Scout: Rapid Exploration of Interface Layout Alternatives through High-Level Design Constraints

Amanda Swearngin , Chenglong Wang , Alannah Oleson , James Fogarty , Amy J. Ko

摘要

UI 设计师在界面设计的过程中通常会创建一系列的替代方案，通过斟酌、比较以及迭代，得到最终采纳的方案。现有的自动生成界面布局的方法局限于生成单一的结果，无法生成多种替代方案。Scout 根据设计师给定的元素，和设计师指定的一些设计意图（即高级约束，比如指定顺序，是否突出某个元素，等等），将这些高级约束形式化为一系列的低级约束（对元素的大小和位置的约束），然后通过一个分支界限算法自动生成多种符合约束的界面布局方案，供用户选择和进一步修改。

关键词：界面设计；布局；自动生成；约束

1 引言

替代方案在界面设计中有非常重要的作用。通过提供多个不同的解决方案，设计师拥有更多考虑和选择的空间，并且通过对比不同的方案做出更好的决定。但是人工设计多个解决方案往往是比较困难的，不仅需要更大的工作量，而且需要设计师突破思维的局限。设计师可以通过收集其他设计例子来寻找灵感，但是将设计例子改编为自己的设计方案仍然需要大量的工作，并且涉及到对元素进行低层次的调整，这对设计师的专业能力提出了巨大的挑战。Scout 可以帮助设计师快速地探索界面布局替代方案。设计师只需提供界面元素，并通过交互界面提出一些高级约束（比如顺序、权重、结构），Scout 就会实时生成多种满足要求的布局备选方案，供设计师参考和使用。

2 相关工作

2.1 替代方案生成

目前已有许多生成替代方案的系统。DesignScape[1]使用了基于一些设计原则的能量模型，为布局设计生成替代方案。Brian[2]让设计师探索设计例子，并将风格和元素迁移到自己的设计上。Scout 在这些工作的基础上做了改进，让设计师指定高级约束来引导替代方案的生成。

2.2 基于约束的布局生成

目前已有许多基于约束的布局生成工具[3][4]。这些工作往往关注空间层面的低级约束，比如元素的位置和大小，而 Scout 只需要设计师关注设计层面的高级约束，Scout 会将这些高级约束转化为一系列低级约束。此外，过去的工作主要关注于生成单一的布局，而 Scout 可以生成多个备选方案供设计师参考。

3 本文方法

3.1 本文方法概述

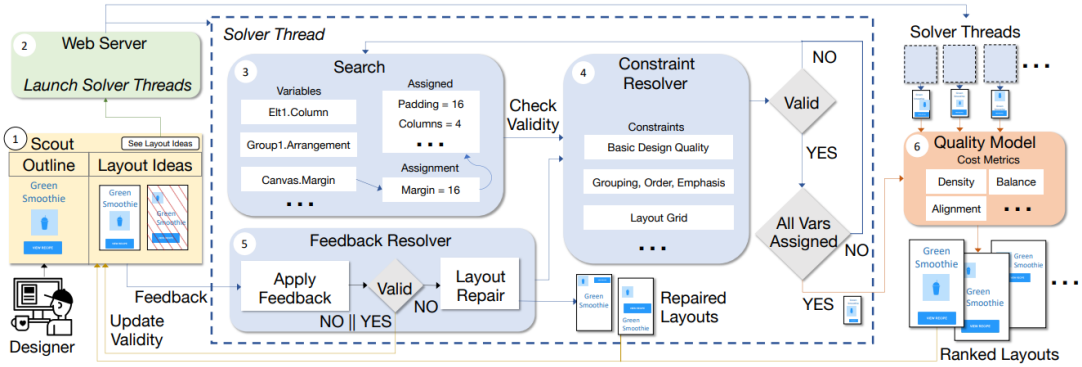

图 1: 方法示意图

Scout 接收用户提供的界面元素和指定的高级约束作为输入，然后启动多个线程来生成布局替代方案。Scout 为每个布局元素赋予一些变量，比如大小、位置、对齐方式等，然后根据已指定的高级约束，为变量生成一系列低级约束，然后启动一个分支界限算法为变量赋值。分支界限算法每次随机从变量列表中选取一个变量，并为其随机赋值，每次赋值结束后检查当前变量是否能够满足约束，如果不能满足，而剪枝，否则继续选取下一个变量赋值，直到所有变量都赋值完毕，就可以得到一个可行解。根据已经生成的解，Scout 会为其生成一个约束，避免同样的解再次出现。最后将生成的解排序输出。

3.2 约束编码和变量设计

Scout 通过给一组变量分配具体的值来生成布局，允许它探索元素排列、对齐、位置和尺寸的多种组合。对于画布、组和元素这三种不同的对象，Scout 分别定义了不同的变量。画布变量包括布局网格、基线网格和边距，组变量包括了排列、对齐和距离，元素变量则包含位置和大小。每个变量都有各自的值域，Scout 从值域中随机选取一个值为变量赋值。Scout 的约束系统包含了五个方面的约束：基本的设计约束，分组和顺序约束、重要性约束，替代约束和重复约束。基本设计约束确保所有元素不会超出画布以外，以及避免重叠。除了基本设计约束之外的约束，都是由用户指定的高级约束。

分组和顺序约束确保所有元素不会超出所属的分组，分组内元素之间的对齐，和分组内元素之间的距离。对于每个分组，用户可以指定是否关注顺序，如果关注顺序，Scout 会按照用户输入元素的顺序来排列元素，否则随机排列元素。

重要性约束是对元素大小的约束。用户可以指定元素的重要性（大，中，小），重要性大的元素尺寸可以变大，而重要性小的元素尺寸可以变小，而普通的元素则尺寸不变。

用户可以指定替代分组，该分组下包含多个可供替换的元素，Scout 会用一个随机变量，决定从中采取哪个元素。重复约束则确保若干个小分组之间的属性一致，包括对齐方式、排列、距离，以及每个小分组下的元素大小相等。加入让这些变量相等的约束即可实现重复效果。

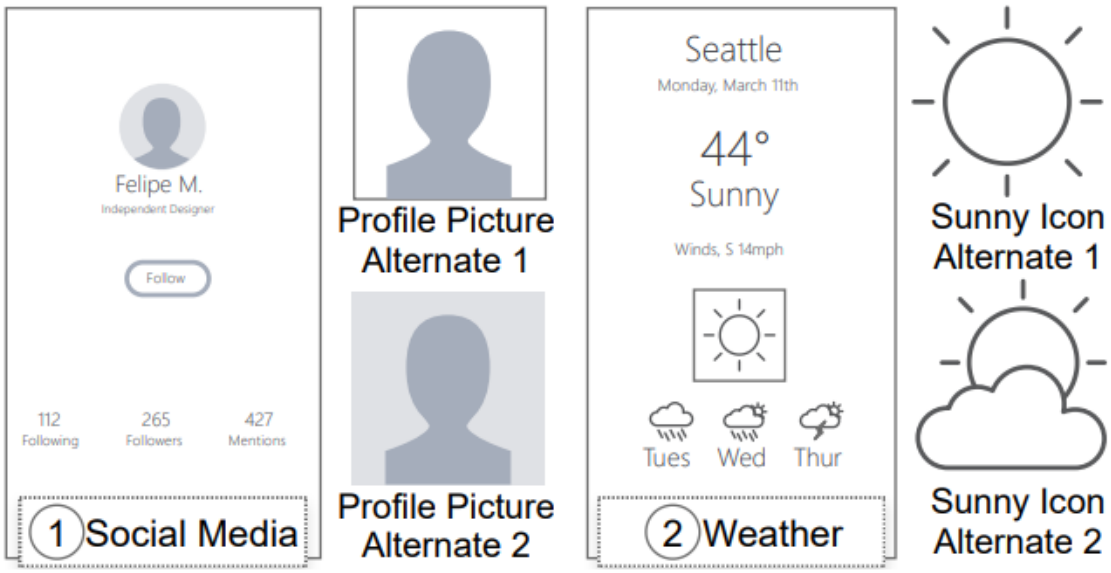

图 2: 替代约束和重复约束实例

4 复现细节

4.1 与已有开源代码对比

复现过程中引用了部分作者提供的源代码，主要是构建基本约束的代码和分支界限法的代码，这些代码在作者源代码的 custom_solver.py 和 constraintd_builder.py 文件里。

作者提供的源代码只是一个简单的 demo，许多高级约束并没有实现。这次复现的主要贡献在于补充了高级约束的实现，包括顺序约束、重要性约束、替代约束和重复约束。这些高级约束都可以转化为一系列低级约束来表示，即对各个元素的坐标和尺寸变量 $(x, y, width, height)$ 的约束。

此外，还实现了一个简单的网页交互界面，用于验证复现的结果。

4.2 实验环境搭建

复现基于 web 环境搭建实验平台。界面基于 Vue.js 2.9.6，服务端基于 Flask 2.2.2.

5 实验结果分析

如图 3，输入若干元素，并为其指定分组和高级约束，由 Scout 生成布局方案。可以观察到顺序约束（Order Group 下面的三个元素）、替代约束（奶昔图案）、重复约束（时间和卡路里图标和文字）都能够满足。

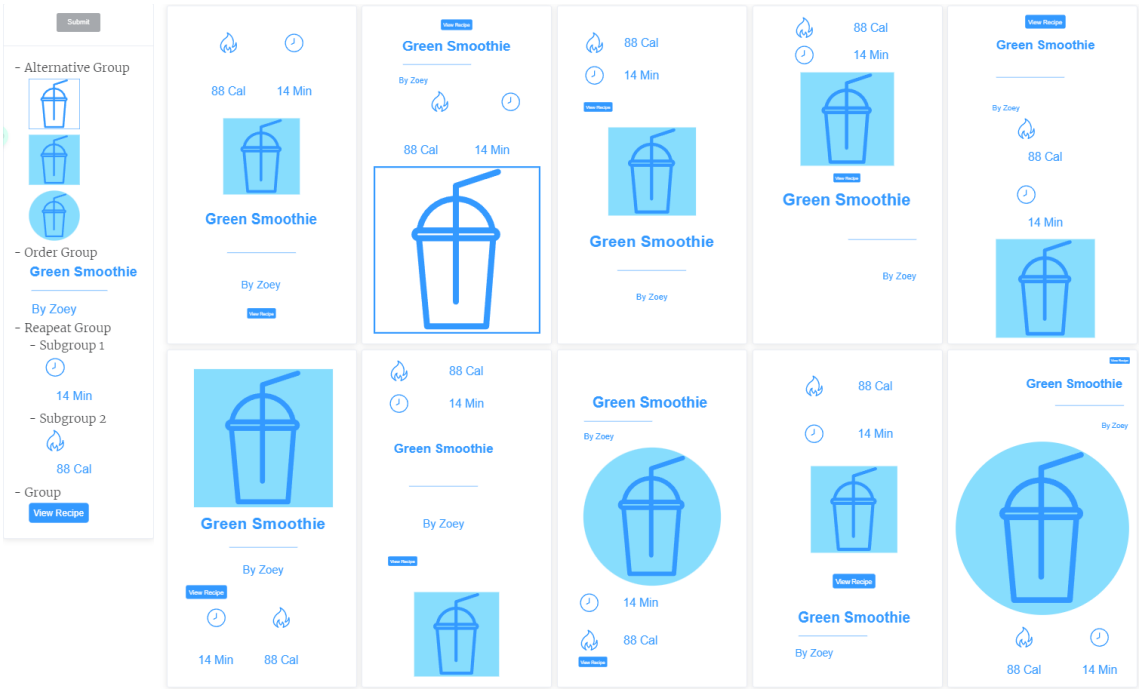

图 3: 实验结果

6 总结与展望

本次复现实现了 Scout 的算法和简单的界面，主要是实现了论文提出的几种高级约束。不足之处在于没有尝试探索论文中没有提到的其他约束，未来可以考虑添加进去。另外还可以尝试把该方法应用在界面设计以外的应用场景上，比如文本和网页的布局设计。

参考文献

[1] O'DONOVAN P, AGARWALA A, HERTZMANN A. DesignScape: Design with Interactive Layout Suggestions[C/OL]//CHI '15: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. Seoul, Republic of Korea: Association for Computing Machinery, 2015: 1221-1224. https://doi.org/10.1145/2702123.2702149. DOI: 10.1145/2702123.2702149.

[2] LEE B, SRIVASTAVA S, KUMAR R, et al. Designing with Interactive Example Galleries[C/OL]//CHI '10: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Atlanta, Georgia, USA: Association for Computing Machinery, 2010: 2257-2266. https://doi.org/10.1145/1753326.1753667. DOI: 10.1145/1753326.1753667.

[3] ZEIDLER C, LUTTEROTH C, STURZLINGER W, et al. The Auckland Layout Editor: An Improved GUI Layout Specification Process[C/OL]//UIST '13: Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology. St. Andrews, Scotland, United Kingdom: Association for Computing Machinery, 2013: 343-352. https://doi.org/10.1145/2501988.2502007. DOI: 10.1145/2501988.2502007.

[4] ZANDEN B V, MYERS B A. The Lapidary Graphical Interface Design Tool[C/OL]//CHI '91: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New Orleans, Louisiana, USA: Association for Computing Machinery, 1991: 465-466. https://doi.org/10.1145/108844.109005.

DOI: 10.1145/108844.109005.