Machine Learning from Prior Designs for Facility Layout Optimization

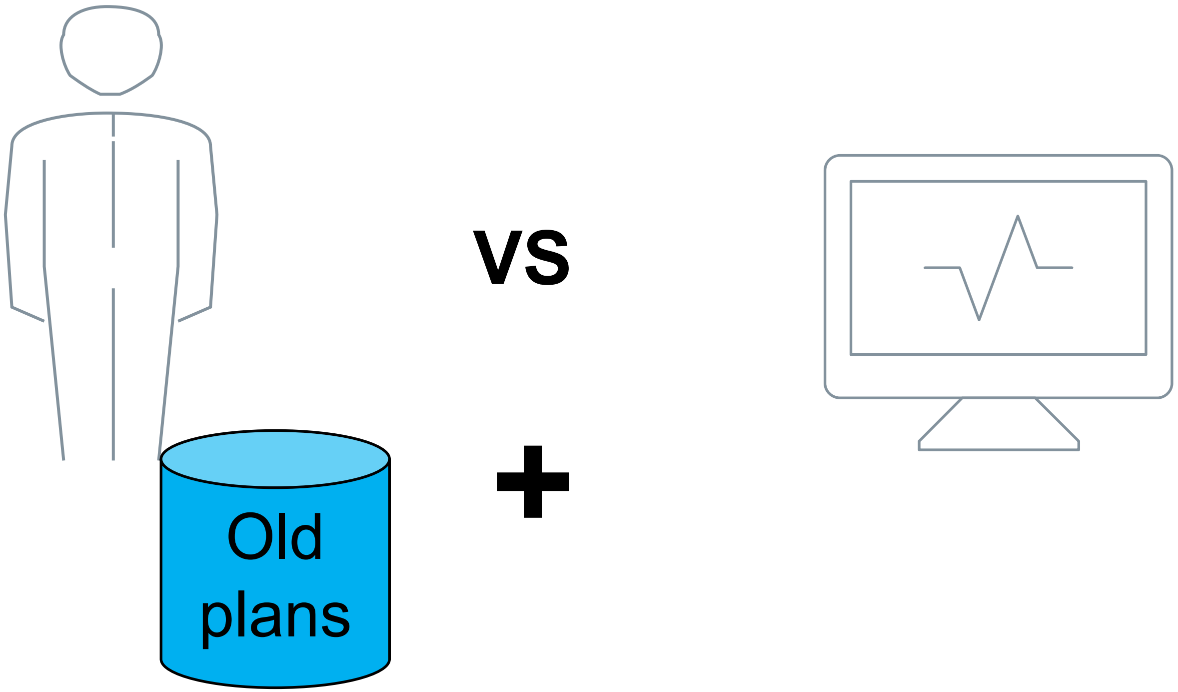
29.10.2018

Project: Engineering Rulez

Hannu Rummukainen

Jukka K Nurminen

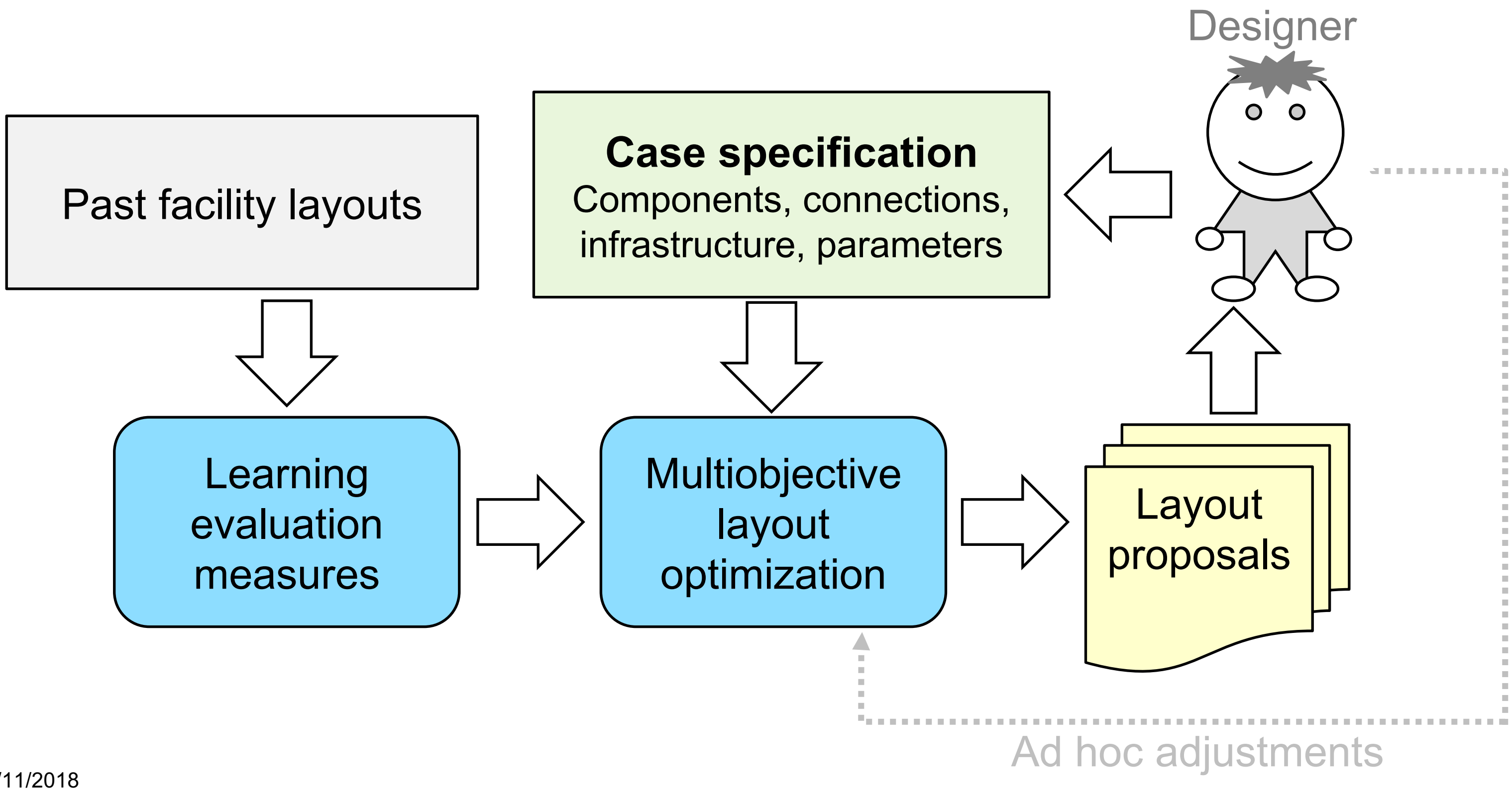
Facility Layout Planning


Old plans
vs
+

Facility Layout in Paper and Pulp Mills

- Where to place the main equipment inside a given building?
- Use case: Speed up the initial design of a new building or a department
    - ➢Provide diverse alternative layouts for designer
- Designer must consider huge number of design rules and objectives
    - Cumbersome to model everything explicitly
- Alternative approach for design automation:
  Learn rules and objectives from sample designs of old projects

Solution overview



Research Questions

- How can we learn the input for constraint-based layout optimization from limited data?
- Can an explicit layout optimization model be combined with implicit rules learned from expert-designed layouts?

- What learning models would be most suitable?
- Can this kind of learning-based layout lead to a practical design tool?

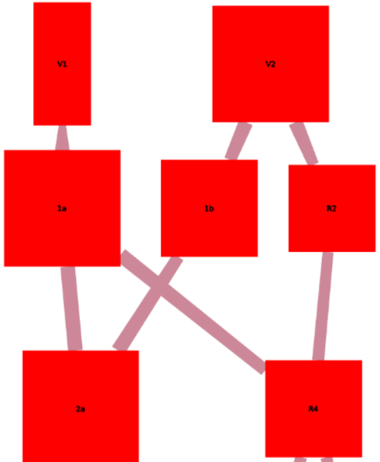
Layout model: Decisions

Given:

- Coordinate grid and bounding walls
- Set of required components, each with one or more alternative patterns:
  - Dimensions of bounding box (width × height)
  - Connection point in the middle
- Connection graph between components
  - Connections may be of different types

To be decided:

- For each required component:
  - Grid position
  - Choice of pattern
  - Orientation at 90° intervals



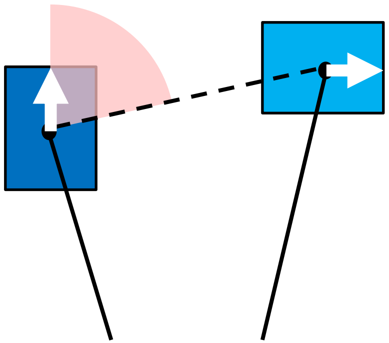
Layout model: Objectives

1. Model-based

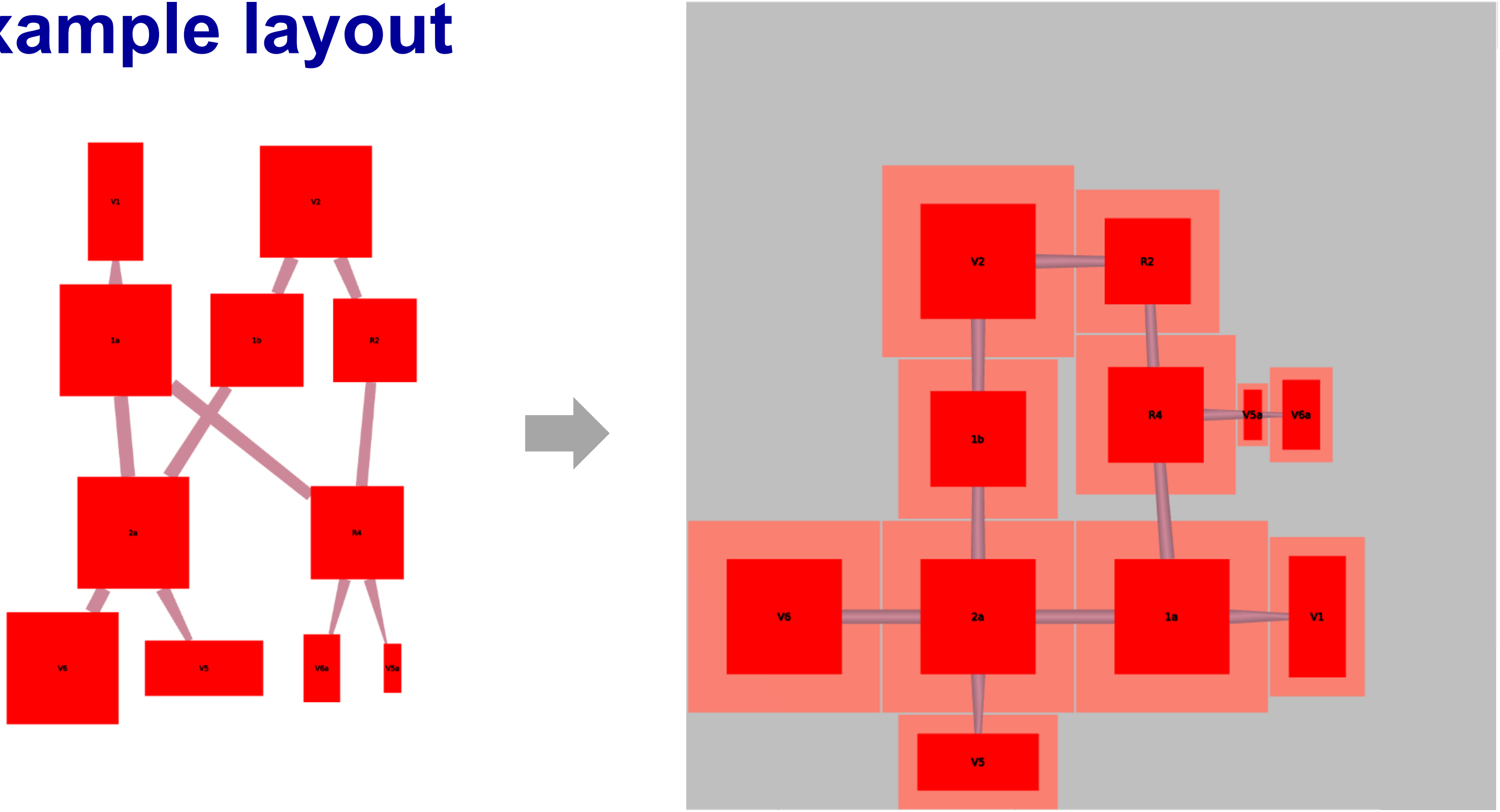
- Connection cost between components
  - Distance between connection points (Manhattan distance, or other metric)
  - Weighted by connection type

2. Learning-based

- Local similarity to reference data
  - Pairwise distance between similar component types
  - Relative angle between similar component types



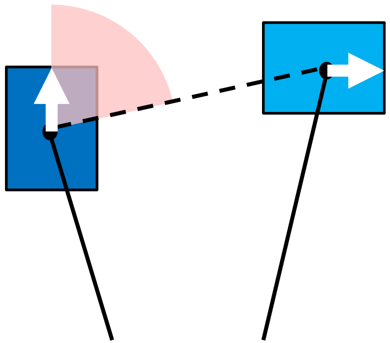
Example layout



Solution approach

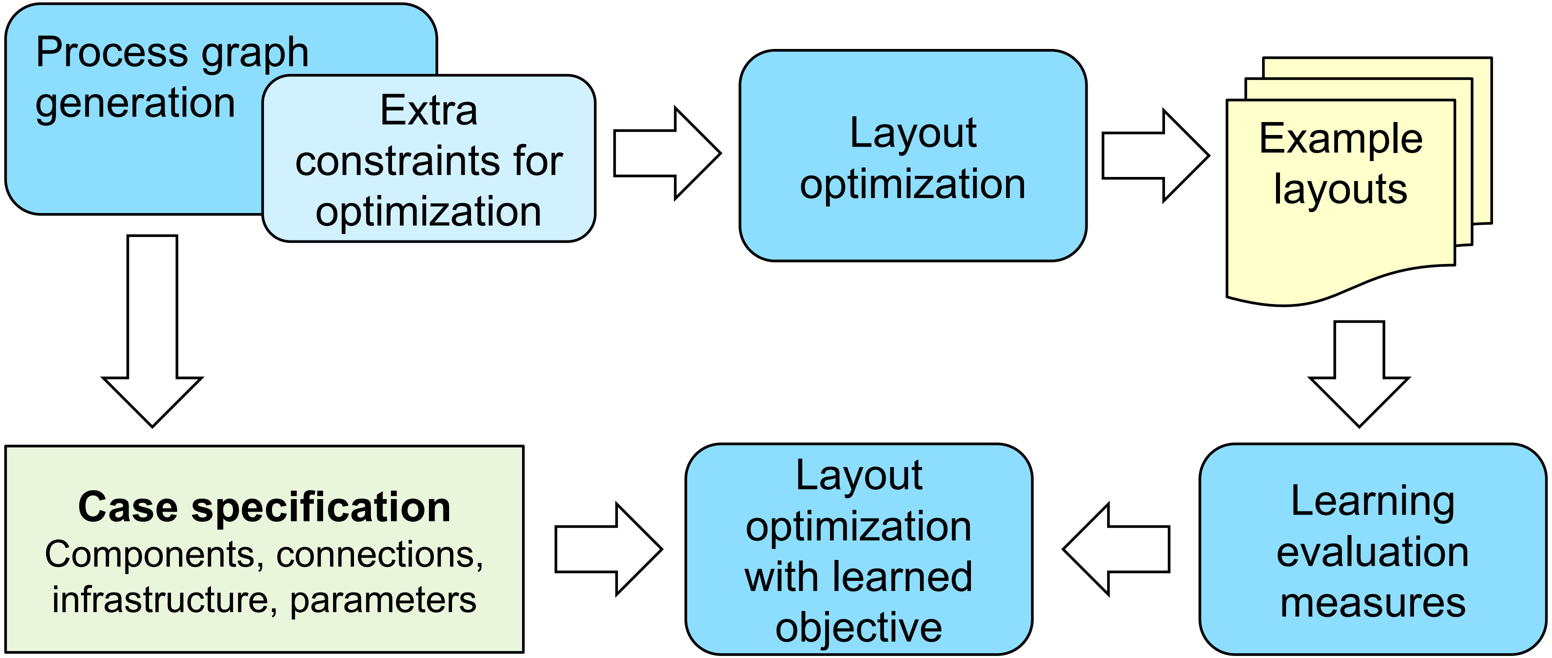
- Similarity measured by likelihood (probabilistic model)
  - Data-based model learned by kernel density estimation
  - Tools: Python statsmodels library
- Layout optimization by constraint programming
  - Algorithm finds 1) feasible solutions and 2) the optimum given enough time
  - Can handle ad-hoc rules added by designers
    - E.g. *“Move X and Y away from each other”, “Put component X on this area”*
  - Tools: MiniZinc modelling language, Chuffed & Yuck solvers (global / local optimization)
- Generate diverse Pareto-optimal solutions by weighting objectives

Similarity model: basic idea



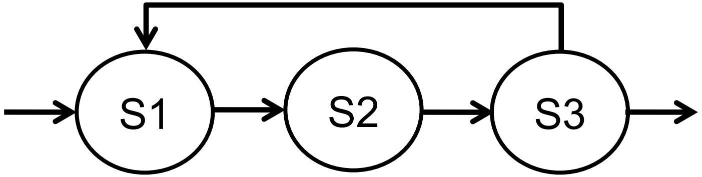
- Independent variables:
  - T_i, T_j – component types
  - Δ – distance in undirected connection graph (1, 2, … steps)
  - O_i – component orientation
- Dependent variables:
  - Θ – angle between component orientation and direction of other component
  - D – distance between connection points
- $\mathrm{P}(\text{layout}) = \prod_{i,j\in\text{components}} \mathrm{P}(\Theta_{ij}, D_{ij})$
- $\mathrm{P}(\Theta_{ij}, D_{ij}) = \mathrm{P}(\Theta_{ij} | T_i, T_j, O_i, \Delta_{ij})\mathrm{P}(D_{ij} | T_i, T_j, O_i, \Delta_{ij})$

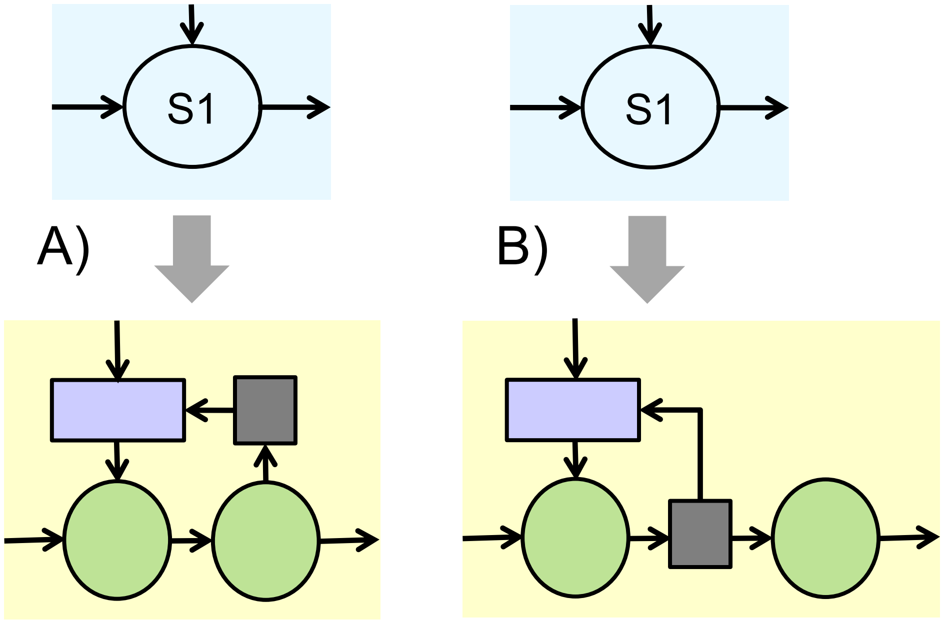
Generating artificial test data



Process graph generation

- Graph built from 3 parts, each with 2 alternatives

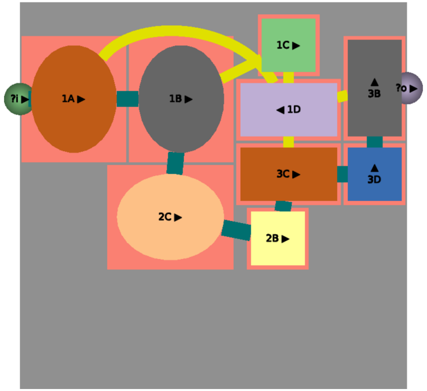


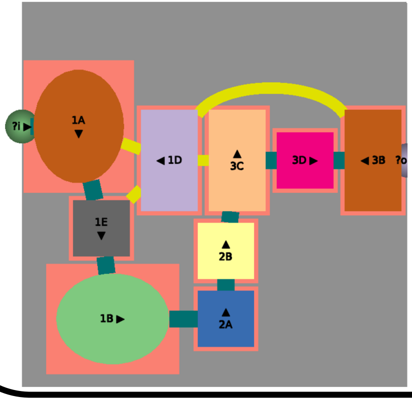


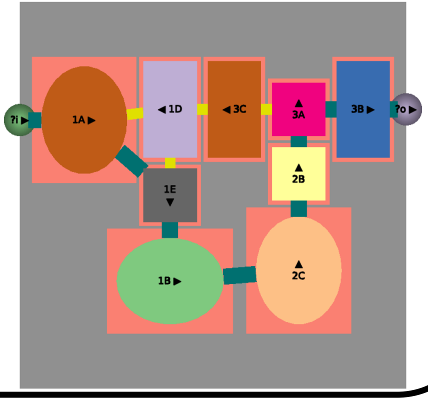
- 3 component types
  - Large cylinder
  - Mid-size rectangle
  - Small square

Generated process graphs

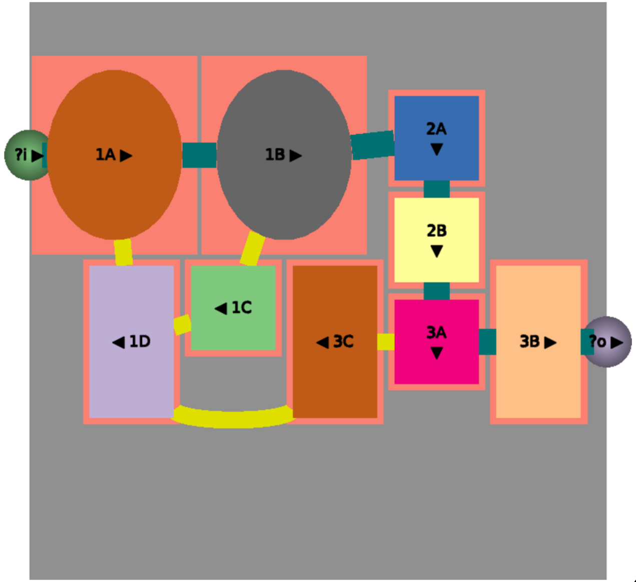
Example graphs







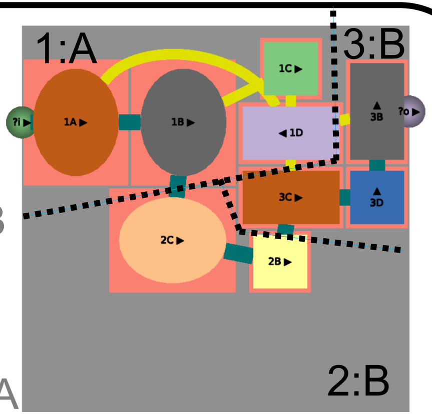
Layout case graph



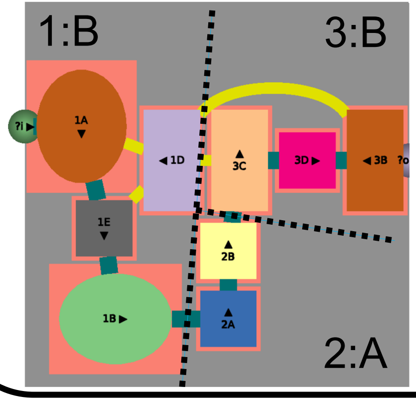
Generated process graphs

Example graphs

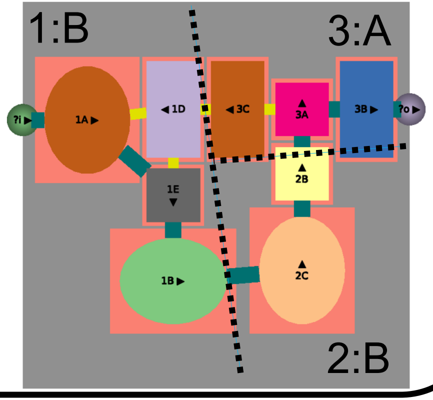
ABB



BAB

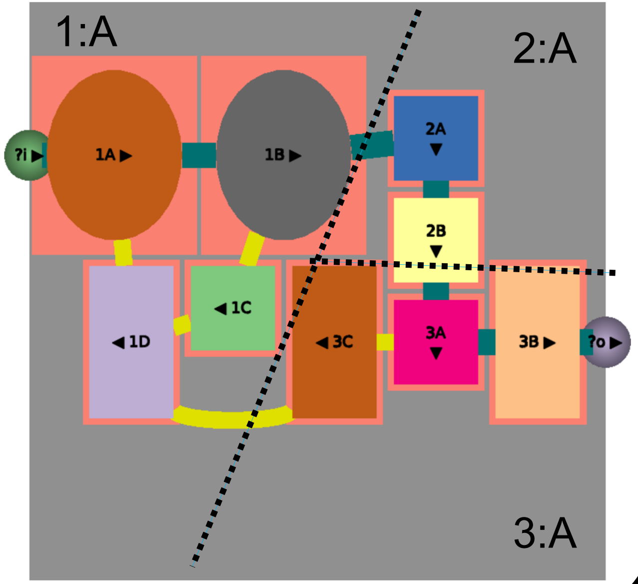


BBA



Layout case graph

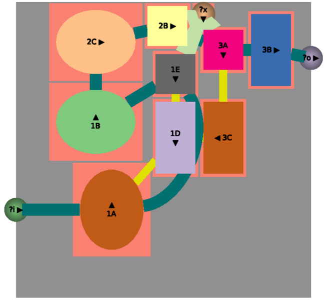
AAA



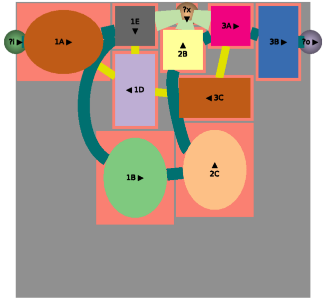
Basic example: Optimizing similarity

- Same process graph in examples and case (BBA)
    - …but examples have extra node to force some component positions
- Example layout goal: Minimize weighted pipe length
- Layout case goal: Maximize similarity
    - No explicit pipe length objective
    - 2 layouts with best similarity measure shown

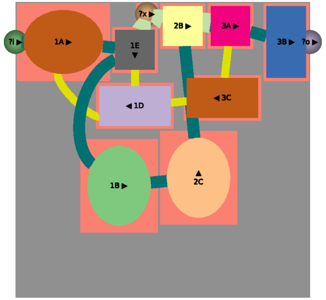
Optimizing similarity: Squares near edge



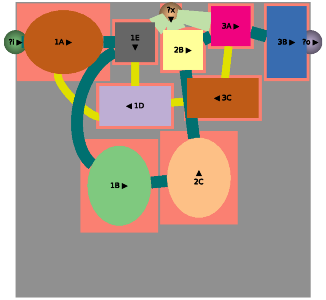
COST=636 : main=468 rej=168 (2.0, 8.0)
SIM=1107 : sim_angle=712 sim_dist=395
PO example-A-layout-1.png



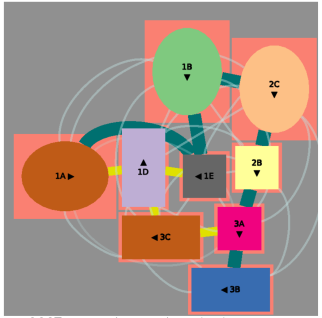
COST=642 : main=462 rej=180 (4.0, 6.0)
SIM=1134 : sim_angle=740 sim_dist=394
PO example-A-layout-2.png



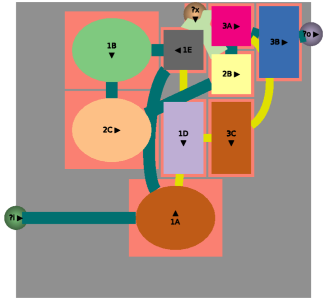
COST=640 : main=456 rej=184 (4.0, 6.0)
SIM=1110 : sim_angle=715 sim_dist=395
PO example-A-layout-3.png



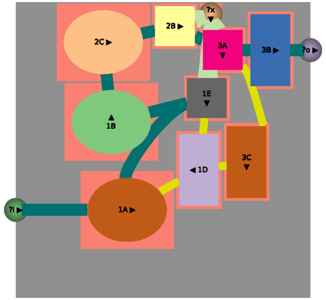
COST=644 : main=456 rej=188 (4.0, 6.0)
SIM=1126 : sim_angle=733 sim_dist=393
PO example-A-layout-4.png



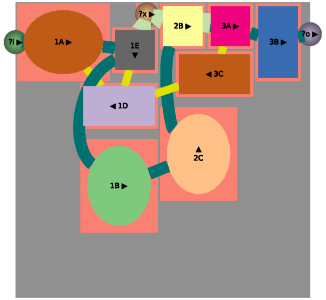
COST=572 : main=420 rej=152 (0, 0)
SIM=1320 : sim_angle=784 sim_dist=536 (5.0, 9.0)
PO learned-A-layout-0.png



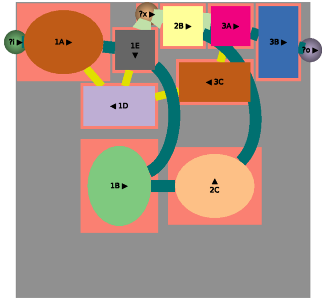
COST=630 : main=462 rej=168 (4.0, 6.0)
SIM=1143 : sim_angle=733 sim_dist=410
example-A-layout-6.png



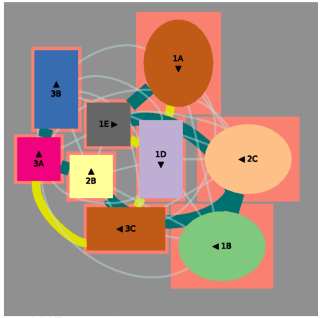
COST=648 : main=456 rej=192 (6.0, 3.0)
SIM=1099 : sim_angle=699 sim_dist=400
example-A-layout-7.png



COST=642 : main=450 rej=192 (4.0, 6.0)
SIM=1101 : sim_angle=701 sim_dist=400
example-A-layout-8.png

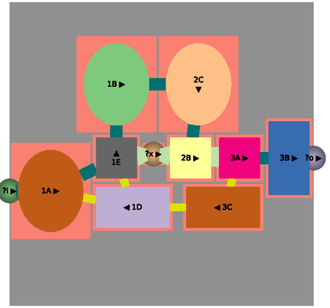


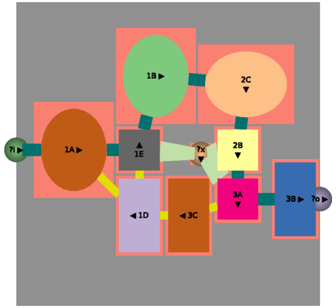
COST=666 : main=474 rej=192 (4.0, 6.0)
SIM=1110 : sim_angle=711 sim_dist=399
example-A-layout-9.png

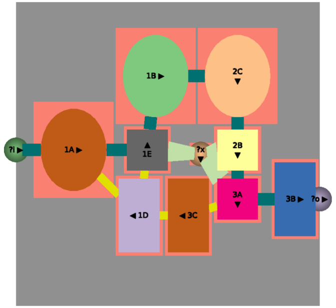


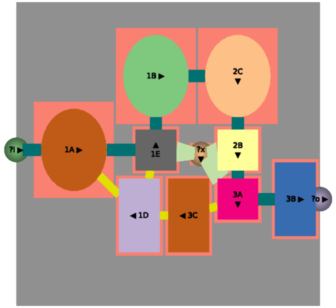
COST=770 : main=558 rej=212 (0, 0)
SIM=1329 : sim_angle=795 sim_dist=534 (1.0, 1.0)
learned-A-layout-1.png

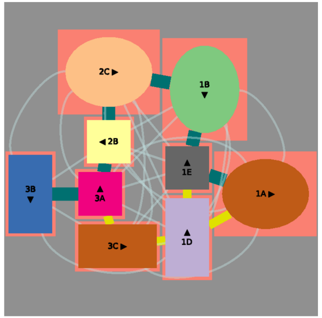
Optimizing similarity: Squares in centre

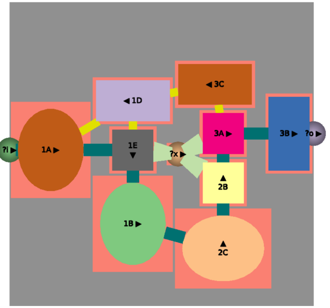

COST=474 : main=318 rej=156 (8.0, 6.0)
SIM=1096 : sim_angle=738 sim_dist=358
PO example-B-layout-1.png

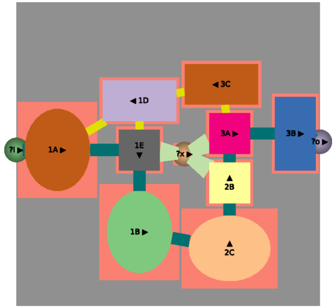

COST=470 : main=318 rej=152 (7.0, 7.0)
SIM=1031 : sim_angle=690 sim_dist=341
example-B-layout-2.png

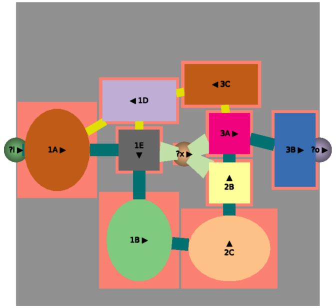

COST=462 : main=306 rej=156 (7.0, 7.0)
SIM=1055 : sim_angle=714 sim_dist=341
example-B-layout-3.png

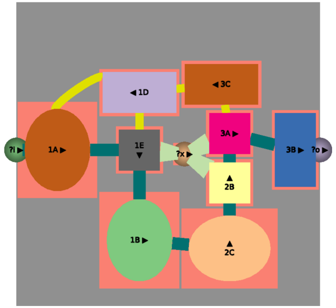

COST=466 : main=306 rej=160 (7.0, 7.0)
SIM=1056 : sim_angle=714 sim_dist=342
example-B-layout-4.png

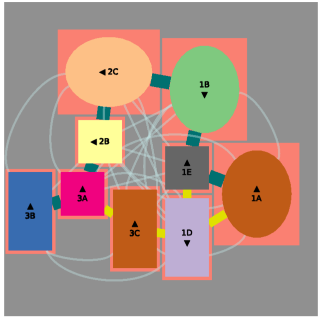

COST=510 : main=354 rej=156 (0, 0)
SIM=1425 : sim_angle=815 sim_dist=610 (0.0, 10.0)
PO learned-B-layout-2.png


COST=494 : main=330 rej=164 (8.0, 4.0)
SIM=1115 : sim_angle=765 sim_dist=350
example-B-layout-6.png


COST=494 : main=330 rej=164 (8.0, 4.0)
SIM=1114 : sim_angle=764 sim_dist=350
example-B-layout-7.png


COST=506 : main=342 rej=164 (8.0, 4.0)
SIM=1118 : sim_angle=764 sim_dist=354
example-B-layout-8.png


COST=510 : main=342 rej=168 (8.0, 4.0)
SIM=1120 : sim_angle=764 sim_dist=356
example-B-layout-9.png

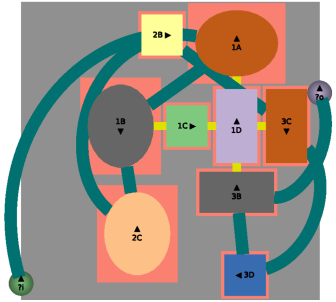

COST=512 : main=360 rej=152 (0, 0)
SIM=1446 : sim_angle=833 sim_dist=613 (0.0, 10.0)
PO learned-B-layout-3.png

Advanced example: Similarity trade-off

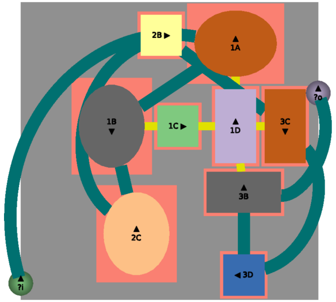
- Three different process graphs in examples
    - Total 3×3 = 9 examples
- Fourth different process graph in case
- Pipes split into two colours
    - **The colour is not known by the learning model!**

- Example layout goal: Minimize yellow pipe length
- Layout case goals: Minimize green pipe length & maximize similarity

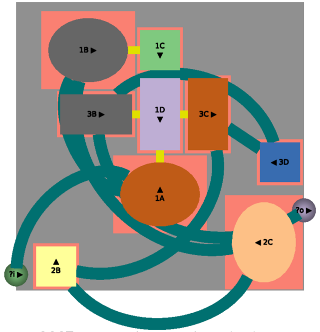
Example layouts minimizing yellow pipes



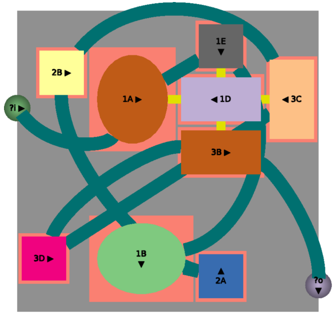
COST=908 : main=756 rej=152 (0, 1)
SIM=1226 : sim_angle=753 sim_dist=473
PO example-VX0-layout-0.png



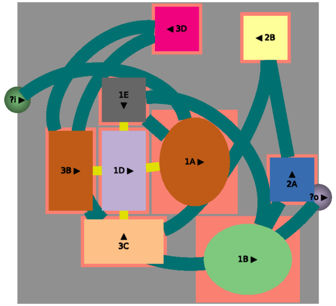
COST=922 : main=762 rej=160 (0, 1)
SIM=1228 : sim_angle=754 sim_dist=474
example-VX0-layout-1.png



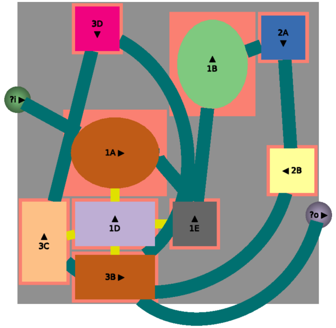
COST=1268 : main=1104 rej=164 (0, 1)
SIM=1254 : sim_angle=756 sim_dist=498
example-VX0-layout-2.png



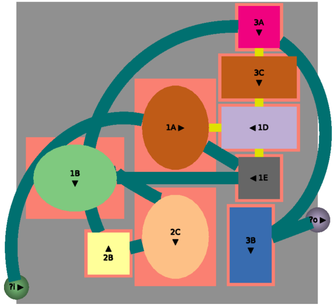
COST=1368 : main=1248 rej=120 (0, 1)
SIM=1299 : sim_angle=805 sim_dist=494
PO example-VX1-layout-0.png



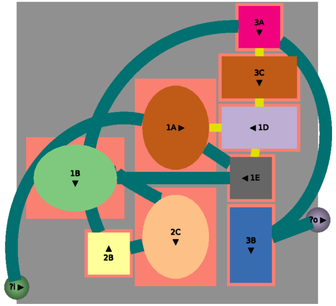
COST=1186 : main=1062 rej=124 (0, 1)
SIM=1290 : sim_angle=803 sim_dist=487
example-VX1-layout-1.png



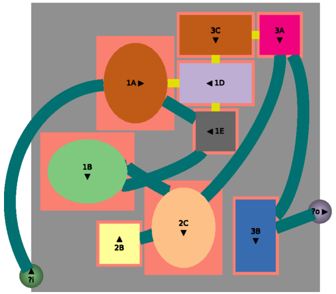
COST=1092 : main=960 rej=132 (0, 1)
SIM=1274 : sim_angle=780 sim_dist=494
example-VX1-layout-2.png



COST=946 : main=834 rej=112 (0, 1)
SIM=1181 : sim_angle=718 sim_dist=463
PO example-VX2-layout-0.png

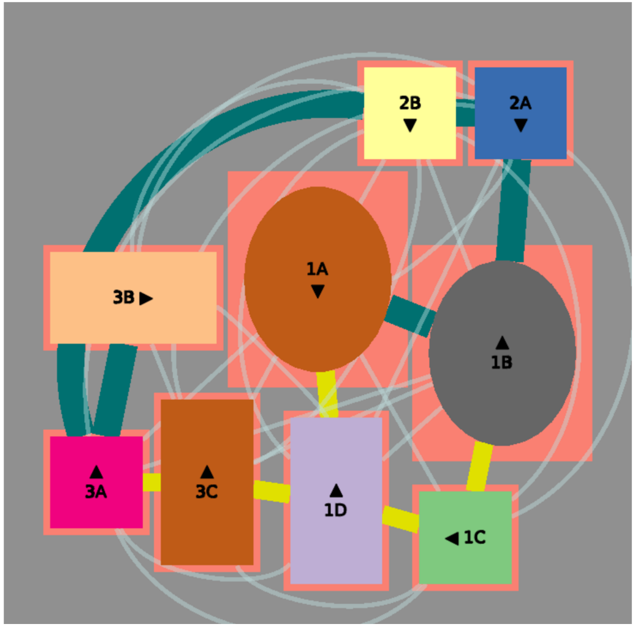


COST=938 : main=822 rej=116 (0, 1)
SIM=1162 : sim_angle=702 sim_dist=460
example-VX2-layout-1.png

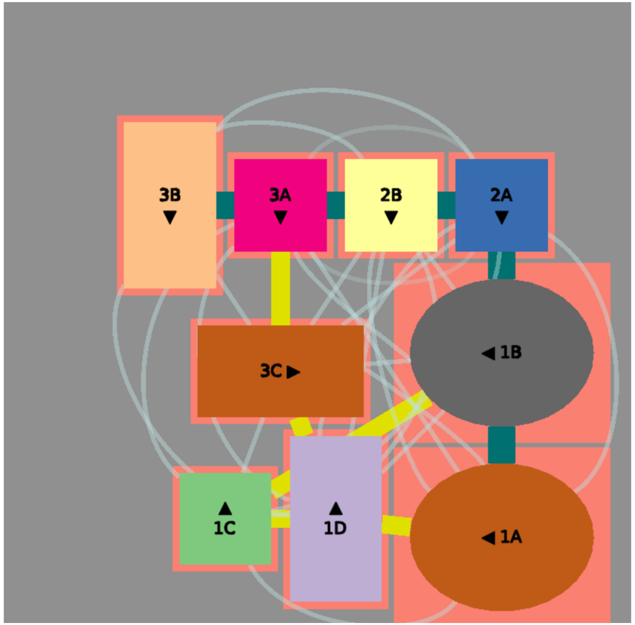


COST=960 : main=840 rej=120 (0, 1)
SIM=1174 : sim_angle=713 sim_dist=461
example-VX2-layout-2.png

Case layouts minimizing similarity vs other pipe type

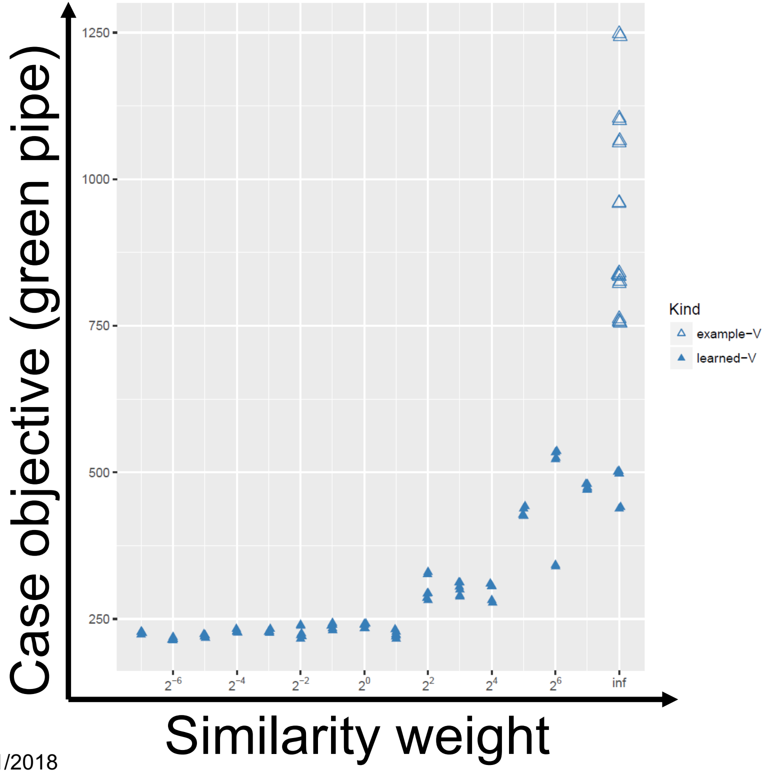


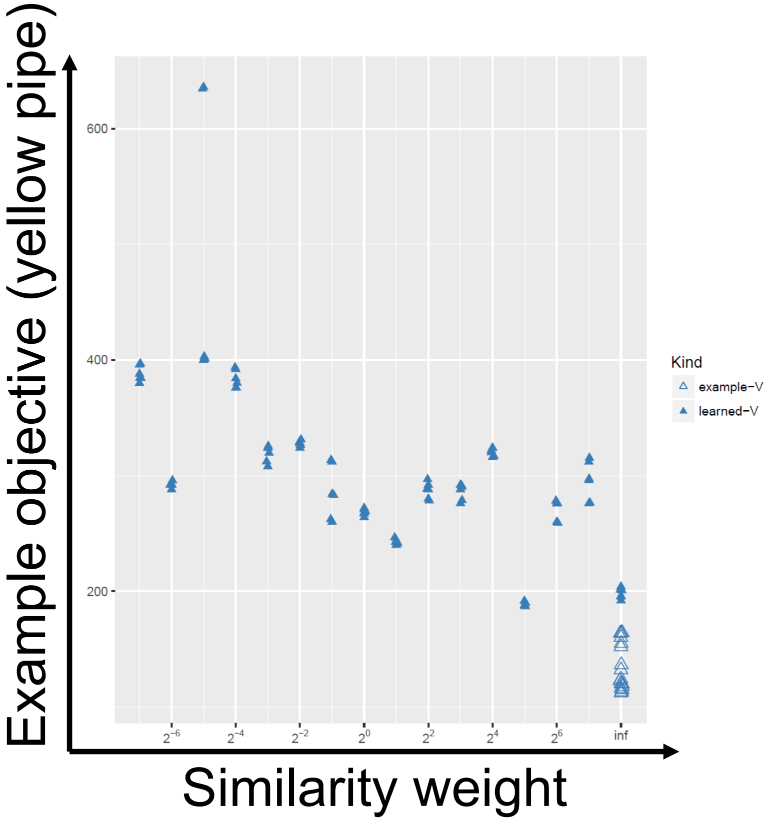
Weighting similarity



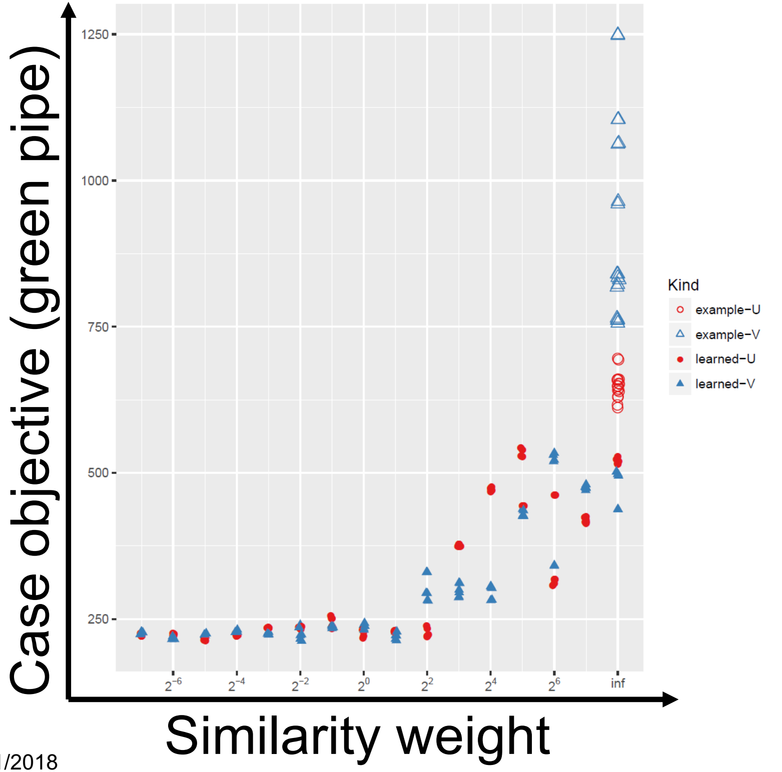
Weighting green pipe length

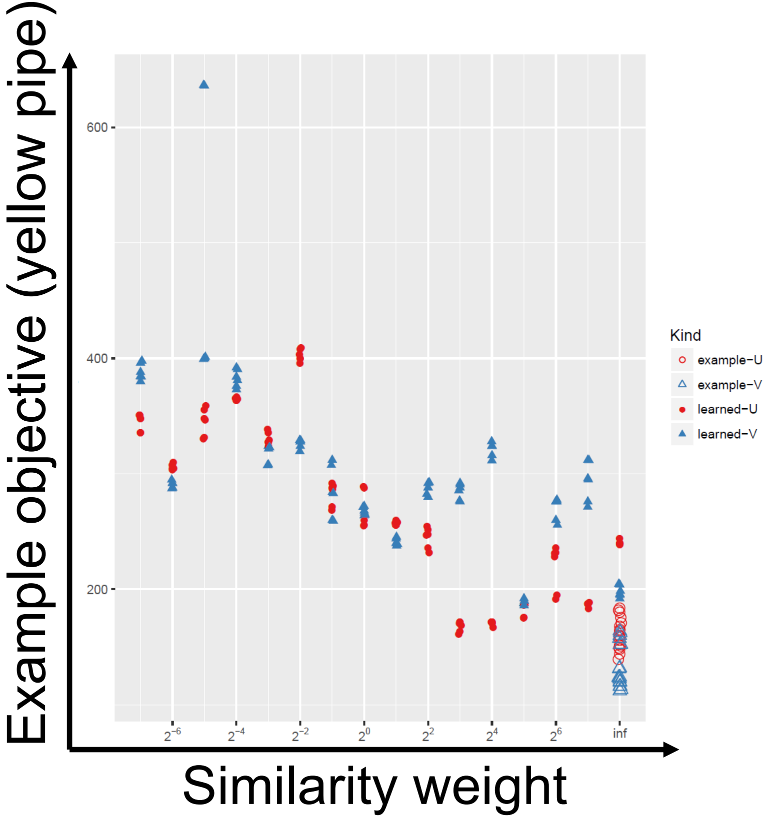
Trading off similarity and objective





With examples with the same topology (red)





Possible future work

- Experiments with real-world facility data
  - Components on multiple floors
  - Leave service space at specific locations around components
  - Larger amounts of components
  - May require performance improvements and/or more computing power
  - Evaluation of results with experienced facility designers
- More flexible learning models
  - E.g. dealing with machine operator walkways is currently hard;
    empty spaces for personnel could be addressed directly in the similarity model, or try black-box learning
- Learn explicit constraints in addition to “soft” objective
  - E.g. minimum safe distances between components (pairwise)


TECHNOLOGY FOR BUSINESS