Exploiting Similar Behavior of Users in a Cooperative Optimization Approach for Distributing Service Points in Mobility Applications*

Thomas Jatschka[1], Tobias Rodemann[2], and Günther R. Raidl[1]

[1] Institute of Logic and Computation, TU Wien, Austria
{tjatschk,raidl}@ac.tuwien.ac.at
[2] Honda Research Institute Europe, Germany
tobias.rodemann@honda-ri.de

Abstract. In this contribution we address scaling issues of our previously proposed cooperative optimization approach (COA) for distributing service points for mobility applications in a geographical area. COA is an iterative algorithm that solves the problem by combining an optimization component with user interaction on a large scale and a machine learning component that provides the objective function for the optimization. In each iteration candidate solutions are generated, suggested to the future potential users for evaluation, the machine learning component is trained on the basis of the collected feedback, and the optimization is used to find a new solution fitting the needs of the users as good as possible. While the former concept study showed promising results for small instances, the number of users that could be considered was quite limited and each user had to evaluate a relatively large number of candidate solutions. Here we deviate from this previous approach by using matrix factorization as central machine learning component in order to identify and exploit similar needs of many users. Furthermore, instead of the black-box optimization we are now able to apply mixed integer linear programming to obtain a best solution in each iteration. While being still a conceptual study, experimental simulation results clearly indicate that the approach works in the intended way and scales better to more users.

Keywords: Cooperative optimization · facility location problem · matrix factorization

1 Introduction

There exists a vast amount of literature regarding setting up service points for mobility applications such as bike sharing systems [1] or charging stations for

* Thomas Jatschka acknowledges the financial support from Honda Research Institute Europe.

electric vehicles [2]. A fundamental ingredient for optimizing the locations of service points is the distribution of existing customer demand to be potentially fulfilled in the considered geographical area. An estimation of this existing demand distribution is usually obtained upfront by performing customer surveys, considering demographic data, information on the street network and public transport, and not that seldom including human intuition and political motives. Unfortunately, this estimation is frequently imprecise and a system built on such assumptions might not perform as well as it was hoped for. Therefore, we have recently proposed the concept of a cooperative optimization algorithm (COA) [3, 4], which, instead of estimating customer demand upfront, directly incorporates potential users in the optimization process by iteratively suggesting them solution scenarios and asking for feedback. Based on this user feedback a machine learning (ML) model is trained, which is used as evaluation function by an optimization component. This optimization core is responsible for generating new promising solution candidates, from which scenarios to be presented to the users are again derived. A major bottleneck in this previous approach is the large ML model consisting of many smaller components—one per considered user and potential service point location— which need to be trained in each iteration, and the used black-box optimization at the core.

In this contribution, we aim to improve the scalability of COA by replacing the ML model as well as the optimization core in a way that allows to exploit similar behavior of users. We refine the user interaction of COA by assuming that each potential user has certain use cases for the system, such as going to work, to a recreational facility, or shopping. The demand of these individual cases can be satisfied by different service points to different degrees, depending on the customer's preferences about the locations of these service points. It is unlikely that two customers have the same needs in all respect, i.e., they have the very same use cases with the same demands; however, given a sufficiently large number of users, it is safe to assume that some customers share some use cases and then have similar opinions on the suitability of service point locations w.r.t. such a use case. Our goal is to exploit these similarities using collaborative filtering techniques, in particular matrix factorization [5], to predict a customer's preferences of service point locations.

Concerning the optimization core in COA, we investigated in [4] a variable neighborhood search and a population-based iterated greedy algorithm, but both act as black-box methods, which do not exploit any structural features except of the ML model used to evaluate candidate solutions. Their scalability to larger instances therefore also is rather limited. Using now the matrix factorization based ML model allows to formulate the optimization problem as mixed integer linear program, which we are able to solve sufficiently fast to proven optimality.

This article is structured as follows. In Section 2 related work is discussed, while Section 3 formalizes the considered service point location problem. Section 4 presents our new approach. In Section 5 we experimentally evaluate the new COA variant based on a user simulation and discuss obtained results. Section 5.2 concludes this work with an outlook on future work.

2 Related Work

The Service Point Distribution Problem (SPDP) we consider here can generally be classified as a variant of the uncapacitated Facility Location Problem (FLP) [6]. For a survey on FLPs see [7]. Although the SPDP is quite generally phrased, we specifically have mobility applications in mind, especially the distribution of charging stations for electric vehicles. While there exists a vast amount of literature for setting up such systems, see e.g. [8–11], to the best of our knowledge all existing work essentially assumes customer demand to be estimated upfront. In our approach we substantially deviate from this traditional way of solving the SPDP by resorting to an interactive approach. Potential future customers are incorporated in the optimization process as an integral part by iteratively providing feedback on meaningfully constructed solution scenarios. In this way we learn user demands on-the-fly and may avoid errors due to unreliable a priori estimations. For a survey on interactive optimization algorithms see [12].

As we cannot expect a user to evaluate hundreds of solutions, a common way to unburden the users is to train a surrogate function [13] with the user feedback which is then used to evaluate intermediate solutions. In this contribution we use matrix factorization [5] as ML model to realize the surrogate function. Matrix factorization is a collaborative filtering technique which is frequently used in recommender systems [14]. The idea of collaborative filtering is to make recommendations for users based on the preferences of similar users, which means in our context to estimate some user demand for a use case by the feedback already provided by other users for similar use cases.

Matrix factorization is based on singular value decomposition which decomposes a matrix into two smaller matrices. Unknown values can then be estimated my multiplying the corresponding rows and columns of the decomposed matrices [14]. The two most popular techniques for decomposing a matrix with missing values are stochastic gradient descent (SGD) [15] and alternating least squares (ALS) [16]. ALS is usually only preferred over SGD for parallelization [5].

3 The Service Point Distribution Problem

The SPDP was originally defined in [3] as follows. We are given a set of locations $V = \{1, \dots, n\}$ at which service points may be built and a set of potential users $U = \{1 \dots, m\}$. The fixed costs for setting up a service point at location $v \in V$ are $z_v^{\text{fix}} \geq 0$, and this service point's maintenance over a defined time period is supposed to induce variable costs $z_v^{\text{var}} \geq 0$. The total construction costs must not exceed a maximum budget $B > 0$. Erected service stations may satisfy an arbitrary amount of customer demand, and for each unit of satisfied customer demand a prize $p > 0$ is earned.

A solution to the SPDP is a binary incidence vector $x = (x_v)_{v \in V}$, where $x_v = 1$ indicates that a service point is to be set up at location v. A solution x is feasible if its total fixed costs do not exceed the maximum budget B, i.e.,

$$z^{\text{fix}}(x) = \sum_{v \in V} z_v^{\text{fix}} x_v \leq B. \tag{1}$$

The objective function $f(x)$ of the problem is not explicitly given but only implicitly by allowing solutions to be evaluated by the users. In the original problem definition a user provides as feedback the estimated amount of demand (e.g., per week) that would be satisfied for him at each service point included in the solution x.

We now refine this user feedback by asking users already initially to specify use cases by a name and the demand each of them induces. Hence, we are also given for each user $u \in U$ the set of use cases E_u and the demand $D_{u,e}$ for each use case $e \in u$. Note, however, that we do not know which users share which use cases, their names have no meaning to us. The number of service points required to satisfy a use case e in general depends on the underlying application scenario. In our experiments in Section 5, we only consider scenarios where a use case requires one suitable service point to be satisfied, such as setting up charging stations for electric vehicles. Our approach, however, is in principle more general. For example when setting up rental stations for a bike sharing system, a use case will typically require two suitable service stations, one close to the origin and one close to the destination of a trip.

The objective is to find a feasible solution that maximizes the expected prizes earned for satisfied customer demands reduced by the variable costs for maintaining the service points, which is in our case

$$f(x) = q \cdot \sum_{u \in U} \sum_{e \in E_u} D_{u,e} \cdot \max_{v \in V} w(u,e,v)\, x_v - \sum_{v \in V} z_v^{\text{var}}\, x_v, \tag{2}$$

where function $w(u,e,v) \in [0,1]$ denotes the suitability of a service point at location v to satisfy the needs of user u concerning his use case e. This objective function assumes that a user chooses for a use case always a location that is most suitable. The objective function $f(x)$ further interprets the determined suitability value for each use case as probability of the actual usage of the system to satisfy the demand $D_{u,e}$.

Note that $w(u,e,v)$ is not known upfront, but respective values can only be partially obtained from the users by providing them sample scenarios for evaluation. The evaluation of scenarios is discussed in more detail in Section 4.2.

As we are in general only able to obtain a small portion of all relevant values for $w(u,e,v)$ from the users, we exploit user behavior similarities and replace $w(u,e,v)$ by an approximation $\tilde{w}(u,e,v)$, yielding the surrogate objective function $\tilde{f}(x)$. This approximation will be realized by a ML model.

4 Cooperative Optimization Algorithm

The basic procedure of our COA remains almost the same as presented in [3], i.e., the framework consists of an evaluation component (EC) (containing the ML model), an optimization component (OC), a feedback component (FC), and a solution management component (SMC). Figure 1 illustrates the communication between the components, and Algorithm 1 shows the main procedure in pseudo-code. We now use, however, different algorithms in these components as explained in the following.

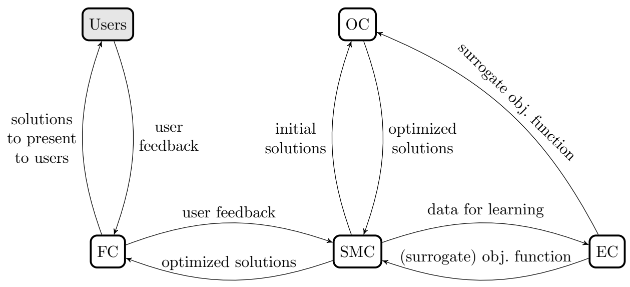

Fig. 1: Components of the COA framework and their interaction.

Algorithm 1: Basic Framework

```
    Input : an instance of the SPDP
    Output: a solution $x = (x_v)_{v \in V} \in \{0,1\}^n$
1:  while no termination criterion satisfied do
2:      Feedback Component:
3:          for $u \in U$ do
4:              for $e \in E_u$ do
5:                  determine set of scenarios $S_{u,e}$ to be evaluated by user $u$;
6:                  let user $u$ evaluate $S_{u,e}$;
7:                  update SMC with ratings obtained from $S_{u,e}$;
8:              end for
9:          end for
10:     Evaluation Component:
11:         train ML model with ratings in $R$, yielding surrogate obj. func. $\tilde{f}(x)$;
12:         re-evaluate all solutions stored in the SMC with new $\tilde{f}(x)$;
13:     Optimization Component:
14:         $x^{\mathrm{OC}} \leftarrow$ generate optimal solution w.r.t. the EC's $\tilde{f}(x)$;
15:         update SMC with $x^{\mathrm{OC}}$;
16: end while
17: return overall best found solution $\tilde{x}^*$;
```

4.1 Solution Management Component

The SMC stores and manages so far considered solutions and evaluations by the users. This includes in particular the set of tuples $R = \{(u,e,v) \mid w(u,e,v) \text{ is known from user feedback}, u \in U,\ e \in E_u,\ v \in V\}$ with the respective ratings $w(u,e,v)$. Moreover, the SMC also maintains the set X of all solutions obtained from the OC over all major iterations with their current surrogate objective values and, if available, their exact objective values. The current best solution is the solution in X with the highest surrogate objective value, denoted by $\tilde{x}^*$. With $V(u,e)$ the SMC also keeps track of the set of all locations $v \in V$ for which $(u,e,v) \in R$, with $u \in U,\ e \in E_u$. Last but not least, through the FC we are also able to obtain upper bounds on ratings $w(u,e,v)$, with $v \in V,\ u \in U,\ e \in E_u$, as explained in the next section. These upper bounds are stored in the SMC as $w^{\mathrm{UB}}(u,e,v)$.

4.2 Feedback Component

The FC generates location scenarios for users to evaluate. Similar to solutions these scenarios are binary incidence vectors $s = (s_1, \ldots, s_n) \in \{0,1\}^n$, however they are not restricted by the budget constraint (1) and can therefore contain an arbitrary number of service points. In each COA iteration we present a set of scenarios to each user $u \in U$ for each of his use cases E_u for evaluation. If a user u selects in a scenario s for a use case e a suitable service point location v, he grades it with a rating $w(u,e,v) \in (0,1]$. If a user u decides that for a use case e there is no suitable service point location in scenario s, he indicates this by selecting no service point location, and we then know that $w(u,e,v) = 0$ for all $v \in V(s)$. Note that the user is required to select a best suited service point in the scenario if not all service points are unsuitable.

The obtained ratings are used in the EC for training the surrogate function. Moreover, the obtained ratings also serve as upper bounds for unknown ratings. As each user selects the best suited service point v in the presented scenario x w.r.t. a use case e, it must hold that $w(u,e,v) \geq w(u,e,k)\ \forall k \in x$. Hence, $w(u,e,v)$ serves as upper bound $w^{\mathrm{UB}}(u,e,k)$ of $w(u,e,k)$. Moreover, $w^{\mathrm{UB}}(u,e,k)$ is updated in the SMC whenever a lower upper bound is obtained.

We use two approaches to generate scenarios that are presented to a user $u \in U$ w.r.t. a use case $e \in E_u$. First, a scenario $s^{\mathrm{V}} = \{v \in V \mid w(u,e,v) \notin R\}$ containing all locations that have not been rated yet w.r.t. u and e is presented to the user. Then, the user is also asked to evaluate the scenario $s^* = \{v \in \tilde{x}^* \mid w(u,e,v) \notin R\}$ containing all locations from the current best solutions that have not been rated yet w.r.t. u and e.

A main goal is to keep the number of presented scenarios per use case as low as possible. For this purpose, we exploit that users may show similar preferences for single use cases, hence, not every user needs to evaluate every location for a use case. Therefore, the scenario s^{V} is presented to u with a probability of 90% and s^* is shown to u with a probability of 20%.

4.3 Evaluation Component

The EC provides the means for evaluating solutions, in particular also within the OC. The real objective function $f(x)$, cf. (2), which contains many unknown user ratings, is approximated by the surrogate objective function $\tilde{f}(x)$ that is defined in accordance to $f(x)$ but makes use of *estimated ratings*

$$\tilde{w}(u,e,v) = \begin{cases} w(u,e,v) & \text{if } (u,e,v) \in R \\ \min\{w^{\mathrm{UB}}(u,e,v),\ \tilde{g}(u,e,v)\} & \text{else,} \end{cases} \tag{3}$$

where $\tilde{g}(u,e,v)$ is an approximate rating of location v for user u w.r.t. use case e.

We use matrix factorization [5] in order to predict unknown ratings. Given matrix $W = (W_{(u,e),v})_{u\in U, e\in E_u, v\in V}$ with $W_{(u,e),v} = w(u,e,v)$ for $(u,e,v) \in R$ and the other values unknown, matrix factorization identifies for each row $(u,e),\ u \in U,\ e \in U_e$ a vector $\xi_{u,e} \in \mathbb{R}^\phi$ and for each column $v \in V$ a vector $\nu_v \in \mathbb{R}^\phi$, respectively, with a space of features $F = \{1,\dots,\phi\}$. The number of features ϕ is hereby a parameter that is chosen, e.g., in dependence of an estimation of the overall number of different use cases. An unknown value in W is approximated via the dot product $W_{(u,e),v} = {\xi_{u,e}}^\intercal \nu_v$, and $\tilde{g}(u,e,v) = W_{(u,e),v}$. The vectors $\xi_{u,e}$ and ν_v are learned by minimizing the loss function

$$\min \sum_{(u,e,v)\in R} \left(W_{(u,e),v} - (\mu + b_{u,e} + b_v + {\xi_{u,e}}^\intercal \nu_v)\right)^2 + \lambda(\|\xi_{u,e}\|^2 + \|\nu_v\|^2 + b_{u,e}^2 + b_v^2), \quad (4)$$

where λ is a regularization parameter which is set to 0.001 in our experiments, $b_{u,e} \in \mathbb{R}$ and $b_v \in \mathbb{R}$ are biases for users and locations, respectively, and μ is the average over all known values in R. For this minimization, stochastic gradient descent is used. In the first iteration of COA, the weights of the model are initialized randomly, while in later iterations, the model is re-trained starting with the values from the previous iteration.

4.4 Optimization Component

The OC solves the following mixed integer programming (MIP) formulation to determine an optimal solution w.r.t. the current surrogate objective function with ratings $\tilde{w}(u,e,v)$ provided by the EC. We use a binary variable x_v to indicate whether or not a location $v \in V$ is in the solution. Continuous variable $y_{u,e} \in [0,1]$ represents the expected degree to which a use case $e \in E_u$ is satisfied for user $u \in U$. Binary variable $z_{u,e,v} \in \{0,1\}$ indicates whether or not a user u would use a service point at location v to satisfy the demand of a use case $e \in E_u$.

$$\max\ q \cdot \sum_{u\in U} \sum_{e\in E_u} D_{u,e}\, y_{u,e} - \sum_{v\in V} z_v^{\text{var}}\, x_v \quad (5)$$

$$y_{u,e} \le \sum_{v\in V} \tilde{w}(u,e,v) \cdot z_{u,e,v} \qquad \forall u \in U,\ e \in E_u \quad (6)$$

$$z_{u,e,v} \le x_v \qquad \forall u \in U,\ v \in V,\ e \in E_u \quad (7)$$

$$\sum_{v\in V} z_{u,e,v} \le 1 \qquad \forall u \in U,\ e \in E_u \quad (8)$$

$$\sum_{v\in V} c_{v,1}^{\text{fix}}\, x_v \le B \quad (9)$$

$$x_v \in \{0,1\} \qquad \forall v \in V \quad (10)$$

$$y_{u,e} \in [0,1] \qquad \forall u \in U,\ e \in E_u \quad (11)$$

$$z_{u,e,v} \in \{0,1\} \qquad \forall u \in U,\ e \in E_u,\ v \in V \quad (12)$$

Inequalities (6) determine the expected degrees of satisfying the use cases in dependence of the user ratings and the location selection variables. Inequalities (7) express that a location can only satisfy demand if it contains a service point. According to (8), the demand of a service point for a user can only be satisfied at a single location. Finally, inequality (9) ensures that a solution does not exceed the budget.

5 Experimental Evaluation

As this contribution is only a conceptual study, we do not test with real users but simulate the user interaction in an idealized manner in certain benchmark scenarios. For this purpose we adopt the user simulation from [3] and extend it to our new needs.

5.1 Benchmark Scenarios

The n possible locations for service stations are randomly distributed in the Euclidean plane with coordinates $\mathrm{coord}(v),\ v \in V$ chosen uniformly from the grid $\{0, \dots, L-1\}^2$, with $L = \lceil 10\sqrt{n}\rceil$. The fixed costs c_v as well as the variable costs z_v for setting up a service station at each location $v \in V$ are uniformly chosen at random from $\{50, \dots, 100\}$. The budget is assumed to be $B = \lceil 7.5 \cdot n\rceil$ so that about 10% of the stations with average costs can be set up.

The number of use cases for each user $u \in U$ is chosen randomly according to a shifted Poisson distribution with offset one and expected value three. Each of these use cases $e \in E_u$ is associated with an individual demand $D_{u,e}$ chosen at random from $\{5,\dots,50\}$ and a particular geographic location $r_{u,e} \in \{0, \dots, L-1\}^2$. In order to model similarities in the users' use cases, these locations are generated in the following dependent way. We first select α *attraction points* A with uniform random coordinates from $\{0, \dots, L-1\}^2$. Then, each use case location is derived by choosing one of these attraction points $(a_x, a_y) \in A$ and adding an individual deviation, i.e.,

$$r_{u,e} = (\lfloor \mathcal{N}(a_x, \sigma_{\mathrm{v}})\rfloor, \lfloor \mathcal{N}(a_y, \sigma_{\mathrm{v}})\rfloor), \tag{13}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes a random value sampled from a normal distribution with the respectively given mean value and standard deviation. Note that coordinates beyond the grid are re-sampled.

A service point location $v \in V$ is generally considered suitable for the use case e if its Euclidean distance to the use case location does not exceed 15.

In this case v receives a positive rating that decreases exponentially with the distance but is also perturbed by a Gaussian noise:

$$w(u, e, v) = \mathcal{N}(e^{-||r_{u,e} - \mathrm{coord}(v)||/10}, \sigma_{\mathrm{r}}). \tag{14}$$

If $w(u, e, v) \notin (0, 1]$, the random sampling is repeated in order to obtain a valid rating.

In our experiments we consider benchmark scenarios with $n = 100$ locations and $m \in \{500, 1000, 1500\}$ users. For each combination we derive three groups of 30 independent instances with different parameters $\alpha \in \{10, 17, 25\}$, $\sigma_{\mathrm{v}} \in \{5, 7, 10\}$, and $\sigma_{\mathrm{r}} \in \{0.03, 0.1, 0.15\}$. All benchmark instances are available at https://www.ac.tuwien.ac.at/research/problem-instances#spdp.

5.2 Computational Results

The whole approach was implemented in Python 3.7. The matrix factorization has been realized with Keras 2.2.4 and TensorFlow 1.13.1 without GPU support. The number of features ϕ of the matrix factorization was set in accordance to the number of attraction points α of the test instances. At each iteration, the model was trained with the SGD optimizer to minimize loss function (4). Each training was done over 300 epochs with a batch size of 32, or until the loss function did not improve within 10 epochs. We use 20% of the training data as validation data with which the loss of the model is calculated.

The MIP is solved with Gurobi 8.1.0. All test runs have been executed on an Intel Xeon E5-2640 v4 2.40GHz machine in single-threaded mode. COA was terminated after five major iterations or when a CPU-time limit of 7200s has been reached and returned as the overall best solution $\tilde{x}^*$, i.e., the solution with the highest surrogate objective value at the end.

We compare our results to optimal solutions obtained by solving the MIP in the OC with exact values $w(u, e, v)$ provided by the user simulation, and with our previous COA variant from [3], here denoted as COA_0. In order make the comparison to COA_0 as fair as possible, the same termination criteria were applied, but otherwise all parameters of COA_0 were set as described in [3].

Table 1 shows the obtained results. Each line lists, for COA as well as COA_0, the average number of iterations $\overline{n_{\text{it}}}$, the average optimality gap $\overline{\%\text{-gap}}$ between the objective value of $\tilde{x}^*$ and the optimal solution, the average percentage error of the surrogate function values of the final solutions $\overline{\%\text{-}\Delta\tilde{f}}$, with $\%\text{-}\Delta\tilde{f} = 100\% \cdot |\tilde{f}(\tilde{x}^*) - f(\tilde{x}^*)|/f(\tilde{x}^*)$, the average ratio of locations the users had to rate during the course of the algorithm per use case and their relevant locations per use case $\overline{\rho}$, and the median computation times in seconds $t[s]$.

The results clearly show that COA is able to converge to very reasonable solutions with small remaining optimality gaps of typically less than 2.3% within only five major iterations. For $\%\text{-}\Delta\tilde{f}$, we can observe that the percentage errors decrease as the number of users increases. This is especially evident for the hardest instance groups C, F, and I where $\%\text{-}\Delta\tilde{f}$ decreases from 8.17% to 4.42% on average. This documents that, given a sufficient amount of users, the surrogate function is able to approximate the real objective function at the end well in the relevant parts w.r.t. the returned solution. The table also shows that not all runs have been completed with five iterations, i.e., COA was aborted due to the time limit for 9 instances from the instance groups H and I. Column $\overline{\rho}$ of COA also shows that in general users do not need to rate more locations than their total number of relevant locations for each of their use cases.

COA_0 is significantly outperformed by COA in all aspects. COA is able to generate better solutions in less time for all instance groups. In many cases COA_0 exceeded the time limit of 7200s already in the first or second iteration which explains the large difference in performance between COA and COA_0. It is not quite easy to compare $\overline{\rho}$ between COA and COA_0 since COA_0 was not able to perform as many iterations as COA. However, in general we can observe

Table 1: Average results of COA and COA$_0$.

						COA					COA$_0$				
Inst.	m	α	σ_v	σ_r	ϕ	$\overline{n_\mathrm{it}}$	$\overline{\%\text{-gap}}$	$\overline{\%\text{-}\Delta\tilde{f}}$	$\overline{\rho}$	t[s]	$\overline{n_\mathrm{it}}$	$\overline{\%\text{-gap}}$	$\overline{\%\text{-}\Delta\tilde{f}}$	$\overline{\rho}$	t[s]
A	500	10	5	0.03	10	5.00	**0.35**	**2.28**	0.86	**751**	1.97	16.40	28.07	0.82	7172
B	500	17	7	0.10	17	5.00	**1.18**	**5.19**	0.88	**888**	2.43	18.37	21.44	1.24	7168
C	500	25	10	0.15	25	5.00	**2.23**	**8.17**	0.84	**1033**	2.07	14.61	26.54	0.89	7190
D	1000	10	5	0.03	10	5.00	**0.39**	**1.94**	0.84	**1540**	2.90	16.93	22.63	1.53	7180
E	1000	17	7	0.10	17	5.00	**1.61**	**4.73**	0.83	**2407**	2.30	13.34	21.91	1.07	7181
F	1000	25	10	0.15	25	5.00	**1.52**	**5.72**	0.86	**3383**	2.53	16.98	20.86	1.32	7191
G	1500	10	5	0.03	10	5.00	**0.26**	**1.73**	0.85	**2579**	2.83	14.78	14.81	1.50	7189
H	1500	17	7	0.10	17	4.90	**1.18**	**3.81**	0.82	**4478**	1.77	17.78	28.88	0.65	7179
I	1500	25	10	0.15	25	4.73	**1.63**	**4.42**	0.80	**5605**	1.97	18.08	26.13	0.83	7189

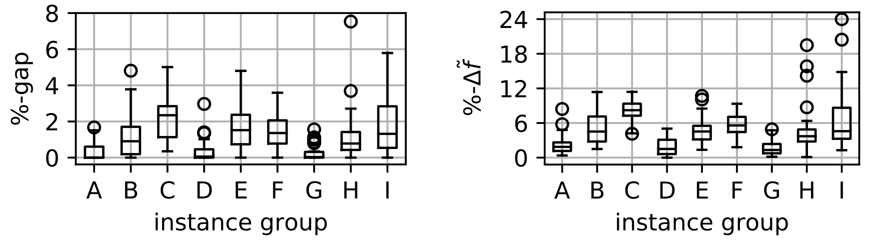

Fig. 2: Distributions of the optimality gaps and surrogate percentage errors of the best found solutions.

that users are required to evaluate significantly more locations with COA$_0$ than with COA.

In Figure 2 we take a closer look at the distributions of the optimality gaps of the obtained solutions and how well our surrogate function is able to learn the behavior of the users. Considering a fixed number of users, the obtained optimality gaps deteriorate as the complexity of the instances (i.e., α, σ_v, σ_r) increases. Interestingly, increasing the number of users does not have a substantial impact on the optimality gaps when the complexity parameters stay the same. For $\overline{\%\text{-}\Delta\tilde{f}}$, however, we can observe that the medians of the percentage errors slightly improve as the number of users increases. The large outliers of the instances groups H and I are from runs that have been aborted due to the time limit.

Generally, Figure 2 indicates that the new approach scales now much better to larger numbers of users, and instead of the users, the number of actually different use cases is now what matters primarily. Thus, the similarity among users is indeed effectively exploited.

In Figure 3 we analyze the computation times of the individual components of COA. Note that we omitted the computation times of the FC in Figure 3 as they are negligible in comparison to the computation times of the EC and the

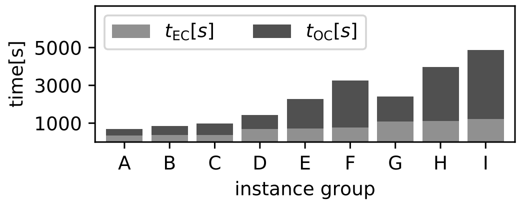

Fig. 3: Computation times of COA grouped by its framework components.

OC. We see that the number of users has the strongest impact on the overall times. However, with an increasing complexity of the test instances, the OC quickly becomes the main bottleneck of our COA, as it generally requires more computation time than the other two components together.

While for COA_0 the EC was a major bottleneck, it now scales very well with an increasing number of users w.r.t. our benchmark instances. Hence, matrix factorization turns out to be an excellent choice as underlying model of our surrogate function.

6 Conclusion and Future Work

In this contribution we have made major progress in improving the scalability of our previously presented COA [3] by using a matrix factorization model as our new surrogate function in the EC. Due to this change we were also able to abandon our previous black box optimization model of the OC and use a MIP instead. The new surrogate function as well as the new optimization core resulted in a major speedup and improvement in the scalability of our COA. Moreover, our new approach also requires a significantly lower number of user interactions.

In future work we aim at improving the approach further by refining in particular the feedback component to further reduce the number of user evaluations that are necessary to obtain reliable results. Moreover, for larger instances solving the MIP becomes the major bottleneck, as we have seen. Hence, a natural step to further improve the scalability is to replace the exact MIP with a reasonable heuristic approach. The loss of the proven optimality does not seriously matter in our application as enough other uncertainties remain. Last but not least, remember that COA was designed with more general applications in mind, and one of our next steps will be to apply it to more complex scenarios like bike sharing station planning, where we have to deal with trips instead of single locations in the use cases.

References

1. Kloimüllner, C., Raidl, G.R.: Hierarchical clustering and multilevel refinement for the bike-sharing station planning problem. In: International Conference on Learn-

ing and Intelligent Optimization. LNCS, vol. 10556, pp. 150–165. Springer (2017)
2. Frade, I., Ribeiro, A., Gonçalves, G., Antunes, A.: Optimal Location of Charging Stations for Electric Vehicles in a Neighborhood in Lisbon, Portugal. Transportation Research Record: Journal of the Transportation Research Board 2252, 91–98 (2011)
3. Jatschka, T., Rodemann, T., Raidl, G.R.: A cooperative optimization approach for distributing service points in mobility applications. In: Liefooghe, A., Paquete, L. (eds.) Evolutionary Computation in Combinatorial Optimization. LNCS, vol. 11452, pp. 1–16. Springer (2019)
4. Jatschka, T., Rodemann, T., Raidl, G.R.: VNS and PBIG as optimization cores in a cooperative optimization approach for distributing service points. In: Computer Aided Systems Theory – EUROCAST 2019". LNCS, Springer (to appear), https://www.ac.tuwien.ac.at/files/pub/jatschka_19a.pdf
5. Bell, R.M., Koren, Y., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(08), 30–37 (2009)
6. Cornuéjols, G., Nemhauser, G.L., Wolsey, L.A.: The uncapacitated facility location problem. In: Mirchandani, P.B., Francis, R.L. (eds.) Discrete Location Theory, pp. 119–171. Wiley, NY, USA (1990)
7. Farahani, R.Z., Hekmatfar, M.: Facility Location: Concepts, Models, Algorithms and Case Studies. Springer (2009)
8. Awasthi, A., Venkitusamy, K., Padmanaban, S., Selvamuthukumaran, R., Blaabjerg, F., Singh, A.K.: Optimal planning of electric vehicle charging station at the distribution system using hybrid optimization algorithm. Energy 133, 70–78 (2017)
9. Cavadas, J., Homem, G.d.A.C., Gouveia, J.: A MIP model for locating slow-charging stations for electric vehicles in urban areas accounting for driver tours. Transportation Research Part E: Logistics and Transportation Review 75, 188–201 (2015)
10. Chung, S.H., Kwon, C.: Multi-period planning for electric car charging station locations: A case of korean expressways. European Journal of Operational Research 242(2), 677–687 (2015)
11. Kameda, H., Mukai, N.: Optimization of charging station placement by using taxi probe data for on-demand electrical bus system. In: König, A., Dengel, A., Hinkelmann, K., Kise, K., Howlett, R.J., Jain, L.C. (eds.) Knowledge-Based and Intelligent Information and Engineering Systems. pp. 606–615. Springer (2011)
12. Meignan, D., Knust, S., Frayret, J.M., Pesant, G., Gaud, N.: A review and taxonomy of interactive optimization methods in operations research. ACM Transactions on Interactive Intelligent Systems 5(3), 17:1–17:43 (2015)
13. Koziel, S., Ciaurri, D.E., Leifsson, L.: Surrogate-based methods. In: Computational Optimization, Methods and Algorithms. Studies in Computational Intelligence, vol. 356, pp. 33–59. Springer (2011)
14. Ekstrand, M.D., Riedl, J.T., Konstan, J.A.: Collaborative filtering recommender systems. Foundations and Trends in Human–Computer Interaction 4(2), 81–173 (2011)
15. Robbins, H., Monro, S.: A stochastic approximation method. The Annals of Mathematical Statistics 22(3), 400–407 (1951)
16. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: Seventh IEEE International Conference on Data Mining. pp. 43–52 (2007)