# Constructive Layout Synthesis via Coactive Learning

**Paolo Dragone***
University of Trento, Italy
TIM-SKIL, Trento
paolo.dragone@unitn.it

**Luca Erculiani**
University of Trento, Italy
luca.erculiani@studenti.unitn.it

**Maria Teresa Chietera**
University of Trento, Italy
maria.chietera@studenti.unitn.it

**Stefano Teso**†
University of Trento, Italy
teso@disi.unitn.it

**Andrea Passerini**
University of Trento, Italy
passerini@disi.unitn.it

## Abstract

Layout synthesis is the task of arranging entities in space subject to a set of constraints. In this paper we address the task of learning to synthesize layouts tailored to the preferences of a user. This learning task arises, for instance, in developing recommendation systems for furniture arrangements or architecture design aiding systems. We employ the coactive learning framework to devise a constructive recommendation system for 2D layout synthesis that is able to efficiently create customized layouts. Experiments show how the system performs on problems with increasing complexity.

## 1 Introduction

Layout synthesis is a design process that involves the arrangement of objects in space subject to a set of constraints. Layout synthesis tasks include furnishing a room [10, 3, 9] or designing urban spaces [1], to mention a few. As other complex design processes, layout synthesis can be cast as a decision task driven by the preferences of a designer or a customer. In furniture arrangement, an apartment may be furnished in a minimalist or industrial style depending on the owner. Most of the previous works are, however, based on optimization [10] or off-line supervised techniques [3, 9], not taking into account the preferences of the target user. Few other systems allow for some limited interaction [1] but do not attempt to generalize from the collected examples. In this paper, we propose a learning method to cast the layout synthesis problem as recommendation task. Differently from classical recommendation tasks, in which a fixed set of candidates is assumed, layouts have to be assembled from their individual components. Therefore finding the best candidate layout for a user may involve the search over an exponentially large space of candidate layouts. The problem can thus be considered as a *constructive recommendation* task. Building up on previous work [8, 7, 6, 2], we propose a constructive method to interactively learn the user preferences and recommend personalized layouts. Our method is based on Coactive Learning [5], an online learning method that allows to learn the user preferences about structured instances (in our case layouts) through manipulative feedback.

In this work we consider the task of arranging 2D objects in a canvas. In this case, layouts can be represented as ensembles of components whose properties (e.g. position, size) are described by constraints. The underlying optimization problem can thus be readily formulated as a constraint satisfaction problem. We demonstrate our method on the task of arranging furniture in a room. We present experimental results to show the quality of the learning algorithm and its computational performance in settings with increasing complexity.

*PD is a fellow of TIM-SKIL Trento and is supported by a TIM scholarship.
†ST is supported by the Caritro Foundation through project 2014.0372.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

**Algorithm 1** Coactive Learning[3] with perceptron update [5]. Interaction with the user happens within the QUERYIMPROVEMENT procedure.

1: **procedure** COACTIVELEARNING($\mathcal{X}, \boldsymbol{\phi}, T$)
2: Initialize $\boldsymbol{w}^1$
3: **for** $t = 1, \ldots, T$ **do**
4: $x^t \leftarrow \text{argmax}_{x \in \mathcal{X}} \langle \boldsymbol{w}^t, \boldsymbol{\phi}(x) \rangle$
5: $\bar{x}^t \leftarrow$ QUERYIMPROVEMENT($x^t$)
6: $\boldsymbol{w}^{t+1} \leftarrow \boldsymbol{w}^t + \boldsymbol{\phi}(\bar{x}^t) - \boldsymbol{\phi}(x^t)$
7: **return** $\text{argmax}_{x \in \mathcal{X}} \langle \boldsymbol{w}^T, \boldsymbol{\phi}(x) \rangle$

## 2 Coactive Learning

Coactive Learning [5] is an interactive framework for learning a utility function from the user feedback. Given a set of structured entities $x \in \mathcal{X}$, henceforth called *configurations*, the preferences of the user over $\mathcal{X}$ are represented by a linear utility function $u(x) = \langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle$. Here $\boldsymbol{\phi} : \mathcal{X} \to \mathbb{R}^m$ is a feature map from the configuration space to some $m$-dimensional feature space, and $\boldsymbol{w} \in \mathbb{R}^m$ are the parameters of the model that have to be learned. As shown by Algorithm 1, at each iteration $t$, the system recommends the configuration $x^t$ with maximum utility with respect to the current estimate of the weights $\boldsymbol{w}^t$ (line 4):

$$x^t = \text{argmax}_{x \in \mathcal{X}} \langle \boldsymbol{w}^t, \boldsymbol{\phi}(x) \rangle \tag{1}$$

To learn the parameters of utility function, the algorithm requires the user to answer to *improvement* queries. After receiving a recommendation $x^t$, if not satisfied, the user has to provide a configuration $\bar{x}^t$ that is a slightly improved version of $x^t$ (line 5). The pair $(x^t, \bar{x}^t)$ is then used as an implicit ranking example to update the utility model (line 6). The true utility of the user is determined by the true weight vector $\boldsymbol{w}^*$: $u^*(x) = \langle \boldsymbol{w}^*, \boldsymbol{\phi}(x) \rangle$. In recommending $x^t$ in place of an optimal configuration $x^*$, the system suffers a regret (utility loss) of $u^*(x^*) - u^*(x^t)$. The goal of a coactive learning system is to minimize the average regret at time $T$:

$$\text{REG}_T = \frac{1}{T} \sum_{t=1}^{T} \left( u^*(x^*) - u^*(x^t) \right) \tag{2}$$

Given reasonable assumptions on the informativeness of the user, Algorithm 1 yields an average regret that is upper bounded at iteration $T$ by $O(1/\sqrt{T})$, see [5] for an in-depth explanation.

## 3 Constructive layout synthesis

Coactive Learning has been employed in recommendation settings where the task was selecting one or more instances from a fixed set of candidates, such as movies [5]. Layouts, on the other hand, are complex entities composed of several pieces that have to comply the given "rules", and thus recommending them equates to assembling them from scratch on the basis of the user preferences. Moreover, previous applications of coactive learning assumed an implicit user feedback, such as clicks on links from a ranked list of urls. In this work, instead, we focus on an interactive design task in which the user is actively involved in designing the improved configurations. In other words, we cast layout synthesis into a *constructive* recommendation task. A layout $x \in \mathcal{X}$ is represented as set of variable attributes encoding the properties of its components, e.g. the position and size of each piece of furniture. A layout is considered feasible only when it satisfies the constraints imposed by the problem. For instance, the pieces of furniture in a room must not overlap and there should be enough space left to walk around the room. The space of feasible configurations $\mathcal{X}$ is composed by all possible assignments of the attributes subject to the problem constraints. The size of $\mathcal{X}$ may therefore be exponential in the number of attributes. To handle this very large search space, we encode the inference problem (Eq. 1) as a constraint satisfaction problem. When the constraints and the feature map are linear in the attributes of $x$, the inference problem becomes a mixed integer linear program (MILP). MILP problems are a class of well studied optimization problems that can be efficiently solved by off-the-shelf solvers. Mid sized problems can be quickly solved to optimality, whereas for larger instances only suboptimal solutions may be affordable. As we show in Section 4, the performance of our method is not significantly affected by these approximations.

[3] Differently from the original formulation, we assume for simplicity that the contextual parameters are hard-coded in the inference problem, and thus we use a "context-less" utility model.

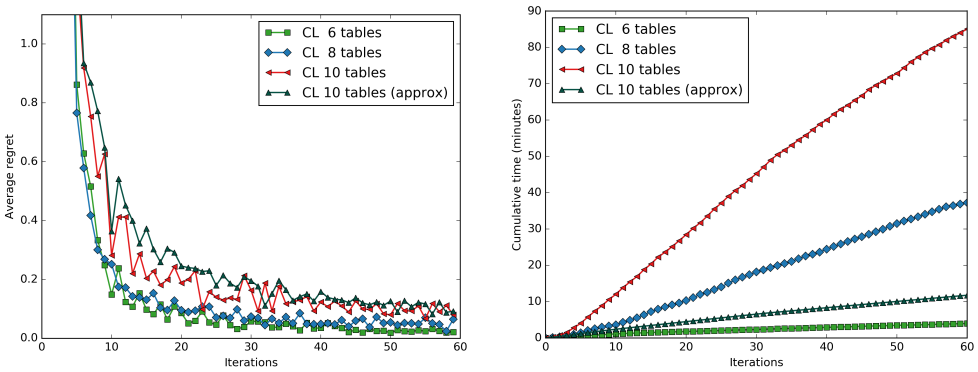


Figure 1: Average regret (left) and cumulative time (right) of the system. Best viewed in color.

## 4 Experiments

As a testbed for our layout synthesis system, we developed a furniture arrangement application[4], whose goal is learning to arrange tables in a room according to the user preferences. A room is a discrete 2D bounding box and the tables are modelled as rectangles of various shapes. The tables are identified with their coordinates in the room and the lengths of their edges. All the tables have to fit in the room and they can not overlap. The constraints for the furniture arrangement task are similar to those of the bin-packing problem, which has been extensively studied in constraint programming and operations research [4], and is well supported by existing MILP solvers. The objective is however different: while in bin-packing the objective is to pack as much stuff as possible in the least number of bins, in our case the goal is to find a layout with maximal utility for the user. Each problem instance is represented by a room with a certain (typically non-rectangular) shape and size, doors on the walls and a certain number of tables to be placed. When placing tables, the system needs to keep a free path connecting doors. In our experiments we also restrict the tables to be of size $1 \times 1$, $2 \times 1$ or $1 \times 2$. We assume that the same room layout is used throughout the learning procedure for each recommendation task, so that the system can be seen as a sort of design support tool for architects or amatours. We plan to extend the system to be able to generalize to different layouts, by learning weights that are insensitive to the specific characteristics of the instance.

Inference (Eq. 1) and improvement problems are tackled using an external MILP solver[5]. Users are simulated by randomly sampling user-specific weight vectors $\boldsymbol{w}^*$. User improvements are generated by solving the following problem:

$$\bar{x}^t = \text{argmin}_{x \in \mathcal{X}} \; u^*(x) \qquad \text{s.t.} \quad u^*(\bar{x}^t) \geq u^*(x^t) + \alpha(u^*(x^*) - u^*(x^t))$$

The parameter $\alpha \in (0, 1]$ represents the informativity of the user improvements (see [5] for more details) and was set to $0.2$ in all our experiments. The feature map $\boldsymbol{\phi}$ is composed of $10$ features including: [i] max and min distance between tables on each of the axes; [ii] max and min distance of tables from external walls (bounding box); [iii] max and min distance of tables from internal walls; [iv] number of $1 \times 1$ tables and $2 \times 1$ or $1 \times 2$ tables.

In our first experiment we focused on a quantitative evaluation of the system performance, for an increasing complexity of the scenarios as given by the number of tables. Intuitively, the more tables to fit in the room, the more variables to handle and constraints to satisfy. The room used in this experiment has size $12 \times 12$ and has the layout of the café showed in Figure 2. We first test our method with $6$, $8$ and $10$ tables using exact optimization, i.e. leaving the solver running until it finds the global optimum. We then run our algorithm with $10$ tables using approximate inference, i.e. by imposing a $20$ seconds time cut-off to the solver and keeping the best solution found. We run the experiment on $20$ users with randomly sampled weight vectors. Figure 1 shows the average results over all users. On the left we plot the median average regret for varying number of tables, on the right the mean cumulative time needed for learning. For all settings, the regret drops from around $20$ at the first iteration (not plotted) to below $1$ in less than $10$ iterations, and then keeps decreasing at a

[4] Available at: `github.com/unitn-sml/constructive-layout-synthesis`

[5] Opturion CPX: `www.opturion.com/cpx`

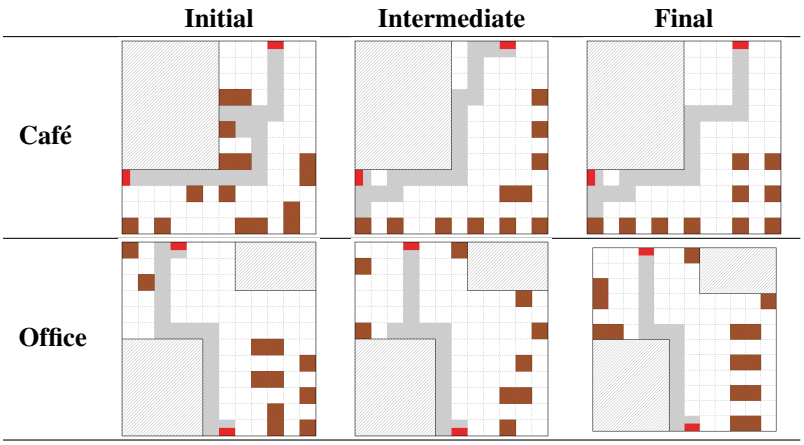


Figure 2: Two examples of configurations built by our system when used by users who are furnishing a café or an office. The figure shows different stages of algorithm while learning the user preferences. The tables are colored in brown and the doors in red. The path between the doors is colored in grey and the walls are represented with black striped sections. Best viewed in colors.

slower rate. As expected, the higher the number of tables, the higher the average regret. Despite the increased complexity, the regret decreases with the same pattern in all cases. We can also see that using approximate inference does not substantially affect performance: the average regret is slightly higher than with exact inference in the first iterations, and the two curves eventually overlap. By observing the right plot we can see that using exact inference, the cumulative time clearly rises with the number of tables. The time needed for 10 tables with exact inference is largely impractical. On the other hand, using a time cut-off to approximate inference, we can keep the running times low and predictable. This shows that our method can scale to more complex problems in predictable running time with a minor loss in performance.

In the second experiment we focused on two particular types of table arrangements, namely cafés and offices. In cafés tables are usually of size $1\times1$ and are positioned in a regular pattern along the walls, tightly packed together to leave room for the people standing. In offices instead there are mostly desks, i.e. tables of size $2\times1$ or $1\times2$, and they are spread inside the room following regular patterns e.g. in rows or columns. We generated a user interested in furnishing a café and one interested in an office, by sampling weights according to the above criteria. Figure 2 shows examples of configurations synthesized by our system while learning from the interaction with these two users. This figure shows how our system is able to learn different kinds of furniture arrangements for different users. In both cases the initial configuration is random since weights are randomly initialized. In the intermediate case, about half way to convergence, some patterns can be observed which resemble the desiderata for cafés and offices respectively. Final recommendations clearly match our above descriptions.

## 5 Conclusion

We presented a coactive learning system for automatic layout synthesis. This approach is suitable for developing constructive recommender systems, such as design aiding tools. We tested our method on a furniture arrangement scenario in which the goal is to furnish a room with tables, e.g. furnishing a café or an office. Results show that our system is able to learn different types of arrangements on the basis of the users preferences and to scale to complex scenarios using approximate inference.

As mentioned in Section 4, the context-less utility model is a simplifying assumption that we aim to remove in the near future. Another future work is the hierarchical modelling of houses, including automatic design of floors, rooms and furniture. In this setting, we could ask the user to "sketch" a local improvement on a single room and have the system learn to generalize the improvement to all rooms of the house, keeping in mind the different roles of the other rooms. Another interesting research direction is extending the system to support quadratic constraints (relying on the recent advances in quadratically constraint quadratic programming solvers), which would allow to encode measures such as Euclidean distances and areas.

## References


[1] D. G. Aliaga, C. A. Vanegas, and B. Benes. Interactive example-based urban layout synthesis. In *ACM TOG*, volume 27, page 160. ACM, 2008.

[2] P. Campigotto, R. Battiti, and A. Passerini. Learning modulo theories for preference elicitation in hybrid domains. *CoRR*, 2015.

[3] M. Fisher, D. Ritchie, M. Savva, T. Funkhouser, and P. Hanrahan. Example-based synthesis of 3d object arrangements. *ACM TOG*, 31(6):135, 2012.

[4] S. Martello, D. Pisinger, and D. Vigo. The three-dimensional bin packing problem. *Operations Research*, 48(2):256–267, 2000.

[5] P. Shivaswamy and T. Joachims. Coactive Learning. *JAIR*, 53:1–40, 2015.

[6] S. Teso, P. Dragone, and A. Passerini. Structured feedback for preference elicitation in complex domains. In *BeyondLabeler Workshop at IJCAI 2016*, 2016.

[7] S. Teso, A. Passerini, and P. Viappiani. Constructive preference elicitation by setwise max-margin learning. In *IJCAI*, pages 2067–2073, 2016.

[8] S. Teso, R. Sebastiani, and A. Passerini. Structured learning modulo theories. *Artificial Intelligence*, 2015.

[9] Y.-T. Yeh, L. Yang, M. Watson, N. D. Goodman, and P. Hanrahan. Synthesizing open worlds with constraints using locally annealed reversible jump mcmc. *ACM TOG*, 31(4):56, 2012.

[10] L. F. Yu, S. K. Yeung, C. K. Tang, D. Terzopoulos, T. F. Chan, and S. J. Osher. Make it home: automatic optimization of furniture arrangement. *ACM Transactions on Graphics (TOG)-Proceedings of ACM SIGGRAPH 2011, v. 30, no. 4, July 2011, article no. 86*, 2011.