GRIDS 论文复现

李祎杰

摘要

在设计中，网格越来越被广泛使用到规范内容排版和内容布局中，规整的网格线作为辅助设计在视觉上的重要性越来越得以凸显。自动排版和布局一直以来面临一些困难，例如如何定义好的设计，如何定义好的约束，设计上的这些标准并不完全是客观统一的。本报告复现的是论文 GRIDS: Interactive Layout Design with Integer Programming[1]，该工作用整数规划模型来自动求解生成内容排版。

关键词：排版；人机交互；整数规划

1 引言

排版或布局在可视化设计中一直是一个重要的问题。网格化的排版在设计中已经越来越多被设计师们所使用，用来辅助他们画草稿或拟初稿。然而，目前他们的设计依然耗时并需要大量的手工操作。该文章提出了使用 MILP 整数规划模型，自动生成基于网格的排版布局。

2 相关工作

早期对于 layout 的研究主要聚焦于交互式的生成布局，对设计过程主要起辅助作用。其中一些工具已经被各大图形设计框架运用，例如对齐命令。利用这些方法，使用者不用精确地放置元素位置，但仍需要手动操作元素或指定约束来确定设计的排版或布局。当前生成排版布局的研究方向主要为自动生成，即希望只通过用户给的少量信息，可以是一些文字，图像元素，或者是高等级的约束 (high-level constraints)，就能生成用户想要的排版或布局，而不需要手动放置或调整每个元素的位置和大小。

自动生成排版的工作主要可以分为两个方向，一是用机器学习或深度学习的方法，利用数据集，自定义损失函数来训练出模型进行自动生成[2-4]；二是用纯算法的方法，根据元素间的数学或几何上的关系来定义约束，从而使自动生成的排版或布局满足条件[5]。

本次复现的工作为 GRIDS: Interactive Layout Design with Integer Programming，主要是使用整数规划的算法，能够应用到布局的生成，增强和补全。整数规划的优点：随机搜索 (黑盒) 下，输出的主要特征能够保证，如果有解，则解一定会被找到。

3 本文方法

3.1 问题描述

将需要排版的内容看作一个个矩形元素，给定一个矩形的集合，以及一个固定尺寸的画布，要求找出合理的解：集合中的所有矩形都在画布内，不超出画布边缘，相互之间没有重叠。

满足以上基本条件后，本文设定了几个目标。

1. 全局对齐。指的是整张画布上的矩形元素要尽可能多的对齐。
2. 矩形轮廓。画布上的所有矩形元素外围组成的轮廓，要尽可能接近矩形。

3. 优先位置。对于一些特定元素，需要有一些全局的或相对的优先位置关系。

3.2 本文方法概述

本文的求解核心就是定义目标函数，定义约束，使用整数规划模型进行求解。

首先定义两个二进制的决策变量 $\Gamma^e_{\bar{e}}, \Pi_{e\bar{e}}$，分别表示元素 e 在元素 $\bar{e}$ 的上方，以及元素 e 在元素 $\bar{e}$ 的左侧，画布上任何非重叠的元素间相对位置关系都能用这两个符号表示。

下面给出禁止元素间重叠的约束。其中，$\mathbb{W}$ 和 $\mathbb{H}$ 分别代表画布的宽度和长度。L, R, B, T 分别代表矩形元素四条边的坐标。

$$1 \leqslant \Gamma^e_{\bar{e}} + \Gamma^{\bar{e}}_e + \Pi_{e\bar{e}} + \Pi_{\bar{e}e} \leqslant 2 \ldots \forall e, \bar{e} \in \mathbb{E}$$

$$T_{\bar{e}} \geqslant B_e + \mathbb{H}\left(\Gamma^e_{\bar{e}} - 1\right) \ldots \forall e, \bar{e} \in \mathbb{E}$$

$$L_{\bar{e}} \geqslant R_e + \mathbb{W}\left(\Pi_{e\bar{e}} - 1\right) \ldots \forall e, \bar{e} \in \mathbb{E}$$

$$\mathbb{W}\Pi_{e\bar{e}} \geqslant L_{\bar{e}} - R_e \ldots \forall e, \bar{e} \in \mathbb{E}$$

$$\mathbb{H}\Gamma^e_{\bar{e}} \geqslant T_{\bar{e}} - B_e \ldots \forall e, \bar{e} \in \mathbb{E}$$

该文提出了对齐组的概念，例如若干个元素的左边对齐，那么他们就同属于一个左对齐组 LG，类似的，还有 RG, TG, BG。将对齐组的对齐边延长，形成网格，这些对齐边固定住了元素的位置，任意一个矩形元素的位置和尺寸都由四条对齐边确定。对于同一个输入条件，越少的对齐组，意味着全局对齐越优越，画布上的矩形元素对齐的越好。

关于其他的约束添加，本报告中不再一一列出。

3.3 解的多样性

对于同一个输入条件，生成尽可能多的不同的解也是本算法的目标之一。文中给出的伪代码如下。

Procedure 1 Procedure to generate grid layouts

Input: Data instance involving n elements
$\mathbb{F} := \{\}$
$\Gamma_{max}, \Gamma_{min}$:= External values of Γ by core MILP
Π_{max}, Π_{min} := External values of Π by core MILP
ε_{min} := Optima value by minimising ε
$\mathbb{R}_{min}$:= Optimal number of cases for overall rectangular outline using core MILP
Use $\varepsilon_{min}, \Gamma_{max}, \Gamma_{min}, \Pi_{max}, \Pi_{min}, \mathbb{R}_{min}$ to augment core MILP formulation
while $||\mathbb{F}||$<*Required number of solutions* **do**
  for *all* $\Gamma_{val} \in [\Gamma_{min}, \Gamma_{max}$ **do**
    for *all* $\Pi_{val} \in [\Pi_{min}, \Pi_{max}]$ **do**
      Enforce Γ_{val}, Π_{val}
      $f \leftarrow$ Optimal solution of core MILP withcurrent constraints
      $\mathbb{F} = (\mathbb{F} \cup \{f\})$
      Remove Γ_{val}, Π_{val} constraints
    end
  end
  Loosen alignment constraint (ε_{min}) by unit value
end
Output: Set $\mathbb{F}$ of feasible grid layouts

4 复现细节

4.1 与已有开源代码对比

Grids 提供了源代码aalto-ui/GRIDS，本次复现基于其源代码进行改进。已有的开源代码固定了元素的尺寸大小，本次复现为实现添加新的约束，放宽了图形元素的尺寸范围，使得元素尺寸可变。为了使得生成结果更美观规范，本次复现在源代码的基础上添加了新的设计约束，如文本栏等宽的约束。源代码的输入数据不含文本类型的元素，本次复现加入了含文本元素的数据。另外，已有的开源代码不包含结果的渲染部分代码，本工作实现了布局的渲染代码，便于结果的观察比较。

4.2 实验环境搭建

实验所需环境如下：

1. python 3.0
2. gurobipy
3. matplotlib
4. numpy

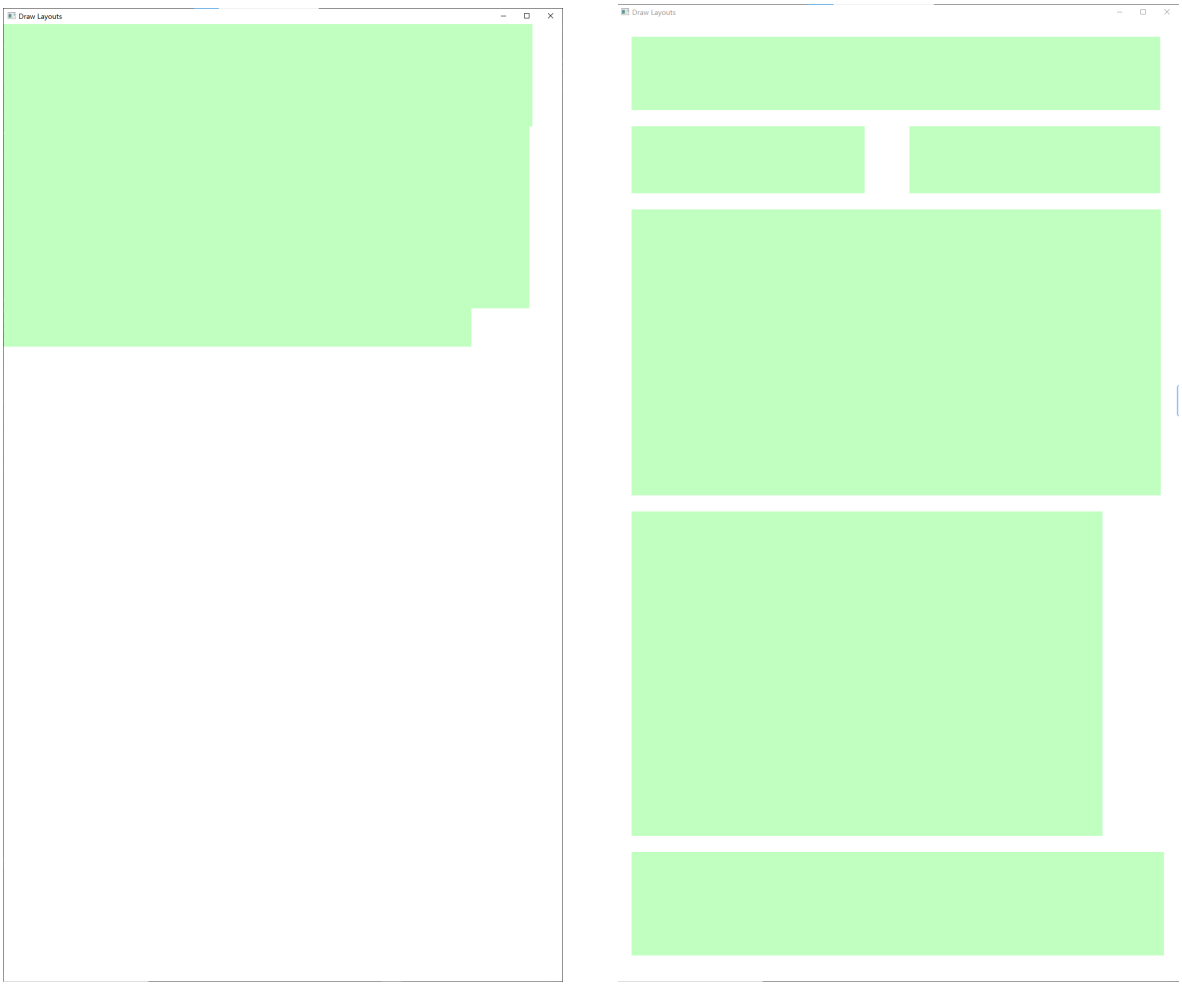
gurobipy 是一个数学优化求解器，可以直接通过 pip install gurobipy 安装。

4.3 界面分析与使用说明

运行求解器首先需要给定输入，目录下有一个 JSON 文件，包含了输入的例子，在终端下输入命令行：

```
python StartMe.py <PATH_TO_JSON_INPUT_FILE>
```

得到的结果将保存在 results 文件夹下，选择任意结果，将结果的目录位置粘贴到脚本 draw.py 中的输入路径下，运行 draw.py 脚本，即能得到求解结果的渲染图片。使用脚本对输入与输出的结果如图 1 所示。



(a) 输入元素　　(b) 输出布局

图 1: 求解前后渲染图

4.4 创新点

本次复现主要完成了两个工作，一是在原有算法基础上添加了文本框等宽的约束进行求解，二是写了一个渲染脚本，将 JSON 格式的输入输出渲染成可视化的结果，便于观察比较。下面主要介绍文本框等宽约束的加入。

在实际应用场景下，例如杂志的排版，为了保证排版的美观易读，设计师通常会保持文本列的宽度一致。因此从设计的角度出发，本次复现在不减少原文约束的前提下，加入文本栏等宽的约束。

本项目源代码中元素宽高固定不可修改，因此给文本元素添加等宽约束后，会造成无解的情况。为了满足文本栏等宽，将放松元素尺寸的限制，这样求出的解能够同时满足原约束以及新加入的约束。具体代码将随报告一并提交。

5 实验结果分析

由于源代码中没有包含文本元素的输入数据，本项目在原有的数据基础上修改，添加了两个文本元素。以项目目录中的 Example1.json 为例，使用 Grids 算法求解，并用脚本渲染结果，加入文本等宽约束前后的结果渲染图如图 2 所示。

(a) 加入文本等宽约束前　　(b) 加入文本等宽约束后

图 2: 约束文本框前后对比

可以发现 (a) 图的两个文本框宽度有差别，而 (b) 图的文本框的宽度相等。此例中由于输入样本限制，对比结果并不明显。实际使用中，源代码求出的解可能出现文本框相差极大的情况，加入文本栏等宽约束，能使得求解结果更为合理，排版更加规范。

6 总结与展望

GRIDS 方法主要是用整数规划的方法，通过添加约束达成设计目的，实现自动排版。本项目基于 GRIDS，对已有工作进行改进。本次改进主要完成了两项工作，首先是算法方面，扩大了源代码中元素大小的范围，使元素尺寸可变。从设计角度出发，加入了文本栏等宽的约束，帮助实际使用中的文字内容规范易读。第二是写了渲染脚本，将 JSON 格式的输入输出渲染为可视化的结果，便于观察和比较。

GRIDS 方法的计算速度可以接受，并且不需要事先训练，在有解的情况下能很好地满足约束条件。在生成结果的多样性上，GRIDS 主要是通过改变对齐边的组数来达成结果多样性。经实践发现，这种做法生成的结果大部分仍然相似，很多生成结果的差别仅仅是个别元素边缘偏移微小的距离。因此在未来的改进中，可以尝试转换思路，从其他角度实现解的多样性。

参考文献

[1] DAYAMA N R, TODI K, SAARELAINEN T, et al. Grids: Interactive layout design with integer pro-

gramming[C]//Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. 2020: 1-13.

[2] PATIL A G, BEN-ELIEZER O, PEREL O, et al. Read: Recursive autoencoders for document layout generation[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020: 544-545.

[3] SUN J, WU W, LIU L, et al. WallPlan: synthesizing floorplans by learning to generate wall graphs[J]. ACM Transactions on Graphics (TOG), 2022, 41(4): 1-14.

[4] NAUATA N, HOSSEINI S, CHANG K H, et al. House-gan++: Generative adversarial layout refinement network towards intelligent computational agent for professional architects[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 13632-13641.

[5] SWEARNGIN A, WANG C, OLESON A, et al. Scout: Rapid exploration of interface layout alternatives through high-level design constraints[C]//Proceedings of the 2020 CHI conference on human factors in computing systems. 2020: 1-13.