Supplemental Material for Personalized and Responsive Web Designs with Integer Programming

January 2021

1 MILP Definitions

1.1 Generating Breakpoint Web Designs

The optimization of web layouts is formulated as a mixed integer linear programming (MILP) model, which provides good computational performance and solution quality. Our implementation ensures a non-overlapping, non-overflowing and perfectly packed grid layout. Continuous decision variables are used to represent the location of all four edges of each element. Continuous decision variables avoid pixel-level discretization, which is important for the performance of the solver. Furthermore, the size of our MILP model depends solely on the number of elements involved, rather than on the canvas size. The main objectives for the optimization are as follows:

1. Layout Similarity: The element order should follow that of the original layout.
2. Layout Quality: The result should be a perfectly packed symmetric grid layout.
3. Layout Usability: The saliency of the most important elements should maximized, while their selection time should be minimized.

The user of the system can control the optimization result by adjusting objectives 1 and 3. The optimization process is divided into four phases: (i) Resolving Element Order, (ii) Optimizing for Layout Quality, (iii) Optimizing for Layout Usability, and (iv) Optimizing for Both Layout Quality and Usability.

1.1.1 Resolving Element Order

In order to achieve a consistent layout across different responsive versions and to improve optimizer performance, the element order is defined irrespective of the screen resolution and layout. The elements are ordered in the ascending order, according to the value of the equation

$$O'_e = S \cdot O_e + (1 - S) \cdot I_e$$

where O_e corresponds to the element's original ordinal, and O'_e corresponds to the new ordinal, both normalized to the range $[0, 1]$. I_e corresponds to the element's normalized inverted importance. This means that the greater the importance of an element, the more likely it is for that element to be first in the layout. However, regardless of the element order, the hierarchical structure of the layout is kept intact, and the order is applied to sibling nodes in the hierarchy.

1.1.2 Optimizing for Layout Quality

A perfectly packed symmetric grid layout is achieved through a combination of MILP constraints that guarantee a perfectly packed grid with an objective for symmetry on each grid row. The constraints and objects are applied iteratively to each level of the layout hierarchy. The following constraints and objectives are thus applied to each group of sibling nodes in the layout.

Although our approach can support multiple different layout models, we present an example implementation of a simplified version of the CSS Flexible Box Layout model, where elements are stacked side by side on one or more rows. In this example, we constrain each group to a fully justified, stretched, and perfectly packed grid, where all elements on the same row have an equal height, and the row extends the full width of its container. As the element order is predefined, the optimizer essentially selects the "line breaks" of the layout. In addition of the overall layout constraints, the size of each element is selected from a predefined set of shapes. This is to prevent content overflow.

As the element order is known, overlap is prevented by placing positional constraints on consecutive elements. Consecutive elements on the same row are required to share top and bottom edge coordinates, and the right edge of the first element must be flush against the left edge of the second element (Constr. 8–11). If consecutive elements fall on different rows, they must be placed at the end and the beginning of their rows, respectively, and the first element bottom edge must be aligned with the top edge of the second one (Constr. 4–7). This results in a non-overlapping, non-overflowing, well-aligned and perfectly packed grid.

- $\xi \in \{0, 1\}$: symmetry
- $\rho \in \{0, 1\}$: on same row
- $\upsilon \in \{0, 1\}$: have same width
- w_l: Layout width (constant)
- $\max(h_l)$: Maximum layout height (constant)

We define following continuous decision variables for each element: λ (left edge), τ (top edge), γ (right edge), and β (bottom edge), as well as binary decision variables for each pair of successive elements: ρ (elements are on the same row) and υ (elements have the same width).

For all $i \in \{1, \dots, |\mathbb{E}_g|\}$

$$\begin{aligned} \lambda_i &\geq \lambda_g \\ \tau_i &\geq \tau_g \\ \gamma_i &\leq \gamma_g \\ \beta_i &\leq \beta_g \end{aligned} \tag{1}$$

Minimize group symmetry:

$$\min \sum_{i=2}^{|\mathbb{E}_g|} \xi_i \tag{2}$$

For all $i \in \{2, \dots, |\mathbb{E}_g|\}$

$$\xi_i \geq \rho_{i-1}^i - \upsilon_{i-1}^i \tag{3}$$

$$\begin{aligned} \neg\rho_{i-1}^i \implies & \lambda_i = \lambda_g \\ & \wedge \gamma_g = \gamma_{i-1} \\ & \wedge \tau_i = \beta_{i-1} \end{aligned}$$

$$\begin{aligned} \lambda_i &\leq \lambda_g + \rho_{i-1}^i w_l && (4) \\ \gamma_{i-1} &\geq \gamma_g - \rho_{i-1}^i w_l && (5) \\ \tau_i &\leq \beta_{i-1} + \rho_{i-1}^i \max(h_l) && (6) \\ \tau_i &\geq \beta_{i-1} - \rho_{i-1}^i \max(h_l) && (7) \end{aligned}$$

$$\begin{aligned} \rho_{i-1}^i \implies & \tau_i = \tau_{i-1} \\ & \wedge \beta_i = \beta_{i-1} \\ & \wedge \lambda_i = \gamma_{i-1} \end{aligned}$$

$$\begin{aligned} \tau_i &\leq \tau_{i-1} + (1 - \rho_{i-1}^i) \max(h_l) && (8) \\ \tau_i &\geq \tau_{i-1} - (1 - \rho_{i-1}^i) \max(h_l) && (9) \\ \beta_i &\leq \beta_{i-1} + (1 - \rho_{i-1}^i) \max(h_l) && (10) \\ \beta_i &\geq \beta_{i-1} - (1 - \rho_{i-1}^i) \max(h_l) && (11) \end{aligned}$$

If $\upsilon^i_j = 1$, elements must be equally wide:

$$\upsilon^i_{i-1} \implies w_i = w_{i-1} \tag{12}$$

$$w_i \leq w_{i-1} + (1 - \upsilon^i_{i-1}) w_l \tag{13}$$

$$w_i \geq w_{i-1} - (1 - \upsilon^i_{i-1}) w_l \tag{14}$$

1.1.3 Optimizing for Layout Usability

After optimizing for layout quality $\mathbb{Q}$, the model objective is replaced with layout usability $\mathbb{U}$, which consists of maximizing the saliency σ and minimizing the selection time δ for important elements in the layout. The complete objective for this stage is:

$$\min \sum_{i=1}^{|\mathbb{E}|} I_i \cdot (\delta_i - \sigma_i) \tag{15}$$

We use Fitts' law to compute the time required to reach a specific element on the screen. Fitts' law is widely used in model-based optimization as an objective function. Selection time ST is a function of target distance D and size W:

$$\begin{aligned} \mathrm{ST} &= a + b \log_2(\frac{2D}{W}) \\ &= a + b(1 + \log_2 D - \log_2 W) \end{aligned} \tag{16}$$

In our case, we assume that the user starts scanning the screen from the top-left corner. To improve the optimizer performance, we use a piece-wise-linear approximation of the logarithm function. For rectangular objects, ST in the in direction of movement equals the ST along the worse of the cardinal directions. Furthermore, for the optimization, the constants a, b, and 1 are not relevant. Our MILP implementation therefore becomes:

$$\delta_i \geq \log_2 \frac{\lambda_i + \gamma_i}{2} - \log_2 w_i \tag{17}$$

$$\delta_i \geq \log_2 \frac{\tau_i + \beta_i}{2} - \log_2 h_i \tag{18}$$

$$\tag{19}$$

Saliency refers to how attention-grabbing an element is given the rest of the page. To measure saliency, we use the area of an element as a proxy, although other factors, such as color could be used. As the Weber–Fechner law states that perceived intensity is proportional to the logarithm of the stimulus, we use the logarithm of the element's area as the saliency function:

$$\begin{aligned}\sigma_i &= \log_2(h_i \cdot w_i) \\ &= \log_2 w_i + \log_2 h_i \end{aligned} \tag{20}$$

Although the layout quality $\mathbb{Q}$ is removed from the objective, a new constraint is added to the model in order to maintain the quality achieved in the previous stage:

$$\mathbb{Q} \leq \mathbb{Q}_1 \tag{21}$$

where $\mathbb{Q}_1$ corresponds to the value achieved in the previous stage. No other restrictions are added to the model at this stage.

1.1.4 Optimizing for Both Layout Quality and Usability

If the optimizer reaches an optimal solution in the previous stage, that is the final layout and this stage is skipped. If an optimal solution has not yet been found, the objectives from the two previous stages are combined to achieve a balance between the aesthetics of the layout and its usability.

First, the usability objective from the previous stage is normalized, so that the new $\mathbb{U}'$ is defined as:

$$\mathbb{U}' = \frac{\mathbb{U}_2}{\mathbb{U}_2 - \mathbb{U}_{\text{lower}}} \tag{22}$$

Where $\mathbb{U}_2$ corresponds to the value achieved in the previous stage, and $\mathbb{U}_{\text{lower}}$ to the lower bound estimated by the solver. The full objective of the final optimization stage becomes:

$$\min(\mathbb{Q} + \mathbb{U}') \tag{23}$$

As $\mathbb{Q}$ is an integral expression, more weight is given for improving the layout aesthetics. However, to prevent the solver from compromising the usability of the layout, a new constraint is added to model:

$$\mathbb{U}' \leq 1.1 \tag{24}$$

This constraint essentially allows the usability objective $\mathbb{U}$ to be decreased 10% of the difference between what was reached at the previous stage and what was estimated as optimal.

1.1.5 Adjusting Results

If the design task instance prioritizes selection time, the optimizer attempts to minimize the predicted time required for important elements. The most obvious effect is that very important elements may become larger and move closer to the top-left corner of the web page. However, the reading order of the elements will not be changed. If the most important elements are not the first elements (i.e., when the similarity weight is high) the first elements can be expected to decrease in size, if their shape sets allow it.