Machine Learning from Prior Designs for Facility Layout Optimization

H. Rummukainen[1], J. K. Nurminen[1], T. Syrjänen[2], and J.-P. Numminen[2]

[1] VTT Technical Research Centre of Finland
hannu.rummukainen@vtt.fi
[2] Pöyry Finland
timo.syrjanen@poyry.com

Abstract. The problem of facility layout involves not only optimizing the locations of process components on a factory floor, but in real-world applications there are numerous practical constraints and objectives that can be difficult to formulate comprehensively in an explicit optimization model. As an alternative to explicit modelling, we present an optimization approach that learns structural properties from examples of expert-designed layouts of other similar facilities, and considers similarity to the examples as one objective in a multi-objective facility layout optimization problem. We have tested the approach on small-scale artificial test data, and the initial results indicate that a layout objective can be learned from example layouts, even if the process structure in the examples differs from the target case.

1 Introduction

We consider the problem of facility layout, that is where to place the various process components, e.g. production machines, in a manufacturing facility. Whether in process industry or in goods manufacturing, the design of a facility is a complex multi-disciplinary effort. The layout of components is directly or indirectly affected by numerous factors starting from the construction, operation and maintenance of relevant production technologies and infrastructure, and including considerations such as production economics, safety, security, environmental issues, and regulatory requirements.

Because the number of influencing factors is large, and often difficult to specify completely, modelling the task explicitly as an optimization problem is difficult or impossible. The key idea behind our work is that earlier plans for different but similar facilities contain a lot of useful knowledge. The experts who have created the earlier plans have used their skills and expertise to consider and balance the multiple influencing factors. As a result, the plans contain a lot of explicit as well as hidden knowledge that can be useful for future planning tasks.

In this work we investigate the use of machine learning to distill facility layout knowledge from old plans and apply it to the design of new facilities. Instead of attempting to include all essential factors to facility layout optimization explicitly we use probabilistic machine learning to generate new goals for the model automatically. In this way we think the optimization model better reflects the real world complexities without expending modelling effort to find, formulate, and balance the different factors.

At the time of initial layout design, the data available about the particular facility is typically limited, and it is not practical to require the user to input large amounts of site-specific parameters. An automated decision support tool should regardless be able to quickly provide plausible layouts that can be used as a starting point for more detailed planning and design. If the designer disagrees with automated layout decisions, it should then be easy to interactively improve the generated designs.

An established design consultant or manufacturing business may have detailed design data about dozens of facilities, which can then be used as source data for machine learning. Our research question is whether such data is sufficient to learn relevant design rules well enough to apply them to new designs. If so, machine learning has the advantage of both reducing the complexity of the layout model and reducing the need for explicit site-specific parameter data.

The key contribution of this paper are:

1. We propose the idea of using knowledge automatically extracted from the plans of old facilities in the design of new ones (Section 1)

2. We formulate a probabilistic similarity model used to extract the essential planning knowledge from old plans (Section 4).
3. We combine a basic facility layout model (Section 3) and the similarity model into an optimization model which uses similarity model as an additional goal (Section 5)
4. We evaluate the idea with small-scale artificial test data (Sections 6–7).

Figure 1 illustrates how the proposed optimization approach would be used in practice in a decision support tool by a facility designer.

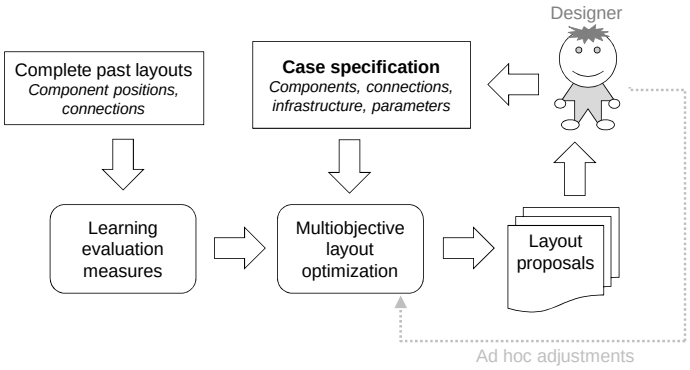


Fig. 1. Concept of a decision support tool for facility layout, based on example layouts. Working on a new facility, the designer outlines the basic process and relevant constraints to the decision support tool, and gets a small, varied selection of plausible layouts as a starting point for more detailed design work.

2 Related work

In the operations research literature, facility layout has typically been addressed as a combinatorial optimization problem, where the most important layout rules are explicitly modelled [1–3]. Typically the primary objective is to minimize material transport costs derived from inter-component distances. Additional objectives and constraints may be used to model e.g. safety considerations [4]. However, it would be complicated to fully model all relevant design considerations. Due to the limitations of existing models, automatic facility layout models have made little headway as a practical design tool to date [5]: The initial design of a process plant is in practice performed by an experienced designer, and detailed design is performed by a multi-disciplinary team of experts.

The issue of how to include human expertise in automated facility layout has been commonly addressed by actively involving expert designers either in the model specification or in the solution process. In our work we do not require experts to directly or indirectly weight different objectives. Instead, we implicitly learn those weights from old plans, which we suggest contain the expert judgement used to create them. To the best of our knowledge we are the first to attempt facility layout optimization with similarity to expert-designed layouts as an explicit optimization goal.

Because of the difficulty of modelling and weighing all relevant factors, researchers have suggested using expert opinions to rate multiple plans generated by computational tools [6]. Similarly, García-Hernández et al. [7, 8] involve a human expert designer in facility layout via interactive multiobjective optimization: After each iteration of a genetic algorithm, a representative subset of solutions are scored by the expert. An explicit quantitative objective (based on weighted transport distance) and the expert scores are treated as separate objectives, and Pareto-optimal solutions are then selected for the genetic algorithm to work on in the next iteration.

In order to automate processing of *qualitative* design rules, Grobelny and Michalski [9] apply fuzzy set theory to formalize layout rules expressed in semi-formal natural language, and then run simulated annealing to optimize the mean fuzzy truth value of the rules.

Chung [10] presents a neural network based method to generate new facility layouts that are similar to given examples. The method does not consider any explicit objectives or constraints on the generated layouts. Ahmad et al. [11] train a neural network to reproduce expert evaluations

of given layouts, and propose that the resulting model would be useful for optimization; however the network was limited to four quantitative measures of the layout as inputs, and as many as 500 manually evaluated layouts were used for training data.

Merrell et al. [12] generate building layouts by training a Bayesian network on example layouts. Their method generates new layouts in two stages: room shapes, sizes and adjacencies are sampled from the trained Bayesian network, and then the rooms are positioned by a separate optimization procedure with an explicitly modelled objective. Our approach does not need a separate sampling stage, but embeds a probabilistic similarity model directly in the final layout optimization model.

Machine learning has been applied to many planning tasks. Approaches to use machine learning for VLSI circuit design are reviewed in [13].

The problem of deriving a constraint programming model from feasible and infeasible example solutions is in general quite challenging, but has been addressed by several methods [14] including machine learning [15]. Mizoguchi and Ohwada [16] present an inductive logic programming method that derives constraints from examples of feasible solutions only, and apply the method to a floor layout problem. In contrast, we derive an evaluation function instead of hard constraints, and we avoid the complexity of the general constraint acquisition problem by focusing on the modelling of geometric similarity. In optimization literature, the problem of deducing an objective function that makes the given solutions optimal is known as inverse optimization, and again the general problem is hard [17, 18]. We know of no applications of inverse optimization to facility layout problems.

Lombardi et al. [19] present a methodology for embedding a machine learning model into a combinatorial optimization model. Our work can be seen as an application of their methodology, extending it by 1) embedding a *probabilistic* machine learning model in a constraint programming model, and 2) using a learned objective in addition to an explicit model objective.

3 Facility layout model

Here we define the basic layout model that will be combined with the similarity model in Section 5.

The production process is described by a directed graph (V, E), called *process graph*, where the nodes V are identified with components, and the edges $E \subseteq V \times V$ indicate material flows between components. Note that directedness is not needed for the layout model, but we use the information later in the similarity model of Section 4. For each edge $(i, j) \in E$, the distance between components i and j contributes to the objective function with a cost coefficient $c_{ij} \geq 0$.

Each component $i \in V$ can be realized as a number of different rectangular patterns, denoted by the set P_i: the choice of pattern $p \in P_i$ determines the width w_p and height h_p of the component. Different patterns may arise when a component is rotated (by 90 degrees), or is replaced by another interchangeable component of different dimensions.

We use a discrete model with integer coordinates on a $W \times H$ grid. Distances between components are measured by rectilinear (taxicab) distance between component centre points; note that we double the distances to avoid half-integral values. We use the following decision variables:

$x_i \in \{0, \ldots, H\}$	Lower left corner x-coordinate of component $i \in V$
$y_i \in \{0, \ldots, W\}$	Lower left corner y-coordinate of component $i \in V$
$p_i \in P_i$	Choice of pattern for node $i \in V$
$d_{ij} \in \mathbb{N}$	Distance from component i to j, for edge $(i, j) \in E$

We start natural numbers $\mathbb{N}$ from 0.

The basic facility layout model is now:

$$\min z_C = \sum_{(i,j)\in E} c_{ij} d_{ij} \tag{1}$$

$$d_{ij} = \left|(2x_i + w_{p_i}) - (2x_j + w_{p_j})\right| + \left|(2y_i + h_{p_i}) - (2y_j + h_{p_j})\right| \qquad \forall (i,j) \in E \tag{2}$$

$$x_i + w_{p_i} \leq x_j \vee x_j + w_{p_j} \leq x_i \vee y_i + h_{p_i} \leq y_j \vee y_j + h_{p_j} \leq y_i \qquad \forall i, j \in V \tag{3}$$

The objective (1) is the sum of edge distance costs. Equations (2) link the distance variables with the component coordinates and pattern choices. Equations (3) ensure that no two components overlap each other.

4 Similarity model

We develop a probabilistic model that assigns a probability density to each potential layout of a given problem instance, based on examples of expert-designed layouts in different problem instances. We expect that the number of observed example layouts is on the order of dozens. Due to the low number, instead of considering the layout as a whole, we focus on pairwise geometric relationships between components. The production processes in the example cases may differ, but we assume a shared classification of process components is available: for example, each component could be classified as either a large tank, a small tank, a pump, or "other component". For best results, all components in the same class should be of approximately the same dimensions.

4.1 Probabilistic layout model

Each example layout is assumed to be drawn independently from a random distribution of "well-designed" process layouts. An example comprises both the process graph described in Section 3 and the positions and dimensions of the components. In addition, we assume that we can observe an orientation for each component, indicating which of the four cardinal directions the component is facing. Note that we assume no knowledge of any objective functions or constraints under which the example layouts could be considered optimal solutions.

In the following, $\mathcal{T}$ denotes the set of shared component types, and $\mathcal{O} = \{(0,1),\ (1,0),\ (0,-1),\ (-1,0)\}$ is the set of orientation vectors. The random variable L represents a process layout from the random distribution of well-designed layouts. We observe the following random variables, which are technically functions of L.

$\mathsf{N} \in \mathbb{N}$		The number of nodes in the process graph
$\mathsf{V} = \{1, \ldots, \mathsf{N}\}$		The index set of nodes in the process graph
$\mathsf{E} \subseteq \mathsf{V} \times \mathsf{V}$		The edges in the process graph
$\mathsf{T}_i \in \mathcal{T}$	$i \in \mathsf{V}$	Type of component i in the shared classification
$\mathsf{C}_i \in \mathbb{R}^2$	$i \in \mathsf{V}$	Position of the midpoint of component i
$\mathsf{O}_i \in \mathcal{O}$	$i \in \mathsf{V}$	Orientation vector of component i

We omit the subscripts when convenient to refer to vectors or matrices formed by subscripted random variables. This is somewhat of an abuse of notation considering that the ranges of the subscripts are not fixed.

Commensurate with distances (2) in the basic constraint model, we define the length of vector $c \in \mathbb{R}^2$ as $d(c) = 2\|c\|_1$. The oriented angle between direction vectors $c, c' \in \mathbb{R}^2$ is denoted by $\alpha(c, c') \in (-Q, Q]$, where the constant Q represents a half turn. For technical reasons we use a nonstandard angle measurement, described in Section 5; however the difference to e.g. radians is immaterial for the similarity model.

To consider locality purely on the graph level, we define a distance measure in the undirected graph corresponding to the directed process graph (V, E): Let $\delta_{ij}(V, E)$, or simply δ_{ij} when clear in the context, be the number of edges between components i and j in the graph, ignoring edge direction. Further, we define the augmented edge set

$$\tilde{E} = \{(i, j) \in V \times V : 0 < \delta_{ij} \leq \delta^{\max}\}, \tag{4}$$

where the distance bound $\delta^{\max}$ is 3 in our tests. Other bounds are quite possible, considering the trade-off that $\tilde{E} = E$ requires the least computational effort, while $\tilde{E} = V \times V$ results in the most complete similarity model.

The similarity model is built on the following auxiliary random variables:

$\tilde{\mathsf{E}}$	The edge set E augmented as above (4)
$\mathsf{H} \in \tilde{\mathsf{E}}$	Uniform random edge from the augmented random graph $(\mathsf{V}, \tilde{\mathsf{E}})$
$\mathsf{I} \in \mathsf{V}$	The source node of edge H
$\mathsf{J} \in \mathsf{V}$	The target node of edge H
$\mathsf{A}_{\mathsf{IJ}} = \alpha(\mathsf{O}_\mathsf{I}, \mathsf{C}_\mathsf{J} - \mathsf{C}_\mathsf{I})$	Angle between orientation of component I and the direction to component J
$\mathsf{D}_{\mathsf{IJ}} = d(\mathsf{C}_\mathsf{J} - \mathsf{C}_\mathsf{I})$	Distance between midpoints of components I and J
$\Delta_{\mathsf{IJ}} = \delta_{\mathsf{IJ}}(\mathsf{V}, \mathsf{E}) \in \mathbb{N}$	Number of edges between nodes I and J

Given a process graph (V, E) and the component types T, we model the conditional density of the layout distribution as

$$f_{\mathsf{C,O|V,E,T}}(C, O \mid V, E, T) = \prod_{(i,j)\in\tilde{E}} f_{\mathsf{A_{IJ}|T_I,T_J,\Delta_{IJ}}}(\alpha(O_i, C_j - C_i) \mid T_i, T_j, \delta_{ij}) \\ f_{\mathsf{D_{IJ}|T_I,T_J,\Delta_{IJ}}}(d(C_j - C_i) \mid T_i, T_j, \delta_{ij}), \tag{5}$$

where we assume that the pairwise geometrical relationships between components are not only independent but adequately modelled by the two conditional density functions $f_{\mathsf{A_{IJ}|T_I,T_J,\Delta_{IJ}}}$ and $f_{\mathsf{D_{IJ}|T_I,T_J,\Delta_{IJ}}}$, which describe the angles and distances between random pairs of components.

The model is independent of the overall orientation of the example layout, and the shape of the floor plan may also vary. By using a simplified model, we avoid the need for large amounts of example data for parameter estimation. Moreover, the simplified density can be relatively easily incorporated in the objective function of a constraint programming model of the layout problem, as described below.

4.2 Estimation

To evaluate the conditional density (5) of "well-designed" layouts, we estimate the conditional density functions $f_{\mathsf{A_{IJ}|T_I,T_J,\Delta_{IJ}}}$ and $f_{\mathsf{D_{IJ}|T_I,T_J,\Delta_{IJ}}}$ from example layout data by kernel density estimation. Specifically, we apply kernel density estimation of mixed categorical and continuous data as described by Racine and Li [20] and Ju et al. [21]. Both conditional density functions are estimated in the same way, so we describe the method for $f_{\mathsf{A_{IJ}|T_I,T_J,\Delta_{IJ}}}$ only.

We construct a sample of the random variables $(\mathsf{A_{IJ}}, \mathsf{D_{IJ}}, \mathsf{T_I}, \mathsf{T_J}, \mathsf{\Delta_{IJ}})$ by observing them on all (augmented) edges of available example layouts. The sample is assumed to be a uniform random sample, following the simplifying assumption we made for the layout density model (5) that the angles $\mathsf{A_{IJ}}$ and distances $\mathsf{D_{IJ}}$ are independent on different edges of the same graph.

Let $\mathcal{L}$ be the set of observed example layouts, and let us indicate observations on layout $L \in \mathcal{L}$ by superscript L. For each edge $(i, j) \in \tilde{E}^L$ in each layout example $L \in \mathcal{L}$, we denote the corresponding multivariate sample point by $(A_{ij}^L, D_{ij}^L, T_i^L, T_j^L, \Delta_{ij}^L)$. We get the total sample size $n = \sum_{L\in\mathcal{L}} \left|\tilde{E}^L\right|$. Note that we have both continuous observations of angle A_{ij}^L and distance D_{ij}^L, and discrete observations of component type T_i^L, T_j^L and path distance Δ_{ij}^L.

Following Racine and Li [20], we use an unnormalized joint density estimate of the form

$$\hat{f}_{\mathsf{A_{IJ},D_{IJ},T_I,T_J,\Delta_{IJ}}}(a, d, t, t', \delta) = \frac{1}{n} \sum_{\substack{L\in\mathcal{L} \\ (i,j)\in\tilde{E}^L}} k(\frac{A_{ij}^L - a}{h_A}) k(\frac{D_{ij}^L - d}{h_D}) l(T_i^L, t, \lambda_T) l(T_j^L, t', \lambda_{T'}) l(\Delta_{ij}^L, \delta, \lambda_\Delta), \tag{6}$$

where $k(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$ is the Gaussian function, and l is the unnormalized Aitchison-Aitken kernel for unordered discrete variables, defined as

$$l(X, x, \lambda) = \begin{cases} 1 & \text{if } X = x, \\ \lambda & \text{if } X \neq x. \end{cases} \tag{7}$$

The smoothing parameters h_A, h_D, λ_T, $\lambda_{T'}$ and λ_Δ are estimated by the least-squares cross-validation method.

The unnormalized estimate of conditional density is derived from estimates of joint densities as

$$\hat{f}_{\mathsf{A_{IJ}|T_I,T_J,\Delta_{IJ}}}(a|t, t', \delta) = \frac{\hat{f}_{\mathsf{A_{IJ},T_I,T_J,\Delta_{IJ}}}(a, t, t', \delta)}{\hat{f}_{\mathsf{T_I,T_J,\Delta_{IJ}}}(t, t', \delta)}, \tag{8}$$

where the unnormalized density in the numerator is derived by eliminating a k factor from (6):

$$\hat{f}_{\mathsf{A_{IJ},T_I,T_J,\Delta_{IJ}}}(a, t, t', \delta) = \frac{1}{n} \sum_{\substack{L\in\mathcal{L} \\ (i,j)\in\tilde{E}^L}} k(\frac{A_{ij}^L - a}{h_A}) l(T_i^L, t, \lambda_T) l(T_j^L, t', \lambda_{T'}) l(\Delta_{ij}^L, \delta, \lambda_\Delta), \tag{9}$$

and the denominator is derived analogously by eliminating both k factors from (6).

5 Similarity in layout optimization

We extend the layout optimization model of Section 3 to address layout similarity. The constraint model is applied to a specific non-random instance of the layout problem, where the process graph (V, E) and other parameters of Section 3 are known. In addition, each component $i \in V$ is assumed to have a type $t_i \in \mathcal{T}$ in the shared classification.

Rotations of components were already represented as alternative patterns. We extend the notion so that each pattern choice $p_i \in P_i$ also determines the orientation of component i, denoting the orientation vector by $(x^O(p_i), y^O(p_i)) \in \mathcal{O}$.

To be able to use integer-only constraint solvers, we use a nonstandard discrete angle measure in the range $\{-Q+1, \ldots, Q\}$, closely related to the one described by Todd [22]. With a solver supporting floating point and trigonometric functions, we could use angles measured in radians instead. In our tests we set $Q = 11$.

The following auxiliary decision variables are used:

$(x^R_{ij}, y^R_{ij}) \in \mathbb{Z}^2$	Direction vector from component i to j, relative to orientation of component i
$a^R_{ij} \in \{-Q+1, \ldots, Q\}$	Angle of direction vector (x^R_{ij}, y^R_{ij}), i.e. direction from component i to component j, relative to orientation of component i

The auxiliary variables are linked to the variables of the basic model by

$$x^R_{ij} = x^O(p_i)(x_j - x_i) + y^O(p_i)(y_j - y_i) \qquad \forall (i,j) \in \tilde{E} \tag{10}$$

$$y^R_{ij} = -y^O(p_i)(x_j - x_i) + x^O(p_i)(y_j - y_i) \qquad \forall (i,j) \in \tilde{E} \tag{11}$$

$$a^R_{ij} = \begin{cases} \frac{Q+1}{2} - \operatorname{trunc}\left(\frac{\frac{Q+1}{2} x^R_{ij}}{1+|x^R_{ij}|+|y^R_{ij}|}\right) & \text{if } y^R_{ij} \geq 0 \\ -\frac{Q-1}{2} + \operatorname{trunc}\left(\frac{\frac{Q+1}{2} x^R_{ij}}{1+|x^R_{ij}|+|y^R_{ij}|}\right) & \text{if } y^R_{ij} < 0 \end{cases} \qquad \forall (i,j) \in \tilde{E} \tag{12}$$

where $\operatorname{trunc} : \mathbb{R} \to \mathbb{Z}$ maps a real number to its integer part, i.e. rounds towards zero.

We note that the angle measurement function α in Section 4.1 can now be determined as the mapping that equations (10)–(12) define from the orientation $(x^O(p_i), y^O(p_i))$ and the offset $(x_j - x_i, y_j - y_i)$ to the angle a^R_{ij}.

In addition to the distance cost objective (1), we define two *similarity* objective functions

$$z_A = \sum_{(i,j) \in \tilde{E}} -\log \hat{f}_{\mathsf{A_{IJ}}|\mathsf{T_I},\mathsf{T_J},\mathsf{\Delta_{IJ}}}(a^R_{ij} | t_i, t_j, \delta_{ij}), \tag{13}$$

$$z_D = \sum_{(i,j) \in \tilde{E}} -\log \hat{f}_{\mathsf{D_{IJ}}|\mathsf{T_I},\mathsf{T_J},\mathsf{\Delta_{IJ}}}(d_{ij} | t_i, t_j, \delta_{ij}), \tag{14}$$

representing the log-densities of the angles and the distances in the solution, respectively. Both are minimization objectives. By the model (5), the sum $-(z_A + z_D)$ is the unnormalized log-density of the solution in the distribution of "well-designed" layouts, conditioned on parameters of the known problem instance.

In a discrete constraint model, the objectives z_A and z_D are naturally implemented by tabulating the values of the log terms for feasible values of a^R_{ij} and d_{ij} on each edge $(i,j) \in \tilde{E}$. Storage requirements are quite low as we only need two one-dimensional arrays for each edge.

6 Experiments

We have implemented a prototype of the proposed similarity model and facility layout model in order to test the viability of the proposed decision support approach. We implemented the similarity model estimation in Python, using multivariate conditional kernel density estimation from the Python statsmodels library [23]. We implemented the facility layout model in the MiniZinc constraint modelling language [24]. We use the local search based constraint solver Yuck [25].

We report here initial tests on artificial test data. There are components of three types: large cylinder, medium-size rectangle and small square. In addition, small round nodes are used to guide the example layout, but not included in the process graph for the similarity model or layout case. We use only four different process graphs, shown in Figure 2, and vary the layouts of the graphs.

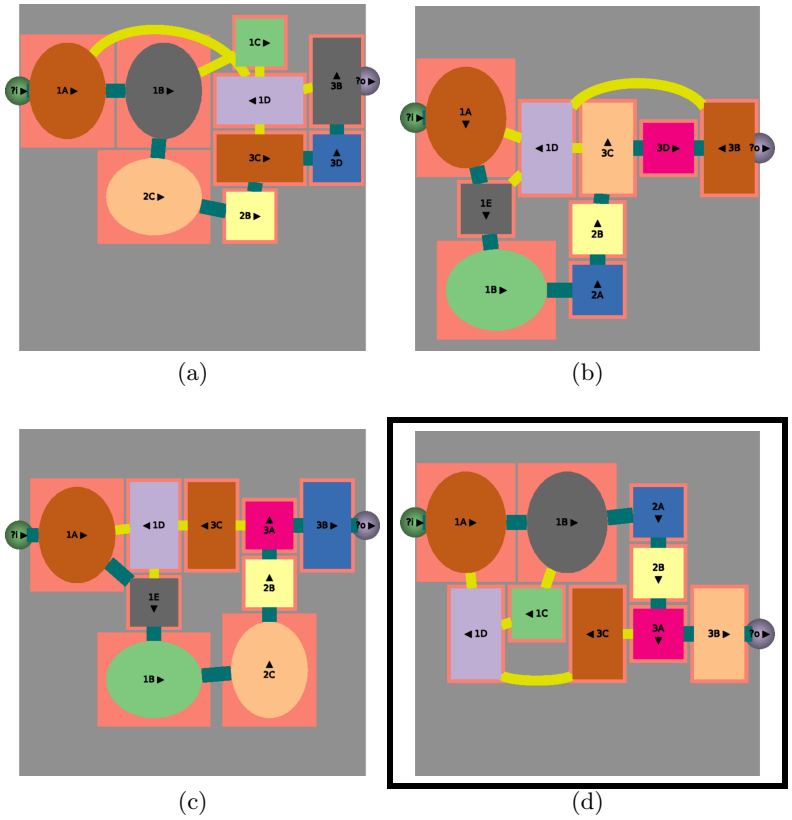


Fig. 2. Artificial process graphs used in the tests. Only the graph structure is of importance, the specific layouts are not. The bounding boxes of the components are shown in pink. Black triangles point in the direction of component orientation. Edge colours indicate which edge we apply different edge weights on. Some edges are curved solely for clarity of visualization.

The goal of all our experiments is to optimize the layout of graph (d) of Figure 2, called the *layout case*. The similarity model is built from *example layouts*, in which the graph structure may differ from the layout case. We report results for two experiments:

U. Learning similarity measure from *uniform* example data: All layout examples as well as the layout case have the same graph structure (d), and only the layout varies. There are nine example layouts of graph (d).
V. Learning similarity measure from *varied* example data: The layout examples use three different graphs (a)–(c), all of which differ from the structure (d) of the layout case. There are three example layouts of each of the graphs (a)–(c), for a total of nine examples.

We generated the example layouts by running the basic layout optimization model of Section 3. We used unit cost on the yellow edges and 0 cost on the green edges: in other words, our example layouts have short yellow edges, and the length of green edges is ignored. We collect all solutions produced by a local search solver, and then pick a diverse subset of solutions within 10 % of the lowest-cost solution found. For component orientations, we pick the major direction closest to the mean direction of the edges connected to each component.

As we optimize the layout case after the construction of the similarity model, we apply two objective functions: 1) the *similarity objective* $z_A + z_C$ measures similarity to the example layouts

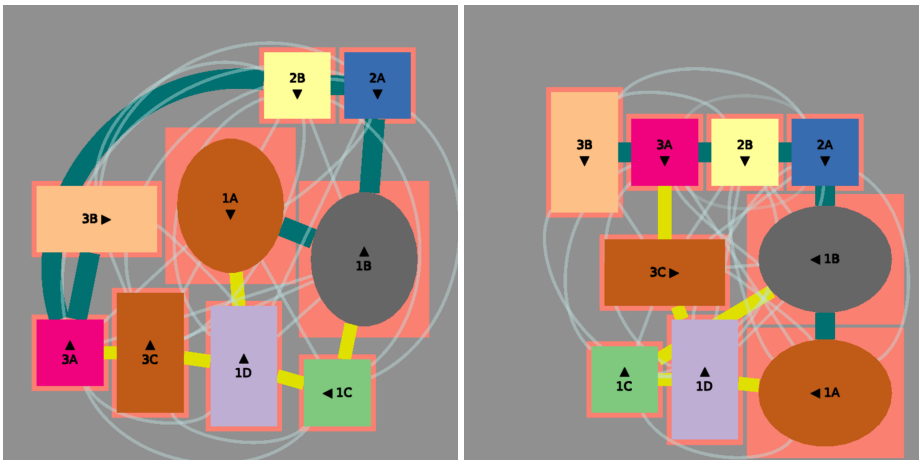
and 2) the *case objective* is the distance cost objective z_C with unit cost on the green edges and 0 cost on the yellow edges. In other words, here the similarity objective implicitly minimizes the length of yellow edges and the case objective explicitly minimizes the length of green edges.

We use a different explicit objective in the layout case and in the examples, so that machine learning is necessary to consider the example objective in the layout case. The intention is that by applying different weights on the similarity and distance cost objectives, we can trade off similarity to the examples and optimality with respect to the case objective.

It is important to note that the edge colour is not a factor in the similarity model: the effect of the different distance cost objectives in the examples and the layout case must be learned indirectly via the types of the edge endpoints.

The optimization of the layout case was run with a range of weights on the similarity objective z_A+z_D and the case objective z_C, i.e. green edge length. The relative proportion of similarity weight to distance weight ranged from 1/64 to 64/1, with an additional run optimizing pure similarity with 0 weight on the case objective. From each optimization run with specific objective weights, we collected three solutions within 10 % of the best weighted objective value.

Two specific case layouts for experiment V are shown in Figure 3: these are the solutions with (a) the best similarity value, found by pure similarity optimization, and (b) the best distance cost value, which in this experiment was found with weights 2/1 on the objectives.



(a) Weighting similarity (b) Weighting green edge length

Fig. 3. Case layouts after learning, minimizing (a) similarity to example layouts with short yellow edges, and (b) the case objective, i.e. green edge length. The thin translucent lines show the augmented edges of the similarity model.

An overview of the distance cost objective values reached with different similarity weights in experiments U and V is shown in Figure 4.

7 Discussion

The similarity model works reasonably well to reproduce geometrical relationships between components, even when the examples are not identical in structure to the current case: In particular, in Figure 3(a) it is clear that optimizing similarity has resulted in short yellow edges at the expense of long green edges, as desired.

In Figure 4 we see that as the similarity weight increases, the case objective (green edge length) tends to increase, and in exchange, the example objective (yellow edge length) decreases. Nevertheless, the case results (learned-U and learned-V in Figure 4) do not fully reach the level of the examples (example-U and example-V), even when optimizing only the similarity objective without regard to the case objective: This indicates that our simplified similarity model does not fully reproduce the example layouts, since the model is limited to pairwise geometrical relationships

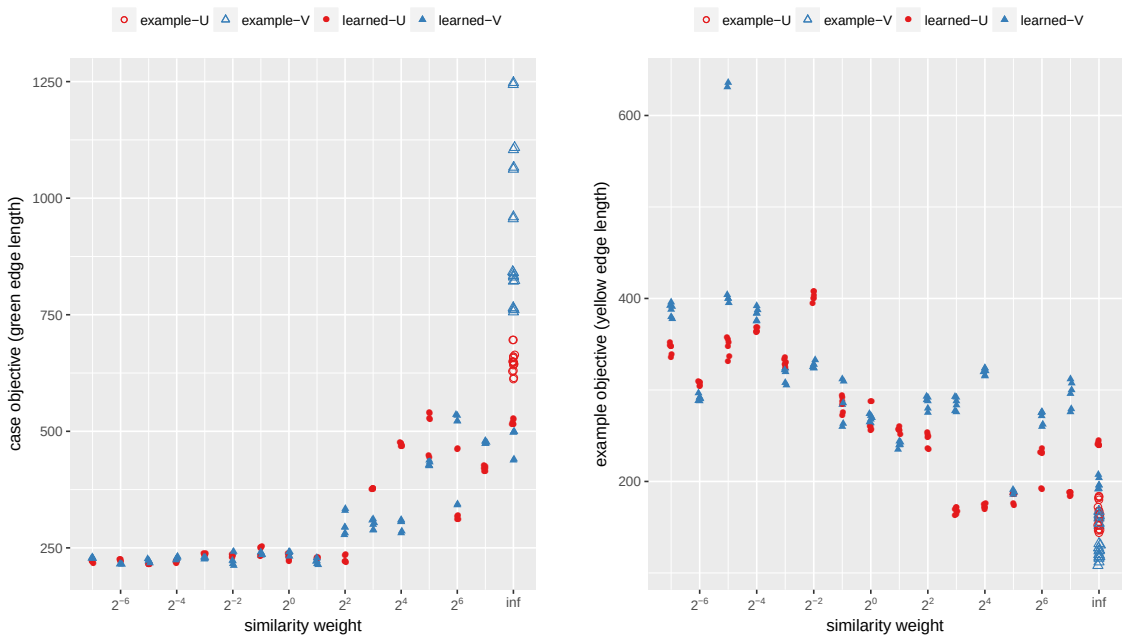


Fig. 4. Objective values of case layout results with different weights on the similarity and case objectives, showing also the objective values of the example layouts. The proportion of similarity weight to case objective weight is on the x-axis, with examples and results of pure similarity optimization at "infinity". The y-axis represents either (a) the explicit case objective, i.e. green edge length or (b) the example layout objective, i.e. yellow edge length.

between components. A more accurate model would likely require far more training data than typically available in facility layout planning.

Looking at the right half of Figure 4, we arrive at the natural conclusion that increasing the weight of the similarity objective gives more consistent reductions in the example objective, when the similarity model is derived from examples of the same process graph as in the case (experiment U), as compared to example of different process graphs (experiment V). In any case, there appears to be more variance in the results with larger weights on the similarity objective; this may be related to the fact that similarity to examples with long green edges conflicts with the minimization of green edge length in the case objective.

The presented similarity model is limited to geometrical features, and more abstract relationships that involve e.g. numerical properties of the process components are not considered: for example, the minimum safe distance from a tank to a furnace might depend both on the tank volume and the stored material.

Another limitation of our model is that it has difficulties reproducing continuous physical spaces between components, e.g. walkways for machine operators. We believe this issue can be solved by alternative similarity models with a more global view of the layout, not just pairwise geometrical relationships between components. It can also be useful to learn hard constraints in addition to the purely objective function based approach employed here.

The general approach shows promise, and we aim to evaluate it further with real-world data, and to verify its usefulness by soliciting feedback from expert facility designers. Although the presented model is limited to basic two-dimensional layouts, it would be relatively straightforward to extend the layout similarity model to three-dimensional layouts on multiple interconnected floors.

Acknowledgement

This work was supported by Tekes -– the Finnish Funding Agency for Innovation, through project Engineering Rulez.

References

1. Armour, G.C., Buffa, E.S.: A heuristic algorithm and simulation approach to relative location of facilities. Management Science **9** (1963) 294–309
2. Anjos, M.F., Vieira, M.V.C.: Mathematical optimization approaches for facility layout problems: The state-of-the-art and future research directions. European Journal of Operational Research **261** (2017) 1–16
3. Hosseini-Nasab, H., Freidouni, S., Fatemi Ghomi, S.M.T., Fakhrzad, M.B.: Classification of facility layout problems: a review study. International Journal of Advanced Manufacturing Technology **94** (2018) 957–977
4. Jung, S.: Facility siting and plant layout optimization for chemical process safety. Korean Journal of Chemical Engineering **33** (2016) 1–7
5. Moran, S.: An Applied Guide to Process and Plant Design. Elsevier (2015) ISBN 978-0-12-800242-1.
6. Cambron, K.E., Evans, G.W.: Layout design using the analytic hierarchy process. Computers & Industrial Engineering **20** (1991) 211–229
7. García-Hernández, L., Pierreval, H., Salas-Morera, L., Arauzo-Azofra, A.: Handling qualitative aspects in unequal area facility layout problem: An interactive genetic algorithm. Applied Soft Computing **13** (2013) 1718–1727
8. García-Hernández, L., Arauzo-Azofra, A., Salas-Morera, L., Pierreval, H., Corchado, E.: Facility layout design using a multi-objective interactive genetic algorithm to support the DM. Expert Systems **32** (2015) 94–107
9. Grobelny, J., Michalski, R.: A novel version of simulated annealing based on linguistic patterns for solving facility layout problems. Knowledge-Based Systems **124** (2017) 55–69
10. Chung, Y.K.: A neuro-based expert system for facility layout construction. Journal of Intelligent Manufacturing **10** (1999) 359–385
11. Ahmad, A.R., Tasadduq, I.A., Imam, M.H., Shaban, K.B.: Automated discovery and utilization of tacit knowledge in facility layout planning and optimization. Journal of Software & Systems Development (2015) Article ID 369029.
12. Merrell, P., Schkufza, E., Koltun, V.: Computer-generated residential building layouts. ACM Transactions on Graphics **29** (2010) Article 181
13. Yu, B., Pan, D.Z., Matsunawa, T., Zeng, X.: Machine learning and pattern matching in physical design. The 20th Asia and South Pacific Design Automation Conference (2015) 286–293
14. O'Sullivan, B.: Automated modelling and solving in constraint programming. In Fox, M., Poole, D., eds.: Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10), AAAI Press (2010) 1493–1497
15. Bessière, C., Coletta, R., Freuder, E.C., O'Sullivan, B.: Leveraging the learning power of examples in automated constraint acquisition. In Wallace, M., ed.: CP'04: 10th International Conference on Principles and Practice of Constraint Programming. Lecture Notes in Computer Science, Springer (2004) 123–137
16. Mizoguchi, F., Ohwada, H.: Constrained relative least general generalization for inducing constraint logic programs. New Generation Computing **13** (1995) 335–368
17. Schaefer, A.J.: Inverse integer programming. Optimization Letters **3** (2009) 483–489
18. Aswani, A., Shen, Z.J., Siddiq, A.: Inverse optimization with noisy data. Operations Research **66** (2018) 870–892
19. Lombardi, M., Milano, M., Bartolini, A.: Empirical decision model learning. Artificial Intelligence **244** (2017) 343–367
20. Racine, J., Li, Q.: Nonparametric estimation of regression functions with both categorical and continuous data. Journal of Econometrics **119** (2004) 99–130
21. Ju, G., Li, R., Liang, Z.: Nonparametric estimation of multivariate CDF with categorical and continuous data. Advances in Econometrics **25** (2009) 291–318
22. Todd, J.: Encoding 2d angles without trigonometry. https://www.freesteel.co.uk/wpblog/2009/06/05/encoding-2d-angles-without-trigonometry/ (2009) Accessed 2018-08-20.
23. Seabold, S., Perktold, J.: Statsmodels: Econometric and statistical modeling with python. In: 9th Python in Science Conference. (2010)
24. Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: MiniZinc: Towards a standard CP modelling language. In Bessiere, C., ed.: Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming. Volume 4741 of Lecture Notes in Computer Science., Springer (2007) 529—-543
25. Marte, M.: Yuck. https://github.com/informarte/yuck (2018) Accessed 2018-10-04.