IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Automated Online Experiment-Driven Adaptation – Mechanics and Cost Aspects

Ilias Gerostathopoulos[1], Frantisek Plasil[2], Christian Prehofer[3], Janek Thomas[4], Bernd Bischl[4]

[1]Vrije Universiteit Amsterdam, Amsterdam, Netherlands
[2]Charles University in Prague, Prague, Czech Republic
[3]DENSO Automotive Germany, Munich, Germany and Technical University of Munich, Munich, Germany
[4]Ludwig Maximilian University of Munich, Munich, Germany

Corresponding author: Ilias Gerostathopoulos (e-mail: i.g.gerostathopoulos@vu.nl).

The work has been partially supported by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. This work was supported by the Bavarian Ministry for Economic Affairs, Infrastructure, Transport and Technology through the Center for Analytics-Data-Applications (ADA-Center) within the framework of "BAYERN DIGITAL II"

ABSTRACT As modern software-intensive systems become larger, more complex, and more customizable, it is desirable to optimize their functionality by runtime adaptations. However, in most cases it is infeasible to fully model and predict their behavior in advance, which is a classical requirement of runtime self-adaptation. To address this problem, we propose their self-adaptation based on a sequence of online experiments carried out in a production environment. The key idea is to evaluate each experiment by data analysis and determine the next potential experiment via an optimization strategy. The feasibility of the approach is illustrated on a use case devoted to online self-adaptation of traffic navigation where Bayesian optimization, grid search, and local search are employed as the optimization strategies. Furthermore, the cost of the experiments is discussed and three key cost components are examined—time cost, adaptation cost, and endurability cost.

INDEX TERMS experimentation, optimization, self-adaptation

I. INTRODUCTION

Large software-intensive systems (LSIS) are becoming more dynamic, adaptive, and data-driven. An example is a smart grid, for which, contrary to the classical power grid, no clear models of consumption and production are available. This is mostly due to the need of high flexibility in distributed energy production and consumption.

With this in mind, we focus on LSIS that need to change their architecture or configuration at runtime in order to adapt to changes in their environment. The objective of such adaptation is to optimize the LSIS's functionality with respect to a specific goal. Adaptation typically assumes the existence of models that describe the behavior of the system and its environment and allow for exploring the (potentially large) space of adaptation options to select from. Nevertheless, a common problem in LSIS is the lack of accurate, efficient, and up-to-date models that can be used in the process of self-adaptation. For instance, in order to optimize the capacity of a highway, a model connecting the average speed and traffic density to adaptation actions such as setting dynamic speed limits or opening and closing extra lanes, would be needed. Typically, this would be a coarse-grained, empirically constructed model. In this context, it is a challenge to tailor such a model to the specifics of a particular LSIS and keep it continuously updated in face of changes in the environment (consider, e.g., a highway close to a city busy with commuters and one with detours in a hilly remote area).

Instead of creating such models, we propose to self-adapt LSIS to meet an optimization goal at runtime via automated online experiment-driven adaptation (AOEDA). We build on large-scale experiments employing A/B testing used by many organizations including Microsoft [1], [2], Google [3], and Uber [4] to evaluate different functionality variants of their systems. AOEDA is a novel approach to optimize a system on-the-fly by evaluating variants of system parameters settings – system configurations. More general than A/B testing, AOEDA (i) is initiated by the system itself (not by its operators), and (ii) aims at the dynamic (online) identification of an optimal configuration and its activation. Similar to A/B testing, AOEDA employs experimentation upon production systems with real users, where the cost of

experimentation is an important factor, potential negative user reactions included (e.g. dropping the use of a system service). In particular, we identify three key components of experimentation cost: (i) Cost related to the time needed for an optimization round of the LSIS in question (time cost); (ii) cost of applying a new configuration to the LSIS (adaptation cost); (iii) cost related to the user dissatisfaction potentially caused by a new configuration (endurability cost).

In this paper, we describe the main ingredients of AOEDA, focus on the three essential components of experimentation cost, showcase the related trade-offs in the use of different optimization strategies, and discuss our insights on the use of runtime optimization/adaptation via online experimentation.

The rest of the paper is organized as follows. Sec. II presents the use case that will be used for motivating and illustrating AOEDA. Sec. III describes the basic concepts and challenges of AOEDA, whilst Sec. IV elaborates on three optimization strategies that can be used with AOEDA. Sec. V demonstrates the different cost aspects of our approach on the use case and Sec. VI discusses key learnings from applying AOEDA and recommended best practices. Finally, Sec. VII discusses other approaches focusing on cost aspects, and the concluding Sec. VIII summarizes the contribution. .

II. USE CASE: TRAFFIC NAVIGATION

As a simple LSIS use case, we consider CrowdNav [5]—a navigation service used by cars to help optimize their routing from an origin to a destination in a street network. The optimization goal is to reduce the average trip time by smartly dispensing the street traffic, even though this may lead to forcing some cars to take sub-optimal routes.

The navigation service runs the Dijkstra algorithm to determine the routes with the smallest sum of routing weights (we could also have used other graph search algorithms such as RRT [6]). The weights are assigned to streets according to map information (statically) and traffic intensity (dynamically). The weight of a street is proportional (i) to its length and inversely proportional to the maximal speed allowed in it, and (ii) to the average time to pass through the street (a proxy for traffic intensity), based on data reported by cars.

CrowdNav features a number of system parameters that can be changed at runtime and control the evaluation of routing weights. For example, one system parameter controls the importance of map information while another one of traffic intensity. To set appropriate values of the system parameters (configurations), the impact of their particular selection on the optimization goal needs to be modeled. Notice that such a model would be specific to the situation at hand: an appropriate configuration when many cars are in operation does not have to work in the case of few cars. Similarly, the model would depend on the actual status of the street network, day of the week, behavior of drivers, etc.

To optimize CrowdNav (our production system) functionality at runtime, we use, in an iterative way, a series of online experiments that yield a proposition of a new configuration, which is then applied, and its effect is measured by car trip durations and drivers' convenience. This assumes determining the value domain for each system parameter and setting an initial configuration (Table 1). The default values and ranges for each parameter were set based on the experience of performing past experiments with CrowdNav [7].

Table 1. CrowdNav parameters.

Parameter	Value Range	Default Value	Description
static information importance	[1,2.5]	1	Controls the importance of static information (i.e. max speed, street length) on routing
dynamic information importance	[1,2.5]	1	Controls the importance of traffic information (i.e. average duration to traverse a street) on routing
exploration importance	[1,2.5]	1	Controls the magnitude of detours
route randomization	**[0-0.3]**	**0.1**	**Controls the random noise introduced in the router to avoid giving same routes**
exploration percentage	[0-0.3]	0.1	Controls the ratio of cars that make detours
data freshness threshold	**[100-700]**	**400**	**Threshold for considering observed traffic-related data as stale and disregard it**
rerouting frequency	[10-70]	50	Controls how often the router should be invoked to reroute a car

For simplicity, we focus in this paper on optimizing CrowdNav by tuning only two of its system parameters. The first one is "route randomization" which controls the amount of noise that is introduced to the routing weight. Its range is 0 to 0.3; in simple terms, a value of 0.1 means that there is a possible 10% increase or decrease of routing weight. Larger values lead to more diverse routes for the same origin-destination pairs, which helps avoid bottlenecks. On the other hand, high values may add some less-optimal routes due to added noise. The second system parameter is the "data freshness threshold" that determines the acceptable level of staleness in traffic data. Consider the case of a number of cars that need to move in a peak time from an area A (e.g., a residential zone) to area B (e.g., downtown). If all of them are given the same route, congestion might occur; nevertheless, too much randomization could yield several suboptimal routes. Similarly, the threshold of data freshness influences the efficiency of the routing process.

CrowdNav is bundled with a microscopic traffic simulator (SUMO [8]) that allows simulation of different car trips in realistic environments. For instance, to perform the demonstration and evaluation in Sec. V, we deployed 500 cars in the city of Eichstädt in Germany.

III. AOEDA KEY CONCEPTS

A. MAIN IDEA

AOEDA is based on the following key assumptions:

1. The LSIS features a configuration space (a set of its configurations) that can be explored at runtime by its adaptation interface.
2. The LSIS can be adapted online by choosing and

applying a new configuration.

3. The LSIS provides runtime data on its behavior—system outputs—which allow quantifying the effect of applying a particular configuration.
4. There is an Adaptation Manager that collects the system outputs, processes and analyses them, and periodically performs a series (pursuit) of online experiments by choosing and applying specific configurations to the LSIS with the goal to optimize its functionality.

Figure 1 shows the main idea of AOEDA. Essentially, AOEDA follows the principles and phases of MAPE-K loop, a classical approach to self-adaptation [9]. Concretely, the LSIS features a number of system parameters, including those characterizing the context (environment) of LSIS, which are continuously **monitored** and **analyzed** to trigger a pursuit (**plan**) and finally applied (**execute**) as the optimal configuration found in the pursuit. The inner MAPE-K loop in Figure 1 executes an experiment as a pursuit's step, monitors and analyzes its results, and plans the next experiment to run in the pursuit. The analysis phase involves obtaining multiple readings of system outputs. In a similar vein, the outer MAPE-K loop in Figure 1, controls the execution of the pursuit; specifically, its termination is determined either by meeting an optimization goal or by exceeding a predefined number of experiments.

When the pursuit finishes, the (sub-)optimal configuration found is applied to the LSIS (the execute phase of the outer loop) and saved in the Knowledge base, so that this configuration can be reused if the LSIS resides in the same situation in the future.

In CrowdNav, the system outputs are trip overheads and driver complaints. For each trip, the overhead is calculated by dividing the actual trip duration by its theoretical duration assuming no other traffic is present and cars move always at the maximum permitted speed. Practically, trip overheads range from 1 to 30[1]. An experiment in CrowdNav applies a new configuration and monitors the system outputs. It terminates after 5000 samples of trip overhead are collected. A pursuit consists of potentially many such experiments and is triggered by monitoring a single system (context) parameter — the number of cars. Whenever this number reaches a certain threshold, a new pursuit is triggered. The goal of a pursuit in CrowdNav is to identify a minimum or close to minimum median trip overhead. The pursuit ends by alternatively (a) meeting the optimization criterion "median(trip overhead) < 1.2", or reaching a configuration that cannot be improved any further (local optimum is found), (b) reaching the pursuit budget (maximum number of experiments).

B. CHALLENGES

In essence, AOEDA performs pursuits that change configurations at runtime and measure the effect of the changes, for different environments or contexts. In this setting, a basic challenge is to select the best optimization strategy to drive the pursuit. This depends on many factors including the nature of system inputs (e.g. discrete or continuous), the size and shape of the configuration space, the potential necessity to find the global optimum, the time budget available for the pursuit, and the tolerance levels of users. Clearly, no strategy is optimal in all cases, as also indicated by the "no free lunch" theorems for optimization [10].

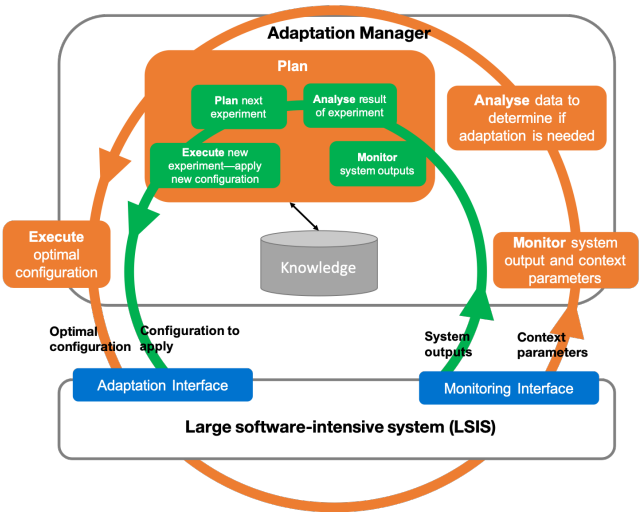


Figure 1: High-level illustration of AOEDA.

The above challenge is exacerbated when the comparison of the optimization strategies and the selection of the optimal one for a certain environment is learned once versus continuously. In the latter case, mechanisms have to be in place for switching strategies at runtime and assessing their effectiveness in an automated way, as discussed in our previous work [11].

Anyhow, an overarching challenge in AOEDA is that all its phases have to be performed in a real-world setting, i.e. after the system has been deployed to production. This is different to the testbeds where optimization algorithms (e.g. numeral optimization, genetic algorithms) are typically benchmarked [12]–[14] and creates the extra requirement of handling the different types of cost components associated with AOEDA, described below (Sect. III.C).

C. EXPERIMENTATION COST

Compared to other adaptive systems, we argue that costs of experimentation need to be considered differently for AOEDA. As mentioned before, a pursuit is associated with three key components of experimentation cost, which are neither system parameters nor system outputs, but a property

[1] Trip overhead of 1 means that its duration was equal to its theoretical duration.

of the optimization strategy:

1. Time cost. This is related to both the size of configuration (sub)space to be explored and the number of experiments in a pursuit. Obviously, the higher the size of configuration space and the number of experiments, the higher the time cost.
2. Adaptation cost. Every time an experiment starts, the LSIS is adapted eventually. This cost component is proportional to the computational complexity of the required adaptation actions. As an example, consider applying to CrowdNav new router settings. Here the adaptation cost is given by the burn-in time needed to ensure that these settings have been picked up by all cars.
3. Endurability cost. This is the cost of running configurations which lead to user dissatisfaction [15], which we call harmful configurations. Clearly, different users will have different sources of dissatisfaction: longer trip times, more fuel consumption, less driving comfort, etc. The probability of inducing endurability cost, as well as its magnitude is difficult to predict; controlling the magnitude of this cost amounts to controlling the risk of online experimentation [7].

The significance of each experimentation cost component depends on a particular LSIS. When optimizing a web service, the time cost or adaptation cost do exist but they are often not significant. On the contrary, endurability cost is important due to potential loss of user engagement. In a similar vein, when optimizing a routing request issued by an ambulance responding an accident, the time cost is primarily important, since the optimization needs to take just few seconds, or at most a minute, to be relevant.

IV. OPTIMIZATION STRATEGIES

Different optimization methods can be used to drive a pursuit in AOEDA, i.e. to select the series of online experiments to be performed to optimize the system. The optimization we consider takes the form of finding the minimum of a response $y = f(x_1, x_2, \dots, x_n) + \varepsilon$ where f is the (unknown) response function, $x_1, x_2, \dots, x_n$ are the input parameters (independent variables, set specifically in each experiment) and ε is the statistical error representing other sources of variability not accounted by f.

The parameters x_1, x_2, …, x_n are elements of the domain X_1, X_2, …, X_n forming a configuration space CS. The response function f with its domain CS and the statistical error ε determine the potential values of y – a response surface RS. Since f is unknown in general, RS is typically approximated, e.g., by a surrogate model, or even (in discrete settings) fragments of it are created on the fly. A variety of methods for creating such an approximation are known. They are commonly based on carrying out experiments. Designing experiments basically means to determine for which parts of CS the response surface RS is to be approximated. An experiment involves traversing the approximated parts of RS (this process is called search) with the aim to find a domain point yielding an optimal value of y.

Some methods specify/design all experiments before the pursuit starts; others do so on the fly. Each method has different effect on reducing a particular cost component, while potentially increasing others. We illustrate this trade-off by optimizing CrowdNav via the three methods listed below. These have been chosen (no only) since each of them inherently aims at reducing one out of the three cost components.

1. **Grid search** over the configuration space (factorial design)—mainly reduces adaptation cost;
2. **Sequential model-based optimization** (SMBO), also referred to as Bayesian optimization— mainly reduces time cost;
3. **Local search** starting from a predefined configuration— mainly reduces endurability cost.

A. GRID SEARCH

Grid search systematically goes through all the possible configurations of a discretized configuration space CS. It starts from the configuration at an edge of CS and changes configuration by increasing one parameter $x_i$value at a time. Using design-of-experiments (DoE) terminology [16], this method corresponds to full factorial design. Here, all experiments (configurations to be tested) are designed (specified) a priori. The results of grid search can be analyzed with factorial ANOVA to determine which of the parameters $x_1, x_2, \dots, x_n$or combinations of them affect the response. The results can also be used for fitting a first- or second-order polynomial model that approximates the response surface RS [17]. Since grid search always slightly modifies a single parameter x_i , the search only makes small moves in the configuration space, resulting likely in low adaptation cost.

B. SMBO

In each experiment, SMBO builds a surrogate model (e.g. Gaussian process, decision tree) of the response surface RS and chooses the next experiment—the next configuration to try out [18], [19]. To choose the next experiment, SMBO uses an acquisition function that balances exploration with exploitation: it tries to find a globally optimal configuration with the least number of experiments by exploring less visited regions in CS and exploiting the learned knowledge by evaluating close-to-known good configurations. Since the focus is to converge fast, SMBO inherently reduces the time cost of experimentation. However, adaptation and endurability costs may be compromised due to jumps in CS.

C. LOCAL SEARCH

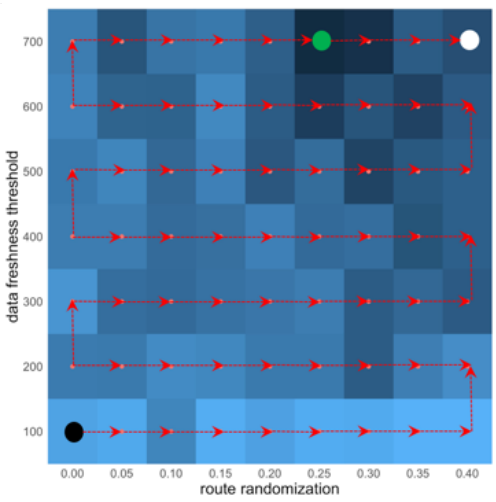
Local search is a simple method, inspired by Evolutionary Operation by Box [20], that aims at finding a better configuration (local optimum) in a neighborhood in CS. Local search starts from evaluating a starting configuration

(current one), tries out all the neighboring configurations and moves to the configuration that performs best, and, of course, better than the current configuration. If no such neighbor exists, the search terminates. The bottom line is that local search does not aim at a globally optimal configuration. Since it incrementally improves the system output by making only small modifications at a time, it does not bring high risk of running harmful configurations; so there is a good chance that it does not worsen the endurability cost.

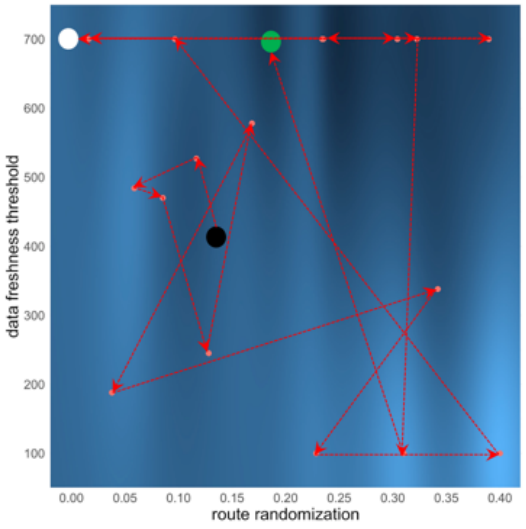
V. CONCEPT EVALUATION

The goal of this section is to demonstrate AOEDA and to compare the different optimization strategies and their effect on the different cost components. In particular, we evaluate the applicability of AOEDA in the optimization of the traffic navigation use case introduced in Section II. Since we aim at showcasing the different cost components, we benefit from the flexibility of the simulator bundled with CrowdNav, as other self-adaptation approaches did in their evaluations [21]–[23].

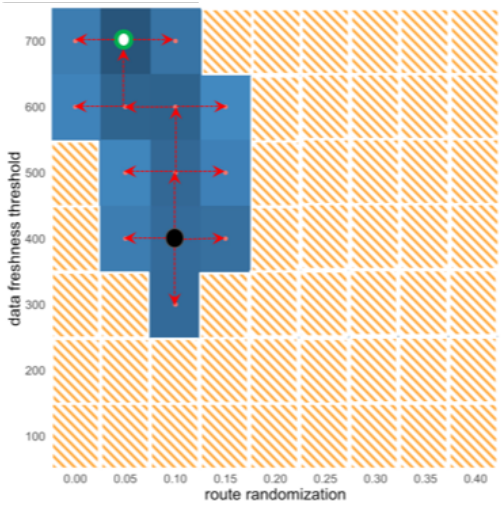
In CrowdNav we simulated random trips of 500 cars in the city of Eichstädt, Germany. In case of a high trip overhead (higher than 2.5), a driver complaint was issued with some probability (we used 50%). The output of an experiment was the median of trip overheads of 5000 trips and the number of complaints issued in the meantime. Such



(a) Grid search for 9 values of route randomization and 7 values of data freshness threshold.



(b) Sequential Model-Based Optimization (SMBO) with 20 iterations.



(c) Local search: The algorithm visits the 4 neighbors of a configuration and moves towards the best performing neighbor.

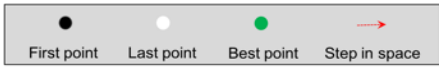


Figure 2: Comparison of the three pursuit strategies on CrowdNav. Darker areas of the response surface indicate lower median values (better) of trip overheads.

a large number of recorded overheads resulted in a median value that was rather stable and did not suffer from measurements errors (outliers, etc.). The simulation results reported in the rest of this section serve for demonstration of the different cost types and informal comparison of the pursuit strategies.

Recall from Sec. II that we consider configuration of two parameters: "route randomization" and "data freshness threshold". The latter controls the length of the time window for which traffic data is considered relevant in the routing process. In Figure 2, the configuration of each experiment (represented as a point) along with the median of trip overhead per experiment is depicted. In particular, Figure 2.a shows the pursuit using the SMBO strategy with Gaussian processes as the surrogate model, 3.b using grid search, and 3.c using local search. For each of these subfigures, the first, last, and best configuration are marked in black, white, and green, respectively. Furthermore, each pursuit yielded a different optimal configuration and value of cost components as described below. Recall that a cost component in our case is neither a system parameter nor a system output, but a property of the optimization strategy.

Optimal configurations. In the case of both the SMBO and grid search, the pursuit stopped when it exhausted its budget (20 experiments for SMBO, 63 for grid search), whereas local search terminated not being able to find better local optimum in neighboring configurations. Among the pursuit strategies, grid search yielded the optimal configuration. At the same time, SMBO yielded the second-best optimal configuration (a local optimum), whereas local search was less efficient. Although the three strategies yielded different results, all of them have matched our intuition that some route randomization boosts the routing efficiency, whereas more randomization hinders it.

Time cost. SMBO is faster in finding a good configuration than grid search: with a pursuit budget of 20 experiments, SMBO reports a local optimum that is close to the best-known optimum reported by grid search. However, grid search needs a much higher pursuit budget of 63 experiments. Finally, local search needs a low pursuit budget of 13 experiments.

Adaptation cost. On the other side, grid search and local search carry out only small changes in CS (low adaptation cost), whereas SMBO occasionally exhibits big jumps in the configuration space. Given that

$$adaptation\ cost = \sum_{experiments} cost\ of\ a\ single\ experiment$$

and the cost of a single experiment in CrowdNav is always a small constant, the big jumps in CS do not lead to higher adaptation cost in case of SMBO.

Endurability cost. A harmful configuration leading to higher trip durations will also yield more user complaints—and thus increase the endurability cost. In case of SMBO, this can happen due to the fact that the pursuit will explore less visited regions of CS. We measured the number of issued complaints and found out that in the case of grid search 15,405 complaints were reported in total (average complaints-to-trips ratio: 4.89%), compared to only 4,176 complaints in total in case of SMBO (average complaints-to-trips ratio: 4.17%). This is of course related to the less experiments (20) performed for SMBO compared to grid search (63). When using local search by starting from a configuration of "route randomization" of 0.1 and "data freshness threshold" of 400 (Figure 2.c.), 3,226 complaints were reported in total (average complaints-to-trips ratio: 4.61%). Although the total number of complaints was less than in the case of SMBO, we note that the efficiency of local search highly depends on the initial configuration. A main goal of the evaluation was to illustrate the trade-offs between different cost factors for production settings, where adaptation cost and endurability cost can be very significant.

In adaptive systems research (including our earlier work), extensive evaluations of the effect of the different strategies on the optimization objective and the cost components confirm the above observation. For instance, in [11] we compared the effectiveness of Bayesian optimization with two variants of genetic algorithms on optimizing CrowdNav at runtime. In [7], we combined Bayesian optimization with factorial design (grid search), A/B testing and binomial testing (for detecting harmful configurations). In [24], we performed experiments in a microservice-based production environment to collect data with the goal of continuous optimizing services deployment in an edge-cloud scenario. Finally, in [25] we used Bayesian optimization in combination with reinforcement learning to optimize machine learning pipelines.

VI. DISCUSSION OF KEY LEARNINGS

In this section, we summarize the lessons we have learned when applying AOEDA to CrowdNav and discuss our recommendations when applying AOEDA in general.

A. LESSONS LEARNED

Trade-offs between different pursuits and cost components need to be considered for each case. Local search can be good in reducing endurability cost if a good starting point is chosen and a good or best local optimum can be found. Grid search is exhaustive and slow, but has minimal cost for adaptation in each step. Both Grid and local search work best if the configuration space is continuous, without sharp peaks. SMBO explores the RS more widely and can detect more local optima (and the global optimum at best). Still, any application of SMBO needs to consider all the three cost components, and especially the endurability cost as it makes bigger jumps in the configuration space.

Experimentation cost depends on the application. As we showcased in Sec. IIIC, one needs to account for the different cost components associated with the choice of a pursuit

strategy. However, cost is also associated with the target LSIS itself: For instance, applying AOEDA in optimizing a car-sharing system by modifying the position of some of the available cars will most probably have higher adaptation cost than changing the values of CrowdNav parameters, since moving cars around comes with additional effort. A good idea before starting to optimize an LSIS is to first estimate the size of all cost components. Time cost can be estimated by looking at the variance of system outputs, with higher variance pointing to longer experiments and higher time cost. Adaptation cost may be estimated offline by calculating the cost of each potential experiment. Finally, endurability cost is hard to estimate offline; small-scale pre-experiments should be used here to gauge the amount of endurability cost that can be expected.

B. RECOMMENDATIONS

Harmful experiments should be aborted early. Time and endurability costs' evaluation can be used not only for assessing different optimization strategies, but also for aborting an experiment if it incurs a high time and/or endurability cost. This is analogous to the abortion criteria used in A/B testing for stopping an online experiment before many users are exposed to a harmful configuration [26].

Large configurations spaces should be pruned offline. In order to reduce the time cost of a pursuit strategy, CS may have to be pruned offline to reduce the number or range of parameters considered in the pursuit. One way to achieve this is to learn (offline) which parameters have the strongest effect on the system output and use them to form the CS considered in the pursuit [22].

Humans in the loop. Ideally, AOEDA should be performed in complete autonomy in which continuous monitoring of the LSIS is used for triggering a pursuit when the current system configuration is considered suboptimal with respect to the runtime situation (e.g. high traffic in the city). Nevertheless, when automated pursuit triggering is difficult to achieve, humans can be in the loop to possibly trigger new pursuits. Indeed, automating the triggering of new pursuits is an important subject of our future work.

VII. COST ASPECTS IN SELF-ADAPTIVE AND ONLINE EXPERIMENTATION APPROACHES

Our new approach for online experimentation integrates adaptation, search and optimization, and experimentation for systems with a larger configuration space, where no suitable model of the system and environment exists. As there is a vast body of related work, we compare AOEDA to different related approaches in detail below.

The main goal of self-adaptive systems is adaptation in response to changes in their internal state and their operating environment [27], [28]. Considerable research efforts focus on finding the most suitable new configuration [27], [29] based on a model (e.g. a Markov decision process). In AOEDA, we use adaptation to explore the configuration space by experiments without such a model. Thus, we have to consider the cost of several adaptation steps, which leads to our novel cost classification. The notion of cost that appears in the self-adaptive literature mainly focuses on the utility or suitability of a single target configuration, compared to others in a model. Also, other works consider cost in the context of monitoring, e.g. [30].

In the area of adaptive systems, approaches exist that employ online planning to find the best adaptation actions at runtime [22], [31]. A number of algorithms have been employed to this end: Hill climbing has been used to implement a search-based feedback loop [32]; genetic programming and genetic algorithms (including NSGA-II and novelty search) have been advocated as part of the vision of genetic improvement for adaptive software engineering [33] and used in determining optimal configurations [11], [23], [34]–[36]; finally, multi-armed bandits [37] and Bayesian optimization [7], [11] have been employed for online planning in self-adaptive systems.

Although the emphasis in these works is on the quality of the found solution, the time needed to find such solution is also evaluated and reported, as, e.g., in [21], [23], [36]. Elapsed time is indeed related to our time cost; however, we emphasize that in our approach, runtime experiments are performed with the system itself, not with a model of it. In this setting, model-less approaches using multi-criteria optimization have also reported time costs as a significant factor, but do not mention adaptation or endurability costs [11]. For specific applications of adaptive systems, the adaptation cost is considered, e.g. in [38], [39] the effort to start up virtual machines is relevant for adaptation.

In the recent literature on online experimentation [1], the focus is on improving connected software applications. Here, endurability cost is emphasized over adaptation or time cost, since there is a high risk in user exposure to suboptimal functionality [40], [41]. Online experimentation is a generalization of A/B testing [26] and, in this line, typically a discrete set of options is considered, not the exploration of a search space with possibly continuous variables.

In summary, a main contribution of our new AOEDA approach is the cost classification suitable for experiment-driven adaptation in AOEDA. We show how to use different multi-criteria optimization strategies in this cost model, including incremental optimization based on Gaussian Processes.

VIII. CONCLUSIONS

In this paper, we presented our new approach for automated online experiment-driven adaptation (AOEDA). This integrates concepts of optimization, experimentation and adaptation for systems with a larger configuration space, where no suitable model of the system and environment exists. To overcome the problem of missing model, we used online experimentation as in A/B testing, but for larger configuration spaces where A/B testing does not scale.

Furthermore, we argued that cost needs to be considered differently for AOEDA. In this line, we presented a novel classification of the different cost factors which are most relevant for AOEDA, and showed the connection to related approaches in self-adaptive systems and online experimentation. Based on a use case of traffic management, we compared different options for controlling the online experiments with different cost focus and drew several learnings for AOEDA.

ACKNOWLEDGMENT

The research leading to these results has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783221.

REFERENCES

[1] R. Kohavi *et al.*, "Online experimentation at Microsoft," in *Data Mining Case Studies and Practice Prize III*, 2009, vol. 11, Accessed: Jun. 21, 2017. [Online]. Available: http://www.appliedaisystems.com/papers/DMCS2009_Workshopproceedings4.pdf#page=11.

[2] S. Gupta, L. Ulanova, S. Bhardwaj, P. Dmitriev, P. Raff, and A. Fabijan, "The Anatomy of a Large-Scale Experimentation Platform," in *Proc. of ICSA 2018*, Apr. 2018, pp. 1–109.

[3] D. Tang, A. Agarwal, D. O'Brien, and M. Meyer, "Overlapping experiment infrastructure: More, better, faster experimentation," in *Proc. of SigKDD 2010*, ACM, 2010, pp. 17–26, Accessed: Jun. 13, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=1835810.

[4] "Uber Experimentation Platform," Nov. 01, 2019. https://eng.uber.com/tag/experimentation/.

[5] S. Schmid, I. Gerostathopoulos, C. Prehofer, and T. Bures, "Self-Adaptation Based on Big Data Analytics: A Model Problem and Tool," in *Proc. of SEAMS 2017*, IEEE, May 2017, pp. 102–108.

[6] Steven LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, 98–11, Oct. 1998. Accessed: Feb. 15, 2021. [Online]. Available: http://cs.brown.edu/courses/cs1951r/assignments/motionplanning/rtpaper.pdf.

[7] I. Gerostathopoulos, C. Prehofer, and T. Bures, "Adapting a System with Noisy Outputs with Statistical Guarantees," in *Proc. of SEAMS 2018*, 2018, pp. 58–68.

[8] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *Int. J. Adv. Syst. Meas.*, vol. 5, no. 3 & 4, pp. 128–138, Dec. 2012.

[9] J. Kephart and D. Chess, "The Vision of Autonomic Computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[10] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997, doi: 10.1109/4235.585893.

[11] E. M. Fredericks, I. Gerostathopoulos, C. Krupitzer, and T. Vogel, "Planning as Optimization: Dynamically Discovering Optimal Configurations for Runtime Situations," in *2019 IEEE 13th International Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, Jun. 2019, pp. 1–10, doi: 10.1109/SASO.2019.00010.

[12] N. R. Herbst, S. Kounev, A. Weber, and H. Groenda, "BUNGEE: An Elasticity Benchmark for Self-Adaptive IaaS Cloud Environments," in *Proceedings of the 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, USA, May 2015, pp. 46–56, doi: 10.1109/SEAMS.2015.23.

[13] B. Bischl, J. Richter, J. Bossek, D. Horn, J. Thomas, and M. Lang, "mlrMBO: A Modular Framework for Model-Based Optimization of Expensive Black-Box Functions," *ArXiv Prepr. ArXiv170303373*, 2017.

[14] B. Bischl, O. Mersmann, H. Trautmann, and M. Preuß, "Algorithm selection based on exploratory landscape analysis and cost-sensitive learning," in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12*, Philadelphia, Pennsylvania, USA, 2012, p. 313, doi: 10.1145/2330163.2330209.

[15] I. Gerostathopoulos, C. Prehofer, L. Bulej, T. Bures, V. Horky, and P. Tuma, "Cost-Aware Stage-Based Experimentation: Challenges and Emerging Results," in *Proc. of ICSA 2018*, 2018, pp. 72–75.

[16] S. Ghosh and C. R. Rao, Eds., *Handbook of Statistics 13: Design and Analysis of Experiments*, 1 edition. Amsterdam: North-Holland, 1996.

[17] D. Baş and İ. H. Boyacı, "Modeling and optimization I: Usability of response surface methodology," *J. Food Eng.*, vol. 78, no. 3, pp. 836–845, Feb. 2007, doi: 10.1016/j.jfoodeng.2005.11.024.

[18] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *J. Glob. Optim.*, vol. 13, no. 4, pp. 455–492, Dec. 1998, doi: 10.1023/A:1008306431147.

[19] P. I. Frazier, "A Tutorial on Bayesian Optimization," *ArXiv180702811 Cs Math Stat*, Jul. 2018, Accessed: May 19, 2019. [Online]. Available: http://arxiv.org/abs/1807.02811.

[20] G. E. P. Box and N. R. Draper, *Evolutionary Operation: A Statistical Method for Process Improvement*, Y First printing edition. New York: Wiley-Interscience, 1998.

[21] T. Chen, K. Li, R. Bahsoon, and X. Yao, "FEMOSAA: Feature-Guided and Knee-Driven Multi-Objective Optimization for Self-Adaptive Software," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 2, pp. 1–50, Jul. 2018, doi: 10.1145/3204459.

[22] P. Jamshidi, J. Cámara, B. Schmerl, C. Kästner, and D. Garlan, "Machine learning meets quantitative planning: enabling self-adaptation in autonomous robots," in *Proc. of SEAMS 2019*, May 2019, pp. 39–50, Accessed: Sep. 13, 2019. [Online]. Available: http://dl.acm.org/citation.cfm?id=3341527.3341534.

[23] C. Kinneer, Z. Coker, J. Wang, D. Garlan, and C. L. Goues, "Managing uncertainty in self-adaptive systems with plan reuse and stochastic search," in *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '18*, Gothenburg, Sweden, 2018, pp. 40–50, doi: 10.1145/3194133.3194145.

[24] L. Bulej *et al.*, "Managing latency in edge–cloud environment," *J. Syst. Softw.*, vol. 172, p. 110872, Feb. 2021, doi: 10.1016/j.jss.2020.110872.

[25] X. Sun, J. Lin, and B. Bischl, "ReinBo: Machine Learning Pipeline Conditional Hierarchy Search and Configuration with Bayesian Optimization Embedded Reinforcement Learning," in *Machine Learning and Knowledge Discovery in Databases*, Cham, 2020, pp. 68–84, doi: 10.1007/978-3-030-43823-4_7.

[26] R. Kohavi, R. Longbotham, D. Sommerfield, and R. M. Henne, "Controlled experiments on the web: survey and practical guide," *Data Min. Knowl. Discov.*, vol. 18, no. 1, pp. 140–181, Feb. 2009, doi: 10.1007/s10618-008-0114-1.

[27] C. Krupitzer, F. M. Roth, S. VanSyckel, G. Schiele, and C. Becker, "A survey on engineering approaches for self-adaptive systems," *Pervasive Mob. Comput.*, vol. 17, pp. 184–206, Feb. 2015, doi: 10.1016/j.pmcj.2014.09.009.

[28] B. Cheng *et al.*, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in *Software Engineering for Self-Adaptive Systems*, Springer Berlin Heidelberg., 2009, pp. 1–26.

[29] K. Angelopoulos, A. V. Papadopoulos, and J. Mylopoulos, "Adaptive predictive control for software systems," in *Proceedings of the 1st International Workshop on Control Theory for Software Engineering*, New York, NY, USA, Aug. 2015, pp. 17–21, doi: 10.1145/2804337.2804340.

[30] E. Zavala, "Towards Adaptive Monitoring Services for Self-Adaptive Software Systems," in *Service-Oriented Computing – ICSOC 2017 Workshops*, Cham, 2018, pp. 357–362, doi: 10.1007/978-3-319-91764-1_31.

[31] T. Zhao, "The Generation and Evolution of Adaptation Rules in Requirements Driven Self-Adaptive Systems," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*, Sep. 2016, pp. 456–461, doi: 10.1109/RE.2016.18.

[32] P. Zoghi, M. Shtern, and M. Litoiu, "Designing search based adaptive systems: a quantitative approach," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, 2014, pp. 7–16.
[33] M. Harman *et al.*, "Genetic improvement for adaptive software engineering (keynote)," in *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Hyderabad, India, Jun. 2014, pp. 1–4, doi: 10.1145/2593929.2600116.
[34] Z. Coker, D. Garlan, and C. L. Goues, "SASS: Self-Adaptation Using Stochastic Search," in *Proceedings of the 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, USA, May 2015, pp. 168–174, doi: 10.1109/SEAMS.2015.16.
[35] G. G. Pascual, M. Pinto, and L. Fuentes, "Run-time adaptation of mobile applications using genetic algorithms," in *2013 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, May 2013, pp. 73–82, doi: 10.1109/SEAMS.2013.6595494.
[36] S. Y. Shin, S. Nejati, M. Sabetzadeh, L. C. Briand, C. Arora, and F. Zimmer, "Dynamic Adaptation of Software-defined Networks for IoT Systems: A Search-based Approach," *SEAMS 2020*, p. 12, 2020.
[37] B. Porter and R. R. Filho, "Distributed Emergent Software: Assembling, Perceiving and Learning Systems at Scale," in *SASO 2019*, 2019, p. 10.
[38] D. Kusic, J. O. Kephart, J. E. Hanson, N. Kandasamy, and G. Jiang, "Power and Performance Management of Virtualized Computing Environments Via Lookahead Control," in *2008 International Conference on Autonomic Computing*, Jun. 2008, pp. 3–12, doi: 10.1109/ICAC.2008.31.
[39] Q. Zhang, Q. Zhu, M. F. Zhani, and R. Boutaba, "Dynamic Service Placement in Geographically Distributed Clouds," in *2012 IEEE 32nd International Conference on Distributed Computing Systems*, Jun. 2012, pp. 526–535, doi: 10.1109/ICDCS.2012.74.
[40] R. Kohavi, D. Tang, and Y. Xu, *Trustworthy Online Controlled Experiments: A Practical Guide to A/B Testing*. Cambridge, United Kingdom ; New York, NY: Cambridge University Press, 2020.
[41] Y. Xu, W. Duan, and S. Huang, "SQR: Balancing Speed, Quality and Risk in Online Experiments," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, New York, NY, USA, Jul. 2018, pp. 895–904, doi: 10.1145/3219819.3219875.

ILIAS GEROSTATHOPOULOS is an Assistant Professor of Computer Science at Vrije Universiteit Amsterdam, Netherlands. He obtained a Ph.D. in Computer Science from the Department of Distributed and Dependable Systems, Faculty of Mathematics and Physics, Charles University in Prague. He has spent time as a postdoctoral researcher in the Department of Informatics at the Technical University of Munich. His research focuses on software engineering, software architecture, and self-adaptive systems.

FRANTIŠEK PLÁŠIL is professor of software engineering in the Department of Distributed and Dependable Systems (D3S). Charles University, Prague. In his research, he focuses on component-based software architectures, and also on allocation of formal methods in software systems. He has led several D3S research teams in a number of research projects such as ITEA OSMOSE, ITEA OSIRIS, EU FP7 Q-ImPrESS, and ASCENS. He co-authored over 100 refereed articles in international journals and proceedings of international conferences, and also served on the program committees of numerous international conferences, and editorial boards of several international journals. In the course of his carrier, he has had visiting positions in US at the University of Denver, Wayne State University, University of New Hampshire, and in Austria at the University of Linz.

CHRISTIAN PREHOFER holds a MS degree from the University of Illinois at Urbana-Champaign and obtained his Ph.D. and habilitation in computer science from TU München in 1995 and 2000. Currently, he is director at DENSO Germany and lecturer at TU München. Before this, was leading research groups at fortiss and Fraunhofer. He also was acting as professor in Computer Science at LMU München and Chang'an University. Before 2009, he held different management and research positions in the mobile communication industry. He is author of more than 150 publications and 34 granted patents.

JANEK THOMAS is group leader of the Fraunhofer IIS group AutoML & XAI funded by the Ada Lovelace Center. The group is closely connected to the Working Group Computational Statistics as well as the Chair of Database Systems and Data Mining of Ludwig Maximilian University (LMU) of Munich. He finished his PhD at the Working Group Computational Statistics of LMU in April 2019 focusing on Automated Machine Learning and Gradient Boosting. During his PhD he did research internships at the Microsoft Cloud and Information Services Lab and H2O.ai.

BERND BISCHL is Professor for Statistical Learning and Data Science at the Department of Statistics at the Ludwig Maximilian University of Munich. He obtained his PhD in 2013 from the Department of Statistics at TU Dortmund, Germany. He works in data science, machine learning and computational statistics.