2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

XF-OPT/META: A Hyperparameter Optimization Framework Applied to the H-SPPBO Metaheuristic for the Dynamic TSP

Daniel Werner
Department of Computer Science
Leipzig University, Germany

Fatma Turna
Swarm Intelligence and Complex Systems Group
Department of Computer Science
Leipzig University, Germany
fatma.turna@uni-leipzig.de

Hoang Thanh Le
Swarm Intelligence and Complex Systems Group
Department of Computer Science
Leipzig University, Germany
lht@informatik.uni-leipzig.de

Martin Middendorf
Swarm Intelligence and Complex Systems Group
Department of Computer Science
Leipzig University, Germany
middendorf@informatik.uni-leipzig.de

***Abstract*—This paper has two objectives. Firstly, to introduce a new framework XF-OPT/META for testing and comparing Hyperparameter Optimization (HPO) methods. The framework supports model-free methods, e.g., Random Search (RS), as well as model-based methods, such as Bayesian Optimization (BO), with various surrogate models. Due to the generalized and modular structure of the XF-OPT/META framework, it can be easily extended to other optimization methods for different optimization problems. The second objective is to empirically compare the performance of various HPO methods for population-based metaheuristics. For that the XF-OPT/META framework is used to apply HPO methods to the Hierarchical Simple Probabilistic Population-Based Optimization (H-SPPBO) metaheuristic for the Dynamic Traveling Salesperson Problem (DTSP) and to calculate high-performing parameter values for H-SPPBO. Promising results are obtained using the parameter values found by BO. In particular, a parameter set obtained with Gradient-Boosted Regression Trees (GBRT) outperforms a reference parameter set for H-SPPBO from an existing study.**

***Index Terms*—Hyperparameter Optimization, Metaheuristics, Parameter Tuning, Dynamic Traveling Salesperson Problem, Hierarchical Simple Probabilistic Population-Based Optimization (H-SPPBO), Bayesian Optimization**

I. Introduction

In the recent years, powerful algorithms have been developed for a variety of problems, including many NP-hard optimization problems. However, these algorithms' versatility and high performance stem from parameters that can be set by the user to carefully adjust the algorithm's behavior. In order to thoroughly utilize an algorithms potential, efficient and easily applicable methods are needed to determine high-performing parameter values for these algorithms. *Hyperparameter Optimization* (HPO) refers to the optimization problem of finding such parameter values in order to maximize an algorithm's performance. For this problem, the framework *XF-OPT/META* is presented in this work which provides various methods for HPO as well as tools to compare and evaluate HPO algorithms. The framework is used to empirically compare different HPO methods.

For a comparison the HPO methods are applied to the *Hierarchical Simple Probabilistic Population-Based Optimization* (H-SPPBO) in order to tune its parameters for the dynamic TSP (DTSP). H-SPPBO is a metaheuristic framework developed by Kupfer et al. [1] for designing algorithms that incorporate populations, i.e., sets of previously generated solutions. It extends the SPPBO framework [2] which unifies various population-based algorithms, such as the Population-Based Ant Colony Optimization (PACO) [3] and Simplified Swarm Optimization (SSO) [4]. The central component of both frameworks is formed by Solution Creating Entities (SCE, similar to ants in an ACO algorithm). In H-SPPBO the SCEs are organized in a hierarchical tree structure where swaps occur if an SCE obtains better solutions than its parent SCE. The swaps can be used to detect dynamic changes [1], making H-SPPBO suitable for dynamic optimization problems, such as the DTSP.

Metaheuristics belonging to the H-SPPBO framework feature a large number of parameters to adjust the algorithm's behavior, such as the weights for different populations and the strength of the influence of heuristic components or of randomness on constructing new solutions [1], [2]. Thus, H-SPPBO metaheuristics provide a promising case for comparing HPO methods and using the XF-OPT/META framework. So far there is not much research available on comparing HPO methods for tuning metaheuristics.

The remainder of this paper is structured as follows. In Section II, an overview of related work is given. Methods to optimize hyperparameters for metaheuristics are covered in

Section III. Experimental results are presented in Section IV and conclusions are given in Section V.

II. Related Work

The use of metaheuristics for dynamic optimization problems began more than 20 years ago. One of the earlier works that discusses the theoretical concept of the dynamic TSP using evolutionary computation is [5]. Another example is [6] which shows that adapting an Ant Colony Optimization (ACO) algorithm to dynamic changes of TSP instances is faster than simply restarting the algorithm. Guntsch and Middendorf [7] proposed pheromone modification strategies for ACO to react to dynamic insertions or deletions of cities of a TSP instance. Similar methods are presented in [8]. Since then numerous studies have been performed on dynamic optimization. A survey is given by Mavrovouniotis et al. [9] who review existing studies on Swarm intelligence dynamic optimization (SIDO). They group the most important applications of SIDO into continuous or discrete problems while discussing and classifying strategies that enhance Swarm intelligence algorithms to cope with dynamic changes. Another survey is by Yazdani et al. [10], [11] on evolutionary continuous optimization over the last two decades.

The problem of choosing appropriate values for parameters of a metaheuristics (such parameters are also called hyperparameter) has always been an important consideration in research. Eiben et al. [12] give a thorough review and analysis of the different options in the field of parameter tuning. Although they focus on Genetic Algorithms, they provide a taxonomy for parameter optimization. Talbi [13] classifies parameter tuning methods into offline methods and online methods. The offline methods are further split into meta-level and base-level methods, whereas for online methods a distinction is made between dynamic and adaptive updating methods, which are also further classified.

In a review of parameter optimization by Wong et al. [14], it is concluded that parameter tuning plays an important role in the exploratory and exploitative behavior of ACO. The most popular reference in manual parameter tuning for ACO is the original work by Dorigo et al. [15], [16]. They performed several experiments with different parameter value combinations on TSP instances and the obtained parameter values serve as a baseline for many subsequent studies. An updated version of these "good" parameter values and variations of the original ACO were published in [17]. A self-adaptive control scheme has been proposed by Hao et al. in [18] which constructs a combinatorial problem of the parameter value search problem and applies PSO to optimize the obtained values, leading to promising results in TSP benchmarks.

Another promising approach to optimize parameters as a type of black-box optimization problem is *Bayesian optimization* (BO) which was first introduced in [19]. Since then, BO has been applied to a variety of problems, including hyperparameter optimization problems. For example, Snoek et al. [20] consider the auto-tuning problem in the framework of Bayesian optimization, in which the generalization performance of a learning algorithm is modeled as a Gaussian process (GP). The traceable backscatter induced by the GP allows the information gathered from previous experiments to be efficiently used, providing optimal choices about which parameter values to try next. In a study performed by van Hoof and Vanschoren [21], a new surrogate model based on gradient boosting is proposed. This model uses quantitative regression to provide optimistic estimates of the performance of unobserved hyperparameter settings which is combined with distance measurements between unobserved and observed hyperparameter settings to aid in tuning exploration. Finally, an approach similar to this work, but more narrowly focused, is investigated by Yin and Wijk [22]. They use Random Search and Bayesian optimization (BO) to tune the parameters of a classical ACO algorithm on multiple instances of the asymmetric TSP. Their approach to tuning parameters leads to promising results, without requiring a priori knowledge of the problem or the algorithm.

III. Hyperparameter Optimization For Metaheuristics

Based on the description of [23] to the problems in this study, the hyperparameter optimization problem for metaheuristics can be formulated as follows: Let A be an optimization algorithm (here, a metaheuristic) and Λ be the set of possible (hyper)parameter configurations (i.e., a vector of parameter values) for A. For a vector $\lambda \in \Lambda$ the algorithm A with parameter values according to λ is denoted by A_λ. Let f_A be a function that evaluates the "performance" of algorithm A when it is executed with parameter values λ. As an example, $f_A(\lambda)$ can be defined to be the best obtained solution quality of A_λ during a run with some problem instance. In the case of metaheuristics, typically, A incorporates stochastic elements so that $f_A(\lambda)$ is a random variable.

Assuming that small values for f_A are desirable, the hyperparameter optimization problem is to find an optimal parameter configuration λ that minimizes the expected value of $f_A(\lambda)$:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} E[f_A(\lambda)] \tag{1}$$

Usually it is not known how f_A changes depending on the used parameter values λ. Furthermore, it is possible that f_A (as a black-box function) can be computationally expensive to evaluate. Due to this, approaches like *Random Search* (RS) where configurations from Λ are randomly sampled might require a large computational effort before promising parameter values are obtained.

An approach to deal with this is to approximate f_A by using a surrogate model $\hat{f}_A$ that is easier to compute and "learned" by sampling the original function f_A at selected parameter configurations λ for the metaheuristic A. Afterwards, the model $\hat{f}_A$ is primarily used instead of f_A to test parameter values. In the following, let $\mathcal{D}_0$ be the set of pairs (λ_i, f_i) with sampled parameter values λ_i and observed values $f_i = f_A(\lambda_i)$ that is used to construct $\hat{f}_A$.

In the context of *Bayesian optimization* (BO), which is not just an algorithm used for HPO, but a complete framework for the global optimization of (expensive) black-box functions, the function $\hat{f}_A$ constructed using the sampled values λ_i is referred to as a *prior distribution*. It provides information on "uncertainty" in unsampled areas of Λ as well as information on promising areas that potentially lead to a low value for f_A. Based on this information new data points (λ_j, f_j) in Λ are sampled, which, in combination with the data in $\mathcal{D}_0$ leads to a larger data set $\mathcal{D}'$ that is used to construct a new function $\hat{f}'_A$ (*posterior distribution*).

The selection of new parameter configurations used for the new data set $\mathcal{D}'$ is determined by an *acquisition function* $u : \Lambda \to \mathbb{R}_{>0}$. Depending on the choice of u, it is possible to adjust the sampling process to focus more on exploration (reducing uncertainty in unsampled areas) or exploitation (i.e., more sampling in well-performing areas). In the next iteration, $\hat{f}'_A$ becomes the prior distribution used to sample new data points and construct a new posterior distribution that takes all previously observed data points into account. This process is repeated for a number of $\texttt{n}_{\texttt{calls}}$ iterations.

In contrast to Random Search (as a model-free method), Bayesian Optimization incorporates all previously tested data points and thus the knowledge obtained during the optimization process to construct surrogate models which can lead to more promising parameter values. Furthermore, evaluating a surrogate function $\hat{f}_A$ is often computationally easier than running the algorithm A to evaluate f_A. In this work, the following surrogate models are used and compared:

- Gaussian Process (GP), which is developed as an extension of the multivariate Gaussian distribution [22],
- Extra-Trees (ET) proposed by Geurts et al. [24] and
- Gradient Boosted Regression Trees (GBRT) as an ensemble method that uses decision trees as base learners to produce an ensemble prediction [25].

In short, three HPO methods based on Bayesian optimization with these three models are used in the computational experiments in the next section. In addition, Random Search (RS) [26] is used as a baseline algorithm leading to a total of four *HPO algorithms* tested in this work. Regarding acquisition functions, the Probability of Improvement (PI) [27] and Expected Improvement (EI) [28] are used. The choice of these functions and models is based on the Machine Learning library `scikit-optimize` [29], which also provides ET and GBRT as regression is commonly used in the field of ensemble learning.

For all of the methods named above, this work provides a extensive software package, written in Python and called ***EX**perimentation **F**ramework and (Hyper-)Parameter **Opt**imization for **Meta**heuristics* (XF-OPT/META). Every aspect of this package is modular (allowing for easy replacement of components) and straightforward to configure (allowing for adaptation to algorithms other than H-SPPBO and other optimization problems outside of the dynamic TSP). Furthermore, it provides a variety of logging functionalities and methods to evaluate and visualize results. The code for this package is available at [30] with an extensive documentation to increase the comprehensibility and reproducibility of the results presented in this work.

IV. Computational Results

In this section, the XF-OPT/META framework is used to apply the hyperparameter optimization methods presented above to the H-SPPBO metaheuristic on the dynamic TSP. In the following, the methodology for performing the experiments and evaluating the results is presented.

A. Choice of Problem Instances

For the experiments, the same problem instances as in [1] and [2] from the TSPLIB benchmark were used to compare the results. Furthermore, additional instances were selected to investigate the performance of the methods on instances with different characteristics and a higher number of nodes. In total, 10 TSP instances are used which are split into 5 groups that each contain a "smaller" instance ($50 \leq n \leq 250$) and a "larger" instance ($250 < n \leq 450$). The instances are chosen such that instances within the same group have similar characteristics, but differ from instances in other groups. A full list of the used instances and a description of the clustering methodology used to select them can be found in [30].

B. Experimental Setup for H-SPPBO

The basis of the experiments in this section is the H-SPPBO algorithm A which is executed for 2600 iterations with given parameter values λ on a DTSP instance. During a so-called *H-SPPBO execution* A_λ, the H-SPPBO algorithm first runs for 2000 iterations without changes in the problem instance in order to minimize effects caused by randomness during the initial iterations and to allow the algorithm to reach a steady state. After 2000 iterations, $n \cdot (C/2)\%$ pairs of distinct nodes are selected randomly and the node positions in each pair are swapped, with C being a parameter of the DTSP instance. This dynamic change occurs 6 times every 100 iterations, starting at iteration 2000. This type of dynamic and the length for each run is based on the study [1] where an H-SPPBO algorithm is also used.

With the H-SPPBO executions on dynamic TSP instances as described above, the HPO methods presented in Section III were used to determine good parameter configurations for the algorithm. In particular, the following parameters were tuned using the algorithms implemented in the XF-OPT/META framework:

- $w_{persprev} \in [0.001, 0.99]$: the weight for $P_{persprev}$ (the SCE's personal previous solution),
- $w_{persbest} \in [0.001, 0.99]$ the weight for $P_{persprev}$,(the SCE's best personal solution),
- $w_{parentbest} \in [0.001, 0.99]$: the weight for $P_{parentbest}$ (the personal best solution of the SCE's parent),
- $\alpha \in [0, 10] \cap \mathbb{N}_0$: influence of the probabilistic/non-heuristic component,
- $\beta \in [0, 10] \cap \mathbb{N}_0$: influence of the heuristic component,

- $\theta \in [0.1, 0.5]$: threshold for detecting a change within the SCE tree,
- $H \in \{H_{full}, H_{partial}\}$: type of reaction algorithm used when dynamic changes in the instance are detected.

Since the dynamic TSP is considered as the optimization problem for the H-SPPBO algorithm, it is not sufficient to evaluate the performance of an H-SPPBO execution using only the solution quality at the end of the run (after 2600 iterations). The underlying TSP instance changes multiple times during an execution. Due to this, the function f_A used to evaluate the performance of an H-SPPBO execution is defined as an average value that, besides the performance at the end of the run also takes into account the obtained solution quality right before the dynamic changes at iterations $t \in \{2000, 2100, 2200, 2300, 2400, 2500\}$ occur.

Since the length L^* of an optimal TSP tour is known for all instances used in this study and since the value L^* does not change with the considered dynamic, it is possible to use the *relative percentage difference* $RPD = (L - L^*)/L^*$ to describe the quality of a TSP tour with length L when calculating f_A. This makes it possible to evaluate the parameter optimizers over different problem instances with varying optimal lengths L^*.

The experiments performed in this section can be divided into two main parts. In the first part, all four HPO algorithms $O \in \{\text{RS}, \text{BO-GP}, \text{BO-ET}, \text{BO-GBRT}\}$ are run $k_1 = 3$ times each, where each *HPO run* independently optimizes the parameter values of the H-SPPBO algorithm using $\texttt{n}_{\texttt{calls}} = 30$ H-SPPBO executions. During the HPO runs, the improvement in f_A over time is logged and averaged over the $k_1 = 3$ HPO runs, with the goal of determining the most suitable HPO algorithm $\hat{O}$. Algorithm $\hat{O}$ is then executed multiple times to calculate promising parameter values $\hat{\lambda}$ for H-SPPBO. The second part consists of running H-SPPBO multiple times with the obtained parameter values $\hat{\lambda}$ and comparing the results with other results from an existing study [1]. Whenever an HPO algorithm with Bayesian optimization is used, the size of the initial data set $\mathcal{D}_0$ to construct the initial surrogate model $\hat{f}_A$ is set to $|\mathcal{D}_0| = 10$. For the acquisition functions [27], [28] the trade-off parameter ξ is set to $\xi = 0.01$.

C. Choosing the Hyperparameter Optimization Algorithm

The first part of the experiments was performed as described above, with using the 5 smaller instances in each group and the dynamic intensity set to $C = 0.25$, leading to a total of 1800 H-SPPBO executions. The averaged convergence curves for the compared HPO algorithms are shown in Fig. 1. The curves show that GBRT is the best performing method in individual comparisons according to convergence behavior among HPO algorithms. It can be seen that the average convergence behavior of GBRT provided a lead of approximately 1.5% in the final RPD over the next best method, ET.

Apart from visually inspecting the convergence behavior of the four HPO algorithms, it is also possible to look at the area under the curve (AUC) of the plots and the best RPD value obtained, i.e., the RPD value at $\texttt{n}_{\texttt{calls}} = 30$, denoted as RPD^* in the following. These values are shown in Table I for all HPO algorithms, averaged over all $k_1 = 3$ HPO runs, for the "small" TSP instances in each of the 5 groups, and their mean.

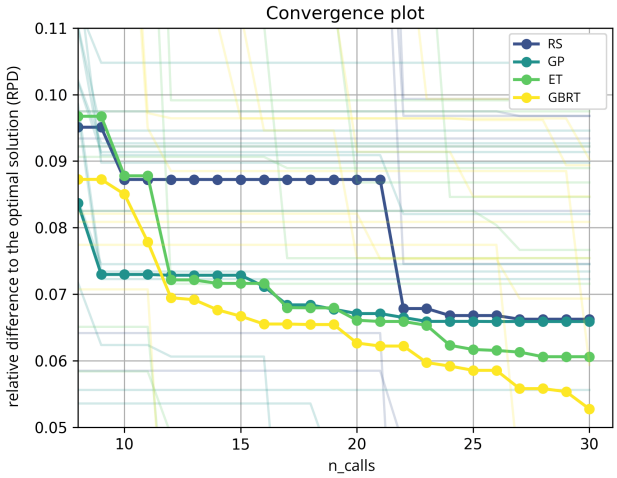

Fig. 1. Convergence plot of the average over TSP instances eil51, berlin52, pr136, pr226, d198, comparing for the 4 HPO algorithms. The bold lines show the averaged values, whereas the semitransparent lines show the convergence curves used to calculate the averaged curve for each HPO algorithm.

TABLE I
THE AUC AND MINIMUM RELATIVE SOLUTION QUALITY (RPD^*) AT THE LAST H-SPPBO EXECUTION FOR EACH HPO ALGORITHM AND SMALLER TSP INSTANCE, AVERAGED OVER ALL $k_1 = 3$ OPTIMIZATION RUNS, AND THE MEAN OVER ALL INSTANCES. THE BEST VALUES FOR EACH COLUMN ARE SHOWN IN BOLD.

	eil51		berlin52		pr136	
	AUC	RPD^*	AUC	RPD^*	AUC	RPD^*
RS	0.830	0.026	0.347	**0.001**	2.266	0.097
GP	0.900	0.034	0.650	0.019	1.850	0.093
ET	0.819	**0.020**	0.227	0.005	**1.809**	**0.075**
GBRT	**0.601**	0.024	**0.159**	0.003	1.948	**0.075**

	pr226		d198		mean	
	AUC	RPD^*	AUC	RPD^*	AUC	RPD^*
RS	1.837	0.075	2.139	0.092	1.484	0.058
GP	**1.394**	0.072	**1.698**	0.082	1.298	0.060
ET	1.547	0.072	1.940	0.085	1.268	0.051
GBRT	1.497	**0.042**	1.745	**0.081**	**1.190**	**0.045**

The mean also show that GBRT is the algorithm that converges fastest ($\overline{AUC} = 1.190$) to the best performing parameter configuration ($RPD^* = 4.5\%$). Even in instances like pr226 and d198 with a higher number of nodes, it performed at least second best or better, implying good search consistency across the parameter space Λ.

To validate the results above, statistical tests were performed using the measured data. One test is the KruskalWallis H test, with each of the four HPO algorithms as a sample group, separated by the five instances used. The results for each instance are shown in Table II. A significance level of 0.05 was chosen to reject the null hypothesis that the data for all sample groups originates from the same distribution. Based

TABLE II
RESULTS OF THE KRUSKALWALLIS TEST, INCLUDING THE TEST STATISTIC H AND p-VALUES APPLIED TO ALL FOUR HPO ALGORITHMS, SEPARATED BY TSP INSTANCE.

Instance	H	p-value
eil51	47.893	2.244×10^{-10}
berlin52	46.960	3.544×10^{-10}
pr136	29.898	1.450×10^{-6}
pr226	57.998	1.574×10^{-12}
d198	46.514	4.410×10^{-10}

on the resulting p-values, the null hypothesis can be rejected, suggesting that the four HPO algorithms show a significantly different convergence behavior from each other.

Furthermore, a post-hoc ConoverIman test was then performed to gain more insight into which specific pairs of optimization algorithms differ and how. The resulting test statistics are shown in Table III. The sign of the test statistic indicates whether the difference is positive or negative, and statistically significant differences are highlighted in green. It can be seen that GBRT obtains significantly smaller RPD^* values than the other algorithms for the instance eil51. For the other instances, the differences are significant to RS and GP for most of the instances, whereas for for the comparison between GBRT and ET a significant difference can only be seen for the instance berlin52 and eil51.

Furthermore, the differences between ET and GP are statistically significant, confirming the observations from the convergence plot in Fig. 1. Regarding the instance berlin52, GBRT also performs better than GP and ET, but interestingly, the observed difference was not significant between GBRT and RS. However, the convergence plot in Fig. 1 as well as the results for the instances pr136, pr226, and d198 indicate that RS tends to perform worse than the other three HPO methods.

To summarize the results of the first part, Bayesian optimization using Gradient Boosted Regression Trees (GBRT) shows the fastest convergence and tends to reach the best solution quality, while providing fairly robust results over the considered instances. Therefore, it is used as HPO algorithm $\hat{O}$ that performs the second part of the experiments.

D. Comparison with Existing Parameter Values

In the second part, as described above, the best performing HPO algorithm $\hat{O}$ = GBRT from the first part is executed $k_2 = 6$ times on the smaller TSP instances in each group with $\texttt{n}_{\texttt{calls}}$ set to $\texttt{n}_{\texttt{calls}} = 60$ to calculate suitable parameter values $\hat{\lambda}$ for H-SPPBO. Furthermore, different dynamic intensities $C \in \{0.1, 0.25, 0.5\}$ are used so that the HPO algorithm is used to find parameter configurations for these TSP instances with different dynamics. In total, 5400 executions of the H-SPPBO are performed. By running the HPO algorithm $k_2 = 6$ times, 6 different parameter configurations are obtained for each of the 5 smaller TSP instances and dynamic intensities C. From these, the best performing configuration with the best value RPD^* was selected, leading to a total of 15 parameter configurations $\hat{\lambda}$ in total. A full list of these parameter configurations is available online at [30].

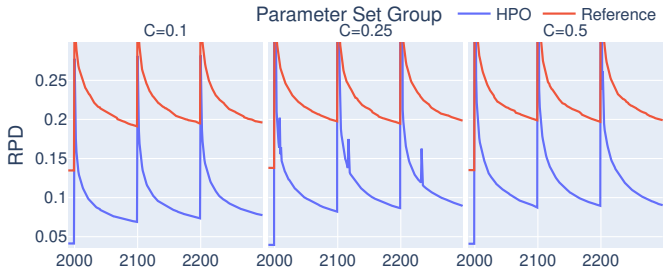

Fig. 2. Line plots showing the relative solution quality RPD over the first 300 dynamic iterations (starting just before iteration 2000) for both parameter configurations $\hat{\lambda}$ and λ_{ref} with a plot for each dynamic intensity $C \in \{0.1, 0.25, 0.5\}$, averaged over all ten TSP instances.

Afterwards, the H-SPPBO algorithm is executed multiple times with the parameter configuration $\hat{\lambda}$ obtained above for each of the 5 groups and the results are compared with the parameter values used in an existing study [1]. Similar to that study, the dynamic intensities $C \in \{0.1, 0.25, 0.5\}$ are used. The H-SPPBO algorithm is now executed on both smaller and larger instance in each group (10 instances in total, with the parameter values in each group being the same) using the three values for C and the results are averaged over 20 replications, leading to 600 executions of the H-SPPBO algorithm. For the larger instance in each group, the same parameter configuration was used that was obtained for the smaller instance in the same group. This provides insight on how suitable the parameter values are for instances with similar characteristics, even when the size of the instance differs. In order to compare the results with the study [1], the same runs are also executed using a generally well-performing reference set λ_{ref} from that study, leading to an additional 600 runs and a total of 1200 H-SPPBO executions.

Fig. 2 shows the results for λ_{ref} and $\hat{\lambda}$ for each dynamic intensity $C \in \{0.1, 0.25, 0.5\}$, averaged over all TSP instances. It can be seen that just before each dynamic event, the reference parameter configuration λ_{ref} consistently produces almost the same solution quality $RPD_{ref} = 0.2$. However, the parameter values obtained by HPO clearly outperform them at every point in the dynamic runtime, including the times just before a dynamic change occurs.

V. CONCLUSION

In this work the XF-OPT/META framework was developed which provides various methods for Hyperparameter Optimization (HPO), such as Random Search (RS) or Bayesian optimization (BO) with various surrogate models for use with metaheuristics. This framework was used to experimentally compare different HPO methods by tuning the parameters of an Hierarchical Simple Probabilistic Population-Based Optimization algorithm (H-SPPBO) for the Dynamic Traveling Salesperson Problem (DTSP).

The results show that on average, Bayesian optimization with Gradient Boosted Regression Trees (GBRT) obtains the best results in this study when compared to other surrogate

TABLE III
VALUES FOR THE TEST STATISTIC OF THE POST-HOC CONOVERIMAN TESTS FOR ALL FOUR HPO ALGORITHMS AS A SAMPLE GROUP, SEPARATED BY TSP INSTANCE. A GREEN CELL INDICATES THAT THE NULL HYPOTHESIS IS REJECTED (AFTER BONFERRONI CORRECTION).

Instance	RS vs. GP	RS vs. ET	RS vs. GBRT	GP vs. ET	GP vs. GBRT	ET vs. GBRT
eil51	−1.644	1.099	8.358	2.743	10.002	7.259
berlin52	−7.486	−1.002	2.668	6.485	10.154	3.669
pr136	5.400	6.223	4.491	0.823	−0.909	−1.731
pr226	14.432	7.766	8.290	−6.666	−6.142	0.524
d198	10.207	3.936	6.083	−6.271	−4.125	2.146

models. Not only did the parameter configurations obtained by Bayesian optimization perform well in the dynamic phase of the DTSP, but the tuned algorithms also obtained improved results during the static phase before the problem instance is changed and outperformed reference parameter values for H-SPPBO from an existing study.

Furthermore, promising results were reached when using the same parameter values for larger instances with similar characteristics. This indicates that HPO is able to use smaller instances to find parameter configurations that generalize to larger instances, illustrating the potential that lies in the combination of HPO and metaheuristics. Moreover, the XF-OPT/META framework provides modularity and interfaces for each of the main modules, in particular the underlying optimization problem, the HPO algorithm, and the metaheuristic to be tuned, allowing straightforward extensions of the framework to other problems and algorithms for further research.

REFERENCES

[1] E. Kupfer, H. T. Le, J. Zitt, Y.-C. Lin, and M. Middendorf, "A hierarchical simple probabilistic population-based algorithm applied to the dynamic TSP," in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2021, pp. 1–8.

[2] Y.-C. Lin, M. Clauss, and M. Middendorf, "Simple probabilistic population-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 245–262, 2015.

[3] M. Guntsch and M. Middendorf, "A population based approach for ACO," in *Applications of Evolutionary Computing*. Springer Berlin Heidelberg, 2002, pp. 72–81.

[4] W.-C. Yeh, "A two-stage discrete particle swarm optimization for the problem of multiple multi-level redundancy allocation in series systems," *Expert Systems with Applications*, vol. 36, no. 5, pp. 9192–9200, 2009.

[5] Z.-C. Huang, X.-L. Hu, and S.-D. Chen, "Dynamic traveling salesman problem based on evolutionary computation," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, vol. 2. IEEE, 2001, pp. 1283–1288.

[6] D. Angus and T. Hendtlass, "Ant colony optimisation applied to a dynamically changing problem," in *Developments in Applied Artificial Intelligence: 15th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE 2002 Cairns, Australia, June 17–20, 2002 Proceedings 15*. Springer, 2002, pp. 618–627.

[7] M. Guntsch and M. Middendorf, "Pheromone modification strategies for ant algorithms applied to dynamic TSP," in *Applications of Evolutionary Computing: EvoWorkshops 2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM Como, Italy, April 18–20, 2001 Proceedings*. Springer, 2001, pp. 213–222.

[8] C. J. Eyckelhof and M. Snoek, "Ant systems for a dynamic TSP: Ants caught in a traffic jam," in *Ant Algorithms: Third International Workshop, ANTS 2002 Brussels, Belgium, September 12–14, 2002 Proceedings*. Springer, 2002, pp. 88–99.

[9] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: Algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.

[10] D. Yazdani, R. Cheng, D. Yazdan, J. Branke, Y. Jin, and X. Yao, "A survey of evolutionary continuous dynamic optimization over two decadespart a," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 609–629, 2021.

[11] ——, "A survey of evolutionary continuous dynamic optimization over two decadespart b," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 630–650, 2021.

[12] Á. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, 1999.

[13] E.-G. Talbi, *Metaheuristics: from design to implementation*, ser. Wiley Series on Parallel and Distributed Computing. Hoboken, NJ: Wiley-Blackwell, 2009.

[14] K. Y. Wong and Komarudin, "Parameter tuning for ant colony optimization: A review," in *2008 International Conference on Computer and Communication Engineering*. IEEE, 2008, pp. 542–545.

[15] M. Dorigo, A. Colorni, and V. Maniezzo, "Ant system: An autocatalytic optimizing process," *Dipartimento Di Elettronica, Politecnico Di Milano, Milan, Italy*, 1991.

[16] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[17] M. Dorigo and T. Sttzle, *Ant Colony Optimization*. Cambridge, MA: The MIT Press, 06 2004.

[18] Z.-F. Hao, R.-C. Cai, and H. Huang, "An adaptive parameter control strategy for ACO," in *2006 International Conference on Machine Learning and Cybernetics*. IEEE, 2006, pp. 203–206.

[19] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization Techniques IFIP Technical Conference Novosibirsk, July 1–7, 1974*, G. I. Marchuk, Ed., 1975, pp. 400–404.

[20] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, vol. 25. Curran Associates, Inc., 2012.

[21] J. van Hoof and J. Vanschoren, "Hyperboost: Hyperparameter optimization by gradient boosting surrogate models," *arXiv preprint arXiv:2101.02289*, 2021.

[22] E. Yin and K. Wijk, "Bayesian parameter tuning of the ant colony optimization algorithm: Applied to the asymmetric traveling salesman problem," Bachelor's Thesis, KTH Royal Institute of Technology. School of Electrical Engineering and Computer Science, 2021.

[23] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Springer International Publishing, Cham, 2019, pp. 3–33.

[24] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 2006.

[25] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *The Annals of Statistics*, pp. 1189–1232, 2001.

[26] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization." *Journal of Machine Learning Research*, vol. 13, no. 2, 2012.

[27] H. J. Kushner, "A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, 1964.

[28] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global Optimization*, vol. 13, no. 4, pp. 455–492, 1998.

[29] T. Head, M. Kumar, H. Nahrstaedt, G. Louppe, and I. Shcherbatyi, "scikit-optimize/scikit-optimize: v0. 8.1," *Zenodo*, 2020.

[30] "XF-OPT/META: Experimentation framework and (hyper-)parameter optimization for metaheuristics," https://github.com/Bettvorleger/XF-OPT-META, accessed: 2023-06-17.