VNS and PBIG as Optimization Cores in a Cooperative Optimization Approach for Distributing Service Points[⋆]

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 present a cooperative optimization approach for distributing service points in a geographical area with the example of setting up charging stations for electric vehicles. Instead of estimating customer demands upfront, customers are incorporated directly into the optimization process. The method iteratively generates solution candidates that are presented to customers for evaluation. In order to reduce the number of solutions presented to the customers, a surrogate objective function is trained by the customers' feedback. This surrogate function is then used by an optimization core for generating new improved solutions. In this paper we investigate two different metaheuristics, a variable neighborhood search (VNS) and a population based iterated greedy algorithm (PBIG) as core of the optimization. The metaheuristics are compared in experiments using artificial benchmark scenarios with idealized simulated user behavior.

Keywords: Cooperative optimization · facility location problem · metaheuristics.

1 Introduction

Usually, locations for setting up charging stations for electric vehicles are optimized by some algorithm w.r.t. previously estimated customer demand. However, estimating the demand of a customer upfront is challenging, as this task is usually based on various uncertain data and uncertain assumptions about the behavior of customers. Among other things, customer demand may be fulfilled in alternative ways that cannot all be predicted in advance. Sometimes, customers might not even completely be aware themselves at which conditions their demands can be fulfilled best until they can see an actual system configuration. At this point, however, it is usually too late to make any further impactful changes.

Therefore, we propose in [4] a *Cooperative Optimization Approach (COA)* for solving this problem by incorporating potential customers into a combined data

[⋆] Thomas Jatschka acknowledges the financial support from Honda Research Institute Europe.

acquisition and optimization process. The method iteratively generates solutions that are presented to the customers for evaluation. Based on the customer's feedback a surrogate objective function is trained and used by an optimization core to generate new, improved solutions. This process is iterated on a large scale with many potential customers and several rounds until a satisfactory solution is reached.

In this contribution, we focus in particular on the optimization part of our approach. A proper configuration of the used optimization core is vital for the cooperative approach to work. We investigate a variable neighborhood search (VNS) and a population based iterated greedy algorithm (PBIG) for this purpose. First, however, we formally introduce the problem to be solved – the Service Point Distribution Problem (SPDP).

2 The Service Point Distribution Problem

While this paper considers the distribution of charging stations for electric vehicles as particular example, we define the underlying problem we consider – the SPDP– in a more general way to be independent of the actual application scenario. In the SPDP we are given a set of locations V at which service points may be built and a set of users U for which the service points are built. 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.

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 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

$$f(x) = p \cdot \sum_{u \in U} \sum_{v \in V} d(u, v, x) - \sum_{v \in V} z_v x_v, \tag{1}$$

where $d(u, v, x)$ specifies the demand of user $u \in U$ fulfilled at location $v \in V$ in solution x. The problem is incompletely specified in the sense that we do not know how to calculate $d(u, v, x)$. The function can only be evaluated by presenting the solution x to user u and collecting the user's feedback. Clearly, the number of candidate solutions that are evaluated in this interactive way must be kept low, as we cannot confront each user with hundreds of evaluation requests. Hence, we additionally make use of a surrogate function $\tilde{d}(u, v, x)$ trained by obtained user feedback, which is actually used by the optimization core for evaluating solutions.

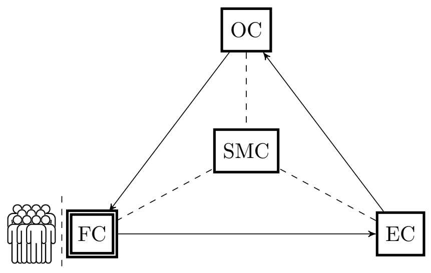

Fig. 1: The cycle of the COA framework. Users evaluate candidate solutions provided by the FC. The feedback is used to train a surrogate function in the EC which is used by the OC to find new optimized solutions.

3 Related Work

The SPDP is a variant of the *uncapacitated Facility Location Problem* (uFLP) [3]. The uFLP generally deals with selecting a subset from a set of potential facility sites in order to serve a set of demand points w.r.t. some optimization goal subject to a set of constraints.

With the rise of electric vehicles, the problem of finding optimal locations for charging stations has gained increased attention recently. There already exists a large number of studies concerning this topic. Moreover, these studies all have to address the problem of how to determine the demands of potential customers. Chen et al. [2] derive the customer demand for charging electric vehicles by using parking demand gained from a travel survey. In [5] charging stations for an on-demand bus system are located using taxi probe data of Tokyo.

For a survey on interactive optimization algorithms see [9]. Continuous user interactions will eventually result in user exhaustion [7], negatively influencing the reliability of the obtained feedback. A way to unburden the users is to use a surrogate-based approach. Surrogate models are typically used as a proxy of functions which are either unknown or extremely time consuming to compute [6].

4 Cooperative Optimization Approach (COA)

In this section, we summarize the COA as proposed in [4], which consists of: an *evaluation component (EC)*, an *optimization core (OC)*, a *feedback component (FC)*, and a *solution management component (SMC)*, see also Figure 1.

The SMC stores and manages solutions and all associated data such as user feedback or surrogate objective values. All framework components can access the SMC.

The FC provides the interface to the users and is responsible for deriving an individual set of solutions for each user that is then presented to the user for evaluation. Solutions are derived from the so far best found solutions stored in the SMC with the purpose of identifying new relevant locations for a user and determining the relationship between a user's relevant locations. Each user gives feedback to the proposed solutions by stating how much of the user's demand would actually be satisfied at which locations, i.e., a user $u \in U$ returns the values $d(u, v, x)$ for all $v \in V$ w.r.t. a solution x. Weaker preferences of locations are hereby expressed by smaller values.

Once the feedback is obtained from the users, the EC builds a surrogate function $\tilde{d}(u, v, x)$ based on machine learning (ML) models $g_{u,v}$ for each pair $(u, v) \in U \times V_u$, where V_u is the set of so far identified relevant locations of user $u \in U$, i.e., the set of locations for which a user has expressed a positive demand at least once. Each $g_{u,v}$ gets as input a binary incidence vector $x = (x_w)_{w \in V_u, w \neq v}$, with $x_w = 1$ indicating that a service point exists at location w. Initially, each $g_{u,v}$ starts out as a linear regression model. However, once the mean squared error (MSE) of $\tilde{d}(u, v, x)$ exceeds some threshold $\tau = 0.075$, the model is upgraded to a perceptron. The model can be further upgraded to a neural network with a single hidden layer and initially two neurons in the hidden layer. Afterwards, whenever the MSE of $\tilde{d}(u, v, x)$ exceeds τ, an additional neuron is added to the hidden layer of the neural network until a maximum number of neurons is reached.

In the last step of a COA iteration, new, improved solutions are generated in the OC using the surrogate function for evaluating solutions. The OC is implemented as a black-box optimization model and returns one or multiple close-to-optimal solutions w.r.t. $\tilde{d}$. In the next section we investigate two different methaheuristics serving as core optimization of COA.

4.1 VNS and PBIG as core optimization of COA

A proper choice of optimization algorithm and corresponding configuration is vital for the COA to work. The optimization algorithm should not only provide close-to-optimal solutions but should also not need too many candidate solution evaluations as they are rather time-expensive and the OC needs to be repeatedly performed. For this purpose we consider two different metaheuristics – a VNS and a PBIG – as OC.

The VNS follows the classical scheme from [10]. An 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 uses an exchange & complete neighborhood following a first improvement strategy. In the first step of the neighborhood, a location in the solution is replaced by an unused location, i.e., a location at which no service point exists. As this exchange might decrease the current budget of the solution, we afterwards add further unused locations in a random order to the solution as long as the budget allows it. The k-th shaking removes k randomly selected locations from the solution and then iteratively adds unused locations in a uniform random order until no more locations can be added. Note that the VNS considers only feasible solutions.

For the general principles of the PBIG we refer to [1]. An initial population of solutions is generated via the same randomized construction used for generating the initial solution of the VNS. Then, in each major iteration a new solution is derived from each solution in the current population by applying a destroy & recreate operation. The best solutions from the joint set of original and newly derived solutions are accepted as new population for the next iteration. Our destroy & recreate operation first removes a number of selected locations from the solution and then again iteratively adds unused locations in a uniform random order, such that the solution stays feasible and no more stations can be added. We reuse the exchange & complete neighborhood as well as the shakings operators of the VNS as destroy & recreate operations of the PBIG. Hence, the PBIG also considers only feasible solutions. Note that the PBIG returns its final population while the VNS only returns a single solution as result of the optimization.

5 Experimental Evaluation

As previously mentioned, the COA is tested in a proof-of-concept manner on artificial benchmark scenarios using an idealized simulation of all user interaction.

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 a grid with height and width $\lceil 10\sqrt{n} \rceil$. 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 $\{50, \ldots, 100\}$. The budget is chosen in such a way that about 10% of the service points can be set up on average.

Each user has a set of *use case locations* distributed in the same grid as the potential service point locations V. The number of use case locations is determined by a shifted Poisson distribution with offset one and expected value three. The use case locations themselves are determined with a set of randomly chosen attraction points, i.e., the closer a location is to an attraction point, the more likely the location is to be chosen as a use case location. Each use case location is associated with a maximal demand that can be partially fulfilled by service points within a maximal walking distance. The satisfied demand is calculated by a distance decay function. It is assumed that a user always chooses the service point closest to a use case location, i.e., the service point that can satisfy most of the use case location's demand.

For each combination of n and m 30 independent scenarios were created. They are available at www.ac.tuwien.ac.at/research/problem-instances/#spdp. The instances were also specifically designed with the ability in mind to calculate proven optimal solutions to which we will compare the solutions of our framework.

Table 1: VNS vs. PBIG.

		VNS				PBIG			
n	m	$\overline{\%\text{-gap}}$	$\sigma_{\%\text{-gap}}$	$\overline{n_{\text{it}}^{\text{best}}}$	t[s]	$\overline{\%\text{-gap}}$	$\sigma_{\%\text{-gap}}$	$\overline{n_{\text{it}}^{\text{best}}}$	t[s]
50	50	**0.26**	0.48	**29**	**3**	0.28	1.22	696	8
50	60	**0.20**	0.61	**35**	**4**	0.31	1.36	656	9
50	70	**0.00**	0.02	**32**	**4**	0.10	0.42	634	8
50	80	0.31	0.74	**31**	**4**	**0.09**	0.28	631	9
50	90	**0.14**	0.42	**32**	**4**	0.37	1.01	648	11
50	100	0.37	1.03	**33**	**4**	**0.01**	0.07	706	15
60	50	0.25	0.64	**43**	**4**	**0.07**	0.34	862	9
70	50	0.34	0.62	**54**	**5**	**0.28**	0.57	1113	17
80	50	0.43	0.54	**56**	**6**	**0.24**	0.51	1389	23
90	50	0.30	0.42	**64**	**7**	**0.19**	0.60	1628	30
100	50	**0.37**	0.47	**65**	**9**	0.42	0.82	1754	39

Table 2: COA[VNS] vs. COA[PBIG].

		COA[VNS]				COA[PBIG]			
n	m	$\overline{\%\text{-gap}}$	$\sigma_{\%\text{-gap}}$	$\overline{n_{\text{it}}}$	t[s]	$\overline{\%\text{-gap}}$	$\sigma_{\%\text{-gap}}$	$\overline{n_{\text{it}}}$	t[s]
50	50	0.28	0.70	11	**2259**	**0.20**	0.45	**10**	2662
50	60	0.73	1.27	**9**	**2343**	**0.06**	0.18	**9**	2643
50	70	0.14	0.37	**10**	**3107**	**0.09**	0.44	**10**	3764
50	80	0.19	0.36	**10**	**3588**	**0.04**	0.15	**10**	3919
50	90	0.42	0.68	10	**3596**	**0.19**	0.71	**9**	4516
50	100	0.12	0.26	**10**	**4391**	**0.02**	0.08	**10**	4995
60	50	0.48	0.72	11	**2460**	**0.05**	0.11	**10**	2944
70	50	0.46	0.66	11	**2533**	**0.13**	0.43	**10**	3658
80	50	0.22	0.58	12	**2864**	**0.11**	0.28	**11**	4810
90	50	0.37	0.52	**11**	**2910**	**0.26**	0.65	**11**	5435
100	50	0.49	1.02	12	**3460**	**0.07**	0.15	**11**	7197

5.1 Computational Results

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. All test runs have been executed on an Intel Xeon E5-2640 v4 with 2.40GHz machine.

Initially, we determined the parameter configurations of standalone variants of the VNS and the PBIG, in which it is naively assumed that users can evaluate all intermediate solutions. The parameters have been determined with irace [8] on a separate set of benchmark instances and have been tuned with the goal to minimize the number of iterations it takes the metaheuristics to generate near optimal solutions, i.e., solutions with an optimality gap of 0.5%. For the VNS we obtained to best use shaking neighborhoods $k \in \{1, 2\}$, for PBIG $k \in \{1, \dots, 10\}$ with a population size of 100. The termination criteria of the metaheuristics have been chosen according to the number of iterations that were necessary on average to find the close-to-optimal solutions, which was 60 iterations without improvement for the VNS, and 300 iterations (or three generations) without improvement for PBIG. Table 1 shows a comparison of the standalone variants of the metaheuristics using above parameter configurations. The table shows for each instance group with n locations and m users the average optimality gap ($\overline{\%\text{-gap}}$) and their corresponding standard deviation ($\sigma_{\%\text{-gap}}$) of the metaheuristics. Moreover, the table also shows the iteration in which the best solution has been found on average ($\overline{n_{\text{it}}^{\text{best}}}$) and the median of the total computation times (t[s]).

PBIG produces slightly better optimality gaps but also needs significantly more time than the VNS. Moreover, it takes PBIG much more iterations to find the best solution as opposed to the VNS.

Next, we tested COA in conjunction with the VNS (denoted as COA[VNS]) and the PBIG (denoted as COA[PBIG]), respectively, as OC. Further tests have shown that COA[PBIG] yields slightly better results when using the so far best found solutions stored in the SMC as initial population. In case there are not enough solutions available, the remaining solutions are generated by the randomized construction heuristic. The other parameters remain unchanged. The COA terminated after five iterations without improvement or after two hours. The comparison can be seen in Table 2. The table shows again the average

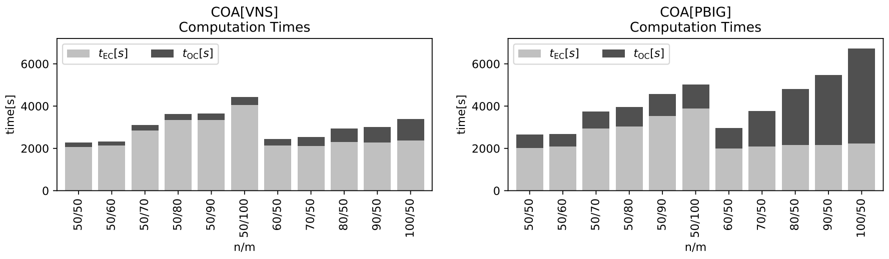

Fig. 2: Median computation times of the COA components for each instance set

optimality gaps ($\overline{\text{\%-gap}}$) and their corresponding standard deviations ($\sigma_{\text{\%-gap}}$). Moreover, the table also shows the average number of COA iterations ($\overline{n_{\text{it}}}$) and the median of the total computation times (t[s]). While the optimality gaps in Table 1 are somewhat comparable between the VNS and the PBIG, COA[PBIG] clearly outperforms COA[VNS] w.r.t. the optimality gaps. This difference can be explained by the number of solutions returned by the VNS and the PBIG. While the OC of COA[VNS] only returns one solution in every COA iteration, the OC of COA[PBIG] returns 100 solutions in every iteration. A higher number of solutions in the SMC results in more diversified solutions not only in the FC but also in the EC. Hence, the accuracy of the surrogate function increases w.r.t. larger areas of the search space, whereas for less diversified training data the accuracy of the surrogate function usually only increases in a small part of the search space. Moreover, the high number of shakings in the PBIG additionally increases the diversity of the solutions returned by the OC. Note however that COA[PBIG] does not scale so well w.r.t. computation times, especially for an increasing number of locations. The reason for this is the generous termination criterion of the PBIG in comparison to the VNS, since it takes the PBIG much more time to find a near optimal solution.

Finally, Figure 2 shows the computation times of the individual components of the COA framework. The total computation time is primarily split between the EC and the OC while the FC has barely any impact. Moreover, the figure also shows that the computation time of the EC mainly depends on the number of users, while the computation time of the OC primarily scales with the number of service point locations. Finally, Figure 2 also shows large differences in computation times between COA[VNS] and COA[PBIG].

6 Conclusion and Future Work

We considered a Cooperative Optimization Approach (COA) for distributing service points within a geographical area in mobility applications under incomplete information. Instead of estimating user demands upfront, our method directly incorporates potential customers in the optimization process. In this contribution we specifically focused on the optimization part of COA. We could show that for our instances a variable neighborhood search (VNS) as well as a population based iterated greedy algorithm (PBIG) reliably generate near optimal solutions in short time. However, within COA, not quality matters: PBIG is naturally able to return multiple different high quality solutions that can all be further exploited. While COA[PBIG] has the edge over COA[VNS] w.r.t. to optimality gaps, COA[PBIG] does not scale as well as COA[VNS] for instances with a large number of service point locations.

In future work, we aim at adapting COA to also work for larger instances in a reasonable of time. Alternative models will be considered for the surrogate function, and we plan to change the optimization core (OC) from a black-box optimization to a white-box or at least a gray box model.

References

1. Bouamama, S., Blum, C., Boukerram, A.: A population-based iterated greedy algorithm for the minimum weight vertex cover problem. Applied Soft Computing **12**(6), 1632–1639 (2012)
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. 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)
4. 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). https://doi.org/10.1007/978-3-030-16711-0
5. 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)
6. 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)
7. 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)
8. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
9. 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)
10. Mladenović, N., Hansen, P.: Variable neighborhood search. Computers & Operations Research **24**(11), 1097–1100 (1997)