Pareto Front Approximation Results for Bus Driver Scheduling with Complex Constraints

Nikolaus Frohner[a,*], Esther Mugdan[a], Lucas Kletzander[a] and Nysret Musliu[a]

[a]Christian Doppler Laboratory for Artificial Intelligence and Optimization for Planning and Scheduling
TU Wien, Vienna, Austria

Abstract. Bus driver scheduling deals with assigning drivers to vehicles which satisfy customers' transport needs on tours. Safety concerns and labor laws dictate hard constraints like imposing a maximum driving time and taking required breaks. Bus operators seek for efficient schedules in terms of costs which should also be accepted by the drivers, for instance by avoiding duty parts cluttered over the day with long breaks and too many tedious vehicle changes. Decision makers want to learn about the trade-offs and the interaction of such conflicting goals. To this end, we compare different multi-objective optimization approaches to approximate the up to five-dimensional Pareto frontiers (PFs) for two real-world bus driver scheduling variants with complex sets of hard and soft constraints. We observe that a mutation-only NSGA-II already provides a solid baseline, while a population-based simulated annealing variant is often superior under short time limits when tuning it towards heavy restarting from the current PF approximation.

1 Introduction

Automated planning approaches to bus driver scheduling (BDS) have been studied over the last decades, see, e.g., Wren and Rousseau [23] in 1995, Lourenço et al. [18] in 2001, and Kletzander and Musliu [11] in 2020. It is still gaining attraction due to its plethora of problem variants given varying labor laws over the world and its relevance in the increasing demand of public transportation to promote the green transformation and sustainability goals. Furthermore, finding personnel becomes increasingly difficult, making a trade-off between a cost-efficient and ecological bus operation and providing schedules well-received by employees even more challenging.

In this work, we study real-world Central European BDS variants with complex break constraints and conflicting goals, which have so far mainly been tackled by weighted-sum approaches. We investigate classical solution archive based many-objective optimization approaches Pareto simulated annealing [2] and NSGA-II [4] to quickly approximate up to five-dimensional Pareto frontiers. They allow us to assess the trade-offs between the different objectives, qualitatively and quantitatively. The approximated fronts may also serve as basis for more sophisticated decision support systems, like the reference point method R-XIMO by Misitano et al. [21] and a Shapley value based weight setting approach by Mischek and Musliu [20].

We describe the considered problem variants in more detail in Section 2 and discuss related multi-objective work for bus driver and crew scheduling in Section 3. The adopted many-objective combinatorial optimization approaches are described in Section 4. The main contribution of this work is a detailed computational study in Section 5, where we tune parameters on a training set and validate the approaches on a test set of real-world inspired BDS instances and one real-world instance, after which we conclude in Section 6. A key finding is that a population-based simulated annealing (PSA) greatly benefits from frequent restarting from the current non-dominated solution set. This mechanism enables PSA tuned with *irace* [17] to beat the solid mutation-only NSGA-II baseline algorithm in terms of normalized hypervolume of non-dominating solution sets and minimizing the distance to a manually tuned trade-off reference point on the Pareto front on three to five-dimensional problem versions.

2 Problem Description

Bus Driver Scheduling (BDS) is a part of crew scheduling in the process of operating bus transport systems [9]. It deals with the feasible combination of pieces of work to duties abiding labor regulations. The duties can then be repeated over a planning horizon and subsequently be assigned to concrete drivers also taking their preferences into account. This is the Bus Rostering Problem (BRP), a separate problem not considered in this work. The BDS has been tackled by many authors as already reviewed by Wren and Rousseau [23] in 1995. Most of them focused on costs [9, Tab. 7], with Lourenço et al. [18] and Kletzander and Musliu [11] being notable exceptions. We study the latter problem variant with complex break constraints (CBC) as explicit multi-objective optimization (MOO) problem based on Pareto dominance instead of using a weighted-sum approach, henceforth abbreviated MO-BDS-CBC. We now provide a concise overview of the original problem, for a more detailed description see Kletzander and Musliu [11].

A problem instance consists of bus routes which are given as a set L of individual bus legs. Each leg $\ell \in L$ is associated with a tour ν_ℓ corresponding to a particular vehicle, a start time t^{s}_ℓ, an end time t^{e}_ℓ, a starting position x^{s}_ℓ, and an end position x^{e}_ℓ. The driving time for leg ℓ is $d_\ell = t^{\mathrm{e}}_\ell - t^{\mathrm{s}}_\ell$. A solution to a MO-BDS-CBC instance is an assignment of exactly one (abstract) employee to each bus leg, i.e., a mapping $A\colon L \to E$ which induces the assignment of legs to employees $A^{-1}\colon E \to 2^L$. The legs are ordered by their starting times. A possible solution to an example instance is shown in Figure 1.

A tour change occurs when a driver has an assignment of two consecutive bus legs i and j with $\nu_i \neq \nu_j$. The time it takes to change from position x to position y when not actively driving a bus (passive ride time), is $\delta_{x,y}$ for $x \neq y$. The diagonal elements $\delta_{x,x}$ represent

* Corresponding Author. Email: nikolaus.frohner@tuwien.ac.at

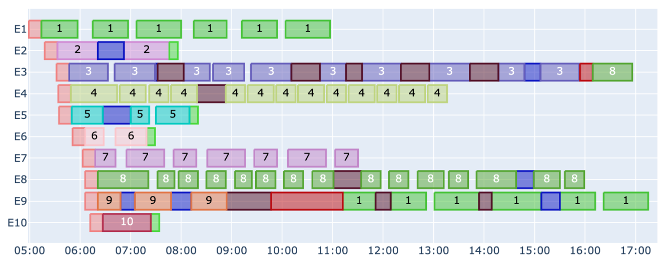

Figure 1. Ten partial example duties (E1–E10) for ten bus tours (1–10). Numbered rectangles are the actively driven bus legs, brown are unpaid breaks, violet are paid breaks, red is passive riding time to a relief point, light red and green setup and postprocessing times.

the time it takes to switch tour at the same position, but is not considered passive ride time. Example duties E3 and E9 in Figure 1 each contain one tour change. Each position x is further associated with an amount of working time for starting a shift (d^{s}_x) and ending a shift (d^{e}_x) at that position.

Constraints are imposed on duties separately. The only implicit overall constraint is that there is maximum number of duties $|E|$. This permits to calculate hard and soft constraint violations $v_h(A), v_s(A)$ of an assigment A as sum of separate calculations of constraint violations for employees (duties) $e \in E$. The problem has the following set of hard constraints $\mathcal{H}$:

- *Overlap* (h^{over}): No overlapping leg assignments and enough changing time in case of a tour change.
- *Max span* (h^{span}): Hard maximum $O_{max} = 14$ hours for the total span of a shift.
- *Max work* (h^{work}): Hard maximum $W_{max} = 10$ hours for the total paid working time per shift.
- *Max drive* (h^{drive}): Hard maximum $D_{max} = 9$ hours for the total driving time per shift.
- *Driving breaks* (h^{breaks}): Driving breaks after at most 4 hours of driving time, with the options of one break of at least 30 minutes, two breaks of at least 20 minutes each, or three breaks of at least 15 minutes each.
- *Rest breaks* (h^{rest}): Rest breaks of at least 30 minutes are required for shifts between 6 and 9 hours, and of at least 45 minutes for shifts of more than 9 hours.

Furthermore, MO-BDS-CBC considers six different objectives as sums over the duties. Given a schedule A, $W_e(A)$ corresponds to the paid working time for employee e and $T_e(A)$ to the duty's span (spreadover) including unpaid duty parts. The objectives are also called soft constraints as they all should be minimized, $\mathcal{S}$, with a given maximum number of duties $n = |E|$ as instance input:

- *Working time* (s^{work}): The total amount of paid working time $v_{s^{\mathrm{work}}} = \sum_{e \in E} W_e$ excluding additional paid working time used to fill up shifts below $W_{min} = 6.5$ hours.
- *Minimum working time* (s^{mwork}): The additional amount of paid working time to fill up shifts below W_{min}, obtained by $v_{s^{\mathrm{mwork}}} = \sum_{e \in E} \max\{W_{min} - W_e;\ 0\}$.
- *Span* (s^{span}): The sum of all spans $v_{s^{\mathrm{span}}} = \sum_{e \in E} O_e$.
- *Passive ride* (s^{ride}): The sum of passive ride times.
- *Tour changes* (s^{change}): The total number of tour changes.
- *Shift splits* (s^{split}): The total number of shift splits (breaks of at least 3 hours which are always unpaid, but not rest breaks).

Hard constraint violations are explicitly considered. The goal is to find "interesting" non-dominated (ideally Pareto-optimal) solutions over (a subset of) the soft constraints, our vector-valued objective. Feasible solutions always dominate infeasible ones, i.e., we have two lexicographically ordered sets of objectives $\mathcal{H}$ and $\mathcal{S}$. The vector-valued violation tuple is therefore:

$$\boldsymbol{v}(A) = (\boldsymbol{v}^{\mathrm{h}}(A), \boldsymbol{v}^{\mathrm{s}}(A)), \tag{1}$$

where $\boldsymbol{v}^{\mathrm{h}}(A)$ amounts to the hard constraint and $\boldsymbol{v}^{\mathrm{s}}(A)$ to soft constraint violations vector of schedule A. We seek to find the set $\min_A \boldsymbol{v}(A)$ w.r.t. the described dominance relation.

3 Related Work

Lourenço et al. [18] in 2001 introduce a many-objective bus driver scheduling (MO-BDS) problem variant with the objectives costs (paid labor time), overcover of legs, number of duties (employees, drivers), tour changes, and duties with single pieces of work. They use a set cover formulation where a set of feasible duties (in the ten-thousands) is already given as input corresponding to the columns with which to cover the rows (legs/pieces of work, in the hundreds). Efficient solutions are generated with sophisticated variants of tabu search and genetic algorithms using a GRASP to solve set covering subproblems. They find a trade-off relation between overcover and flexible but undesirable single-piece-of-work duties, the latter is similar to our minimum working time penalizing short duties. In contrast, we use a direct formulation where each bus leg (in the hundreds/thousands) is assigned exactly once to a driver, a search space of all possible duties (feasible or infeasible) without overcover where hard constraints need to be considered explicitly.

Li and Kwan [16] propose a hybrid genetic algorithm to solve a bi-objective BDS considering operational costs and number of duties. They use a set cover formulation and apply multi-start greedy construction where weights for the greedy criterion combining structural aspects of a duty are learned using the aforementioned genetic algorithm. We also use a normalized evaluation function, transforming objectives into a hypercube defined by an ideal and a reference point. The same objectives are tackled by De Leone et al. [3] who suggest a three-index formulation for an Italian BDS problem with hard regulatory constraints, where pieces-of-work are directly placed at a specific position of a duty having a bounded set of duties. They combine the costs and number of duties as two objectives in a weighted sum and compare different (extreme) weight settings. They introduce a parameterized randomized construction heuristic and different neighborhood structures which they combine in a GRASP approach to tackle the instances out of reach for their exact solving method. In our problem variant, we observe that reducing the number of duties generally supports reducing operational costs and vice versa.

The problem variant we study here was introduced by Kletzander and Musliu [11] with the aforementioned six different objectives. They are combined to a single weighted-sum objective function to create desirable schedules performing manual weight tuning with feedback from practitioners. To address the issue of setting weights, Kletzander and Musliu [12] study an automated weight learning approach driven by goals in the objective space. The current state-of-the-art approaches are based on a sophisticated branch-and-price solver by Kletzander et al. [14], which is incorporated as repair operator into an (adaptive) large neighborhood search by Mazzoli et al. [19] to find new best feasible solutions for medium to large instances.

A generalization of driver scheduling is crew scheduling. Suitably skilled people are assigned together to, probably longer, pieces-of-work like it is the case for airline and railway travel. Ehrgott and Ryan [6] study exact ε-constraint based methods for a bi-objective

Algorithm 1: Population-based algorithmic framework borrowing terminology from evolutionary algorithms.

```
Input: Instance $I$, number of individuals $N^{\text{ind}}$, time limit $\tau$,
       algorithm-specific parameters $\boldsymbol{\theta}$
Result: Non-dominated solution set $\mathcal{S}$
1  $\mathcal{S} \leftarrow \{\}$, $\mathcal{G} \leftarrow$ init($N^{\text{ind}}$), evaluate($\mathcal{G}$);
2  pareto-update($\mathcal{S}, \mathcal{G}$);
3  while elapsed time $\leq \tau$ do
4      $\mathcal{G}' \leftarrow$ select-parents($\mathcal{G}$);
5      mutate($\mathcal{G}'$), evaluate($\mathcal{G}'$);
6      pareto-update($\mathcal{S}, \mathcal{G}'$);
7      $\mathcal{G} \leftarrow$ select-survivors($\mathcal{G}, \mathcal{G}'$);
8      replenish($\mathcal{G}, \mathcal{S}$);                // if $|\mathcal{G}| < N^{\text{ind}}$
9      update-strategy-parameters($\mathcal{G}$);
10 end
11 return $\mathcal{S}$
```

airline crew scheduling problem. They quantify the inverse relation between operational costs and a robustness measure—less robust schedules are more prone to propagation of delay due to flight changes of crews with tight ground time windows. Considering robustness aspects related to traffic congestion issues might be interesting future work for the BDS as well.

Jolai and Assadipour [10] present a concise study applying cellular genetic algorithms to a MO crew scheduling problem with the objectives costs, delay, and unbalanced utilization. Banerjee et al. [1] apply an extended NSGA-II variant to an interval-valued crew scheduling problem with uncertainties.

4 Optimization Approaches

Neighborhood search based metaheuristics like simulated annealing (SA), tabu search (TS), and iterated local search (ILS) have proven to work well on the BDS problem domain, see [11, 15]. Therefore, we follow the pattern of a multi-trajectory local search framework with an archive of currently non-dominated solutions. It is outlined as pseudo-code in Table 1 with terminology borrowed from evolutionary algorithms (EAs), within which both considered variants of NSGA-II and population-based simulated annealing can be formulated. The current generation of individuals are kept in the set $\mathcal{G}$ together with algorithm-dependent strategy parameters and historical information. Every new solution sampled from a neighborhood is checked for non-dominance and potentially goes into the solution archive $\mathcal{S}$, which is eventually returned.

Initialization. Having diversity in mind, we create an initial population of N^{ind} solutions by repeated application of a randomized greedy construction heuristic based on [11]. It iterates over all tasks (bus legs) in ascending order by their start time and randomly assigns a task to a driver whose duty can be feasibly extended. If no feasible extension is possible, a new driver is added and assigned the task. If the maximum number of drivers is hit, the task is then assigned overall randomly to an existing driver. The following legs of the same tour are also added to the selected driver as long feasibly possible to bias towards fewer tour changes. The population initialization is the same for NSGA-II and PSA.

Pareto simulated annealing. We adopt a population-based simulated annealing, called *Pareto simulated annealing* proposed by Czyżżak and Jaszkiewicz [2]. Multiple so-called *generating solutions* (corresponding to EA's individuals) $A \in \mathcal{G}$ traverse the solution space by applying a random move (corresponding to a mutation) operator every generation and collect non-dominated solutions along their way. As neighborhood structure, we use a parameterized composite leg swap neighborhood [11]. Two different duties are randomly selected and a random consecutive subsequence of legs (block) of the first duty is moved to the second duty. The overlapping legs of the second duty are then moved back to the first duty. The quality of the duties are taken into account (e.g., swap from lower-quality duties with higher likelihood) and the block length is also a random variable. The underlying probability distributions are parameters and are discussed and tuned in the computational study.

The cooling schedule is global for all individuals. It starts with an initial temperature $T = T_0$ and cools down T every N^{equi} generations by multiplying with $\beta^{\text{c}} < 1$. We reheat to the initial temperature T_0 when reaching the final temperature T_{final}. A neighboring solution x' proposed by a move applied on x is accepted in four cases: if it dominates the current solution, if it is a newly found (global) non-dominated solution, if it is better according to its current search direction, or probabilistically depending on the current temperature T by the Metropolis criterion. Otherwise, the generating solution x remains unchanged for the given generation.

To facilitate different search directions, the individuals carry sum-to-1 weights $w_j(A)$ for all the objectives $j \in 1, \ldots, |\mathcal{S}|$, their strategy parameters. This amounts to individual preferences, where some objectives are more important than others. As the weights are different for each individual and may also be updated during the search, a broad range of preferences is covered.

An optional repulsion-of-nearest-neighbors mechanism [2] in the objective space is employed based on weight updates to promote exploration of the search space and avoid clustering. This is done by comparing the objectives of an individual with its nearest non-dominated neighbor A'' (not to be confused with a neighbor from the search neighborhood, denoted as x') and multiplying the objectives' weights where the objective is better with $\alpha \geq 1$ and divide by the same α for those which are worse:

$$w_j(A) \leftarrow \begin{cases} w_j(A) \cdot \alpha & \text{if } v_{s_j}(A) \leq v_{s_j}(A'') \\ w_j(A)/\alpha & \text{otherwise} \end{cases} . \tag{2}$$

We take the Euclidean distance in normalized objective space where all objectives have similar scale. The weights are initialized randomly $\sim U[0,1]$ and are kept above a threshold of 10^{-3} during normalization s.t. they do not vanish [20].

To foster intensification, generating solutions that do not contribute to the approximate Pareto front $\mathcal{S}$ for a while are restarted randomly at a currently non-dominated solution, as suggested by Drexl and Nikulin [5]. We use as tunable parameters $\boldsymbol{\theta}_{\mathcal{R}}$ a fixed threshold of non-improving generations n^{re} after which such a restart is performed with probability p^{re} and optionally reinitializing the individual's weights randomly. Note that this restart is not the overall reheat of PSA where the solutions remain unchanged but the global temperature is set to its initial value, but a local restart of an individual by copying a solution deemed more promising.

NSGA-II. As a competing baseline approach, we further study a minimal NSGA-II [4, 22] variant with binary tournaments for parent selection considering the non-dominance rank and crowding distance, and the same rules for survivor selection among parents and offspring. As mutation operator we take the move operator from PSA which swaps leg blocks between duties. A single mutation move is applied on every parent, no crossover operation is performed. Solutions are encoded as a list of duties sorted by increasing start time of their first legs, where each duty itself is encoded as list of legs, again ordered by increasing start time. More sophisticated approaches are justified by beating this rather straightforward baseline.

Evaluation. As our objectives are of different scales and units, we let the algorithms operate in a normalized objective space (the "tilde"-space). To this end, we approximate the ideal points $\boldsymbol{l}$ by optimizing each objective separately for a given problem instance, which then corresponds to $\mathbf{0}$, the origin. Then, we let the decision maker define a reference point at $\mathbf{1}$ by defining the worst acceptable values $\boldsymbol{u}$ for the objectives. The transformation to normalized space is performed by $\tilde{v}_{s_j} = (v_{s_j} - l_j)/(u_j - l_j)$, i.e., calculting the difference between ideal and reference point to rescale the objective and taking the difference to its ideal. This rescaling is important for distance calculations, the update rules using a weighted sum over the objectives, and to compare performance indicators as the hypervolume [7] which measures the objective space covered by a non-dominated solution set relative to a reference point.

As we have many hard constraints that are difficult to fulfill, how to treat infeasible solutions becomes a challenging question to answer. For the baseline NSGA-II, we set infeasible solutions to this reference point. For SA, we keep a separate weight for hard constraints violations as tunable parameter. Equation 3 shows the objective function used to calculate the acceptance probability for a child related to a parent solution via the Metropolis criterion's exponent $-|\tilde{f}(A') - \tilde{f}(A)|/T$, i.e., the log-probability to accept a worse solution A' when currently at solution A. The weighted sum objective is then defined as:

$$\tilde{f}(A; \boldsymbol{w}) = M \cdot \sum_{i=1}^{|\mathcal{H}|} v_{h_i}(A) + \sum_{j=1}^{|\mathcal{S}|} w_{s_j} \cdot \tilde{v}_{s_j}(A). \qquad (3)$$

(Hyper-)parameter tuning. There are quickly many different algorithmic parameters $\boldsymbol{\theta}$ involved, in machine learning referred to as hyperparameters to avoid confusion with model parameters. For PSA, we consider population size N^{ind}, initial temperature T_0, quadratic cooling rate β^{c}, equilibrium iterations N^{equi}, final temperature T_{final} after which we reheat back to T_0, hard weight M, restarting parameters $\boldsymbol{\theta}_{\mathcal{R}}$, and neighborhood structure parameters $\boldsymbol{\theta}_{\mathcal{N}}$, which we will specify in detail in the computational study. For our minimal NSGA-II, in the spirit of a baseline algorithm with few tunable parameters, we only consider the population size N^{ind}. We first explore the parameter space manually with the goal to get first ideas of the impact of and relations between parameters in preliminary experiments on a training data set. After that, we define a configuration space, in which we let the iterative racing software *irace* [17] do its work to efficiently make use of a given computational budget of r runs. The tuned algorithms are then finally studied on unseen test instances.

5 Computational Study

We now present the application of the described MOO approaches to two BDS variants. Our computational testbed was a cluster running Ubuntu 22.04.2 LTS with 2× Intel Xeon CPU E5-2650 v4 (2.2 GHz, 12 physical cores, no hyperthreading). The algorithms were implemented in Python 3.9 and run with the fast PyPy interpreter.[1] NSGA-II was implemented with the Python toolbox DEAP by Fortin et al. [8] using a solution decoding/encoding pattern. We mainly study the problem variant by Kletzander and Musliu [11] with publicly available test instances to be comparable with the literature.[2] Furthermore, we generated another batch of instances from the same distribution to act as training set for exploratory experiments and parameter tuning. They are made available under the same link, together with the instance-wise maximum number of employees and objective ideals. The reference point calculation is described in the next subsection, using the maximum number of employees and instance-independent constants as input. Ideals and reference points are in particular important to calculate the normalized hypervolumes, our main figure of merit. We conclude the study by presenting results on a single instance from a real-world problem variant to further provide evidence for the approaches' usefulness for decision makers.

[1] https://www.pypy.org/

5.1 *Preparation*

In the preparatory phase, we first analyze instance characteristics and derive for each instance a maximum number of duties. Furthermore, we calculate ideals and a reference point for each of the objectives.

Instance characteristics. Previous work [11] has shown that the maximum number of employees (maximum duties) has a strong impact on the solutions, in particular costs, as one would expect. As the work to be performed remains the same, fewer employees necessarily lead to more efficient use of the available labor force but finding feasible solutions may become more complicated or even impossible. However, undesired patterns like shift splits, which can cover two spread demand peaks by, e.g., joining a morning and afternoon spell with a long unpaid break in between using only one employee may then emerge or even become necessary. Reducing the maximum number of employees can also be seen as a separate objective—we take it as input parameter provided along with the problem instance. Multiple runs with different numbers can then be performed to study the impact of changing the maximum size of workforce. This leads to the questions how to sensibly set this parameter for the given instances, which we answer in the remainder of this subsection.

We mainly perform experiments with a set of 30 training instances $\mathcal{I}^{\text{train}}$ of the MO-BDS-CBC. A final validation is performed on an equally-sized test set $\mathcal{I}^{\text{test}}$ and on a different problem variant for which have a single real-world instance of a project partner. The instance characteristics of the training instances are shown in Table 1. The gross leg durations are defined as the end of a tour's last leg minus the start of its first leg plus 25 minutes setup/postprocessing time summed over all tours in units of eight hours. This should give a first coarse estimate of the number of employees required and provide some means to normalize objective values, e.g., how much work time does a solution require in units of gross leg durations.

[2] https://cdlab-artis.dbai.tuwien.ac.at/papers/sa-bds/

Table 1. Characteristics of training instance set $\mathcal{I}^{\text{train}}$ with $|V|$ the number of tours, $|L|$ number of legs/pieces of work, and d and $\hat{d}$ corresponding net and gross durations of legs 8 hour blocks.

$\lvert V\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert L\rvert$	$d(L)_{8\text{h}}$	$\hat{d}(L)_{8\text{h}}$	$\lvert E\rvert$	s^{change}	s^{split}
8	5	76.8	7.0	9.7	12.8	5.4	2.6
17	5	168.2	14.5	20.4	25.6	11.0	3.2
29	5	255.4	23.2	31.7	42.2	20.6	6.8
39	5	355.8	30.5	43.1	55.0	32.4	11.8
50	5	457.8	39.5	55.5	71.6	39.0	13.0
58	5	542.6	47.5	66.0	84.8	47.4	14.8

Construction heuristics. To retrieve more information and a better estimate regarding required employees in relation to desirable goals, we make use of randomized construction heuristics as described in the previous section. In Table 1, we add the maximum number of employees required over ten randomized construction runs, averaged for each instance group. We further take the corresponding solution with the maximum number for each instance and calculate the average number of tour changes and shift splits per instance group. E.g., for the largest instance group every $\sim$ 6th employee faces a split duty.

Ideals and reference points. In our approach, we require, apart from the problem instance in form of the bus legs to be assigned to drivers and distances between stations, further input: The maximum number of drivers available, an ideal point and a reference point for normalization. To bootstrap the maximum number of drivers $|E|$, we perform for each instance another 30 runs for each with the randomized construction heuristic and take the maximum number over the feasible solutions. We perform multiple simulated annealing runs [11] with decreasing number of employees starting from the maximum enabling only one objective at a time to approximate the ideals $\mathbf{l}$ (lower) for the objectives. With the reference point $\mathbf{u}$ (upper), we introduce first preferences of the decision maker. It is calculated based on the maximum number of employees and limits of what can be tolerated: $1.5 \times |E|$ for the tour changes, $0.5 \times |E|$ for shift splits, 2 hours per employee for ride time and minimum work time, 10 hours work time and 12 hours span per employee. With this, we transform the objectives v_{s_i} by $(v_{s_i} - l_i)/(u_i - l_i)$. A value close to 0 lies close to the ideal, close to 1 close to the reference point. As a relative performance measure for non-dominated solutions sets, the normalized hypervolume [7] $|\tilde{H}|$ is calculated in this transformed objective space, for which a value closer to 1 is better. Further considered metrics are the size of the non-dominated solution set $|\mathcal{S}|$ and the minimum Euclidean distance to a reference point $\tilde{d}_{\text{ref}}^{\text{min}}$ calculated with single-trajectory SA runs using the weighted-sum from [13].

5.2 *Training and tuning*

On the training set $\mathcal{I}^{\text{train}}$, we perform first exploratory experiments to learn about the behavior of the algorithms and the relation between different objectives. Furthermore, parameter tuning is performed, meta-optimizing the hypervolume in a time-limited setting as one main goal is to quickly find good Pareto front approximations.

Three-dimensional experiments. In preliminary tuning experiments, we consider the three objectives minimum working time, ride time and span. The motivation is that a reduced span aims for a more efficient use of the available time, also avoiding long shift splits, while too short duties are penalized by the minimum work time. Minimizing ride time acts also as a proxy for either fewer tour changes or favors more efficient changes where drivers meet at the same or close positions. As runtime limit, we use 15 minutes per unit of instance size, i.e., 15 minutes for the smallest and $6 \times 15 = 90$ minutes for the largest instances.

For PSA, we set the hard weight to a relatively high 1 to focus on feasible moves, the initial and final temperature to 0.001 and 10^{-7} respectively, and cooling factor $\beta = 0.99$ with 10 equilibrium iterations. For the swap neighborhood, two duties e_1, e_2 and a subsequence of legs in e_1 need to be sampled. We consider two options, biased and uniform sampling. For biased sampling, the duties are sorted decreasing by their weight-dependent objective value. Then an index is sampled from Beta[1, β]—increasing β favors sampling of duties with higher objective value, which may benefit more from a swap. If $\beta = 1$, the sampling is uniform. We start our experiments with uniform sampling, also because our NSGA-II individuals do not carry any weights. The automated tuning of PSA later may make use of biased sampling. Based on practitioners experience with sampling the leg subsequence, with 5% probability the first leg of a duty is taken, otherwise a leg uniformly at random. Starting from this leg, the length is sampled with 50% probability from $U[2, b]$, otherwise from $U[2, |e_1|]$. The probability weight for multi interval swaps is set to a high $w^{\text{ms}} = 10$ vs. weight 1 for single interval swaps.

Table 2 shows the performance of NSGA-II and different PSA variants with different population sizes on the training set. To initialize a population, we make use of the randomized construction heuristic from before, where seeding ensures that all algorithms start with the same individuals. We observe a small improvement when increasing the population size from 20 to 100, both in terms of being closer to a Pareto-optimal reference point and normalized hypervolume. NSGA-II performs slightly better than PSA-basic, while there is a strong impact restarting from a random solution of the current approximate Pareto front after a number of non-contributing iterations of a generating solutions. For this restarting variant *PSA-restarting*, we roll a die every $n^{\text{re}} = 100$ non-contributing iterations, restarting from a such a solution with probability $p^{\text{re}} = 0.5$ or just resetting the counter without restarting to also allow longer, more diverse randomized walks of the search. We do not randomize the weights on restart but add this later as binary parameter $I_{\text{rnd}}^{\text{re}}$ for the automated tuning. In Table 2, we see the increase of about 10% in mean nor-

Table 2. Training set: Comparison 3D (min-work, ride time, span) normalized hypervolume $|\tilde{H}|$, minimum normalized distance to a PF reference point $d_{\text{ref}}^{\text{min}}$ for different numbers of individuals N^{ind}.

		$\lvert\mathcal{S}\rvert$	$\tilde{d}_{\text{ref}}^{\text{min}}$				$\lvert\tilde{H}\rvert$
config	N^{ind}	mean	mean	std	mean	std	median
NSGA-II	20	454	0.20	0.08	0.68	0.10	0.66
	100	633	0.18	0.09	0.69	0.12	0.66
PSA-basic	20	2 435	0.24	0.15	0.64	0.19	0.57
	100	2 873	0.22	0.13	0.66	0.17	0.60
PSA-restarting	20	2 278	0.15	0.10	0.72	0.15	0.68
	100	2 403	0.14	0.10	0.74	0.14	0.70
PSA-tuned	172	2 270	**0.13**	0.09	**0.74**	0.13	**0.72**

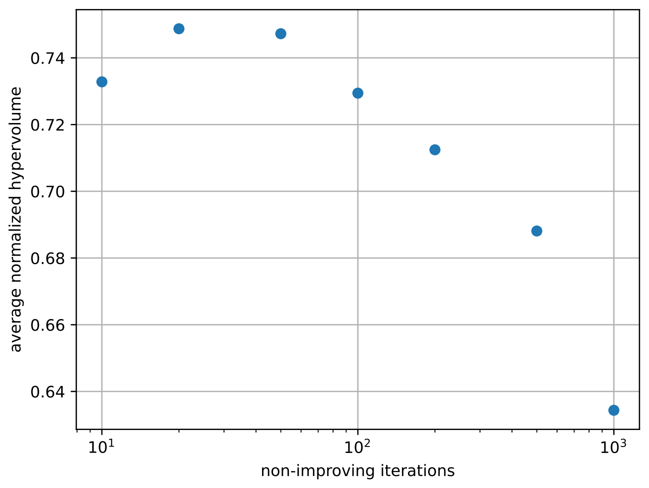

Figure 2. Average normalized hypervolume over non-improving iterations limit for PF-restarting for PSA with fixed number iterations.

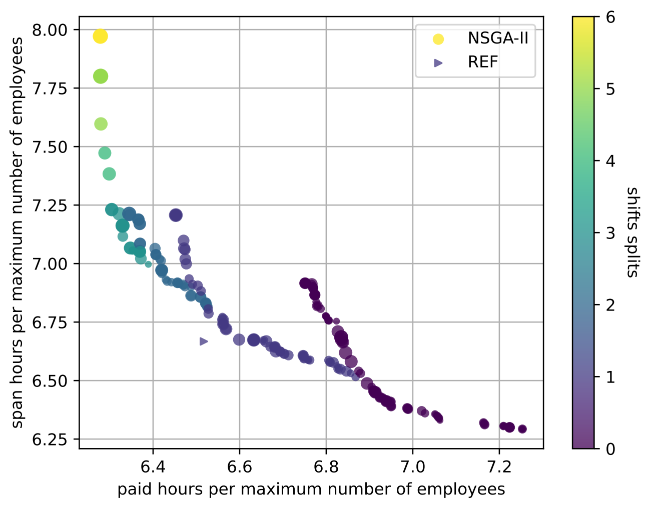

Figure 3. 3D non-dominated solutions created by NSGA-II for an instance with a maximum number of 26 duties. Point size corresponds to number of tour changes, smaller means fewer.

malized hypervolume when using restarting on our training set with 30 instances vs. the basic PSA variant.

To study the impact of the restarting parameter in more detail, we present in Figure 2 average normalized hypervolumes on the training set for PSA runs with different PF-restarting limits and fixed number of iterations, e.g., a limit of 10 randomly resets a generating solution to the current PF after ten non-improving iterations of this solution. We observe a sweet spot at a, somewhat surprisingly, low value of around 20, under an otherwise fixed remaining configuration. A catch is that the runtime increases substantially with more restarting due to more copy operations and the sampling procedure, so the actual optimum parameter value in time-limited runs is shifted to the right.

We now open up most of the PSA parameters, also for the repulsion mechanism, and run irace on the 30 training instances with a computational budget of $r = 3\,000$ and our manual configuration as seed configuration. The parameters and irace results are summarized in Table 3. The performance on the training instance set is shown in Table 2. Hypervolume mean and median and distance to the reference point look slightly better, also with less variance, the hypervolume is better in 19/30 instances. A subsequent Wilcoxon signed rank sum test confirms a significant difference with $\alpha_{\text{test}} = 0.05$.

Regarding the nearest neighbor repulsion mechanism to update the weights during the search, we could not find suitable parameters where the search significantly benefits from these updates, in particular given the increased runtime due to finding the nearest neighbor. Also the elite configurations for irace always kept weight updates disabled. It could be that the effects are too small, require a speedup in the implementation of the nearest neighbor finding using k-d trees or event different weight update rules.

Paid time vs. other criteria. To reduce the dimensionality of the problem, we introduce the sum $s^{\text{paid}} = s^{\text{mwork}} + s^{\text{work}}$, which are the monetary costs for the bus operator. It corresponds to the paid labor time due to the paid work time and, for too short duties, paying the difference time to a minimum work time of 6.5 hours. An exemplary run with NSGA-II shows the inverse correlation between paid time and span together with shift split. To reduce the paid time, too short duties are avoided and fewer, longer duties with shift splits increasing the span are created to cover the same demand more cost-efficiently. This can be observed in an example instance with maximum number of duties 26 in Figure 3. We see multiple sub-fronts with different numbers of shift splits. The size of points indicates tour changes, smaller means fewer. The costs vary within one hour of paid time per duty. The single triangle is a reference point derived with the weighted-sum simulated annealing approach from [11]. We see that a slight simultaneous reduction in paid time and span is still possible with only one shift split, but the front already came quite close to the reference point in this example run.

In further experiments, we add tour changes (4D) and ride time (5D) in the normalized objective space, see Table 4. We take the manually tuned parameters from before but grant 30 minutes of runtime per instance size unit, twice as before. For 4D, PSA with restarting performs substantially better than NSGA-II with +10% in hypervolume and being twice as close to the reference point. On the other hand, NSGA-II only performs half a million function evaluations, while PSA performs about 10 million in the 4D case. As we expect that the implementations can still be improved, the time-limited comparison has to be taken with a grain of salt. For 5D, this difference visibly decreases as more time might be required for convergence. Designing a crossover operator and studying long runs with the hypervolume slope as convergence criterion are planned continuations.

5.3 *Generalization performance*

Test instances. We validate our manually and automatically tuned algorithmic configurations for PSA with restarting on 30 unseen test instances $\mathcal{I}^{\text{test}}$ in three dimensions. Table 5 exhibits a similar trend as Table 2 for the training data, using the configurations from Table 3. Another Wilcoxon signed rank sum test shows this time no significant difference between PSA-restarting with 100 individuals and PSA-tuned, although a type II error should be kept in mind. Based on all tuning experiments, the starting temperature and the restarting parameter seemed to have the most impact on the performance. Performing a similar tuning and validation for 4D/5D is part of ongoing experiments.

Real-world problem instance. We finally compare the approaches on a real-world instance of a public transportation company of a medium sized Central European city derived from a cooperation with our business partner XIMES. We omit some details due to business secrecy reasons. The hard constraints are similar, while the objectives are somewhat different from the previous problem variant and designed together with domain experts. They are the sums to be minimized over the following employee-wise criteria:

- excess tour changes over one tour change
- long breaks excess over 90 minutes
- span, i.e., the duty duration including breaks
- deviation from a minimum work time of 6.5 hours
- deviation from a minimum duty part length of two hours (breaks split a shift into duty parts)

The tour changes and duty parts are penalized slightly superlinearly with an exponent of 1.2 to have a balancing effect over the employees. Long breaks excess relates to the undesirable shift splits from before, where longer breaks are linearly penalized. The minimum work time deviation is the main solution-dependent cost contribution.

We first compare runs optimizing for only the first three objectives. The population sizes are set to 200 individuals for both NSGA-II and PSA, otherwise we use the parameters tuned with irace from before and grant the algorithms one hour runtime. In Figure 4 we

Table 3. Tunable parameters for PSA with manual and raced configurations.

	PSA-basic									PSA-restarting		
	N^{ind}	T_0	β^{c}	T_{final}	M	N^{equi}	Beta[1,β]	b	w^{ms}	n^{re}	p^{re}	$I^{\mathrm{re}}_{\mathrm{rnd}}$
manual-3D	100	10^{-3}	0.99	10^{-7}	1	10	1	5	10	100	1.0	0
irace-3D	172	$2 \cdot 10^{-3}$	0.995	10^{-9}	0.1	20	1	12	13	63	1.0	1

Table 4. Training set: 4D/5D comparison non-dominated set size $|\mathcal{S}|$, minimum normalized distance to a PF reference point $\tilde{d}^{\mathrm{min}}_{\mathrm{ref}}$, and normalized hypervolume $|\tilde{H}|$ for different configurations and individuals N^{ind}.

			$\|\mathcal{S}\|$	$\tilde{d}^{\mathrm{min}}_{\mathrm{ref}}$		$\|\tilde{H}\|$		
dim	config	N^{ind}	mean	mean	std	mean	std	median
4D	NSGA-II	20	295	0.19	0.09	0.70	0.10	0.69
		200	668	0.17	0.09	0.69	0.15	0.65
	PSA-restarting	20	1 815	**0.08**	0.05	0.79	0.13	0.76
		200	1 940	0.08	0.05	**0.80**	0.12	**0.81**
5D	NSGA-II	20	464	0.20	0.08	0.65	0.11	**0.62**
		200	1 224	0.23	0.12	0.60	0.17	0.57
	PSA-restarting	20	5 560	0.17	0.12	0.61	0.21	0.55
		200	6 217	**0.16**	0.10	**0.65**	0.20	0.60

see resulting non-dominated solutions in three dimensions for such a NSGA-II and a *PSA-restarting* run, where the point size indicates wasteful costs by too short duties (smaller means less) and the too short duty parts are not shown. As expected, span and long breaks can be optimized quite well together, as less break time reduces the span and less span calls for a higher work to break time ratio. Tour changes and long breaks compete with each other, since fewer tour changes reduce the flexibility of duties to cover the demand which is then reclaimed by creating duties that are more dispersed. As mentioned, the size of the markers related to deviation from the minimum work time of 6.5 hours (not optimized for in this case), for which the bus operators need to wastefully pay this difference to the employee. Fewer long breaks seem also to lead to short duties as they bring flexibility in how to cover demand fluctuations.

In the final experiment, we consider all five objectives in 10 NSGA-II runs, which performed slightly better in 3D, yielding a mean/max normalized hypervolume of 0.56/0.61. The best run out of ten is visualized in a parallel coordinates plot in Figure 5. We observe that overall more time is required (span per employee) and more tour changes if we want to avoid too short shifts, i.e., reducing operation costs. Filters can be set on the objectives by the decision maker to learn about trade-offs and select a suitable solution to deploy.

6 Conclusions

We presented and compared two multi-objective optimization approaches based on Pareto simulated annealing (PSA) and NSGA-II for bus driver scheduling problems with up to five objectives. We qualitatively analyzed and numerically quantified different trade-off relations between costs for bus operators and shift convenience features for drivers like fewer tour changes and shift splits. We tuned our approaches manually and with irace on a training set and validated them on a test set and one real-world instance of a Central European city provided by a project partner. Both approaches allow to enable decision makers to perform more informed choices. A mutation only NSGA-II provides a solid baseline with few tunable parameters, while a tuned PSA variant with heavy randomized restarting from the non-dominated solution performs overall best in terms of average normalized hypervolume and distance to a reference point. Future work is concerned with investigating the impact of other randomized construction heuristics and implementing a crossover operator for NSGA-II based on the duty blocks of the parent individuals. Furthermore, learning informed non-dominated-solution selection strategies as part of the restarting, e.g., biasing towards regions where more gain in hypervolume is expected, could be an interesting research path.

Table 5. Test set: 3D comparison non-dominated set size $|\mathcal{S}$, minimum normalized distance to a PF reference point $\tilde{d}^{\mathrm{min}}_{\mathrm{ref}}$, and normalized hypervolume $|\tilde{H}|$ for different configurations and individuals N^{ind}.

			$\|\mathcal{S}\|$	$\tilde{d}^{\mathrm{min}}_{\mathrm{ref}}$		$\|\tilde{H}\|$		
dim	config	N^{ind}	mean	mean	std	mean	std	median
3D	NSGA-II	20	418	0.18	0.07	0.68	0.10	0.67
		100	636	0.19	0.09	0.67	0.13	0.63
	PSA-restarting	20	2 093	0.16	0.11	0.71	0.14	0.69
		100	2 216	0.15	0.10	0.72	0.14	0.70
	PSA-tuned	172	2 066	**0.14**	0.09	**0.73**	0.13	**0.71**

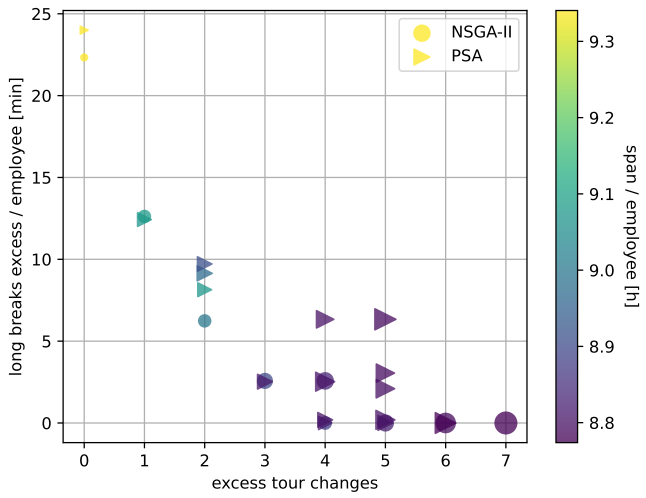

Figure 4. Real-world instance: 3D non-dominated solutions created by NSGA-II and PSA. The size of the points relates to the wasteful costs (smaller means less) in terms of deviation from a minimum duty span.

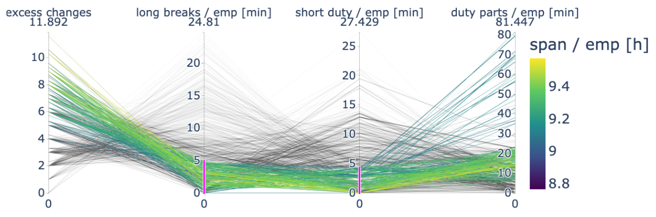

Figure 5. Real-world instance: 5D non-dominated solutions as parallel coordinates plot for real-world instance with filters set on long breaks excess and short work time.

Acknowledgements

The financial support by the Austrian Federal Ministry for Digital and Economic Affairs, the National Foundation for Research, Technology and Development and the Christian Doppler Research Association is gratefully acknowledged.

References

[1] T. Banerjee, A. Biswas, A. A. Shaikh, and A. K. Bhunia. An application of extended NSGA-II in interval valued multi-objective scheduling problem of crews. *Soft Computing*, pages 1–18, 2022.

[2] P. Czyzżak and A. Jaszkiewicz. Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization. *Journal of multi-criteria decision analysis*, 7(1):34–47, 1998.

[3] R. De Leone, P. Festa, and E. Marchitto. A bus driver scheduling problem: a new mathematical model and a GRASP approximate solution. *Journal of Heuristics*, 17(4):441–466, 2011.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[5] A. Drexl and Y. Nikulin. Multicriteria airport gate assignment and pareto simulated annealing. *IIE Transactions*, 40(4):385–397, 2008.

[6] M. Ehrgott and D. M. Ryan. Constructing robust crew schedules with bicriteria optimization. *Journal of multi-criteria decision analysis*, 11 (3):139–150, 2002.

[7] C. M. Fonseca, L. Paquete, and M. López-Ibánez. An improved dimension-sweep algorithm for the hypervolume indicator. In *2006 IEEE international conference on evolutionary computation*, pages 1157–1163. IEEE, 2006.

[8] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[9] O. J. Ibarra-Rojas, F. Delgado, R. Giesen, and J. C. Muñoz. Planning, operation, and control of bus transport systems: A literature review. *Transportation Research Part B: Methodological*, 77:38–75, 2015.

[10] F. Jolai and G. Assadipour. A hybrid cellular genetic algorithm for multi-objective crew scheduling problem. In *Hybrid Artificial Intelligence Systems: 5th International Conference, HAIS 2010, San Sebastián, Spain, June 23-25, 2010. Proceedings, Part I 5*, pages 359–367. Springer, 2010.

[11] L. Kletzander and N. Musliu. Solving large real-life bus driver scheduling problems with complex break constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 421–429, 2020.

[12] L. Kletzander and N. Musliu. Dynamic weight setting for personnel scheduling with many objectives. *Proceedings of the International Conference on Automated Planning and Scheduling*, 33(1):509–517, 2023.

[13] L. Kletzander, N. Musliu, J. Gärtner, T. Krennwallner, and W. Schafhauser. Exact methods for extended rotating workforce scheduling problems. *Proceedings of the International Conference on Automated Planning and Scheduling*, 29:519–527, 2019.

[14] L. Kletzander, N. Musliu, and P. Van Hentenryck. Branch and price for bus driver scheduling with complex break constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11853–11861, 2021.

[15] L. Kletzander, T. M. Mazzoli, and N. Musliu. Metaheuristic algorithms for the bus driver scheduling problem with complex break constraints. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 232–240, 2022.

[16] J. Li and R. S. Kwan. A fuzzy genetic algorithm for driver scheduling. *European Journal of Operational Research*, 147(2):334–344, 2003.

[17] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016.

[18] H. R. Lourenço, J. P. Paixão, and R. Portugal. Multiobjective metaheuristics for the bus driver scheduling problem. *Transportation science*, 35(3):331–343, 2001.

[19] T. M. Mazzoli, L. Kletzander, P. Van Hentenryck, and N. Musliu. Investigating large neighbourhood search for bus driver scheduling. In *Proceedings of the Internat. Conference on Automated Planning and Scheduling*, volume 34, pages 360–368, 2024.

[20] F. Mischek and N. Musliu. Preference explanation and decision support for multi-objective real-world test laboratory scheduling. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 378–386, 2024.

[21] G. Misitano, B. Afsar, G. Lárraga, and K. Miettinen. Towards explainable interactive multiobjective optimization: R-XIMO. *Autonomous Agents and Multi-Agent Systems*, 36(2):43, 2022.

[22] S. Verma, M. Pant, and V. Snasel. A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. *IEEE access*, 9:57757–57791, 2021.

[23] A. Wren and J.-M. Rousseau. Bus driver scheduling—an overview. In *Computer-Aided Transit Scheduling: Proceedings of the Sixth International Workshop on Computer-Aided Scheduling of Public Transport*, pages 173–187. Springer, 1995.