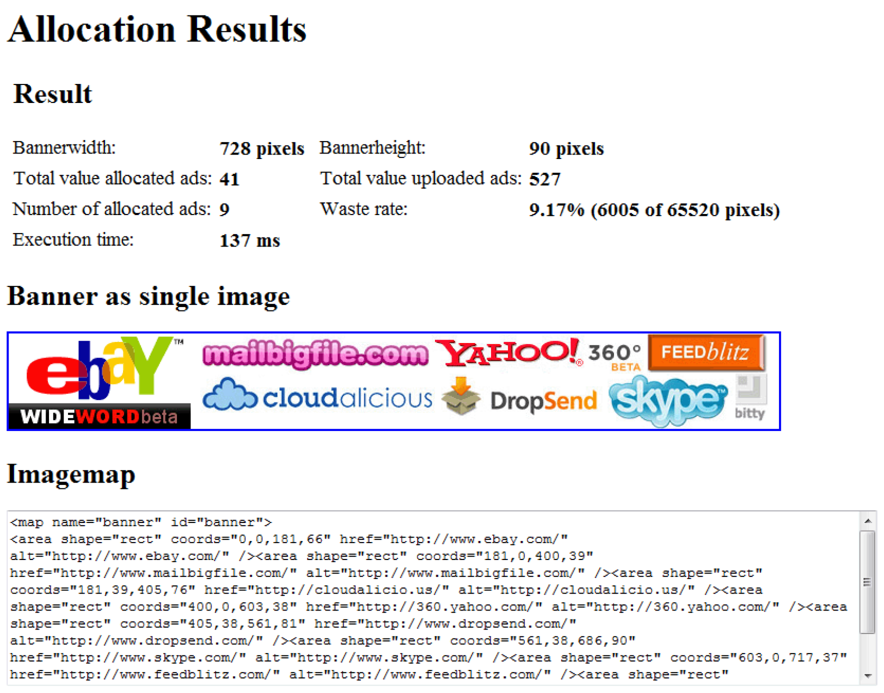
Fig. 5. Advertisement allocator frontpage

rated values file holds information on every advertisement in the format `id; filename; price; URL;`. The `filename` corresponds to a picture from the zip file, belonging to the advertisement. Furthermore, the user has to set parameters for the banner and the sorting criteria used. The user can select a standard banner size, manually set the dimensions, or upload an existing banner. For sorting, the user can specify the primary and secondary sorting criteria in either ascending or descending order.

The advertisement allocator provides the allocation results with some statistics. It returns the allocated banner as a single image and gives a corresponding imagemap in HTML. An imagemap is a list of coordinates relating to a picture. On the coordinates the specified `URL` is set. This makes it easy for Webmasters to implement their pixel advertisement banner. An example of such an result is displayed in Fig. 6.

6 Conclusion

Our main objective was to find a heuristic algorithm that generates advertisement allocation patterns that maximize profit. The best algorithm for our purposes is the *orthogonal* algorithm. Sorting the advertisements based on the width, height, and total area in descending order yields the best results. This algorithm



Allocation Results

Result

Bannerwidth:	**728 pixels**	Bannerheight:	**90 pixels**
Total value allocated ads:	**41**	Total value uploaded ads:	**527**
Number of allocated ads:	**9**	Waste rate:	**9.17% (6005 of 65520 pixels)**
Execution time:	**137 ms**		

Banner as single image

Imagemap

```
<map name="banner" id="banner">
<area shape="rect" coords="0,0,181,66" href="http://www.ebay.com/"
alt="http://www.ebay.com/" /><area shape="rect" coords="181,0,400,39"
href="http://www.mailbigfile.com/" alt="http://www.mailbigfile.com/" /><area shape="rect"
coords="181,39,405,76" href="http://cloudalicio.us/" alt="http://cloudalicio.us/" /><area
shape="rect" coords="400,0,603,38" href="http://360.yahoo.com/" alt="http://360.yahoo.com/" /><area
shape="rect" coords="405,38,561,81" href="http://www.dropsend.com/"
alt="http://www.dropsend.com/" /><area shape="rect" coords="561,38,686,90"
href="http://www.skype.com/" alt="http://www.skype.com/" /><area shape="rect" coords="603,0,717,37"
href="http://www.feedblitz.com/" alt="http://www.feedblitz.com/" /><area shape="rect"
```

Fig. 6. Advertisement allocator result

was able to generate the patterns with the highest profit. It did not have the lowest execution times, but these were still well within the predefined time boundaries. The Java implementation showed that its performance did not influence the Web application's responsiveness.

This research also uncovers possible future work directions. Our research is limited to the allocation algorithms we have used. Better results may be achieved when using more intelligent algorithms. These algorithms should consider a few steps ahead and reach a better allocation while keeping revenue in mind. The GRASP algorithm showed great promise as an efficient algorithm with its low execution times. The implementation of the improvement phase as suggested in the original paper [15] may prove worth the effort in the future. It may also be more realistic to give different positions on a banner different prices. In our research we have a predefined set of advertisements with different prices regardless of the position they get allocated. The Eyetrack III [16] research investigates people's eye movements over Web pages. More frequently watched areas in the banner may be assigned a higher price.

In [3] is described that the "Million Dollar Homepage" concept has some weak points. The major problem of the original concept is that visitors do not return, because the content is never changed. Making the content dynamic will increase the effectiveness of pixel advertisement. Therefore, we propose further research adding time constraints to the pixel advertisement problem. Until now, related work only focused on scheduling advertisements side-by-side. Instead, it may be interesting to schedule pixel advertisement banners. This will make the present, static pixel advertisements more dynamic and increase user attention.

References

1. Interactive Advertising Bureau: Internet Advertising Revenue Report 2008, `http://www.iab.net/insights/_research/530422/adrevenuereport`
2. Tew, A.: Million Dollar Homepage, `http://www.milliondollarhomepage.com/`
3. Wojciechowski, A.: An Improved Web System for Pixel Advertising. In: Bauknecht, K., Pröll, B., Werthner, H. (eds.) EC-Web 2006. LNCS, vol. 4082, pp. 232–241. Springer, Heidelberg (2006)
4. Wojciechowski, A., Kapral, D.: Allocation of Multiple Advertisement on Limited Space: Heuristic Approach. In: Mauthe, A., Zeadally, S., Cerqueira, E., Curado, M. (eds.) FMN 2009. LNCS, vol. 5630, pp. 230–235. Springer, Heidelberg (2009)
5. Wäscher, G., Haußner, H., Schumann, H.: An Improved Typology of Cutting and Packing Problems. European Journal of Operational Research 183(3), 1109–1130 (2007)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co., New York (1990)
7. Adler, M., Gibbons, P.B., Matias, Y.: Scheduling Space-Sharing for Internet Advertising. Journal of Scheduling 5(2), 103–119 (2002)
8. Dawande, M., Kumar, S., Sriskandarajah, C.: Performance Bounds of Algorithms for Scheduling Advertisements on a Web Page. Journal of Scheduling 6(4), 373–394 (2003)
9. Freund, A., Naor, J.S.: Approximating the Advertisement Placement Problem. Journal of Scheduling 7(5), 365–374 (2004)
10. Dawande, M., Kumar, S., Sriskandarajah, C.: Scheduling Web Advertisements: A Note on the Minspace Problem. Journal of Scheduling 8(1), 97–106 (2005)
11. Amiri, A., Menon, S.: Efficient Scheduling of Internet Banner Advertisements. ACM Transactions on Internet Technology 3(4), 334–346 (2003)
12. Menon, S., Amiri, A.: Scheduling Banner Advertisements on the Web. INFORMS Journal on Computing 16(1), 95–105 (2004)
13. Kumar, S., Jacob, V.S., Sriskandarajah, C.: Scheduling Advertisements on a Web Page to Maximize Revenue. European Journal of Operational Research 173(3), 1067–1089 (2006)
14. Interactive Advertising Bureau: Ad Unit Guidelines, `http://www.iab.net/iab/_products/_and/_industry/_services/1421/1443/1452`
15. 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)
16. Outing, S., Ruel, L.: The Best of Eyetrack III: What We Saw When We Looked Through Their Eyes. Poynter Institute, `http://poynterextra.org/eyetrack2004/main.htm`