

# Single Pattern Generating Heuristics for Pixel Advertisements

Alex Knoops   Victor Boskamp   Flavius Frasincar  
Erasmus University Rotterdam, the Netherlands  
`{alex.knoops, victorboskamp}@gmail.com`  
`frasincar@ese.eur.nl`

Adam Wojciechowski  
Poznan University of Technology, Poland  
`adam.wojciechowski@put.poznan.pl`

# Outline

## Introduction

- Research question

## Simulation

- Configuration

- Size of the banner

- Size of the advertisements

- Sorting of the advertisements

## Heuristic algorithms

- Left justified algorithm

- Orthogonal algorithm

- GRASP algorithm

## Analysis

## Software

## Conclusion

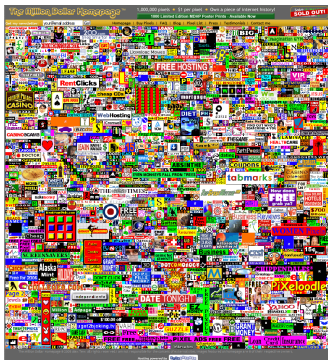
## Further reading

# Introduction

- ▶ Web advertising revenues 2008 in USA \$ 23.4 billion (Interactive Advertising Bureau)
- ▶ 22% of \$ 23.4 billion is banner advertising
- ▶ Pixel advertisement: make use of multibanners for advertisement
- ▶ Multibanner: banner containing small pictures (pixels blocks)

# Introduction

- ▶ Alex Tew's Million Dollar Homepage
- ▶ 21 years old UK student
- ▶ 1000 x 1000 pixels
- ▶ \$1 for 10,000 blocks of 10 x 10 pixels



# Introduction

- ▶ Commercial improvement required:
  - ▶ Static content
  - ▶ No search capabilities
  - ▶ Possibly out-of-date information
  - ▶ Possibly broken links
- ▶ Allow content to be dynamically generated

For more information on suggestions for improvements see [1]

# Research question

*How to arrange rectangular pictures of different sizes and different prices for advertisement on a banner, in order to maximize revenue?*

## Problem description

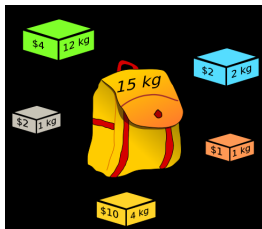
- ▶ Two-dimensional, single, orthogonal, knapsack problem
- ▶ Offline: the set of advertisements is predefined
- ▶ NP-hard

## Research objective

- ▶ MAXSPACE: Maximize total banner revenue (more pictures than the banner can accommodate)
- ▶ Time constraints for Web application
  - ▶ Usability
  - ▶ Server resources

# Knapsack problem

*Which boxes should be chosen to maximize the amount of money while still keeping the overall weight under or equal to 15 kg?*



## Our problem

- ▶ Knapsack is replaced by banner
- ▶ Boxes are replaced by advertisements
- ▶ Weight is replaced by the size of the pictures
- ▶ 1 dimension constraint extended to 2 dimensions constraints

# Configuration

## Setup

- ▶ One allocation pattern per simulation cycle
- ▶ Simulation parameters
  - ▶ 9 banner sizes
  - ▶ 6 maximum ad sizes
  - ▶ 120 sort orders
  - ▶ 3 algorithms
- ▶ Total 19,440 simulation cycles
- ▶ 1.5 to 2 as many advertisements as banner size can accommodate
- ▶ Platform: Intel Core 2 Duo CPU P8400 at 2.26 GHz

## Price of advertisements

Uniform distribution between 9 and 11 per advertisement pixel

## Price of banner

Fixed value of 4 per (unallocated) banner pixel



# Size of the banner

## 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

- ▶ All banner sizes are also reverted except for the “Square Button”
- ▶ 9 banner sizes

## Size of the advertisements

- ▶ Minimum width and height 10 pixels
- ▶ Maximum width  $w_{max}$
- ▶ Maximum height  $h_{max}$

... as fraction of the banner width and the banner height.

$$\{w_{max}, h_{max}\} \in$$

$$\{\{1/5, 1/2\}, \{1/2, 1/2\}, \{1/3, 1/3\}, \{1/5, 1/5\}, \{1/2, 1/5\}, \{1, 1\}\}$$

- ▶ 6 maximum advertisements sizes

# Sorting of advertisements

- ▶ Ascending and descending
- ▶ Primary and secondary attribute
  1. Price per advertisement pixel ( $p$ )
  2. Width ( $w$ )
  3. Height ( $h$ )
  4. Total area ( $w \times h$ )
  5. Flatness ( $w/h$ )
  6. Proportionality (  $|\log(w/h)|$  )
- ▶  $A_{12}^2 - (6 + 6) = 120$  sort orders

# Heuristic algorithms

## Left justified algorithm

1. Start with ordered set of advertisements
2. Select next advertisement from the set of advertisements
3. Begin on the left column
4. Search column top to bottom for free space
5. When free space found, try to place advertisement
6. When there is no space, move to the next column
7. Go to step 2

# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm




# Heuristic algorithms

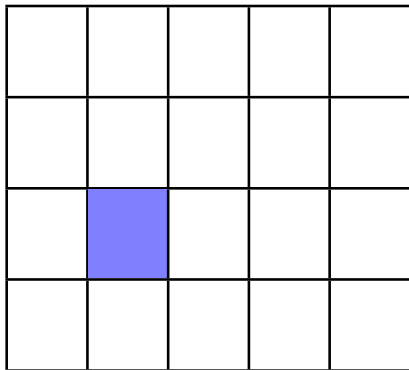
## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


A 4x5 grid with one cell highlighted in blue. The grid is composed of 4 rows and 5 columns. The cell at row 3, column 2 is filled with a solid blue color. All other cells are white with black borders.

# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

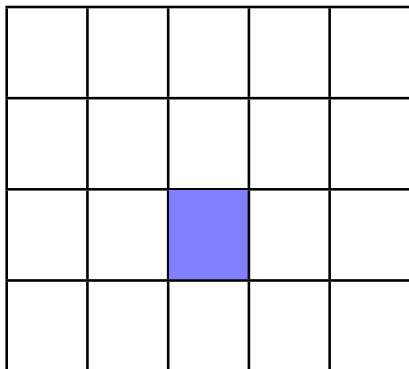
## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm



# Heuristic algorithms

## Left justified algorithm




# Heuristic algorithms

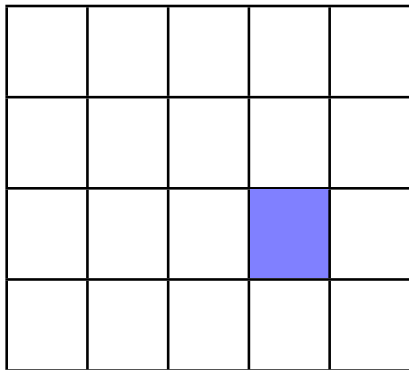
## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm



# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm


# Heuristic algorithms

## Left justified algorithm




# Heuristic algorithms

## Orthogonal algorithm

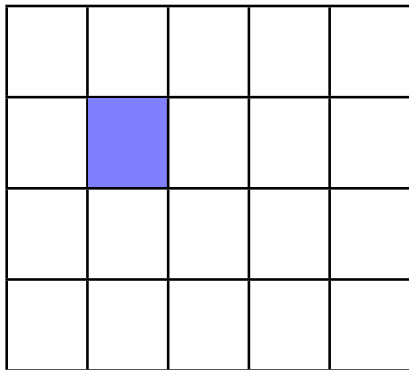
1. Start with ordered set of advertisements
2. Select next advertisement from the set of advertisements
3. Begin in the top-left corner
4. Move cursor diagonally down
5. Search from left and top border to cursor position for free space
6. When free space found, try to place advertisement closest to top-left corner
7. When there is no space, move cursor further diagonally down
8. Go to step 2

# Heuristic algorithms

## Orthogonal algorithm

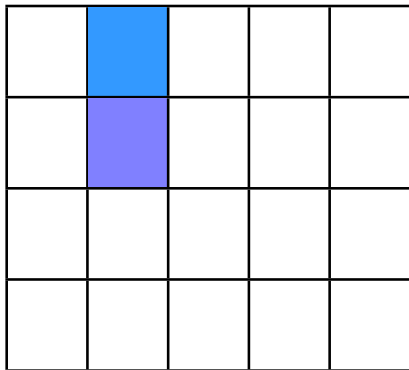

# Heuristic algorithms

## Orthogonal algorithm



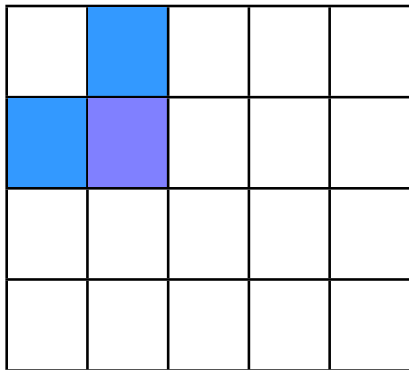
# Heuristic algorithms

## Orthogonal algorithm



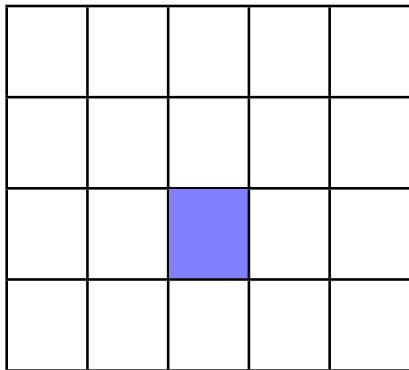
# Heuristic algorithms

## Orthogonal algorithm



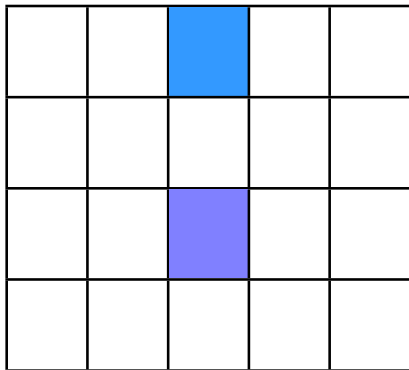
# Heuristic algorithms

## Orthogonal algorithm



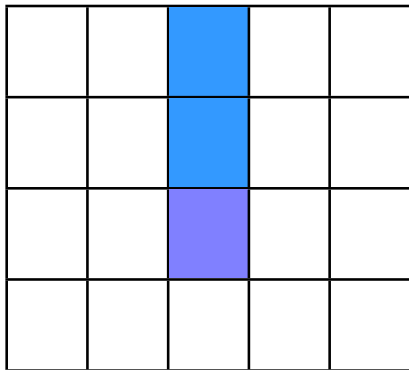
# Heuristic algorithms

## Orthogonal algorithm



# Heuristic algorithms

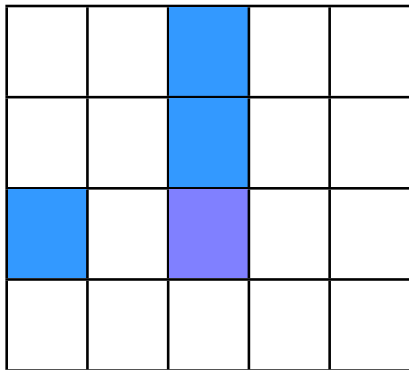
## Orthogonal algorithm





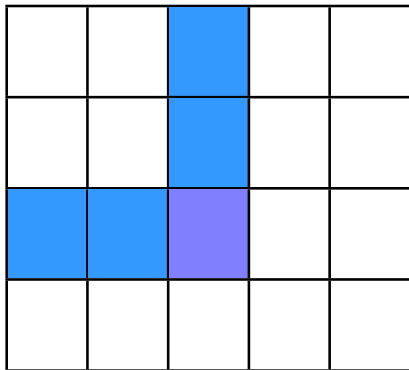
# Heuristic algorithms

## Orthogonal algorithm



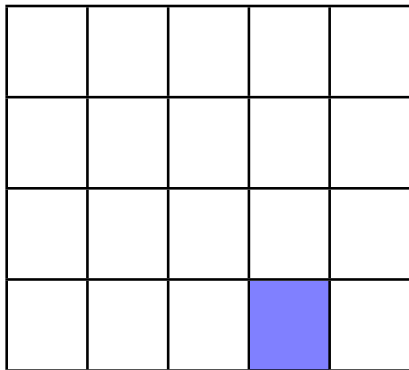
# Heuristic algorithms

## Orthogonal algorithm



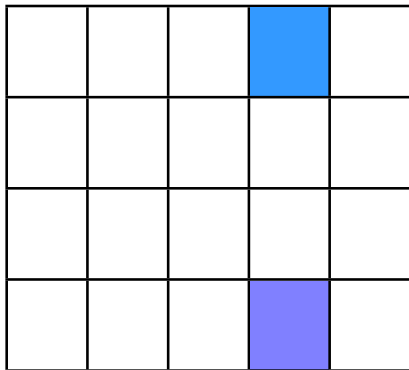
# Heuristic algorithms

## Orthogonal algorithm



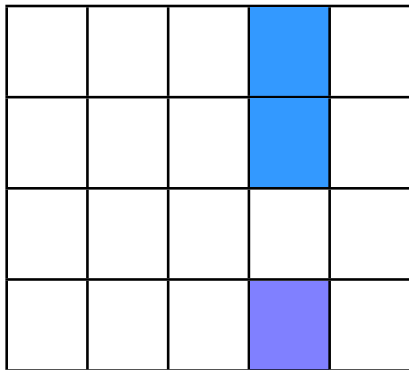
# Heuristic algorithms

## Orthogonal algorithm



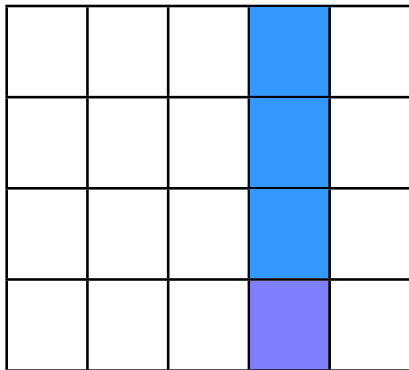
# Heuristic algorithms

## Orthogonal algorithm



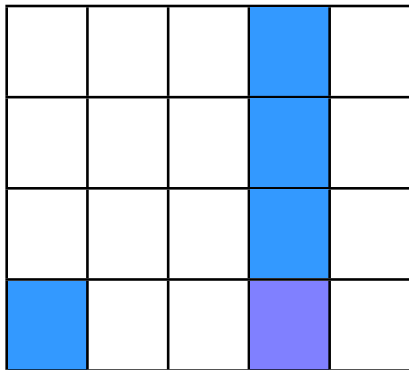
# Heuristic algorithms

## Orthogonal algorithm



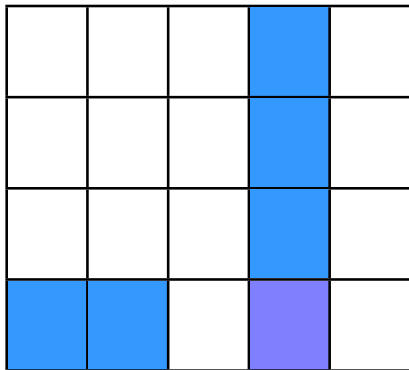
# Heuristic algorithms

## Orthogonal algorithm



# Heuristic algorithms

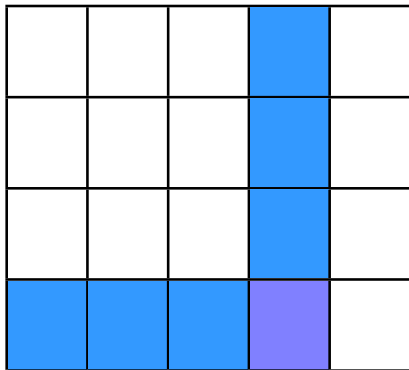
## Orthogonal algorithm





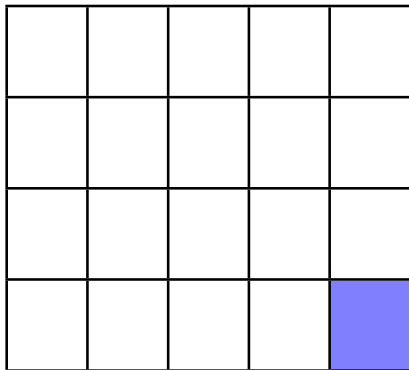
# Heuristic algorithms

## Orthogonal algorithm



# Heuristic algorithms

## Orthogonal algorithm



# Heuristic algorithms

## Orthogonal algorithm


# Heuristic algorithms

## Orthogonal algorithm


# Heuristic algorithms

## Orthogonal algorithm


# Heuristic algorithms

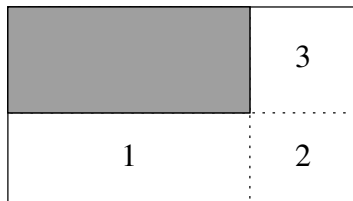
## Greedy Randomized Adaptive Search Procedure (GRASP)

1. Start with ordered set of advertisements
2. Select next advertisement from the set of advertisements
3. Place advertisement in the smallest *rectangle*  
(in a rectangle corner that is closest to a corner of the banner)
4. Divide free space in rectangles in the current *rectangle*
5. Merge rectangles that yield the largest area
6. Go to step 2

For more info see original paper [2]

# Heuristic algorithms

## Greedy Randomized Adaptive Search Procedure (GRASP)



# Analysis

## Profit per banner pixel per algorithm

Algorithm	Minimum	1 <sup>st</sup> Quartile	Median	Mean	3 <sup>rd</sup> 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

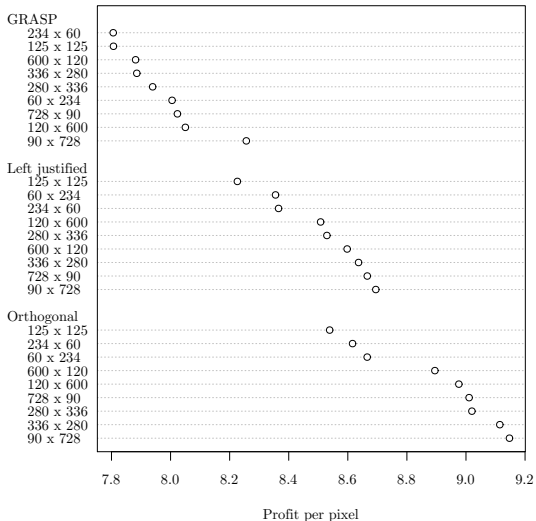
## CPU time per algorithm in seconds

Algorithm	Minimum	1 <sup>st</sup> Quartile	Median	Mean	3 <sup>rd</sup> 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



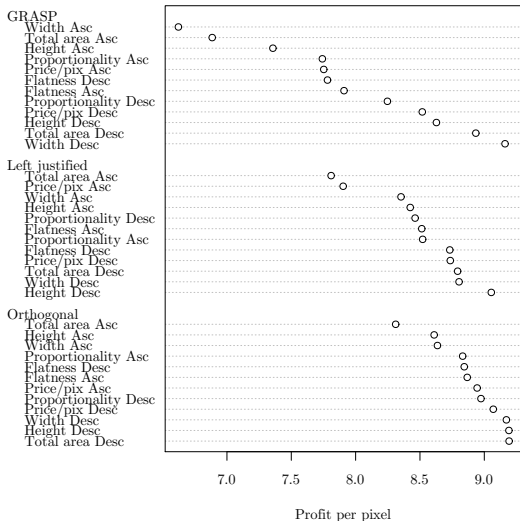
# Analysis

## Mean profit per pixel grouped by banner size



# Analysis

## Mean profit per pixel grouped by sorting



# Software

- ▶ Requirements
  - ▶ Usable results: image + imagemap
  - ▶ Responsiveness
- ▶ Web application demo
- ▶ Url: `http://headshredder.homelinux.net:8080/java/`

Input sample:

```
1;amazon.png;5;http://www.amazon.com/;  
2;bittorrent.png;7;http://www.bittorrent.com/;  
3;bitty.png;9;http://www.bitty.com/;  
4;blogburst.png;6;http://www.blogburst.com/;
```

## 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.

### Advertisement

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:

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:

**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:

Banner height:

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

**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:

Secondary sort:

## 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://cloudalicious.us/" alt="http://cloudalicious.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">
```

# Conclusion

## Conclusion

- ▶ Orthogonal algorithm best for our purpose
- ▶ Sorting by descending total area, width, or height

## Future work

- ▶ Improvement of pattern generating algorithms
  - ▶ Lookahead step
  - ▶ Left justified and orthogonal algorithm (efficiency): memorize previously visited places
  - ▶ GRASP algorithm (effectiveness): implement the improvement phase
- ▶ Online: advertisements placement requests come on the fly
- ▶ Make pixel advertisement banner dynamic (scheduling)

## Future work

- ▶ Assign different prices to locations (Eyetrack research)



## Further reading



[1] Wojciechowski, A.

An Improved Web System for Pixel Advertising.

EC-Web 2006, Springer LNCS **4082** (2006) 232–241



[2] 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** (2005)  
414–425



# Questions

Any questions?