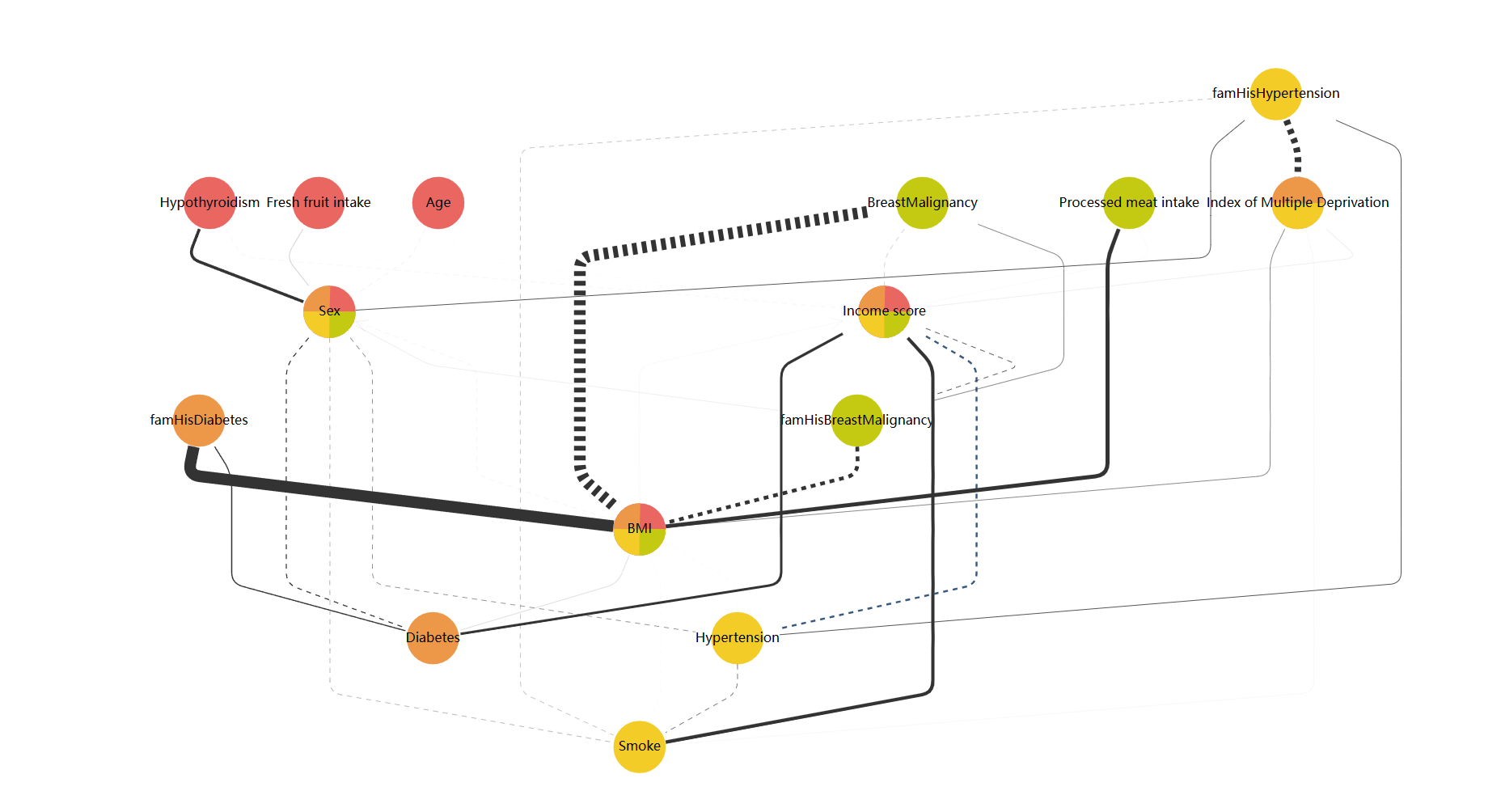
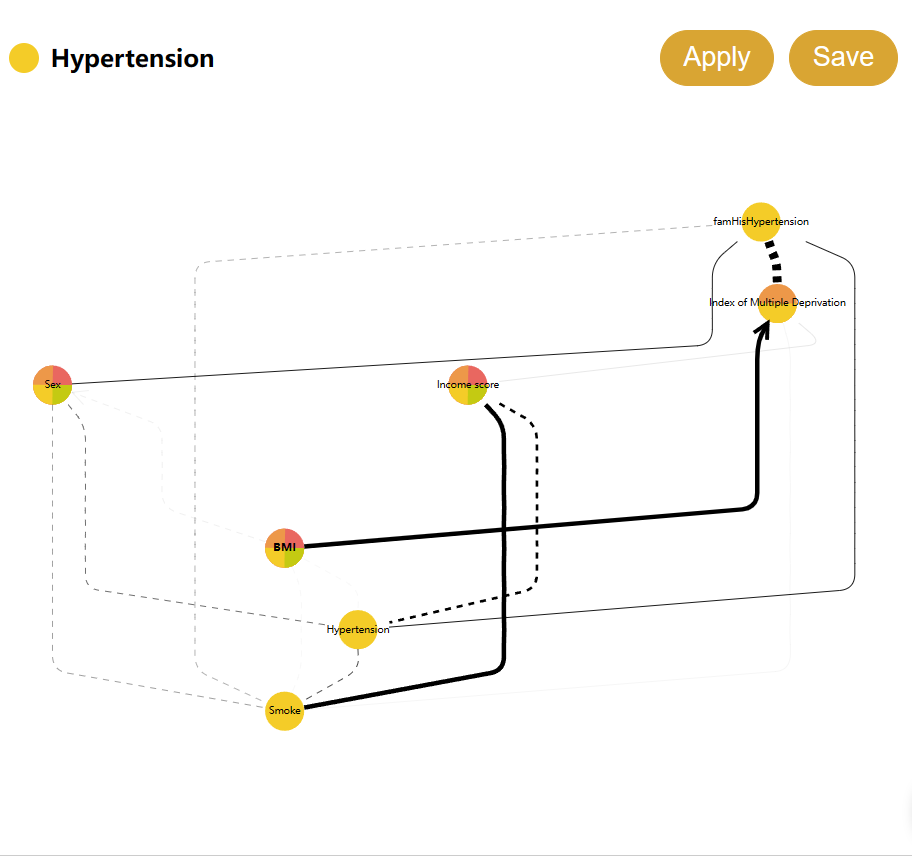
使用Dagre-d3工具（紧致树算法）获取Super Graph（上下方向），获得Super Graph中所有坐标（y坐标和x坐标可分别用来计算每个节点的rank属性和在该层内的节点排序），然后使用Jointjs重新绘图



Extracted Graph：

获取点的**绝对坐标**，绘制每个子图；每个子图根据绘图区域大小进行了合适的整体缩放。



Optimal Graph：

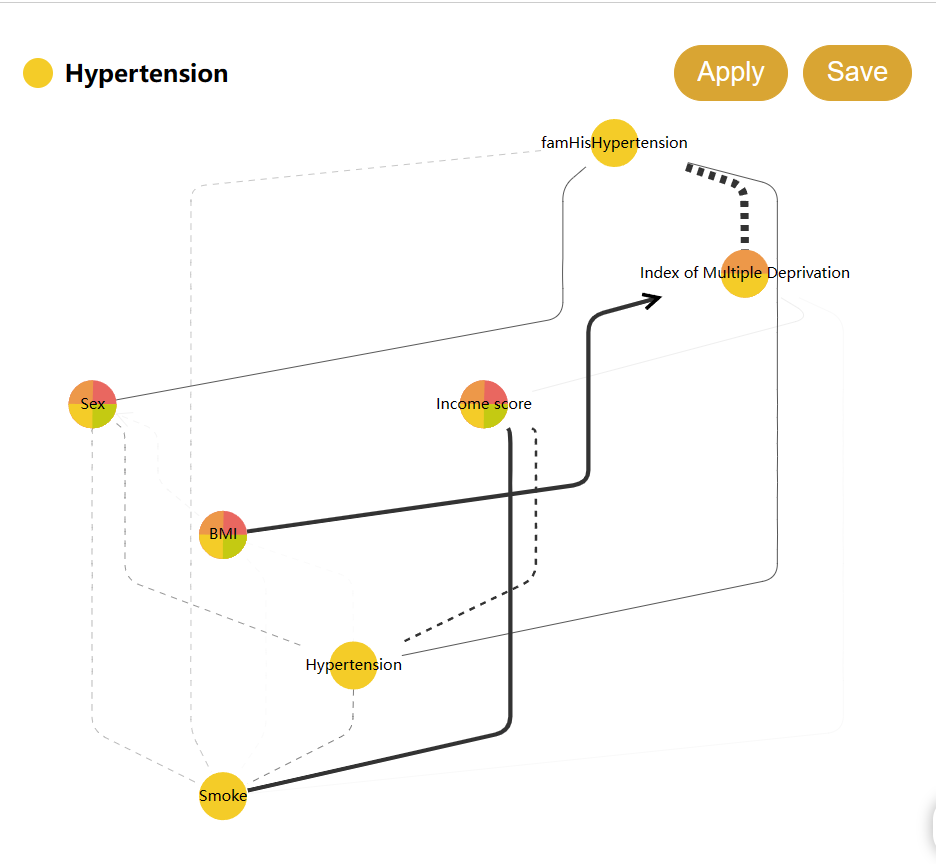
获取点的绝对坐标，并计算其原始layer和rank。

纵轴压缩：去掉子图中所有没有点的layer；

横轴压缩: 重新获得每一层的点排序，但不改变原始的垂直对齐（比如下图中的Hypertension仍然在BMI和income score之间），去除不必要的rank

根据绘图区域大小计算x-gap和y-gap，计算出每个点的坐标

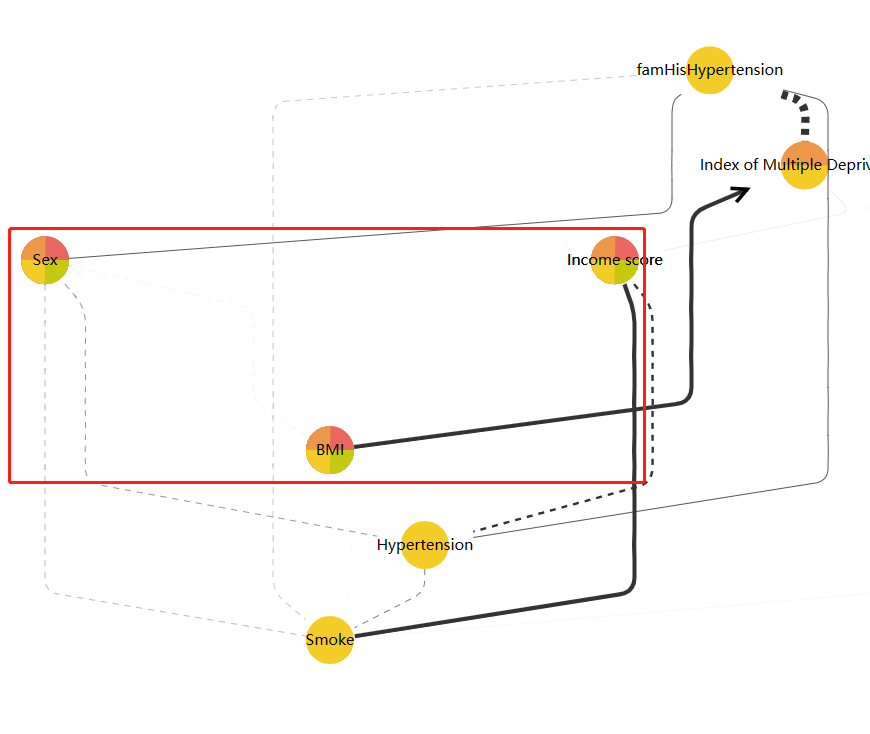
线的拐点按照在与最近的layer和rank的相对距离进行坐标的计算



Tightened Graph：

基本处理方案与Optimal类似，区别在于，不对共用点的layer和rank进行处理。根据最初获得的layer和rank锚定共用点，共用点划定区域[Min\_X~Max\_X]（[Min\_Y~Max\_Y]），其余点x坐标在此范围内则不改变（y坐标在此范围内则不改变，在此范围之外的点同样去除不必要的rank（layer），即完成共用点之外的点的坐标压缩。

（下图中彩色方框为后来添加，不代表绘图效果）

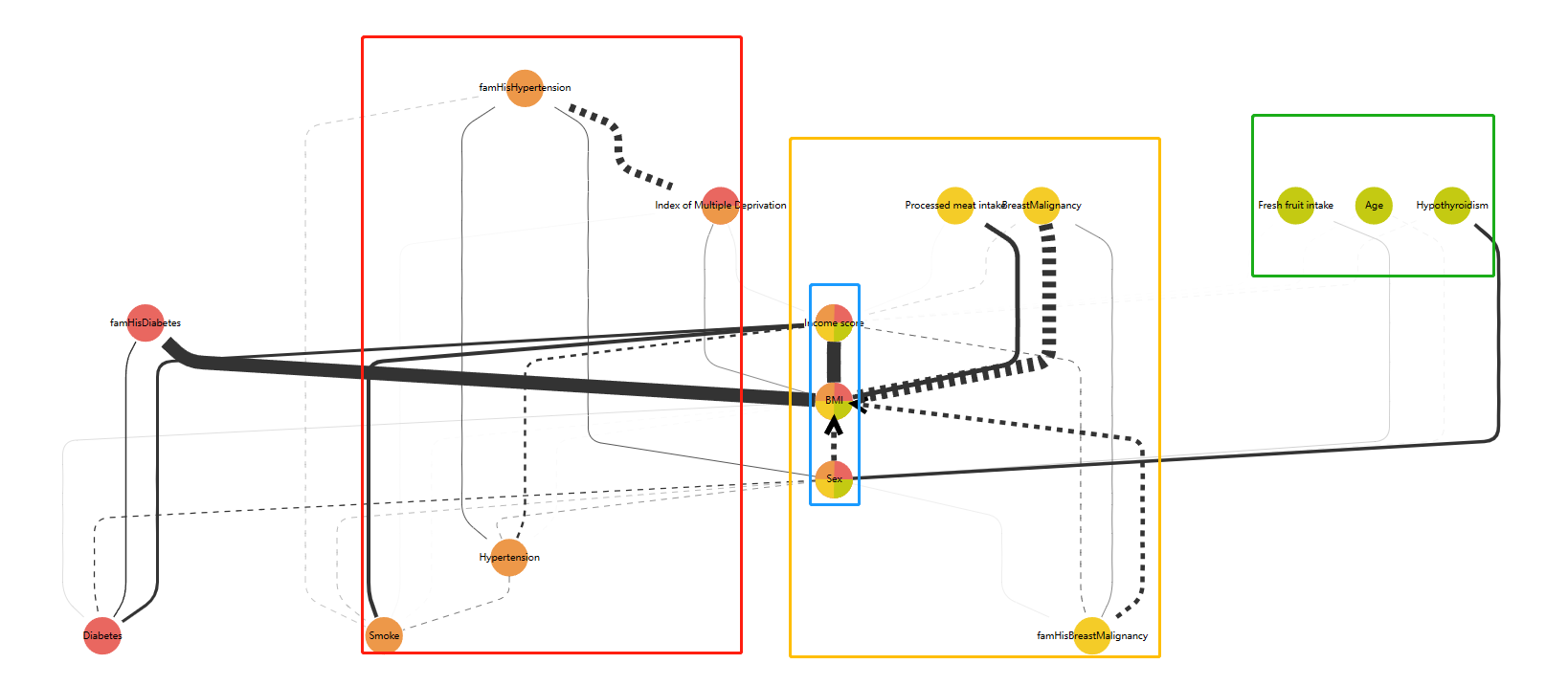


Original Graph：

子图单独使用dagre计算并绘制

**新方案：**

使用Dagre Cluster绘制分层复合超图，即对点进行分组后再使用dagre绘制，以使每个子图包含的点位置紧密，同时共用点的位置紧密（图中彩色方框为后来添加，不代表绘图效果）



使用此方案，Tightened Graph能够有比较明显的效果（图中彩色方框为后来添加，不代表绘图效果）

