In the age of big data, humans need new tools to visually and interactively explore their ever-growing data sets.

**Adaptive layout engines** will enable this by adjusting runtime behavior based on available resources and properties of the problem at hand.

ex: treemap visualizations

500 data points

versus

95,000 data points

Using a declarative domain-specific language, visualization designers specify layout problems as **attribute grammars**.

With a fine-tuned constraint model, an SMT solver takes less than a minute (rather than 8+ hours) to find valid **traversal schedules**, which compile to layout engines.

-ex: sequential schedule

\[
\text{post}\{\text{height, width}\} \quad \text{;; pre}\{\text{bottom, right}\}
\]

-ex: parallel schedule

\[
(\text{post}\{\text{height}\};\text{pre}\{\text{bottom}\}) \quad || \quad (\text{post}\{\text{width}\};\text{pre}\{\text{right}\})
\]

**Adaptation happens both...**

**Ahead of time**

By augmenting synthesis with a **cost model**, the search can find a set of schedules, each optimal under different circumstances.

**Just in time**

Given an input tree, a **cost predictor** will dispatch to one of the layout engines based on properties such as tree size, shape, number of cores, cache size, etc.