A Bayesian Optimization Approach for Tuning a Genetic Algorithm Solving Practical-Oriented Pickup and Delivery Problems

Cornelius Rüther
Department of Operations Research
University of Hildesheim
Hildesheim, Germany
ruether@bwl.uni-hildesheim.de

Julia Rieck
Department of Operations Research
University of Hildesheim
Hildesheim, Germany
rieck@bwl.uni-hildesheim.de

***Abstract*—The Multi Depot Pickup and Delivery Problem with Time Windows and Heterogeneous Vehicle Fleets combines many real-world constraints and is therefore a strongly practical-oriented routing problem. Due to its complexity, a solution approach with sufficiently good quality has to include several operators which are called with certain probability. However, the parameters modelling the probabilities strongly depend on the data structure of given instances. Thus, selecting the best parameter configuration for each given instance is a natural problem to enhance the overall solution quality. We present a Bayesian Optimization approach with Gaussian Processes in order to optimize the parameter configuration of a Grouping Genetic Algorithm. The parameter tuning is evaluated on five data sets with different numbers of depots and 1,200 problem instances each. It is shown that the parameter tuning approach is a good choice in improving the performance of the considered meta-heuristic over all instances in each data set. In addition, the best parameter configuration per problem class with the same characteristics is able to improve both the frequency of finding the best solution as well as the relative error to this solution significantly.**

***Note to Practitioners*—In the field of Less-Than-Truckload (LTL) transportation, goods have to be forwarded on the same vehicle from one customer to another. In practice, several restrictions are of interest when transportation requests are executed. The Multi-Depot Pickup and Delivery Problem with Time Windows and Heterogeneous Vehicle Fleets (MDPDPTWHV) considered in this paper is a routing problem in the LTL-field including many practical constraints. In order to solve instances of such a problem, a sufficiently good meta-heuristic can be applied. However, the variety of constraints makes it hard to solve problem instances of different data structure regarding comparative (solution) quality. In particular, the quality depends on the interaction of the meta-heuristic's operators, which can be controlled by configuration parameters. Thus, customizing these parameters for certain data structures of different problem instances can enhance the solution quality and consequently produce reasonable results for complex routing problems such as the MDPDPTWHV. In the paper at hand, a sophisticated Genetic Algorithm is presented at first in order to solve the considered routing problem. Secondly, a solution approach is introduced which finds sufficiently good configuration parameters of the Genetic Algorithm's operators for a given data structure so that the problem instances which are of this structure can be solved efficiently. This approach can be adopted to any meta-heuristic which controls its operators by configuration parameters. Nevertheless, in order to find appropriate parameters, the approach has to be performed before the actual routing problem can be solved. This can be time-consuming, which is a clear limitation in the LTL-field. Consequently, as future research, the approach at hand can be used in order to find suitable parameters for predefined classes of data structures. New instances can then be classified to one class, from which the previously calculated configuration parameters are applied to the considered meta-heuristic.**

***Index Terms*—Bayesian Optimization, Gaussian Processes, Grouping Genetic Algorithm, Parameter Configuration**

I. Introduction

In Germany, almost all of the top 10 transportation companies (with respect to their revenue) operate in the Less-Than-Truckload (LTL) sector in which several transportation requests are transported together in one truck. Usually, the freight of each request has to be forwarded from an origin (pickup customer) to a destination (delivery customer). At each customer location, the corresponding goods have to be loaded by employees. As this is a time-sensitive process due to the availability of employees and loading ramps, it is important to consider time windows in order to reduce waiting times. Since different types of goods are transported, e. g., palletized goods, but also lattice boxes or cable reels, LTL carriers have to maintain different vehicle types with regard to capacity, speed or even pollution emission. Large carriers typically have several depot locations, so that a multi-depot problem must be taken into account. Vehicles are available at each depot, from which they start and end their routes [1].

The described problem can be modelled as a Multi-Depot Pickup and Delivery Problem with Time Windows and Heterogeneous Vehicle Fleets (MDPDPTWHV). Due to the problem's complexity, an extensive solution approach must be applied to achieve good solution quality (see Subsect. VII-C). To do so, the Grouping Genetic Algorithm (GGA) presented in [1] is adopted and extended by new mutation and repair operators (see Sect. V).

This work was supported by the BMBF project "L2O: Learning to Optimize" [grant number 01IS20013A].

However, since the solution quality of a stochastic algorithm strongly depends on the *correct* combination of the executed operators, configuring the parameters that control the probabilities for the operator selection in the GGA is essential. The automatic optimization of these parameter configurations is defined as Parameter Tuning Problem (PTP) (see Sect. II). An efficient configuration approach to tackle this problem is Bayesian Optimization (BO), which is a global optimization approach and a suitable method for parameter tuning due to its convergence properties [2]. In this paper, BO is implemented for tuning the parameters of the GGA by using Gaussian Processes (GP) as probabilistic model which builds a surrogate model for the function to be optimized and predicts the utility of evaluating unknown parameter settings.

The paper is structured as follows: First, a short introduction into the PTP is given in Section II, while Section III gives an overview of related approaches for parameter tuning. In order to describe the problem's complexity, the MDPDPTWHV is introduced in Section IV and the GGA for solving this problem is described in Section V in detail. The BO approach for tackling the parameter tuning is outlined in Section VI and its results are presented in Section VII. Finally, the paper is closed with a short discussion in Section VIII.

II. Parameter Tuning Problem

The Parameter Tuning (or Parameter Configuration) Problem is the determination of appropriate parameters for algorithms so that their effectiveness and efficiency is optimized [3]. The Parameter Tuning Problem can be described generally according to Eiben and Smit [4].

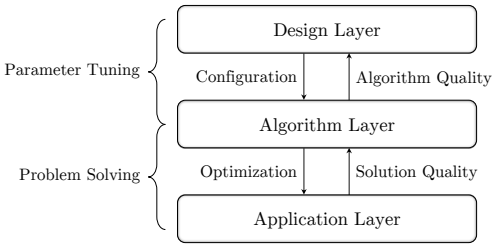

Fig. 1. Control flow (left) and information flow (right) through the different layers in parameter tuning [4]

As Fig. 1 shows, different layers of the problem structure are considered and connected with a control and an information flow. The *application layer* describes the optimization problem at hand (here the MDPDPTWHV as in Section IV), which is solved by an algorithm (*algorithm layer*, here, e. g., the GGA from section V). How this particular algorithm is designed with respect to parameter configuration is described in the *design layer*. The quality of the parameter configuration chosen for an algorithm is evaluated in terms of solution quality.

The PTP arises, among others, in the optimization of parameter settings for meta-heuristics [3]. Thus, many optimization methods such as a Genetic Algorithm (GA) depend on various known parameters that can be optimally chosen so that their performance can be improved in terms of finding good solutions [2]. Although it is obvious that the performance of a meta-heuristic depends on the configuration of their parameters, the configuration problem has not been formally addressed in scientific publications until the end of the last century [5]. According to [6], parameter selection while developing meta-heuristics is typically done manually, i. e., after evaluating different configurations, the one that gives the best result was chosen. The selection was often made according to *expert knowledge* about the optimization problem at hand. However, according to the No Free Lunch Theorem, there is no universal algorithm whose computations have the same solution quality for all optimization problems [7], so parameter tuning is by no means a one-time problem and it makes sense to automate it [3].

III. Related Work

From the end of the last century, a number of approaches have been developed to automate parameter configuration. Since the parameter tuning approach in this paper is designed for a GA, this overview focuses on approaches to optimize procedures of the Evolutionary Algorithm (EA) class. In [4], an overview of methods with respect to tuning parameters for EAs is given. A more general survey on automatic parameter tuning for meta-heuristics can be found in [3].

In [8], the Relevance Estimation and Value Calibration (REVAC), a population-based Estimation of Distribution Algorithm (EDA), is introduced for EAs. It is based on the principle of solving parameter tuning problems by estimating parameter relevance with normalised Shannon entropy. REVAC is an iterative algorithm starting with the assumption of a uniform distribution regarding the EDA and estimates the distributions of promising parameter values for each parameter within the configuration space. A different continuous black-box optimization method is Covariance Matrix Adaptation Evolution Strategy (CMA-ES) which is based on the concepts of self-adapting evolution strategies and is applied in [9]. In each of its iterations, a set of candidate configurations from a multivariate Gaussian distribution is sampled, whose co-variance matrix is adapted cumulatively. In [4], the authors compare REVAC and CMA-ES for tuning EA's parameter. Here, they combine the approaches also with racing, e. g., Iterated F-Race from [10]. In I/F-Race, a small set of parameter configurations is sampled according to a probabilistic model and executed on several instances. Thereby, parameters are discarded as soon as enough statistical evidence is gathered against them. A Bayesian Optimization method to tune the parameter configuration of an EA solving a flow-shop scheduling problem is introduced in [11]. The authors show that the approach is able to improve the efficiency of the meta-heuristic by optimizing six EA parameters considered.

IV. Mathematical Model for the Multi-Depot Pickup and Delivery Problem with Time Windows and Heterogeneous Vehicle Fleets

In order to get an impression of the problem structure for the Pickup and Delivery Problem at hand (see Sect. I), we intro-

duce a two-index model formulation. The main advantage of the model in comparison to a typical three-index formulation is the significant reduction of decision variables. This reduction enables to calculate solutions with exact solvers even for large, complex problem instances [1]. Moreover, the heterogeneity, as capacity and speed, is modelled with a concept based on real valued variables (cf. [12], [13]). To do so, we introduce capacity and speed variables $w_i, u_i \in \mathbb{R}^{\geq 0}$ which values are associated with the vehicle serving customer i and forwarded to the following customer on the same route $v_i \in \mathbb{R}^{\geq 0}$. For solvers like CPLEX, this formulation provides support since the relaxation of the problem is easier due to more real and less binary variables [14]. As a consequence compared to [1], we only model the decision whether an arc (i, j) is used or not with a binary variable x_{ij}. In detail, parameters and variables are listed in Table I.

TABLE I
VARIABLES AND PARAMETERS FOR A COMPACT TWO-INDEX MODEL FORMULATION OF THE MDPDPTWHV

indices and sets:	
$d \in \mathcal{D}$	set of depots, whereas each vehicle is modelled as an artificial depot with a start depot d_s and end depot $d_e = 2n+\delta+d_s$; δ is the number of depots (and vehicles) $\Rightarrow$ set of start depots $\mathcal{D}_s = \{0, \ldots, \delta-1\}$, set of end depots $\mathcal{D}_e = \{2n+\delta, \ldots, 2n+2\delta-1\}$
$i, j \in \mathcal{N}$	set of customer nodes, $\mathcal{N} = \mathcal{N}_p \cup \mathcal{N}_d = \{\delta, \ldots, 2n+\delta-1\}$
$i, j \in \mathcal{N}_p$	set of pickup locations, $\mathcal{N}_p = \{\delta, \ldots, n+\delta-1\}$
$i, j \in \mathcal{N}_d$	set of delivery locations, $\mathcal{N}_d = \{n+\delta, \ldots, 2n+\delta-1\}$
$i, j \in \mathcal{V}$	set of nodes, $\mathcal{V} = \mathcal{N} \cup \mathcal{D} = \{0, \ldots, 2n+2\delta-1\}$; in addition, we declare $\underline{\mathcal{V}} = \mathcal{D}_s \cup \mathcal{N}$ and $\bar{\mathcal{V}} = \mathcal{N} \cup \mathcal{D}_e$
parameters:	
$[a_i, b_i]$	time window at node $i \in \mathcal{N}$ in which the service has to start
c_{ij} (d_{ij})	variable costs (distance) for travelling from i to $j \in \mathcal{V}$; $c_{ii} = \infty$
d_i	delivery demand at customer $i \in \mathcal{N}$, where $d_i > 0$, $\forall i \in \mathcal{N}_d$, and $d_i = 0$, $\forall i \in \mathcal{N}_p$; note that $d_{i+n} = p_i$, $\forall i \in \mathcal{N}_p$ holds
f_d	fixed costs for using a vehicle associated with depot $d \in \mathcal{D}$
κ_d	capacity of vehicle at depot $d \in \mathcal{D}$
$\bar{\kappa}$	maximum capacity of all vehicles, i. e., $\bar{\kappa} = \max_{d \in \mathcal{D}}\{\kappa_d\}$
l	load factor for vehicle loading measured in time per quantity unit
l^{const}	additional loading time to model preparations before service
M	big M for linearization of time window constraints
ν_d	reciprocal speed of the vehicle at depot $d \in \mathcal{D}$
$\bar{\nu}$	max. reciprocal speed of all vehicles, i. e., $\bar{\nu} = \max_{d \in \mathcal{D}}\{\nu_d\}$
p_i	pickup demand at customer $i \in \mathcal{N}$, where $p_i > 0$, $\forall i \in \mathcal{N}_p$, and $p_i = 0$, $\forall i \in \mathcal{N}_d$
s_i	service time at node $i \in \mathcal{V}$ holding $s_i = l \cdot (d_i + p_i) + l^{\text{const}}$, $\forall i \in \mathcal{N}$, and $s_d = 0$, $\forall d \in \mathcal{D}$
decision variables:	
L_i	current load of the visiting vehicle after serving node $i \in \mathcal{V}$
T_i	beginning of the service at node $i \in \mathcal{V}$
v_i	route number on which customer $i \in \mathcal{V}$ is assigned
w_i	capacity of the vehicle which serves customer $i \in \mathcal{V}$
u_i	reciprocal speed of the vehicle which serves customer $i \in \mathcal{V}$
x_{ij}	binary variable which indicates whether a vehicle uses the arc between nodes i and $j \in \mathcal{V}$ ($x_{ij} = 1$) or not ($x_{ij} = 0$)

$$\min \quad \sum_{i,j \in \mathcal{V}} c_{ij} x_{ij} + \sum_{d \in \mathcal{D}_s} \sum_{j \in \mathcal{N}_p} f_d x_{dj} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in \mathcal{N} \cup \mathcal{D}_e} x_{ij} = 1 \qquad \forall i \in \mathcal{N} \tag{2}$$

$$\sum_{i \in \mathcal{N} \cup \mathcal{D}_s} x_{ij} = 1 \qquad \forall j \in \mathcal{N} \tag{3}$$

$$\sum_{j \in \mathcal{N}_p} x_{dj} \leq 1 \qquad \forall d \in \mathcal{D}_s \tag{4}$$

$$\sum_{j \in \mathcal{N}_p} x_{d_s j} - \sum_{i \in \mathcal{N}_d} x_{i d_e} = 0 \qquad \forall d_e, d_s \in \mathcal{D} \tag{5}$$

$$j x_{dj} - v_j \leq 0 \qquad \forall d \in \mathcal{D}_s, j \in \mathcal{N}_p \tag{6}$$

$$v_j - j x_{dj} \leq n(1 - x_{dj}) \qquad \forall d \in \mathcal{D}_s, j \in \mathcal{N}_p \tag{7}$$

$$v_i - v_j \leq n(1 - x_{ij}) \qquad \forall i \in \mathcal{N}, j \in \bar{\mathcal{V}} \tag{8}$$

$$w_d = \kappa_d \qquad \forall d \in \mathcal{D}_s \tag{9}$$

$$w_j - w_i \leq \bar{\kappa}(1 - x_{ij}) \qquad \forall i \in \underline{\mathcal{V}}, j \in \bar{\mathcal{V}} \tag{10}$$

$$u_d = \nu_d \qquad \forall d \in \mathcal{D}_s \tag{11}$$

$$u_j - u_i \leq \bar{\nu}(1 - x_{ij}) \qquad \forall i \in \underline{\mathcal{V}}, j \in \bar{\mathcal{V}} \tag{12}$$

$$v_i - v_{i+n} = 0 \qquad \forall i \in \mathcal{N}_p \tag{13}$$

$$w_i - w_{i+n} = 0 \qquad \forall i \in \mathcal{N}_p \tag{14}$$

$$u_i - u_{i+n} = 0 \qquad \forall i \in \mathcal{N}_p \tag{15}$$

$$v_{d_s} - v_{d_e} = 0 \qquad \forall d_s, d_e \in \mathcal{D} \tag{16}$$

$$w_{d_s} - w_{d_e} = 0 \qquad \forall d_s, d_e \in \mathcal{D} \tag{17}$$

$$u_{d_s} - u_{d_e} = 0 \qquad \forall d_s, d_e \in \mathcal{D} \tag{18}$$

$$T_i + s_i + d_{ij}\, u_i - T_j \leq M(1 - x_{ij}) \qquad \forall i \in \underline{\mathcal{V}}, j \in \bar{\mathcal{V}} \tag{19}$$

$$a_i \leq T_i \leq b_i \qquad \forall i \in \mathcal{V} \tag{20}$$

$$T_i + s_i + d_{i,i+n}\, u_i \leq T_{i+n} \qquad \forall i \in \mathcal{N}_p \tag{21}$$

$$L_i - d_j + p_j - L_j \leq \bar{\kappa}(1 - x_{ij}) \qquad \forall i \in \underline{\mathcal{V}}, j \in \bar{\mathcal{V}} \tag{22}$$

$$L_i \leq w_i \qquad \forall i \in \mathcal{V} \tag{23}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in \mathcal{V}, i \neq j \tag{24}$$

$$L_i, T_i, v_i, w_i, u_i \in \mathbb{R}^{\geq 0} \qquad \forall i \in \mathcal{V} \tag{25}$$

In the objective function (1), the variable costs and the fixed costs for the vehicles used are minimized. The constraints (2) and (3) ensure that each customer is served exactly once. Each vehicle may start at most once (4). Restrictions (5) model that each started vehicle must arrive at the corresponding end depot. The conditions (6) and (7) set the route index v_j to the index of the customer j served first. The unique route index v_j is passed by the inequalities (8) from the customer i to j if j directly follows i. The capacity variable w_d is set by the conditions (9) to the vehicle's capacity associated with depot $d \in \mathcal{D}$. This is passed from customer i to j by means of restrictions (10) if they are directly consecutive. The reciprocal speed u_i is set for the starting vehicles by restrictions (11). For a vehicle using arc (i, j), the reciprocal speed is passed from node i to j with constraints (12). Equations (13), (14), and (15) ensure the coupling of pickup and delivery on the same vehicle. Similarly, this is modeled for the corresponding start and end depots by (16), (17), and (18). If customer j directly follows customer i, the service start time at j must not precede the service end time at i plus the travel time between i and j due to (19). Further, the service start time has to start within the specified time window $[a_i, b_i]$ of customer i (20). The time precedence relationship between pickup and delivery nodes is

modeled using inequalities (21) through the service start at node i and $i+n$. Restrictions (22) ensure that the load L_j is correctly updated after the customer i is visited. In addition, (23) guarantee that the capacity of the vehicle associated to the served customer i is not exceeded.

Finally, the decision variable x_{ij} describing the transport over the arc (i,j) is defined as binary (24). All other variables like the time variables S_i, T_i, the loading variables L_i as well as the variables for the route indices v_i, the capacity variables w_i, and the (reciprocal) speed variables u_i are restricted to be non-negative real numbers (25).

V. Grouping Genetic Algorithm Framework

In the literature, Genetic Algorithms are known to give good results for complex routing problems [15], [16]. The Grouping Genetic Algorithm introduced in [1] for the MDPDPTWHV achieves promising results. Here, the authors show that the considered GA variants with population management regarding general replacement with elitism promise a good solution quality for the presented problem. However, the complexity of the problem instance's data structure can be various. Hence, in this paper, the GGA is extended by new mutation and repair operators in order to investigate which combination of operators is suitable for a given data structure. To do so, the GGA's general replacement version with binary tournament selection is used. An overview of the procedure of the GGA is given in Fig. 2. For all terms and concepts of Genetic Algorithms, we kindly refer to [17].

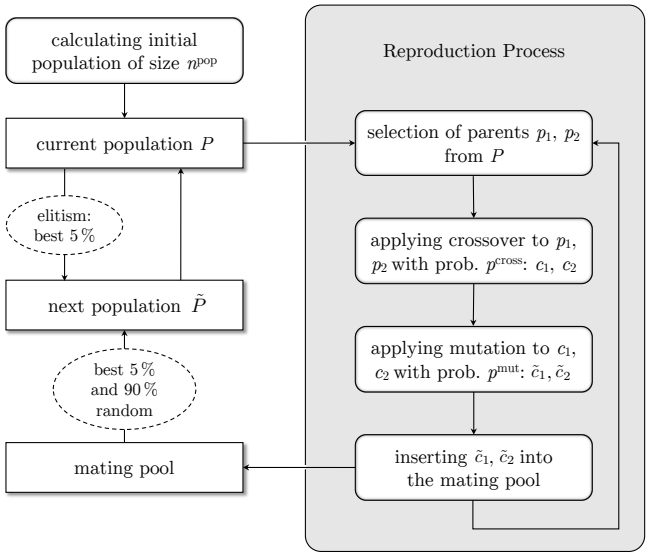

Fig. 2. Grouping Genetic Algorithm framework with its implemented population management: general replacement with elitism

In general, the presented GGA proceeds as follows: Initially, a population P of n^{pop} individuals is determined. We assume that each individual has a fitness value that is equal to the sum of variable and fixed costs of the vehicle routing solution as described in objective function (1) as well as a penalty term for non-served requests, which has to be minimized in the proposed GA variant and can be written as:

$$\sum_{i,j\in\mathcal{V}} c_{ij}x_{ij} + \sum_{d\in\mathcal{D}_s}\sum_{j\in\mathcal{N}_p} f_d x_{dj} + \gamma \sum_{i\in\mathcal{N}_p}\Big(1 - \sum_{j\in\mathcal{N}} x_{ij}\Big)\,. \quad (26)$$

In general for the penalty term in (26), the weight γ is to be chosen in such a way that the consideration of non-served request results in a worse individual than the worst possible solution [18]. Therefore, we set γ as follows:

$$\gamma := 2 \cdot \max_{i,j\in\mathcal{V}}\{c_{ij} + c_{ji}\} + \max_{d\in\mathcal{D}}\{f_d\}\,.$$

Each individual is computed by using several sequential double insertion heuristics that create routes one after another by inserting customers at their best possible position. In order to enhance the diversity of the initial population, we generate 25 % of the individuals with *Best Insertion* that starts the route with one request and inserts the best corresponding request in each iteration, 50 % with *Random Insertion* which chooses a random request for insertion in every iteration (cf. [1]) and 25 % with a *Regret-k* heuristic (cf. [19]) which calculates a regret value to evaluate how much a greedy insertion will cost in the end and selects the request that has the highest costs (see Eq. (28) in Subsect. V-E). A high level of diversity is achieved, as we do not allow duplicates within the population.

As long as the maximum number of generations $\gamma^{\max}$ has not been met, two parents p_1, p_2 for generating offsprings are selected (cf. Subsect. V-B). With a probability of p^{cross}, one crossover operator variant is applied to p_1 and p_2 as in [1] in order to create two children c_1 and c_2 (cf. Subsect. V-C). Moreover, each child is modified by using one of the mutation operators with probability p^{mut} and so two mutated children $\tilde{c}_1$ and $\tilde{c}_2$ are generated (see Sect. V-D). In case no operator is applied, the created offsprings are just clones of their ancestors. Generated individuals are stored in a mating pool from which the next population is selected. Finally, the best individual regarding its fitness is returned. Please note that $n^{\text{pop}}, \gamma^{\max}, p^{\text{cross}}$ and p^{mut} are parameters that are set manually.

In general, the most critical part of a GA is to find a good solution representation or genotype encoding, respectively, such that crossover and mutation operators do not have to be too complex as well as the decoding from genotype to phenotype is not too time expensive. This is why the approach of [1] is implemented as a Grouping GA in which each vehicle is represented by a gene and the pickup and delivery customers are grouped to their request index. A genotype encoding then stores the assignment of a request to the executing vehicle and can be represented by a vector of integers.

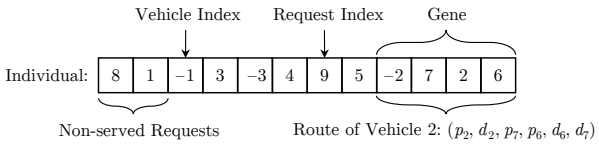

Fig. 3. Genotype Encoding for the Grouping Genetic Algorithm

As demonstrated in Fig. 3, vehicles are represented by negative index values and all subsequent positive integers characterize the requests served by the corresponding vehicle. This encoding implies that the route for each vehicle (e. g., route 2 in Fig. 3) has to be calculated, whenever the genotype decoding is necessary, e. g., for fitness value determination. Since this is a time-consuming procedure, which is with respect to the applied heuristic not necessarily deterministic, we also store the corresponding phenotype of an individual (cf. [1]), i. e., the vehicle routing solution. In this way, good features of an individual are less likely to be removed from the phenotype, when applying crossover or mutation operators. Please note that the phenotype must always be modified together with the genotype during the solution process.

The following Subsections V-A–V-E describe in detail the operators extended compared to [1] and whose control parameters are to be optimized, i. e., mutation and repair operators. The selection and crossover operators as well as the population management are adopted from the GGA of [1].

A. Population Management

Due to the promising results of the GGA with general replacement as population management [1], we have also implemented this variant in the framework at hand. Hence, a mating pool of 1.5 times the generation size is filled with children, whereas duplicates are prevented. To preserve the best solutions from the last generation, elitism is applied by copying the best 5 % of the generation into the next one. Then, the best 5 % of individuals from the mating pool are selected and the rest of the next generation is filled with randomly chosen individuals from the mating pool (cf. Fig. 2).

B. Selection

In order to select parents without duplicates for offspring generation, binary tournament is used, since this selection approach has provided most promising results. Here, two individuals, called a *tournament*, are chosen randomly from the population and the best individual is chosen as first parent. The second parent is selected analogously.

C. Crossover Operator

The implemented group-oriented crossover variants are adopted from the GGA of [1]. A simplified illustration of how the crossover works for a representation with nine requests distributed over three possible vehicles is given in Fig. 4. Two crossover points (both of them between two genes) are randomly chosen for parent 1. Then, we transfer with a probability of 50 % all genes between the two crossover points (inner genes) and otherwise the genes outside of them (outer genes) at a randomly chosen insertion point into parent 2 to generate a child. This requires double-served requests and double-used vehicles coming from parent 2 to be deleted from the child. In doing so, non-served requests have to be re-inserted into the child by repair operators (see Subsect. V-E). In Fig. 4, the inner genes are not transferred into the child. The outer genes provide that vehicles 1 and 2 have to be eliminated from the child. In addition, requests 2 and 7 must be removed. Finally, the requests 5 and 9 remains and have to be re-inserted by a repair operator again. For detailed explanation, we kindly refer to [1].

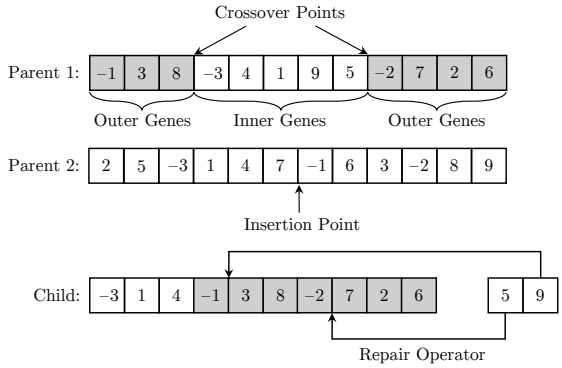

Fig. 4. Crossover operator when transfering genes from parent 1 to parent 2

D. Mutation Operators

Contrarily to [1], the mutation operators are basically classified in *vehicle-based* and *request-based* mutation. The selection of the operator class is done by a specific probability value $p^{\text{choice}}(n)$ that models the likelihood for choosing request-based mutation in generation n:

$$p^{\text{choice}}(n) = 0.1 \cdot \exp\left(\ln(8)\,\frac{n}{\gamma^{\max}}\right) .$$

In this way, the value p^{choice} prefers vehicle-based operators in the beginning of the Genetic Algorithm and request-based operators in the end.[1] In this manner, the trade-off is controlled between solution quality and computational time, since the vehicle-based mutation has a faster and broader investigation in the search space, while request-based mutation has a slower and more accurate search behaviour, which in general describes *exploration vs. exploitation*. Please note that due to the choice of probability p^{choice}, vehicle-based and request-based mutation are not equally weighted.

The vehicle-based mutation is done through selecting a vehicle index and deleting the corresponding vehicle from the genotype and phenotype. All requests served by the erased vehicle have to be re-inserted, which is done by the a repair operator (see Subsect. V-E), such that the solution holds all constraints, i. e., possibly a new vehicle has to be introduced. A simplified scheme of the mutation operator is given in the top of Fig. 5. As in [1], we use the following four vehicle selection mechanisms to tackle the problem of heterogeneity: **Costs/Number-of-Request Ratio.** For each vehicle, the ratio of route costs and the number of served requests is calculated and used to generate a roulette wheel. By spinning the roulette wheel, a vehicle index is selected, i. e., expensive routes with a small number of requests are preferred for removal.

[1]The value p^{choice} starts at 0.1 in the first generation $n = 0$ and ends at 0.8 in the last generation $n = \gamma^{\max}$.

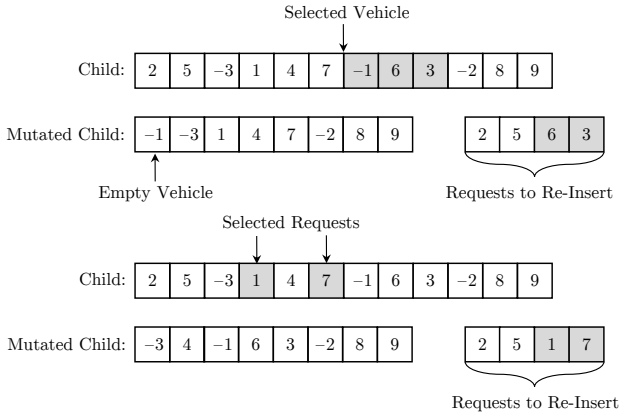

Fig. 5. Mutation operators and mutated childs to be repaired (using an additional empty vehicle if needed) with vehicle-based mutation (top) and request-based mutation (bottom)

Number of Requests. The vehicle with minimum number of requests is used for deletion.

Random Vehicle Index. Here, a random vehicle index is selected, thus all vehicles are equally likely to be chosen.

Random Genotype Position. This variant chooses a random position within the genotype and selects the corresponding vehicle index. Hence, vehicles with many requests are preferred.

Request-based mutation is a newly added concept. Particularly, if instances have many requests per vehicle or if in the course of the search the solution space must be examined more precisely for intensification, it is useful to remove individual requests from the genotype or phenotype. Basically, a number of requests to be released is randomly determined and then eliminated from the individual with one of the mechanisms described below. A simplified procedure is shown in the bottom of Fig. 5.

Historical Request Pair. This concept is adopted from the approach of [20] and uses a matrix $(h_{ij})_{i,j\in\mathcal{R}}$ of weights for all requests $\mathcal{R}$. The matrix stores how promising a combination of two requests on the same vehicle is. To do so, the elite is evaluated after each generation and a matrix element h_{ij} is increased by one if the requests i and j are served together on one vehicle. The matrix $(h_{ij})_{i,j\in\mathcal{R}}$ can be interpreted as long-term memory. Since combinations with high quality in a certain generation do not necessarily have to be globally optimal, the matrix is adjusted each generation through reducing h_{ij} by a factor $\tau \in (0,1)$, i. e., $h_{ij} \coloneqq \tau h_{ij}$. To determine the requests to be eliminated, a score σ_i is specified in which h_{ij} values are summed up if request j is served with i on the same vehicle in the considered individual. Requests with low σ_i are non-promising combinations on the same vehicle and consequently to be removed from the individual.

Similarity Measure. The variant is derived from the similarity measure based removal operator of [19]. Here, the first request is chosen randomly and additionally those requests are removed which are *similar* with respect to the measure $\tilde{\rho}_{ij}$ from Eq. (27), i. e., have the smallest values. Please note that $i, j \in \mathcal{R}$ and request i has pickup customer i and delivery customer $i+n$ (see Sect. IV).

$$\begin{aligned}\tilde{\rho}_{ij} =& \alpha_1(c_{ij} + c_{i+n,j+n}) + \alpha_2(|a_i - a_j| + |a_{i+n} - a_{j+n}|) \\ &+ \alpha_3(|b_i - b_j| + |b_{i+n} - b_{j+n}|) + \alpha_4(|p_i - p_j|) \quad (27)\end{aligned}$$

Thus, requests are defined as similar whose pickup and delivery nodes, start and end time windows, and demands to be transported are close to each other, since for these the exchange to another vehicle offers potential for direct insertion and therefore improvement of the fitness value. Here, for comparability, all values are scaled to the interval $[0, 1]$. The weights $\alpha_k, k = 1, \ldots, 4$, are element of $\mathbb{R}^{\geq 0}$.

We also apply a so-called *swap operator* with a probability of 20 %, after one of the six mutation variants is taken into account. This operator tries to swap used and free vehicles to minimize the sum of fixed costs. In order to keep the search broad within the solution space, the choice of vehicles to be swapped is controlled by a roulette wheel.

E. Repair Operator

When applying crossover or mutation operators, the individual has often to be repaired due to removed requests. During repair, requests need to be re-inserted. Here, we use several insertion methods that determine the best insertion positions for each request with respect to a cost value. Contrarily to [1], we implemented *Regret-k* insertion beside the greedy insertion. The approaches are quite similar to the double insertion heuristics used for generating the initial population. However, the proposed repair operators are parallel insertion heuristics, i. e., the possible insertion positions are evaluated for all vehicles being used. If there are requests that cannot be served by the available vehicles, a new (empty) vehicle has to be introduced whenever feasible.

Greedy Insertion. For each request, the insertion costs for all possible positions in each vehicle are calculated and the position with the lowest costs is chosen (cf. [1]).

Regret-k Insertion. For each request r, the difference is calculated of how much worse it is to insert a request in the k-th best vehicle v_k instead of the best one v_1, while the differences from the best to the k-th best vehicle are summed up. The request $\tilde{r}$ with the maximum value is supposed to be inserted into its best position in the current iteration, since later on the insertion costs increase (cf. [19]). Formally, this can be written as in (28):

$$\tilde{r} = \arg\max_r \left\{ \sum_{l=1}^{k} (\Delta_r^{v_l} - \Delta_r^{v_1}) \right\}. \quad (28)$$

The Regret-k insertion is applied for $k = 2, 3, 4, \delta$.

VI. Bayesian Optimization for Tuning the Parameters of the Grouping Genetic Algorithm

A common drawback of many parameter tuning approaches for meta-heuristics (see, e. g., the methods in Section III) is the considerable number of evaluations of the objective function required to determine the best parameter configuration [11]. As the performance of a Genetic Algorithm is sensitive to

the parameter setting and the structure of a problem instance, the evaluation of specific parameter configurations can be very time-consuming. Moreover, optimizing the parameters of meta-heuristics is a *black-box* problem, since the relationship between the parameter configurations of the algorithm and its performance cannot be measured metrically [3]. Therefore, *Bayesian Optimization* is a reasonable choice as a parameter tuning algorithm, since this method manage to cope with few evaluations of the objective function to be optimized, which also need not be known [11], [21]. Furthermore, BO is well applicable to functions defined in a solution space with dimension $D \leq 20$ (number of parameters) and it tolerates noise with respect to the function evaluation [22], which is a typical behaviour for stochastic methods like a GA. Last but not least, the results of [11] show that BO is in general a good approach in solving the Parameter Tuning Problem for EAs. For a detailed overview of Bayesian Optimization, the reader is kindly referred to [21]–[23].

Let $\varphi : \mathbb{X} \to \mathbb{R}$ be an unknown *black-box function* that cannot be modelled in closed form, i. e., for an argument $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^D$, the function value $\varphi(\boldsymbol{x})$ can only be determined by evaluating φ. In the Parameter Configuration Problem, φ describes the *utility* of a parameter vector $\boldsymbol{x} \in \mathbb{X}$, which can be, e. g., the mean of the objective values over all considered instances. The goal of Bayesian Optimization is to find a configuration $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}\in\mathbb{X}} \varphi(\boldsymbol{x})$ describing the global minimum of φ. The method iteratively learns a probabilistic model that estimates the utility function φ by known function values $\varphi(\boldsymbol{x})$ of different points $\boldsymbol{x} \in \mathbb{X}$ in the parameter space. For this, an *a-priori probability distribution* $P(\varphi)$ over the utility function φ and an *acquisition function* $a_{P(\varphi)} : \mathbb{X} \to \mathbb{R}$ that quantifies the utility of evaluating the function φ at each parameter configuration $\boldsymbol{x}$ are required [24]. Let n data points in the set $\mathcal{D}_n = (\boldsymbol{x}_i, y_i)_{i=1,\dots,n}$ with $y_i = \varphi(\boldsymbol{x}_i)+\varepsilon_i$ be known for function φ. Then, Bayesian Optimization iteratively repeats the following three steps [21]:

1) Find the next configuration $\boldsymbol{x}_{n+1}$ whose evaluation is most promising, i. e., which maximizes the acquisition function under the condition of the known data points $\mathcal{D}_n$:
$$\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x}\in\mathbb{X}} a_{P(\varphi|\mathcal{D}_n)}(\boldsymbol{x})\,.$$
2) Determine $y_{n+1} = \varphi(\boldsymbol{x}_{n+1}) + \varepsilon_{n+1}$ with possible noise ε_{n+1} and add $(\boldsymbol{x}_{n+1}, y_{n+1})$ to the previous data points: $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{(\boldsymbol{x}_{n+1}, y_{n+1})\}$.
3) Update the probability model $P(\varphi \mid \mathcal{D}_{n+1})$ and consequently the acquisition function $a_{P(\varphi|\mathcal{D}_{n+1})}$.

Here, the update of the probabilistic model is done in step 3) via the *Bayes' Theorem* (29), where

$$P(\varphi \mid \mathcal{D}_{n+1}) \propto P(\mathcal{D}_{n+1} \mid \varphi)P(\varphi)\,. \tag{29}$$

The theorem states that the *posteriori* probability distribution $P(\varphi \mid \mathcal{D}_{n+1})$ for the new data points $\mathcal{D}_{n+1}$ is proportional to the a-priori model $P(\varphi)$ multiplied by the similarity of the data points $\mathcal{D}_{n+1}$ to the assumed model for φ.

In condition (29), $P(\mathcal{D}_{n+1} \mid \varphi)$ consequently expresses how likely the data is under the model assumptions we think we know a-priori. If the assumption is that the utility function φ is very smooth and noise free, data with high variance should be recognized as less likely than data that hardly deviate from the mean [21].

The solution quality of Bayesian Optimization significantly depends on the choice of the a-priori probability distribution (see Subsect. VI-A) and on the evaluation of the utility by a surrogate model, i. e., the acquisition function (see Subsect. VI-B). Both components are described in the following subsections.

A. Gaussian Processes

Possible models for a-priori distributions are *Gaussian Processes*, *Random Forests* or also *Bayesian Neural Networks* [24]. Since Gaussian Processes are most often chosen in the literature [22] and, moreover, give promising results as in [11], [24], [25], a Gaussian Process (GP) is used in the presented Bayesian Optimization approach. In the following, we provide some insights into the GP used here. For a detailed description of Gaussian Processes, the reader is kindly referred to [26].

A Gaussian Process describes a stochastic process which considers a probability distribution over functions. In principle, a GP can be understood as a Gaussian distribution in infinitely many dimensions, which defines a multivariate Gaussian distribution in any finite set of dimensions [11]. Thus, a GP represents the natural extension of a multidimensional Gaussian distribution. The Gaussian Process is uniquely determined by a *mean function* $\mathbb{E}[\varphi(\boldsymbol{x})] = m(\boldsymbol{x})$ as well as a *covariance function* $\mathrm{Cov}[\varphi(\boldsymbol{x}), \varphi(\boldsymbol{x}')] = k(\boldsymbol{x}, \boldsymbol{x}')$ (also called *kernel*) over the utility function φ with $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{X}$ [26]. Hence, the kernel contains information about the correlation between function values $\varphi(\boldsymbol{x})$ and $\varphi(\boldsymbol{x}')$ of two parameter configurations [11]. A Gaussian Process is written as:

$$\varphi(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}'))\,. \tag{30}$$

The covariance function is used to determine how the known data points $\mathcal{D}_n$ affect the prediction of the function value $\varphi(\hat{\boldsymbol{x}})$ for new points $\hat{\boldsymbol{x}} \in \mathbb{X} \setminus \mathcal{D}_n$ [24]. Consequently, there are many ways to define the covariance function depending on the model assumptions that are made. A detailed overview can be found in [26], [27].

A typical covariance function is given by the *Automatic Relevance Determination* (ARD) *Squared Exponential* (SE) kernel function and can be described as

$$k_{\mathrm{SE}}(\boldsymbol{x}, \boldsymbol{x}') = \theta_0 \exp\left(-\frac{1}{2} r^2(\boldsymbol{x}, \boldsymbol{x}')\right) \tag{31}$$

with $r^2(\boldsymbol{x}, \boldsymbol{x}') = \sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\theta_d^2}$. The parameters $\theta_d, d = 1, \dots, D$, are scalars and scale the summands in $r^2(\boldsymbol{x}, \boldsymbol{x}')$ and thus express the relevance of the data in each dimension for the output, while θ_0 describes the amplitude [11].

However, this kernel (31) models a very smooth utility function [28] and is thus not suitable for optimizing the parameter configuration of the GGA from section V. For the

problem at hand, it can be assumed that the utility function φ is non-smooth. This is because, different function values $\varphi(\boldsymbol{x})$ can be obtained due to the uniform distributed randomly chosen operators, even if φ is evaluated several times with the same parameters $\boldsymbol{x}$. Kernel functions coming from the class of ARD *Matérn* kernels are more appropriate for the Parameter Tuning Problem presented, since they model less smooth functions [28]. In the BO approach used in this paper, the ARD Matérn $\frac{5}{2}$ kernel is applied, which is defined as follows:

$$k_{\text{M52}}(\boldsymbol{x}, \boldsymbol{x}') = \theta_0 \left(1 + \sqrt{z} + \frac{z}{3}\right) \exp\left(-\sqrt{z}\right) \qquad (32)$$

with $z = 5r^2(\boldsymbol{x}, \boldsymbol{x}')$, while the function $r^2(\boldsymbol{x}, \boldsymbol{x}')$ is defined as in (31). Since the influence of the individual parameters on the solution quality is not known a-priori, we set $\theta_d = 1$ for all $d = 0, \ldots, D$ within $r^2(\boldsymbol{x}, \boldsymbol{x}')$ in the presented approach.

Due to the stochastic nature of the GGA to be optimized, the solution for a given parameter configuration is expected to be subject to noise [3]. To model this noise, the kernel function (33) is used additionally [26]:

$$k_{\text{Noise}}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} \sigma_n^2 & , \boldsymbol{x} = \boldsymbol{x}' \\ 0 & , \text{sonst} \end{cases} . \qquad (33)$$

The parameter σ_n^2 describes the variance of the modelled noise. Altogether, the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$ considered for the Gaussian Process (30) in our BO approach is composed of (32) and (33) (cf. defining GP kernels in [26]):

$$k(\boldsymbol{x}, \boldsymbol{x}') = k_{\text{M52}}(\boldsymbol{x}, \boldsymbol{x}') + k_{\text{Noise}}(\boldsymbol{x}, \boldsymbol{x}') .$$

In order to design the choice of the next parameter configuration to be evaluated in step 1) purposeful, a reasonable acquisition function must be defined. This function makes use of the model assumed by the Gaussian Process to predict the utility for evaluating certain parameter settings $\boldsymbol{x} \in \mathbb{X}$.

B. Acquisition Function

The acquisition function $a_{P(\varphi)}$ is designed to estimate, based on the current data $\mathcal{D}_n$, which areas of the parameter space $\mathbb{X}$ to investigate in order to improve the model's prediction of the Gaussian Process. Therefore, the acquisition function should be constructed to achieve a high value at parameters $\boldsymbol{x} \in \mathbb{X}$, where the predicted utility function value $\varphi(\boldsymbol{x})$ is low or the prediction's quality is poor, i. e., there is too little information of that part of the considered parameter space available for a sufficiently accurate estimation [21]. These two cases describe the classical problem of *exploitation vs. exploration*. This is tackled in several acquisition functions that can be found in the literature, such as *Probability of Improvement*, *GP Upper Confidence Bound*, and *Expected Improvement* (EI) [28]. Since EI automatically controls the balance between exploitation and exploration, and achieves sufficiently good results in BO approaches [11], it is also used in the approach at hand. With respect to the currently observed data $\mathcal{D}_n$, the acquisition function with EI is defined as:

$$a^{\text{EI}}_{P(\varphi|\mathcal{D}_n)}(\boldsymbol{x}) = (\varphi(\boldsymbol{x}^*) - m(\boldsymbol{x}))\Phi(z) + \sigma(\boldsymbol{x})\phi(z) \qquad (34)$$

with $z = \frac{\varphi(\boldsymbol{x}^*) - m(\boldsymbol{x}) - \xi}{\sigma(\boldsymbol{x})}$, where $\boldsymbol{x}^*$ is the best parameter configuration so far, i. e., $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathcal{D}_n} \varphi(\boldsymbol{x})$, and $\sigma^2(\boldsymbol{x})$ describes the predicted variance $k(\boldsymbol{x}, \boldsymbol{x})$. The function ϕ represents the density function and Φ the cumulative distribution function of the normal distribution $\mathcal{N}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}))$. By the acquisition function $a^{\text{EI}}_{P(\varphi|\mathcal{D}_n)}$ in (34), $\boldsymbol{x} \in \mathbb{X}$ are preferred if $m(\boldsymbol{x}) < \varphi(\boldsymbol{x}^*)$, i. e., $\boldsymbol{x}$ could reduce the utility function value, or if there is a high variance $\sigma^2(\boldsymbol{x})$, i. e., the utility function value for $\boldsymbol{x}$ cannot be estimated with sufficient quality based on the current data $\mathcal{D}_n$. To influence exploitation and exploration in addition to the intrinsic control by the two summands of (34), a parameter $\xi \geq 0$ can be selected. The larger ξ is, the more exploration is preferred. Therefore, it is recommended to start with a sufficiently large ξ and to let it approach to zero over the Bayesian Optimization process [21].

To approximate the best parameter configuration determined by the acquisition function $\boldsymbol{x}_{n+1} = \arg\max_{\boldsymbol{x} \in \mathbb{X}} a^{\text{EI}}_{P(\varphi|\mathcal{D}_n)}(\boldsymbol{x})$, which has to be evaluated next for φ, $a^{\text{EI}}_{P(\varphi|\mathcal{D}_n)}$ is evaluated on a sufficiently large number of random parameter configurations $\boldsymbol{x} \in \mathbb{X}$. The best configuration is chosen as $\boldsymbol{x}_{n+1}$.

VII. Evaluation

In this section, the results of the Bayesian Optimization approach regarding the parameter configuration for the Grouping Genetic Algorithm are presented. In doing so, the most promising GGA variant of [1] (described in Section V) is compared to a state-of-the-art algorithm developed to solve a related Pickup and Delivery Problem, the Adaptive Large Neighbourhood Search (ALNS) from [19]. In addition, an optimization approach which randomly selects several parameter configurations and chooses the parameters with best fitness values of the GGA is applied in order to show that the BO performs the optimization of the parameter configuration in a goal-oriented manner.

At first, it is described how the benchmark data sets are created (see Subsect. VII-A). Then, the results for the initial and parameter optimized GGA compared to the ALNS are presented (see Subsect. VII-B). Finally, the different parameter configurations as well as improvements in each problem class are highlighted (see Subsect. VII-C).

A. Data Generation

The evaluation of the meta-heuristics is performed using self-generated problem instances. These were created with a single-depot data set generator (cf. [1]), so that several single-depot problem instances with different depot positions are built and layered on top of one another for the generation of multi-depot instances. In this way, data sets with 1,200 instances each and 1, 4, 6, 8 and 9 depots are constructed. According to our cooperation partner, these data sets represent the structures in reality. Each of the data sets contains 12 problem classes including 100 instances created with same parameters in the data set generator. Thus, 60 classes with different characteristics regarding, e. g., number of depots, time window size, vehicle capacity or constellation of the customer positions, are provided [1]. All data sets are available at [29].

B. Parameter Configuration

The initial parameters used for the GGA have been manually optimized during the development using expert knowledge of the MDPDPTWHV. They are displayed in Table II.

TABLE II
INITIAL PARAMETERS FOR EVALUATING THE GGA VARIANTS

param.	value	description
n^{pop}	50	population size
p^{cross}	1.0	crossover probability
p^{mut}	0.3	mutation probability
$\gamma^{\max}$	250	max. number of generations
ω	1.5	relative mating pool size with respect to population size
λ	0.05	proportion of the elite within the population
p_i^{cross}	0.5	probability of crossover variants $i = 1, 2$
$p_1^{\text{mut}_v}$	0.4	vehicle-based mut. prob. *Costs/Number-of-Request Ratio*
$p_2^{\text{mut}_v}$	0.4	vehicle-based mut. prob. *Number of Requests*
$p_3^{\text{mut}_v}$	0.1	vehicle-based mut. prob. *Random Vehicle Index*
$p_4^{\text{mut}_v}$	0.1	vehicle-based mut. prob. *Random Genotype Position*
$p_1^{\text{mut}_r}$	0.6	request-based mut. prob. *Historical Request Pair*
$p_2^{\text{mut}_r}$	0.4	request-based mut. prob. *Similarity Measure Selection*
p_1^{repair}	0.55	probability for *Greedy Insertion* Repair Operator
p_2^{repair}	0.25	probability for *Regret*-2 Repair Operator
p_3^{repair}	0.10	probability for *Regret*-3 Repair Operator
p_4^{repair}	0.05	probability for *Regret*-4 Repair Operator
p_5^{repair}	0.05	probability for *Regret*-δ Repair Operator

For optimizing the parameters (see Table II), the Bayesian Optimization is applied on the probabilities for mutation and repair operators, since tuning parameters as $n^{\text{pop}}, \gamma^{\max}, \omega$ and λ particularly affect the number of individuals to be generated. Therefore, previous investigations have shown that optimizing these parameters result in the intuitive solution of setting these values to their maximum. Furthermore, the choice for crossover and mutation rate is reasonable with respect to the problem at hand. Finally, due to the similarity of the crossover operators, optimizing the selection probabilities of these variants will not have a huge impact on solution quality.

In order to determine the best configuration of parameters to be optimized, the BO is performed with 20 iterations. Beforehand, for modelling a sufficiently good a-priori probability distribution, the utility function values are determined for 10 randomly drawn parameter combinations. In this way, the Bayesian Optimization is applied to 10 randomly chosen problem instances (i. e., the training data) from each class to identify parameters with which the GGA has the best solution quality regarding the mean of the objective value over these instances. Since the GGA is a stochastic meta-heuristic, each instance is additionally evaluated five times to obtain a stable parameter configuration. In preliminary studies, we figured out that a larger set of training data yields in worse GGA results as well as increased computational time of the BO approach. Hence, the number of instances used for training data is chosen with respect of efficiency.

Finally, the quality of the optimized parameters is tested on the multi-depot data sets by evaluating the GGA with the best found parameters class-wise over all problem instances (which can be seen as the test data).

In order to find parameter configurations with the random optimization, 30 iterations are performed in which a parameter configuration is chosen randomly and applied to the problem instances known as training data. Here, the GGA is also evaluated five times on each instance. The parameter configurations obtained the best fitness are used to evaluate the GGA class-wise for all problem instances of a multi-depot data set as in the BO approach.

C. Results

For evaluating the considered algorithms, they are compared with respect to their computational time and relative error. The latter is calculated for each problem instance $\mathcal{I}$ and each approach $A \in \mathcal{A} = \{\text{GGA}_{\text{BO}}, \text{GGA}_{\text{Rand}}, \text{ALNS}\}$ using

$$\epsilon_{\mathcal{I}}(A) := \frac{f_{\mathcal{I}}(A) - f_{\mathcal{I}}^*}{f_{\mathcal{I}}^*}, \tag{35}$$

where $f_{\mathcal{I}}^*$ describes the objective function value of the best known solution for the problem instance $\mathcal{I} \in \mathcal{I}_{\mathcal{D}}$ (cf. [1]), i. e., $f_{\mathcal{I}}^* := \min_{A \in \mathcal{A}} f_{\mathcal{I}}(A)$. Furthermore, $\mu_\epsilon(A)$ specifies the mean value and $\sigma_\epsilon(A)$ the standard deviation of the relative error (35) for an approach A over all instances $\mathcal{I}$ of a data set $\mathcal{I}_{\mathcal{D}} = \{1\text{D}, 4\text{D}, 6\text{D}, 8\text{D}, 9\text{D}\}$.

TABLE III
MEAN VALUE AND STANDARD DEVIATION OF THE RELATIVE ERROR FOR THE INITIAL AND OPTIMIZED GGA'S PARAMETER CONFIGURATION IN COMPARISON TO THE ALNS AS WELL AS MEAN COMPUTATIONAL TIME

	Approach		$\mu_\epsilon^{\text{init}}$	$\sigma_\epsilon^{\text{init}}$	$\mu_\epsilon^{\text{opt}}$	$\sigma_\epsilon^{\text{opt}}$	$\mu_{\text{exe. time}}^{\text{opt}}$
1D	GGA	BO	0.79 %	1.48 %	0.89 %	1.31 %	15.49 sec.
		Rand			0.99 %	1.39 %	14.67 sec.
	ALNS		1.08 %	1.48 %	1.99 %	1.95 %	17.49 sec.
4D	GGA	BO	0.48 %	1.06 %	0.61 %	1.08 %	17.48 sec.
		Rand			0.66 %	1.12 %	17.45 sec.
	ALNS		0.86 %	1.42 %	1.50 %	1.91 %	19.10 sec.
6D	GGA	BO	0.63 %	1.34 %	0.66 %	1.11 %	34.82 sec.
		Rand			0.80 %	1.24 %	33.15 sec.
	ALNS		1.01 %	1.51 %	2.00 %	2.15 %	48.07 sec.
8D	GGA	BO	0.79 %	1.49 %	0.72 %	1.19 %	66.77 sec.
		Rand			0.88 %	1.23 %	55.56 sec.
	ALNS		0.97 %	1.40 %	2.18 %	2.24 %	100.05 sec.
9D	GGA	BO	0.84 %	1.55 %	0.78 %	1.25 %	83.64 sec.
		Rand			0.91 %	1.30 %	73.14 sec.
	ALNS		1.00 %	1.48 %	2.26 %	2.23 %	135.94 sec.

In Table III, the mean values and standard deviations of the relative error with respect to (35) are displayed for all five data sets. In doing so, the initial and optimized parameter configurations can be compared. First of all, it is worth mentioning that the solution quality of the GGA even with the initial parameter setting is better than of the ALNS (with initial parameters) since $\mu_\epsilon^{\text{init}}$ is smaller for all data sets. Respecting the parameter configuration, it can be seen that both mean value μ_ϵ as well as standard deviation σ_ϵ decrease for the GGA from initial to optimized parameters for all data sets. This shows that optimizing the parameter configuration for the GGA further improves the solution quality in comparison

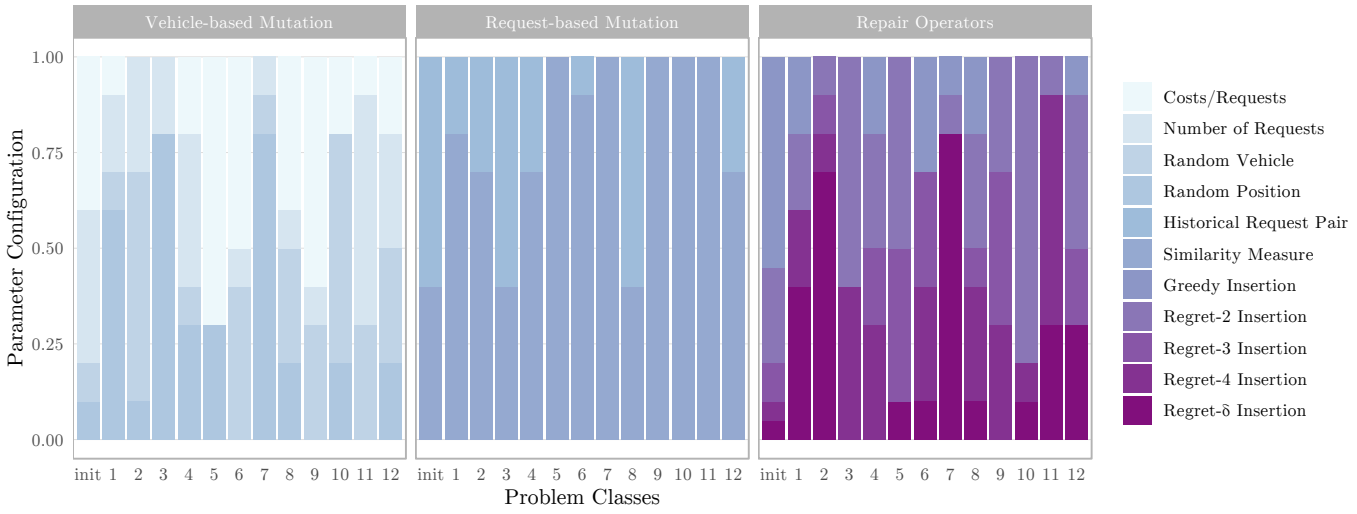

Fig. 6. Different parameter configurations for each problem class compared to the initial parameter setting (*class* init) for the GGA with binary tournament evaluated on the 4-depot data set

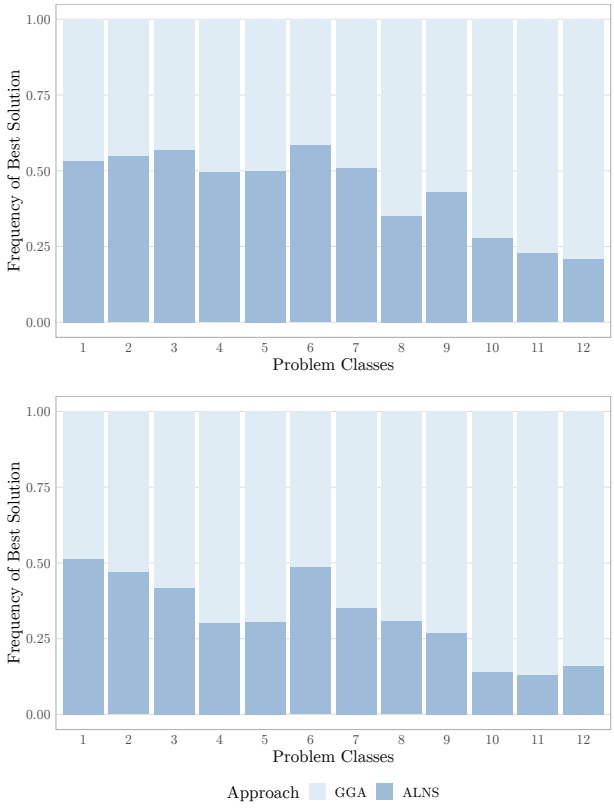

Fig. 7. Frequency of best solution found by the approaches within the 12 problem classes with 4 depots; initial parameter configuration (top) and optimized parameters (bottom)

to the ALNS. Moreover, the GGA becomes more stable in finding sufficiently good solutions which is shown by the smaller standard deviation σ_ϵ of the relative error. Since the corresponding mean and standard deviation values for the ALNS also increases, the GGA finds more best known solutions than initially. In addition, the mean value of the optimized GGA's computational time $\mu^{\text{opt}}_{\text{exe. time}}$ is up to 38 % faster than of the ALNS. Finally, the parameter configurations tuned by the BO results in better and more stable solutions for the GGA than the ones of the randomized optimization. Especially when the complexity of the problem instances increases, i. e., with larger number of depots, the gap of the mean value $\mu^{\text{opt}}_\epsilon$ and standard deviation $\sigma^{\text{opt}}_\epsilon$ of the relative error becomes larger. Hence, parameter configuration with the proposed Bayesian Optimization approach is purposefully reasonable.

The fact that the solution quality of the GGA is improved by the parameter configuration done by the BO approach can also be seen in each problem class. In Fig. 7, for the 4-depot data set, the frequency of how often an approach has found the best solution within each of the 12 problem classes is depicted. It can be observed that the parameter tuning improves the solution quality of the GGA per class, since the frequency of the best solution found increases from the initial (top) to the optimized (bottom) parameter configuration. Similar effects can be shown for the other four data sets.

Finally, the various mutation and repair operators are reasonable for solving the MDPDPTWHV, as they tackle the complexity of the considered Pickup and Delivery Problem in the right manner (see $\mu^{\text{init}}_\epsilon$ in Table III for GGA and ALNS). In Fig. 6, it becomes clear for the 4-depot data set that the selection probabilities of the operators have to be varied over the problem classes for further improvement of the solution quality within each class. This is due to different data structures across the different classes. Similar effects regarding the selection probabilities can be seen in the results for the other data sets.

VIII. Discussion

In this paper, a Bayesian Optimization approach with Gaussian Processes has been considered for the parameter tuning of a Grouping Genetic Algorithm solving practical-oriented Pickup and Delivery Problems. In order to evaluate the improvement of solution quality for the optimized parameter configurations, the BO finds best parameters for each of the five data sets and considered problem classes (with 100 instances each). Thus, 60 parameter configurations are determined, which are then adopted to solve all instances of the same corresponding class.

It could be shown that the BO is able to improve the solution quality of the considered GGA significantly while the computational time of the GGA is kept low in comparison to a state-of-the-art solution approach (see Table III). Additionally, the BO is results in better parameter configurations for the GGA than randomly choosing parameters, thus it is a purposeful approach for improving the solution quality of the GGA. Moreover, the existence of various mutation and repair operators could be proven to be appropriate, since different structures within the instances have to be tackled in a different way (see Fig. 6).

In future research, the knowledge that similar instances (within a class) can be solved efficiently in the same manner is supposed to be included in an a-priori parameter selection approach. To do so, key performance indicators which identify similar instances have to be developed in order to define problem classes in a general way. Then, the best parameter configuration can be determined for each class and be applied on new classified instances. The classification task can be done by, e. g., neural networks.

References

[1] C. Rüther and J. Rieck, "A grouping genetic algorithm for multi depot pickup and delivery problems with time windows and heterogeneous vehicle fleets," in *LNCS, EvoCOP Proceedings*, 2020, pp. 148–163.

[2] J. Mockus, "Bayesian global optimization," in *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds., 2001, pp. 123–127.

[3] C. Huang, Y. Li, and N. X. Yao, "A Survey of Automatic Parameter Tuning Methods for Metaheuristics," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 201–216, 2020.

[4] S. K. Smit and A. E. Eiben, "Comparing parameter tuning methods for evolutionary algorithms," in *IEEE Congress on Evolutionary Computation (CEC)*, 2009, pp. 399–406.

[5] L. Calvet and A. A. Juan and C. Serrat and J. Ries, "A statistical learning based approach for parameter fine-tuning of metaheuristics," *Stat. Oper. Res. Trans.*, 2016.

[6] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 3, no. 2, pp. 124–141, 1999.

[7] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[8] V. Nannen and A. E. Eiben, "Relevance estimation and value calibration of evolutionary algorithm parameters," in *IJCAI International Joint Conference on Artificial Intelligence*, 2007, pp. 975–980.

[9] N. Hansen, "The CMA evolution strategy: A comparing review," *Studies in Fuzziness and Soft Computing*, 2006.

[10] P. Balaprakash, M. Birattari, and T. Stützle, "Improvement strategies for the f-race algorithm: Sampling design and iterative refinement," in *Lecture Notes in Computer Science*, vol. 4771, 2007.

[11] I. Roman, J. Ceberio, A. Mendiburu, and J. A. Lozano, "Bayesian optimization for parameter tuning in evolutionary algorithms," in *IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 4839–4845.

[12] J. Rieck and J. Zimmermann, "A new mixed integer linear model for a rich vehicle routing problem with docking constraints," *Ann. Oper. Res.*, vol. 181, no. 1, pp. 337–358, 2010.

[13] M. G. S. Furtado, P. Munari, and R. Morabito, "Pickup and delivery problem with time windows: A new compact two-index formulation," *Oper. Res. Lett.*, vol. 45, no. 4, pp. 334–341, 2017.

[14] D. Feillet, "A tutorial on column generation and branch-and-price for vehicle routing problems," *4OR*, vol. 8, no. 4, pp. 407–424, 2010.

[15] G. Pankratz, "A grouping genetic algorithm for the pickup and delivery problem with time windows," *OR Spectrum*, vol. 27, pp. 21–41, 2005.

[16] M. I. Hosny and C. L. Mumford, "The single vehicle pickup and delivery problem with time windows: Intelligent operators for heuristic and metaheuristic algorithms," *Journal of Heuristics*, vol. 16, no. 3, pp. 417–439, 2010.

[17] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed., ser. Natural Computing Series. Heidelberg: Springer, 2015.

[18] Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*, 2nd ed. Berlin Heidelberg: Springer, 2004.

[19] S. Ropke and D. Pisinger, "An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows," *Transp. Sci.*, vol. 40, no. 4, pp. 455–472, 2006.

[20] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Comput. Oper. Res.*, vol. 34, no. 8, pp. 2403–2435, 2007.

[21] E. Brochu, V. M. Cora, and N. de Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning," 2010.

[22] P. I. Frazier, "A Tutorial on Bayesian Optimization," 2018.

[23] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the Human Out of the Loop: A Review of Bayesian Optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[24] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast Bayesian hyperparameter optimization on large datasets," *Electron. J. Stat.*, vol. 11, pp. 4945–4968, 2017.

[25] C. Huang, B. Yuan, Y. Li, and X. Yao, "Automatic parameter tuning using bayesian optimization method," in *IEEE Congress on Evolutionary Computation (CEC)*, 2019, pp. 2090–2097.

[26] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, MA, USA: MIT Press, 2006.

[27] D. K. Duvenaud, "Automatic Model Construction with Gaussian Processes," Ph.D. dissertation, University of Cambridge, 2014.

[28] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," 2012.

[29] Betriebswirtschaft und Operations Research Homepage, 2020, last access 28 Feb 2021. [Online]. Available: https://www.uni-hildesheim.de/fb4/institute/bwl/betriebswirtschaft-und-operations-research/