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. We investigate a variant of the facility location problem concerning the optimal distribution of service points with incomplete information within a certain geographical area. The application scenario is generic in principle, but we have the setup of charging stations for electric vehicles or rental stations for bicycles or cars in mind. When planning such systems, estimating under which conditions which customer demand can be fulfilled is fundamental in order to evaluate and optimize possible solutions. In this paper we present a cooperative optimization approach for distributing service points that incorporates potential customers not only in the data acquisition but also during the optimization process. A surrogate objective function is used to evaluate intermediate solutions during the optimization. The quality of this surrogate objective function is iteratively improved by learning from the feedback of potential users given to candidate solutions. For the actual optimization we consider a population based iterated greedy algorithm. Experiments on artificial benchmark scenarios with idealized simulated user behavior show the learning capabilities of the surrogate objective function and the effectiveness of the optimization.

Keywords: Cooperative optimization · facility location problem · surrogate objective function · metaheuristics.

1 Introduction

Identifying optimal locations for setting up charging stations for electric vehicles (EVs), rental stations in public bike or car sharing systems, or, more generally some kind of service stations for mobility applications is always a challenging problem when planning such systems. Usually, the goal is to place stations at locations with high customer demand in order to maximize the usage and revenue of such systems. However, estimating the customer demand that can possibly be fulfilled is challenging. Demographic data is usually interlinked with geographic information, data on public transport, the street network, knowledge on manifold special locations, etc. Additionally, surveys of potential customers are performed. Customer demand information determined in such ways typically is vague, and not uncommonly a system built on such assumptions is not as effective as originally hoped for due to major deviations in reality. The actual usage

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

of a service system by a user will in general depend not only on the construction of service points on a few specific locations but more globally on non-trivial relationships of the user's necessities and preferences in conjunction with larger parts of the whole service system. For example in the case of charging stations for EVs, consider the situation that a station is not built in close proximity to a location a user is interested in, e.g., the user's place of work. Intuitively, one might say that the user's demand cannot be fulfilled. However, such a conclusion may be too naive. It might easily be the case that some other location is covered that provides a reasonable alternative for the user, e.g., by additionally using some public transport. Thus, there might be alternatives for fulfilling demand that cannot all be foreseen or exactly pre-specified by potential users.

Hence, a crucial assumption that makes the situation challenging and appears to be particularly valid in the context of the above mentioned mobility applications is the following: We are in general not able to obtain complete information from potential users about the conditions under which how much demand will be fulfilled, even when assuming absolutely rational users, neglecting uncertainty, and ignoring aspects arising from competition from many users on possibly scarce resources.

To overcome this problem, we investigate a *cooperative optimization approach*. More generally, interactive optimization approaches incorporate potential users on a large scale and more tightly into the data acquisition as well as the optimization process; for a review see [19]. We confront the potential users with carefully selected candidate solutions and ask how these suit the needs. Obtained feedback is used to incrementally gain more knowledge on how much demand may be fulfilled under which conditions. The optimization core relies on a surrogate objective function that approximates the real fulfilled demand. It is based on machine learning models that are trained by the user feedback. Having obtained a new so far best solution from the optimization core, new, more promising candidate solutions can be derived and again be presented to the users. The process is iterated on a large scale with many potential users and several rounds until a satisfactory solution is reached.

We test the approach in a proof-of-concept manner on artificial benchmark scenarios simulating user behavior in an idealized fashion. Results document the learning capabilities of the surrogate objective function and the effectiveness of the optimization, while keeping the number of solutions to which users have to give feedback reasonably low.

The paper is structured as follows. In Section 2 the Service Point Distribution Problem (SPDP) is formally introduced. Section 3 discusses related work. Section 4 introduces the cooperative optimization approach (COA) for solving the SPDP. Finally, in Section 5 we experimentally evaluate the COA and present obtained results.

2 The Service Point Distribution Problem

We now specify the problem we consider more formally. In the *Service Point Distribution Problem (SPDP)* 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 $c_v \geq 0$, and this service point's maintenance over a defined time period is supposed to induce variable costs $z_v \geq 0$. The total construction costs must not exceed a maximum budget $B > 0$. Erected service stations

may satisfy customer demand, and for each unit of satisfied customer demand a prize $p > 0$ is earned. We remark that for simplicity we do not consider here any capacity restrictions at service points. A solution to the SPDP is given by 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.

The problem is incompletely specified in the sense that we do not have a function for calculating the fulfilled demand for a candidate solution upfront. Instead, we assume here that we are only able to evaluate solutions "exactly" by presenting them to the potential customers U and collecting their feedback. These user evaluations are denoted by $d(u, v, x)$ which specifies the demand of user $u \in U$ fulfilled at location $v \in V$ in solution x. If a service station is not ideal for a user but somewhat acceptable for him to be used and there are no better alternatives in solution x, this is modeled by a correspondingly reduced fulfilled demand value $d(u, v, x)$; i.e., the user is less likely to use this service point and therefore the expected fulfilled demand is lower. Clearly, the number of candidate solutions that are evaluated in this interactive way are a major concern. We cannot confront each user with hundreds or thousands of evaluation requests. Instead, we carefully have to select the solutions to be evaluated by each user in an individual fashion, avoiding redundancies as far as possible.

Naturally, the demand fulfilled at any location must always be non-negative and can only be positive when a service point is set up there, i.e.,

$$d(u, v, x) \geq 0, \quad x_v = 0 \rightarrow d(u, v, x) = 0 \qquad u \in U,\ v \in V. \tag{1}$$

A solution x is feasible if its total fixed costs do not exceed the maximum budget B, i.e.,

$$c(x) = \sum_{v \in V} c_v x_v \leq B. \tag{2}$$

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

$$f(x) = p \cdot \sum_{u \in U} \sum_{v \in V} d(u, v, x) - \sum_{v \in V} z_v x_v. \tag{3}$$

3 Related Work

The SPDP can be classified as a variant of the *Facility Location Problem* (FLP). In the FLP a set of potential facility sites and a set of demand points is given. The task is to select a subset of these sites in order to serve the demand points w.r.t. some optimization goal subject to a set of constraints. For a survey on FLPs see [6]. More specifically, our SPDP is closely related to the uncapacitated FLP [4] in which each facility can satisfy an arbitrary amount of demand – with the substantial difference that in our case user demands are not known upfront but must be learned via user interaction.

The problem of optimizing the distribution of charging stations for EVs has gained increased attention recently. An essential question of contributions concerning this topic always is how to determine potential customer demands. Chen et al. [2] substitute charging demand with parking demand in order to identify good locations for public charging

stations. The parking demand is derived from parking information of a travel survey. In [9], a maximal covering model [3] for identifying charging stations is proposed. The demands are estimated using regression analysis based on surveys on the number of cars per household, the average travel distance of cars, the estimated range of an EV etc. In [1], the charging demand of a location is modeled as the expected duration of charging all drivers that need to charge their EV at this location. The number of drivers in a location is derived from a mobility survey as part of a case study of the city of Coimbra, Portugal. In [13] charging stations for an on-demand bus system are located using taxi probe data of Tokyo. While the focus in this paper lies on distributing charging stations for EVs, our approach is a general framework capable of planning service point based systems of any kind, such as bike or car sharing systems [10].

Opposite to the aforementioned contributions, we assume to have essentially no knowledge on customer demand in advance but aim at obtaining this information on the fly in an interactive way by integrating potential customers in the optimization process. More generally, in *interactive optimization algorithms*, humans are typically used to evaluate the quality of solutions; e.g., in [14] an interactive genetic algorithm for designing dresses is proposed. Instead of explicitly defining a fitness function, the fitness of a solution is decided by a user. For a survey on interactive optimization algorithms see [19]. A major disadvantage of interactive algorithms is that their performance strongly depends on the quality of the feedback given by the interactors. Continuous user interactions will eventually result in user exhaustion [17], negatively influencing the reliability of the obtained feedback. Therefore, user interactions should not only be considered time consuming but users also need to be treated as a scarce resource – the interaction should be kept to a required minimum. A common way to overcome this problem is to combine interactive optimization algorithms with a surrogate-based approach [23, 24, 16, 22]. Surrogate models are typically used as a proxy of functions which are either unknown or extremely time consuming to compute. Classic candidates for such surrogates are machine learning (ML) models. In [15] a survey of popular surrogate functions is provided, ranging from polynomial regression [8] to more sophisticated techniques such as neural networks [11] and support vector regression [7].

Our approach also exhibits similarities to so-called *interactive ML* approaches. Such methods are typically used when the number of training samples is not sufficient to properly train an ML model. To compensate this problem, a human is used as a guide to reduce the search space during the learning phase [12]. One way to reduce the search space is to reduce the number of features considered during the learning phase [5]. Our approach can be considered as an interactive ML algorithm in the sense that the ML models in use are continuously improved and corrected through user evaluations.

4 Cooperative Optimization Algorithm

The proposed solution framework, which we call *Cooperative Optimization Algorithm (COA)*, consists of the following interacting components: an *evaluation component (EC)*, an *optimization component (OC)*, a *feedback component (FC)*, and a *solution management component (SMC)*.

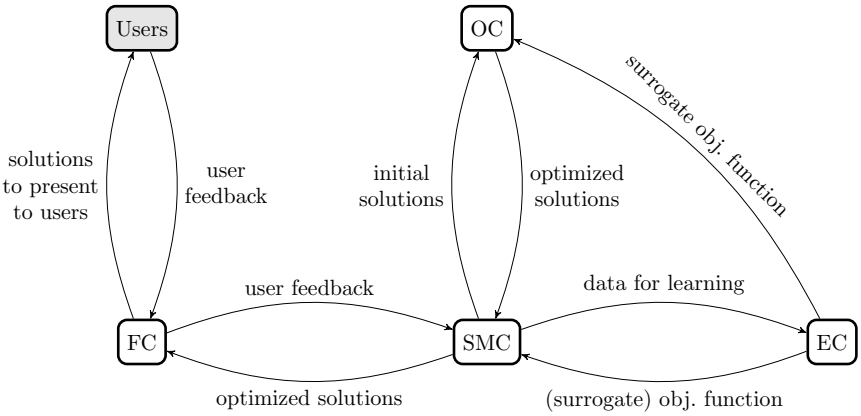


Fig. 1: Components of the framework and their interaction. The framework consists of the feedback component (FC), the evaluation component (EC), the optimization component (OC), and the solution management component (SMC). Users can interact with the framework via the FC.

The FC is responsible for selecting or deriving for each user individual candidate solutions that are then presented to him for evaluation. A user $u \in U$ gives feedback to a proposed solution x by stating how much of his demand would actually be satisfied at which locations, i.e., he returns the values $d(u, v, x),\ v \in V$.

The EC provides a function for evaluating candidate solutions. This in particular also includes intermediate solutions that are not directly evaluated by the users. Should for a solution x the values $d(u, v, x)$ be known for each user $u \in U$ and all locations $\{v \in V \mid x_v = 1\}$, we can calculate the exact objective value $f(x)$ according to (3). Otherwise, we resort to a surrogate objective function $\tilde{f}(x)$ based on an ML model that estimates the real objective value based on the information gained from the users' feedback so far. The respective learning mechanism also is part of the EC.

One call of the OC solves the SPDP by using the EC's current surrogate objective function for evaluating any intermediate solutions and yields one or more optimal or close-to-optimal solutions w.r.t. the current state of the EC. Note that the surrogate objective function never changes during one call of the OC. Instead, the OC is called repeatedly in each major iteration of the framework after having obtained new user feedback and re-trained the EC.

Finally, the SMC efficiently stores and manages information on all candidate solutions that are relevant for more than one of the above components and in particular also the solutions for which users have given feedback.

Figure 1 illustrates the communication between the components, and Algorithm 1 shows how the components of the framework interact with each other within the main program. The algorithm starts with the FC by presenting each user the solution in which all locations are selected, i.e., $x_v = 1$, $v \in V$. While this solution is unrealistic concerning the budget constraint, it allows us to obtain for each user $u \in U$ his maximum total satisfiable demand, denoted by $d_u^{\max}$, in the sense that no other solution will sat-

Algorithm 1: Basic Framework

Input : an instance of the SPDP
Output: a solution $x = (x_v)_{v \in V} \in \{0,1\}^n$

$X_{\mathrm{OC}} = \{(1, \ldots, 1)\}$ // initial solution, later best solution(s) from the OC **while** *no termination criterion satisfied* **do**
  Feedback Component:
    for $u \in U$ **do**
      determine set of solutions X'_u to be evaluated by u from X_{OC} and further data in the SMC;
      let user u evaluate X'_u, update the SMC with evaluated solutions from X'_u;
    end for
  Evaluation Component:
    train surrogate objective function $\tilde{f}(x)$ with data from the SMC;
    re-evaluate all solutions stored in the SMC with the new surrogate objective function;
  Optimization Component:
    adopt so far best solutions from the SMC as initial solutions;
    $X_{\mathrm{OC}} \leftarrow$ perform optimization using the EC's surrogate objective function $\tilde{f}(x)$;
    when possible, calculate exact $f(x)$ for $x \in X_{\mathrm{OC}}$;
    store the solution(s) from X_{OC} in the SMC;
end while
return *overall best found solution* x^* *w.r.t.* $\tilde{f}$;

isfy a larger total demand for the user. The information acquired in this way over all users serves as initial training data for the EC. The following subsections describe each component's functionality in more detail.

4.1 Solution Management Component

As the number of locations $|V|$ is usually much larger than the number of service stations that can actually be built in a feasible solution, candidate solutions in the SMC are compactly represented by the subset of locations chosen for setting up service points. Let us denote this set of locations by $s(x) = \{v \in V \mid x_v = 1\}$.

Next to storing all considered candidate solutions, the SMC also maintains for each solution the users for which the exact fulfilled demand is already known, and for each user $u \in U$ a set of so far identified *relevant locations* V_u, which includes any location for which the user has indicated positive demand in at least one solution. Note that the complete set of relevant locations in general is unknown. However, it is the task of the FC to choose the solutions presented to the users in such a way that as many relevant locations as possible are identified (see Section 4.2). The sets V_u for each $u \in U$ are used in the EC for constructing the surrogate function (see Section 4.3).

Another important task of the SMC is to calculate a user's demand w.r.t. solutions for which the users's evaluation can trivially be derived. Remember that in the first iteration of the FC, each user u is confronted with the artificial solution in which all locations are selected and thus the maximum fulfillable demand $d_u^{\max}$ is determined. By

removing from this solution all locations without a positive fulfilled demand, a minimal optimal solution is obtained. More generally, we define the subset of actually used locations for a solution x and user u by $s_u^{\min}(x) = \{v \in V \mid d(u, v, x) > 0\}$. Throughout the whole algorithm, any encountered minimal optimal solution is stored for each user in a set X_u^{minopt}. Any new candidate solution x' (outside of the OC) is then checked if it is a superset of a solution $x \in X_u^{\text{minopt}}$, in which case we know that x' also is an optimal solution for the user u with total fulfilled demand $d_u^{\max}$ and that $d(u, v, x') = d(u, v, x)$ for $v \in s(x)$ and $d(u, v, x') = 0$ else.

4.2 Feedback Component

In each major generation of Algorithm 1, the FC generates for each user $u \in U$ an individual set of solutions to evaluate. It is assumed here that any user u evaluates each solution in a completely rational way so that the total fulfilled demand is maximal. The number of user evaluations of solutions need to be kept as low as possible to avoid user fatigue [17], we cannot ask real users to evaluate hundreds or thousands of solutions. Thus, each solution presented to a user must be non-redundant and meaningful in the way that we likely obtain new knowledge on his needs that is valuable for finding an overall optimal solution.

It appears natural that a solution presented to a user should be similar to the best solutions identified so far by the OC or, otherwise, provide substantial information gain on locations that are potentially interesting for the user. Next to finding new relevant locations for a user, it is also necessary to gain information on the relationship between relevant locations.

We apply the following combination of strategies for compiling a set of at most κ solutions presented to each user $u \in U$, where κ is a strategy parameter and is set to 15 in the experiments performed for this article.[3] In general, a potential solution x is only presented to a user u if it has not already been evaluated by the user in a previous iteration and if $s(x)$ is not a superset of a solution in X_u^{minopt}; otherwise x is skipped.

Best Solution Strategy. Let $f(x)$ be the evaluation function in which we consider exact fulfilled demands $d(u, v, x)$ and let $\tilde{f}$ be the surrogate function, which will be defined in Section 4.3. Select the γ_1 best feasible solutions w.r.t. to f and the γ_2 best feasible solutions w.r.t. the surrogate function $\tilde{f}$ (see Section 4.3) for which no exact total fulfilled demand is known yet for user u. Hereby, γ_1 and γ_2 are strategy parameters, which are both set to 2 in the experiments performed for this article. This strategy clearly focuses on getting exact evaluations for the currently most promising solutions.

Irrelevant Locations Strategy. This strategy focuses purely on finding new relevant locations for a user u, which might lead to good alternative solutions. For this purpose a solution in which the locations in $V \setminus V_u$ are selected is generated. Note that this solution is not necessarily feasible, which, however, does not immediately matter for the intended purpose.

[3] All parameter values stated in the text have been tuned in comprehensive preliminary tests.

Best Solution Mutation Strategy. This strategy is a combination of the previous strategies and tries to gain information on the relationship between locations by replacing a subset of $s(x) \cap V_u$ of a solution x obtained from the best solution strategy for a user u with a set of locations for which it is so far unclear if they are relevant: A new solution x' is constructed from a copy of an existing solution x by setting $x_v = 0$ with $v \in s(x) \cap V_u$ with a certain probability ξ for each v, where ξ is a strategy parameter which is set to 0.5 in the experiments throughout this article. Afterwards, we set $x_v = 1$ for n' uniformly at random chosen locations $v \in V \setminus s(x)$ with n' being chosen uniformly at random from $\{0, \ldots, |V| - |s(x)|\}$.

Due to its randomization there is a small chance that this strategy yields already existing solutions. Therefore, we apply it to the solutions obtained from the first strategy to reach the intended κ candidate solutions; however, we stop the procedure after at most 5κ iterations.

For the same reasons mentioned in the previous strategy, solutions generated with this strategy do not have to be feasible either.

4.3 Evaluation Component

The EC provides the means for evaluating solutions, in particular also temporary solutions generated within the OC. Calculating the exact objective value of a solution x is trivial if x has already been evaluated by each user $u \in U$, in which case Equation (3) is directly applied.

However, if a solution x has not yet been evaluated by some user, the exact objective value is estimated by a surrogate objective function $\tilde{f}(x)$, which is defined in accordance to $f(x)$ but makes use of *estimated fulfilled demands*

$$\tilde{d}(u, v, x) = \begin{cases} 0 & \text{if } v \notin V_u \vee x_v = 0 \\ \max(0, g_{u,v}(x)) & \text{else} \end{cases} \tag{4}$$

for each user $u \in U$ and each location $v \in V$, where $g_{u,v}(x)$ represents an ML model trained by all solutions so far evaluated by user u. Note that our definition of $\tilde{d}(u, v, x)$ ensures that conditions (1) are always fulfilled and gives function $g_{u,v}(x)$ more freedom in the sense that it may return negative values, which are mapped to zero, and arbitrary values in case of $x_v = 0$. Furthermore, for any location v for which user u has so far never indicated any positive fulfilled demand in any solution, i.e., for any so far not relevant location $v \in V \setminus V_u$, $g_{u,v}(x) = 0$ is assumed and no ML model needs to be maintained.

Similarly to [21], we use an adaptive surrogate function in the sense that the ML model for each $g_{u,v}(x)$ is initially simple and is upgraded to a higher complexity model during the course of the algorithm when the error of the model – measured in terms of the usual mean squared error (MSE) of $\tilde{d}(u, v, x)$ – exceeds a certain threshold τ. In this way we stay as efficient as possible from a computational perspective and substantially reduce problems with overfitting.

Our initial choice for $g_{u,v}(x)$ is the linear model (LM)

$$g^{\mathrm{LM}}_{u,v}(x) = w_{u,v} + \sum_{v' \in V_u \setminus \{v\}} w'_{u,v,v'} \cdot x_{v'}. \tag{5}$$

Ridge regression with a penalization factor of one is used for determining the weights $w_{u,v}$ and $w'_{u,v,v'}$. This model is sufficient for covering simple scenarios where users have independent demands that can be fulfilled at specific locations. Furthermore, it can even accurately represent the case where for a user one demand can be fulfilled at a specific primary location or, with a possibly reduced amount, at one alternative location if no service station is set up at the primary location. More complex dependencies, including in particular more than one alternative location, are, however, beyond the capability of the LM.

In this case, which is detected by a remaining MSE of $\tilde{d}(u, v, x)$ larger than a threshold $\tau = 0.075$, we turn to a neural network, starting with a single layer perceptron with a leaky rectified linear unit (ReLU) activation function [18]. This simple neural network realizes the function

$$g^{\mathrm{NN}}_{u,v}(x) = \phi\left(g^{\mathrm{LM}}_{u,v}(x)\right) \text{ with } \phi(S) = \begin{cases} S & \text{if } S \geq 0 \\ \varepsilon \cdot S & \text{else.} \end{cases} \tag{6}$$

The leaky ReLU activation function ϕ serves as an extension of the LM in the sense that this perceptron takes actively into account that satisfied demands cannot be negative. Due to this non-linearity, it can accurately represent scenarios in which for a user a demand can be fulfilled at an arbitrary number of ordered alternative locations, where a service station at one of these locations will only fulfill a certain amount of the demand when no station is set up at any of the preceding alternative locations in the order. We use here the leaky ReLU function with parameter $\varepsilon = 0.01$ which returns small negative values in case the sum S is negative instead of the more common classical ReLU function (which corresponds to the case $\varepsilon = 0$) in order to avoid the well-known problem of the "dying neuron" due to a zero gradient in case of negative values S.

While the above perceptron is already more powerful, it is still limited when a user has more than one demand that can be fulfilled partly at the same locations, or more generally, when the different demands are related in some way. Again, we detect the insufficiency of the perceptron by a MSE that exceeds τ and turn in this case to a more complex feed forward neural network with one hidden layer that contains initially two hidden neurons. These neurons again make use of the leaky ReLU activation function, while the single output layer neuron corresponds to a simple summation of the inputs. Initially, we use two hidden neurons and increase this number until, after training, either the MSE does not exceed τ anymore or a maximum of $\lambda = 6$ hidden neurons is reached.

Note that in an implementation of the approach, significant time can usually be saved by re-training the ML model for each user and each location only when new data relevant data for the specific model is available. Note that the solutions used for training the models are not required to be feasible, since user evaluations do not consider the budget at all.

4.4 Optimization Component

Remember that the OC is called in each major iteration of the whole framework and makes use of the current surrogate function provided by the EC, which does not change

during each individual run of the OC. The OC is thus supposed to return an optimal or close-to-optimal solution w.r.t. the current surrogate function.

The OC is implemented as a Variable Neighborhood Search (VNS) and follows the classical scheme from [20]. It consists of a randomized construction heuristic, a local search part, and a shaking mechanism for escaping local optima. The initial solution is generated via the randomized construction heuristic that considers all locations in random order and sets up a station at a location as long as the budget is not exceeded.

Our local search follows a first improvement strategy and utilizes a two-exchange neighborhood structure, in which a location in the solution is replaced by a location not contained in the solution. The VNS only considers feasible solutions, hence, we skip all moves in the neighborhood resulting in budget constraint violations. Moreover, after each feasible move, we try to additionally improve the solution by adding stations at further locations to the solution in a random order as long as the budget allows it.

Shaking removes stations from a number of uniformly selected random locations and then iteratively adds stations to other locations in a uniform random order, such that the solution stays feasible and no more locations can be added. The number of stations to be removed corresponds to the index of the shaking neighborhood and varies from one to two.

The VNS terminates if no better solution has been found within 40 iterations.

5 Experimental Evaluation

We test the suggested framework in a proof-of-concept manner on artificial benchmark scenarios using an idealized simulation of all user interaction. To a large degree, the proposed framework is independent of the concrete application as long as our general problem formulation is suitable. The machine learning models in the EC, however, were already designed with a few assumptions on user requirements, as they appear, for example, in the context of setting up charging stations for EVs: Users would like to have certain needs associated with use cases fulfilled that are related to particular geographic locations, such as their home and/or work address or other places they visit regularly. While ideally respective service stations would be set up at precisely these locations, the respective demands can to a certain degree also be fulfilled by service stations located in the vicinity. The degree (amount) of fulfilled demand, however is assumed to decrease with the distance. In this way, we implicitly also consider the convenience for the users. It is generally assumed that for fulfilling a demand, a user always uses a station that is closest to the demand's original location.

5.1 Benchmark Scenarios

The primary parameters for our benchmark scenarios are the number of potential locations for service stations n and the number of users m, and we consider here the combinations $n = 50, 60, \ldots, 100$ with $m = 50$ and $n = 50$ with $m = 50, 60, \ldots, 100$. The n locations correspond to points in the Euclidean plane with coordinates chosen uniformly at random from the grid $\{0, \ldots, L-1\}^2$, where $L = \lceil 10\sqrt{n} \rceil$ is the underlying width and height. It is ensured that all locations have different coordinates. 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 $\{50, \ldots, 100\}$. 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. We assume each of the m users $u \in U$ has ρ_u so-called *use cases*, where ρ_u is chosen randomly according to a shifted Poisson distribution with offset one and expected value three. Each of these use cases $i = 1, \ldots, \rho_u$ is associated with a particular geographical location $r_{u,i} \in \{0, \ldots, L-1\}^2$ and a respective demand $d^*_{u,i}$ that could ideally be fulfilled there. This demand can, for example, be the expected number of usages of a service point in a time period. Here, we choose each $d^*_{u,i}$ uniformly at random from $\{5,\ldots,50\}$. In a real scenario, the locations where demand arises will clearly not be uniformly distributed over the whole considered geographic area. There will be more popular regions as well as less popular ones. We want to consider this aspect and therefore first choose $\alpha = \lceil (L/50)^2 \rceil$ *attraction points* A with uniform random coordinates from $\{0, \ldots, L-1\}^2$ and then derive the location for each use case from a uniformly selected attraction point $(a_x, a_y) \in A$ by

$$r_{u,i} = (\lfloor \mathcal{N}(a_x, 20) \rfloor \bmod L, \lfloor \mathcal{N}(a_y, 20) \rfloor \bmod L), \tag{7}$$

where $\mathcal{N}(\cdot, \cdot)$ denotes a random value sampled from a normal distribution with the respectively given mean value and standard deviation.

For each use case $i = 1, \ldots, \rho_u$ of each user $u \in U$, demand is always only fulfilled at the closest location $v^{\text{clst}}_{u,i}(x) \in V$ w.r.t. the Euclidean distance where a service station is set up in the current candidate solution x (ties are broken according to the locations' natural order) and when a maximum distance, chosen here as 12, is not exceeded. We further assume an exponential decay of the fulfilled demand in dependence of the distance and round down to the closest integer, obtaining

$$d_i(u, v, x) = \begin{cases} \lfloor d^*_{u,i} \cdot e^{-||r_{u,i} - v^{\text{clst}}_{u,i}(x)||/10} \rfloor & \text{if } v = v^{\text{clst}}_{u,i}(x) \wedge ||r_{u,i} - v|| \leq 12 \\ 0 & \text{else,} \end{cases} \tag{8}$$

where $|| \cdot ||$ denotes the L^2 norm. These fulfilled demands for each use case i are finally summed up in order to obtain the overall fulfilled demands $d(u, v, x) = \sum_{i=1} d_i(u, v, x)$ for each user $u \in U$ and location $v \in V$ under candidate solution x. Finally, the prize earned for each unit of fulfilled demand in our objective function is assumed to be $p = 50$.

For each combination of n and m 30 independent scenarios were created, and they are available at https://www.ac.tuwien.ac.at/research/problem-instances. The benchmarks were also specifically designed with the ability in mind to calculate proven optimal solutions to which we will compare the solutions of our framework. Exploiting the complete knowledge of the data and specific structure in a "white-box" manner allows the problem to be expressed as mixed-integer linear programming (MIP) model, which we solved with the MIP-solver Gurobi[4].

[4] http://www.gurobi.com/

5.2 Computational Experiments

The OC was implemented in C++, compiled with GNU G++ 5.5.0, while the remaining components of the framework were realized in Python 3.7. For linear regression scikit-learn 0.17 was used and for the perceptrons and neural networks Keras 2.2.2 on top of Theano 1.0.2 (without GPU support). The perceptrons and neural networks were trained with the adam optimizer (learning rate 0.1) over 5000 epochs with a batch size of 32 in order to minimize the MSE. All test runs have been executed on an Intel Xeon E5-2640 v4 with 2.40GHz machine.

Our framework terminated when no improved solution could be found over five iterations or when the CPU-time limit of 7200s had been reached and returned the overall best found solution x^* w.r.t. the approximate evaluation function $\tilde{f}$.

Table 1 lists average results of COA over all 30 instances for each considered combinations of n and m. Each line shows the average number of iterations n_{it}, the average of the exact objective values of the finally returned solutions $\overline{f(x^*)}$ and the corresponding optimal solutions $\overline{f(x_{\text{opt}})}$ obtained from Gurobi by solving the white-box MIP, the average optimality gap $\overline{\%\text{-gap}}$ and corresponding standard deviation $\sigma_{\%\text{-gap}}$, where the %-gap is calculated for a final solution x^* of COA in relation to an optimal solution x_{opt} as $\%\text{-gap} = 100\% \cdot (f(x_{\text{opt}}) - f(x^*))/f(x_{\text{opt}})$, the average percentage error of the surrogate function values of final solutions $\overline{\%\text{-}\Delta\tilde{f}}$ (serving as an indicator for the quality of the surrogate function) and the corresponding standard deviation $\sigma_{\%\text{-}\Delta\tilde{f}}$, with $\%\text{-}\Delta\tilde{f} = 100\% \cdot |\tilde{f}(x^*) - f(x^*)|/f(x^*)$, and the median of the computation times in seconds. The table shows that COA finds near optimal solutions for almost all instances. Average final gaps to optimal solutions are always less than 0.6%. Moreover, the surrogate function predicts the actual user demand at least for the final solutions excellently; for all instance groups the average percentage error of the surrogate function values is below 0.55%. The percentage errors and computation times tend to slightly increase with an increasing number of users, while the optimization gaps show no such behavior. Neither the number of users, nor the number of locations seems to have an impact on the optimality gaps, indicating that our algorithm is able to also solve larger instances with a similar solution quality. The generally rather high computation times can be explained by the large number of machine learning models that need to be trained in each iteration but also by the fact that the OC is implemented as pure black-box optimization.

Next, in Figure 2, we take a closer look at the percentage errors of the surrogate function. The boxplots in Figure 2 show the distribution of $\%\text{-}\Delta\tilde{f}$ for all instances. The figure shows that the percentage error of the final surrogate function of an instance is almost always below 1%. As observed in Table 1, one can see a slight increase of the percentage errors as the number of users increases, i.e. there seems to be a correlation of the size of the percentage errors and the number of users.

Figures 3a-b visualize for an exemplary run with $n = 100$ and $m = 50$ the best solutions w.r.t. $\tilde{f}$ at the first and at the last iteration of COA, respectively. Blue dots show the locations of users' use cases with their sizes indicating the respective maximal satisfiable demands $d^*_{u,i}$. Diamonds show the potential locations of service stations V, with the larger ones with the discs corresponding to those chosen in the best solution of the iteration. The actually fulfilled demand of a service station is indicated by the size of the diamonds, and the discs illustrate the covered area in respect to the maximum

Table 1: Average results of COA.

n	m	$\overline{n_{\text{it}}}$	$\overline{f(x^*)}$	$\overline{f(x_{\text{opt}})}$	$\overline{\%\text{-gap}}$	$\sigma_{\%\text{-gap}}$	$\overline{\%\text{-}\Delta\tilde{f}}$	$\sigma_{\%\text{-}\Delta\tilde{f}}$	time[s]
50	50	10	56499	56717	0.34	0.67	0.19	0.14	2063
50	60	10	67134	67467	0.38	1.05	0.31	0.19	2594
50	70	11	78562	78845	0.21	0.58	0.33	0.34	2936
50	80	10	88003	88283	0.28	0.51	0.35	0.32	3522
50	90	10	98408	98961	0.56	0.89	0.37	0.34	3867
50	100	11	106604	107020	0.28	0.48	0.41	0.44	4424
60	50	11	59189	59354	0.46	1.43	0.53	1.35	2335
70	50	11	61685	62097	0.49	0.93	0.38	0.48	2447
80	50	12	64425	64690	0.49	0.73	0.27	0.28	2904
90	50	11	66689	66870	0.27	0.40	0.23	0.19	2946
100	50	13	71030	71229	0.26	0.44	0.28	0.27	3889

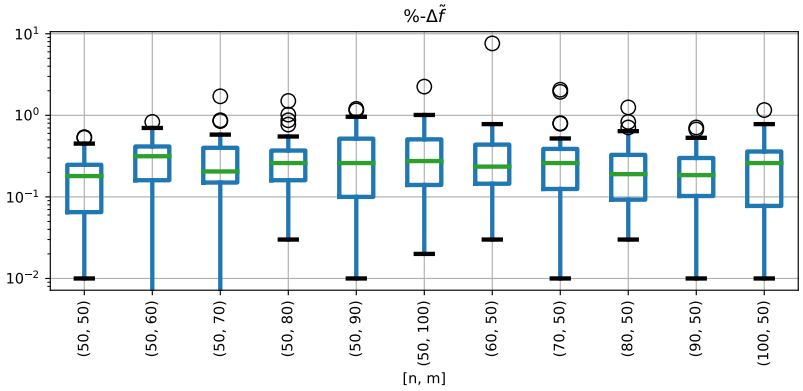
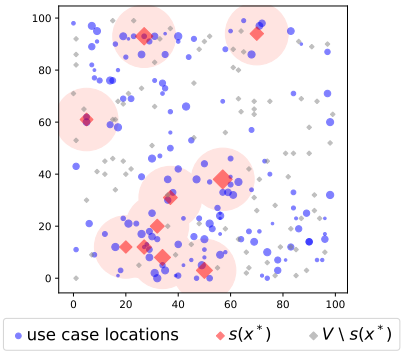


Fig. 2: Distributions of the percentage errors of the surrogate function values of final solutions.

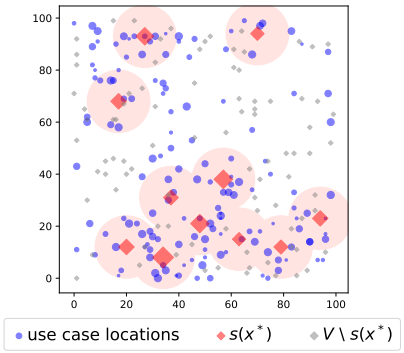
distance of 12. We can see that already the solution obtained in the first iteration is quite meaningful. Although the final solution is similar at the first glance, a closer look reveals a significantly better coverage of demands in the final solution.

This observation is also confirmed by Figure 3c showing the corresponding development of the best solution's exact objective value over the iterations in comparison to the optimal solution value $f(x_{\text{opt}})$. We can see that already the solution of the first iteration has a relatively high objective value, which is continuously improved in few iterations until the optimum is almost reached.

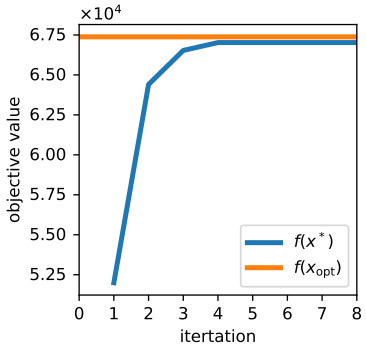
Finally, Figure 3d shows the distribution of the model sizes of the surrogate function's underlying machine learning models at the final iteration of COA. A model size of zero refers to LMs, size one to perceptrons, and larger sizes to neural networks with the respective number of neurons in the hidden layer. The distribution shown in Figure 3d is typical for all instances tested. It shows that the majority of machine learning models is



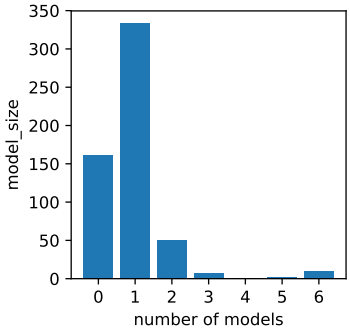
(a) iteration 1



(b) iteration 8



(c) solution quality over iterations



(d) model distribution

Fig. 3: An exemplary run with $n = 100$ and $m = 50$: (a-c) best solutions at different iterations and (d) exact objective value of best solution per iteration.

made up of LMs and perceptrons. Larger size neural networks are rarely needed. However, the figure also shows a small peak at the largest neural network with six neurons in its hidden layer. This peak is caused by unpopular service point locations resulting in training data in which most customer demands are zero. The neural networks often fail to properly learn such data, however, on the other hand, as these locations are the least popular service point locations, they usually have no large impact on the final solution.

6 Conclusion

We proposed a cooperative algorithm framework for distributing service points within a geographical area in mobility applications under incomplete information. Instead of estimating user demands by combining a variety of more or less reliable sources, our method directly incorporates potential customers in the optimization process. Our proof-of-concept implementation is still based on comparatively simple components.

Nevertheless we could show that the machine learning models in our evaluation component are able to learn the non-trivial user behavior of all our benchmark scenarios reliably after relatively few user interactions, and the optimization is able to indeed find solutions with only small remaining optimality gaps. The careful derivation of the candidate solutions to be presented to the users in the feedback component also plays a particularly important role.

In future work we will investigate the approach on more complex scenarios such as bike sharing systems, where a use case always relates to two, usually different locations for renting and returning a bike, respectively. Considering capacity limits and different possible configuration options for the service points is another practically highly relevant aspect. Another challenge is to improve the scalability of the approach towards more potential locations and more users. To this end it seems necessary to replace the individual machine learning models we currently have for each user and each location by a more integrated approach. Even though, we use a surrogate function to unburden the customers from evaluating too many solutions, the current number of solutions a user needs to evaluate is still very high. Additional efforts need to be made to further reduce this number. Last but not least, improvements should also be possible in the optimization component.

References

1. 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)
2. Chen, T., Kockelman, K.M., Khan, M.: The electric vehicle charging station location problem: A parking-based assignment method for Seattle. In: 92nd Annual Meeting of the Transportation Research Board in Washington DC (2013)
3. Church, R., ReVelle, C.: The maximal covering location problem. In: Papers in Regional Science. vol. 32, pp. 101–118. Springer (1974)
4. 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. John Wiley and Sons, Inc, New York, NY, USA (1990)
5. Fails, J.A., Olsen, Jr., D.R.: Interactive machine learning. In: Proceedings of the 8th International Conference on Intelligent User Interfaces. pp. 39–45. IUI '03, ACM, New York, NY, USA (2003)
6. Farahani, R.Z., Hekmatfar, M.: Facility Location: Concepts, Models, Algorithms and Case Studies. Springer (2009)
7. Forrester, A., Andras, S., Keane, A.: Engineering Design Via Surrogate Modelling: A Practical Guide. John Wiley & Sons (2008)
8. Forrester, A.I., Keane, A.J.: Recent advances in surrogate-based optimization. Progress in Aerospace Sciences **45**(1), 50 – 79 (2009)
9. 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)
10. Gavalas, D., Konstantopoulos, C., Pantziou, G.: Design and management of vehicle-sharing systems: A survey of algorithmic approaches. In: Obaidat, M.S., Nicopolitidi, P. (eds.) Smart Cities and Homes, pp. 261–289. Elsevier (2016)

11. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, 2nd edn. (1998)
12. Holzinger, A.: Interactive machine learning for health informatics: when do we need the human-in-the-loop? Brain Informatics **3**(2), 119–131 (2016)
13. 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)
14. Kim, H.S., Cho, S.B.: Application of interactive genetic algorithm to fashion design. Engineering applications of artificial intelligence **13**(6), 635–644 (2000)
15. 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)
16. Lee, J.Y., Cho, S.B.: Sparse fitness evaluation for reducing user burden in interactive genetic algorithm. In: Fuzzy Systems Conference Proceedings, 1999. FUZZ-IEEE'99. 1999 IEEE International. vol. 2, pp. 998–1003. IEEE (1999)
17. Llorà, X., Sastry, K., Goldberg, D.E., Gupta, A., Lakshmi, L.: Combating user fatigue in iGAs: Partial ordering, support vector machines, and synthetic fitness. In: Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation. pp. 1363–1370. GECCO '05, ACM, New York, NY, USA (2005)
18. Maas, A.L., Hannun, A.Y., Ng, A.Y.: Rectifier nonlinearities improve neural network acoustic models. In: Workshop on Deep Learning for Audio, Speech and Language Processing, ICML 2013 (2013)
19. 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)
20. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)
21. Shi, L., Rasheed, K.: Asaga: An adaptive surrogate-assisted genetic algorithm. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation. pp. 1049–1056. GECCO '08, ACM, New York, NY, USA (2008)
22. Sun, X.Y., Gong, D., Li, S.: Classification and regression-based surrogate model-assisted interactive genetic algorithm with individual's fuzzy fitness. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation. pp. 907–914. GECCO '09, ACM, New York, NY, USA (2009)
23. Sun, X., Gong, D., Jin, Y., Chen, S.: A new surrogate-assisted interactive genetic algorithm with weighted semisupervised learning. IEEE transactions on cybernetics **43**(2), 685–698 (2013)
24. Zhou, Y., Gong, D.W., Hao, G.s., Guo, Y.N., Sun, X.Y.: Neural network based phase estimation of individual fitness in interactive genetic algorithm. Control and Decision **20**(2), 234–236 (2005)