Leveraging human strategy in Bayesian optimization for industrial commissioning

Preprint

András Retzler[1,2,*], Tom Lefebvre[1,2], Guillaume Crevecoeur[1,2]

***Abstract*— Commissioning industrial processes is often a tedious and difficult task, because certain settings, such as parameters of a controller, need to be adjusted repeatedly until a certain goal is reached. It is also difficult to pass on the information on how to tune a system from experienced operators to junior colleagues, because such knowledge is often tacit and hard to formalize.**

We approach the tuning problem with Bayesian optimization (BO). We discuss different ways of combining the best of both worlds, i.e., the insights of the expert with the goal-driven exploration of BO. With the goal of using recorded tuning sessions of experts in BO, we dive into two main directions.

First, we explore the effect of context changes on a Human First-Computer Last approach, where the expert does the initial rough tuning, and BO does the fine-tuning afterward. Second, we lay out initial steps toward a subgoal-based BO method, based on the observation that humans often break up complex tasks into multiple subgoals to achieve in a given sequence.

We evaluate these methods on two simulation setups featuring a cascade controller of a pusher-slider robotic system, and a projectile trajectory planning problem.

I. Introduction

Commissioning industrial processes often involves a phase of tuning, where the operator cycles through a series of on-site experiments targeted to find a set of settings for the machine, in order to achieve optimal performance. This comes with two main challenges.

First, such a tuning procedure is often a tedious and difficult task, whose success depends on the knowledge of the operator. A very important part of this knowledge is hands-on experience in how the tuned parameters interact in relation to the performance indicators. If the operator lacks this, the tuning process can take a long time, might lead to a lot of waste and sunk costs, potentially ending up with suboptimal operation of the machine. In extreme cases, an inexperienced operator might even damage the machine.

Second, this experience or knowledge is often hard to formalize or document as concrete instructions that can be transferred to junior operators. Management sciences call this tacit knowledge [1].

Industrial control has many of such problems. Even the Proportional-Integral-Derivative (PID) controller, which can be considered the most common control technique today [2], is still difficult to tune, despite multiple reference works existing on the topic. Due to the non-convex objective landscape it is considered an NP-hard problem [3]. It is not just PID though: most control methods have some parameters that directly affect their performance. For example, in the Model Predictive Control (MPC) feedback loop, both the coefficients inside the cost function and the model parameters can be tuned [4].

Even though our focus in this paper is commissioning industrial machines or processes, configuring complex hardware-software systems with hundreds of parameters, such as solvers [5] or imaging pipelines [6] can also be considered similar problems. Furthermore, it is important to mention that tuning problems also appear in the research and development phase of product design, e.g., the design of chemical synthesis trials [7], aircraft [8], etc.

Now that we see that the automation of tuning processes is an important issue to solve, the question arises: to what extent does the current state of the art handle that? Bayesian Optimization (BO) [9] is a potential candidate to automate various tuning processes. In Machine Learning (ML), BO is extensively used for hyperparameter tuning of models [10], for example, the number of neurons and layers in Neural Networks (NNs), which led to a rise in available BO software tools [11] and techniques, and some companies are even offering it as a service as part of their ML tooling portfolio [12]. While it is just recently that it starts to gain popularity in the field of control as well, BO has been successfully applied to, for example, a multi-loop PID controller in a nonlinear evaporator [13], an active disturbance rejection controller of a throttle valve [14], and a three-level cascade controller of a positioning system [15]. While other approaches such as Reinforcement Learning (RL) [4] exist, BO is attractive due to its sample efficiency and also relatively low computational cost. On the other hand, most of these approaches view the tuned system as a black-box, even though, in real life, humans have additional information and physical insight that helps the tuning process.

However, what is the best way to capture and use that insight? In this paper, we are discussing two out of the many directions to combine human and computer tuning. We have recorded the tuning steps of experts on two Virtual Use Cases (VUCs): a simulated control tuning scenario featuring a cascade controller of a robot arm, and a trajectory planning task. The collected data was used for the initialization of BO (section III) to evaluate the feasibility of such an approach against contextual changes, and for extracting the human strategy (section IV) with a new subgoal-based approach. These are steps toward the grand goal: to combine the best of both worlds, particularly the experience of the human with the efficient exploration of BO.

II. Preliminaries

A. The scope of the paper

It is important to emphasize that even though industrial commissioning often involves tuning, it also often encompasses other tasks that we are not focusing on in this paper. We are interested in a subset of problems that can

[1]The authors are with the Dynamic Design Lab (D^{2}LAB) of the Department of Electromechanical, Systems and Metal Engineering, Ghent University, B-9052 Ghent, Belgium.

[2]The authors are member of core-lab MIRO, Flanders Make, Belgium.

[*]Corresponding author: andras.retzler@ugent.be.

clearly be treated by tuning alone, i.e., that the operator queries different settings to the application, observes the performance, and chooses the next query. This coincides with numerical iterative optimization procedures with the distinction that the performance is evaluated and the next query is chosen by the human. Furthermore, the settings $\boldsymbol{x}$ that are being tuned can be described with real numbers, such that $\boldsymbol{x} \in \mathbb{R}^m$.

For now, we are not treating cases where some of the settings are defined as discrete choices (e.g., "choose between switch positions A, B, and C"), or problem-solving based on specific actions to take in a sequence, but without the element of tuning continuous variables (e.g., "if the pressure is too high, replace the filter by following specific steps in the user manual"), even though these are also very realistic events that can happen during commissioning.

There is one more element to discuss here: the context of the problem. Commissioning is required, because the context (e.g., the wear level of a machine, the environmental temperature, the size of a certain part due to manufacturing inaccuracies) is slightly different every time. A seasoned expert can sense this, and incorporate it into the tuning process. In control engineering, they often try to capture part of the context inside a model of the process, which can be acquired through system identification, and subsequently used during control design. However, the full context is always impossible to capture, due to the complexity of the real world. In this paper, we indicate which parts of the context are known; otherwise, they are considered unknown.

B. Bayesian optimization

Our approach is to apply BO to the tuning problem defined in the previous section. BO is a sequential design strategy tailored to global optimization of black-box functions. It is usually employed to minimize an expensive-to-evaluate function $f(\boldsymbol{x})$, $\boldsymbol{x} \in \mathbb{R}^m$, without the need for its derivatives to be available. As a surrogate-based technique, BO relies on an approximation of the objective function, which, along with its derivatives, is easy to evaluate as opposed to the original objective function.

BO uses a surrogate model, often chosen as a Gaussian Process (GP), to encode a probabilistic belief about the objective function conditioned on available input-output queries, $\mathcal{D}_n = \{(\boldsymbol{x}_0, y_0), ..., (\boldsymbol{x}_n, y_n)\}$ were $y = f(\boldsymbol{x})$, and n is the number of data points used to build the model. During the BO loop, this dataset is extended with new queries, based on which we need to determine where to observe the function f next. This procedure can be formalized by means of the maximization of the α acquisition function, which, considering that the underlying surrogate model can be differentiated and it is cheap and fast to evaluate, can be carried out by nonlinear optimization algorithms such as L-BFGS. An architectural blueprint of BO algorithms is given in Algorithm 1.

In the end, BO is exploring the objective landscape in a way that might look slightly random to the casual observer, though it is taking very systematic steps of sampling points inside uncertain areas where the highest reward is expected. Certain acquisition functions, such as Expected Improvement (EI), allow to set a compromise between exploration and exploitation through a hyperparameter. What brings actual randomness into BO such that we get different results even if we minimize the same f function, is the fact that fitting the GP hyperparameters to the data, along with maximizing the acquisition function, are both non-convex problems. These are tackled with brute-force global optimization algorithms, e.g., L-BFGS local optimization starting from randomly distributed initial points. Furthermore, at the start, BO needs some samples to build the initial surrogate model on, thus, there must be a few initial experiments (typically 4-10) based on a pseudorandom sequence, after which BO takes over. For these reasons, we cannot make conclusions about the performance of BO without running it multiple times.

C. Taking the human back into the loop

After discussing the basic elements of BO, we follow on to the main topic of combining human knowledge or preference with it. However, it makes sense to give some thought to what humans and computers are particularly good and bad at.

First, humans have very versatile, but rough sensing capabilities. For example, if we throw a ball toward a target, from the visual information that our eyes capture, we can extract a single number to tell how close we managed to hit, e.g., "half a meter." However, this observation is often just approximate, and we need instruments for more accurate measurements. Second, we have good planning capabilities. We often build an internal model to predict outcomes of processes, and we come up with a plan to tackle problems. For example, when riding a bicycle, we have a sense of how other moving objects might cross our path, and we modify our trajectory accordingly.

Aside from the sensing precision, we have some other limits, though. In [16], Miller et al. state that "the span of absolute judgment and the span of immediate memory impose severe limitations on the amount of information that we are able to receive, process, and remember," giving that limit as 5-9 chunks of information. This might be a reason why we are not good at keeping track of a lot of trials with different settings, or even just a few trials with a large number of settings: BO is apparently better at these. Keeping these in mind, there exist multiple strategies for combining BO with human knowledge, as described below.

A first group of approaches focuses on guiding the next suggestion of BO so that the human shows the way based on his planning and rough sensing. One way this can be achieved is by altering the acquisition function: both πBO [17] and π^+UCB [18] multiply the acquisition function by a term that contains the user preference. Another logical option is through human-defined bounds constraints, though there is a chance that the human would overconstrain the problem toward a local minimum. The human preference can also be modeled with a GP surrogate model and applied, for example, as constraints [19] to learn an area to be avoided, or as a function to alter the acquisition function with. There are several works considering the human preference

Algorithm 1 Bayesian Optimization

while !converged **do**
  Select design, $\boldsymbol{x}_{n+1} := \underset{\boldsymbol{x}}{\operatorname{argmax}}\ \alpha_n(\boldsymbol{x}, \mathcal{D}_n)$.
  Evaluate performance value, $y_{n+1} := f(\boldsymbol{x}_{n+1})$.
  Update statistical model $p(f|\boldsymbol{x}, \mathcal{D}_{n+1})$ (used internally by α), with the augmented dataset $\mathcal{D}_{n+1} := \{\mathcal{D}_n, (\boldsymbol{x}_{n+1}, y_{n+1})\}$. $n := n + 1$.
end while

directly as the objective, as sometimes the Key Performance Indicator (KPI) to be optimized cannot be determined based on measurements. For example, in [20] they attempt to learn the goal of the users from how they move the sliders corresponding to settings, based on a so-called Preferential BO approach [21].

A second group of approaches are "redirecting" BO after it made the suggestion: they let the human modify it or choose between different suggestions. In [22], they describe a BO algorithm which is capable of generating multiple different suggestions that are reasonably distinct, and they present a choice between these suggestions for the human. This is based on the hypothesis that humans are better at making discrete choices than tuning directly.

A third type of approach is alternating between human tuning and BO. The approach of [23] belongs here, making the logical choice to have the human start with a coarse tuning, and then BO do a fine-tuning.

A fourth type of including human knowledge is through modeling the human strategy. Most related works focus on the aspect that we have a mental process of running an internal simulation of the outcomes while tuning the system. This can be replicated using a physics-based model that can be combined with a statistical model in the surrogate. For example, [24] applies such Bayesian hybrid models to a ballistics decision-making problem. On the other hand, physical modeling comes at a high expense: trained engineers are needed to extract the physical laws of the system.

To lay out the two main directions of this work, we are focusing on a less explored direction, namely that humans also often break down complex tasks into subgoals. On another line of research, we are including the human in the loop in a way that we initialize BO with human knowledge.

III. Human First – Computer Last

Based on the strengths and weaknesses of the human and the computer discussed above, it is a logical avenue to initialize the GP surrogate model of the BO with the first few $(\boldsymbol{x}, y)$ pairs coming from human experts tuning the process, and then continue from there with BO in a fully automatic way, such that the human does an initial rough tuning, subsequently improved by BO fine-tuning. Such a strategy has been introduced in [23], and is called Human First-Computer Last (HF-CL). However, when we implemented this setting, several questions arose.

First, if we have data recorded from multiple human tuning sessions, which expert should we use? Different experts might have different strategies. In [23], they compared descent curves visually (KPI versus cumulative financial cost), and quantitatively based on the total financial cost that the tuning takes. Is that the best way to quantify the information contained in these expert choices?

Second, how many $(\boldsymbol{x}, y)$ iterations should we use from our experts? Analysis by [23] shows that for a particular semiconductor process development problem, there is an optimal number of iterations that one needs to use from the recorded human tuning procedure. Using more or fewer than that results in slower convergence toward the minimum. We are interested in whether we see a similar effect on the problem of tuning an industrial controller.

Third, this is an online strategy. A human is needed in the loop to tune the machine for a different context, which is the case in commissioning. Can we make it offline by reusing the recorded suggestions of an expert if the context is only slightly different? If that works, how far can we push it with respect to the difference in context? To our understanding, this was not analyzed in [23].

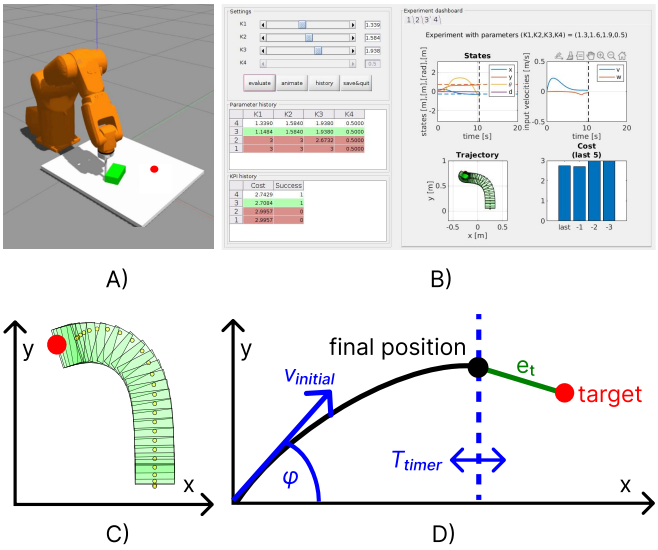


Fig. 1. The virtual experimental setups used in this work. First, the slider-pusher VUC in action (A), along with the GUI provided to the users (B), which visualizes the trajectory (C) of the block and the end effector, and other signals. Second, a sketch of the "moonshot" projectile trajectory planning VUC (D), with the tunable parameters.

In order to answer these questions, we have let 14 experts tune a cascade controller of a slider-pusher robot, and used the data we collected to initialize BO according to a HF-CL strategy.

A. The slider-pusher virtual use case

The setup tuned by the experts is shown in Figure 1 A): it is a robot that pushes a block on a table toward a certain goal position. The experts were tuning the $K_{1...3}$ parameters of the controller in order to minimize a v_{K} KPI, that is calculated from the time v_{T}, the distance v_{d}, the controller efforts v_{ut} and v_{un} (tangential and normal velocities, respectively), and the v_{s} success of the push (i.e., a value of 1 if we got within 0.05m from the target, 0 if not):

$$v_{\mathrm{K}} = \log(v_{\mathrm{T}} + 2v_{\mathrm{d}} + v_{\mathrm{ut}} + 20v_{\mathrm{un}} + 1000(1 - v_{\mathrm{s}})). \quad (1)$$

Note that the value of v_{K} was clamped at a maximum of $\log(20) \approx 3$. In order to utilize their human insight, the C) trajectory of the block was shown to them. Additional graphs depicting how the states and inputs evolved over time, and a history of previous parameter and KPI values were also available to them, as seen on the GUI in B). The MATLAB scripts for this GUI are available[1].

The slider-pusher setup, based on earlier work [25][26], was extended with a model-based cascade controller, the formulas of which are presented in the Appendix. We let the users tune parameters $K_{1...3}$ and fixed K_4 to 0.5. The trajectory of the push and the KPI was always deterministic in the virtual environment to make the case simpler.

We were helped by 14 researchers in our group: we organized a meeting with all the participants, where after a

[1] https://github.com/REXPEK/Robotic-VUC3-Public

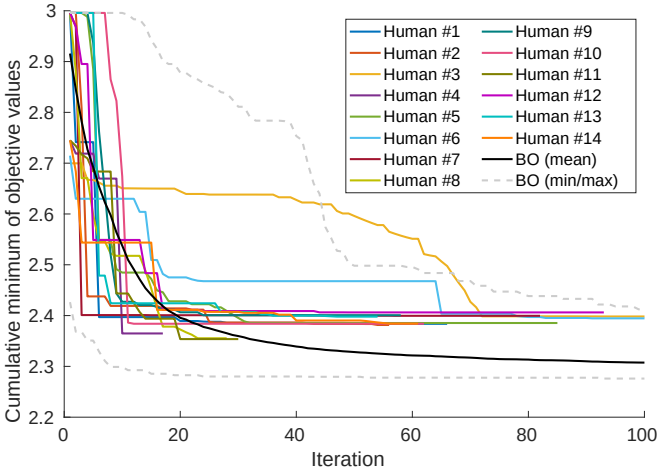


Fig. 2. Humans versus BO, tuning the robotic virtual use case: cumulative minimum values of the objective function per iteration.

short presentation on the task, we let them tune the VUC on their own computer, and then collected the data files that contain all the corresponding $(\boldsymbol{x}, y)$ pairs. All of the participants either had a PhD degree or were working toward it. As our research group deals with control, we assumed that the participants could apply their inherent knowledge to this tuning problem. The participants could stop tuning whenever they were satisfied with the result, though we limited the whole session to approximately 30 minutes.

B. Results on Human First – Computer Last

To set some baselines, let us first compare just human experts versus BO on this problem: this is visualized in Figure 2. We configured BO with an Expected Improvement Plus (EI+) acquisition function [10], 10 uniform random initial points, and an iteration count limit of 100. As there is some randomness in the behavior of BO due to the global search mechanism, we have run it 280 times with the same settings and data. The cumulative minimum curves were extracted, and their mean, minimum, and maximum values were calculated per iteration, which are shown in the graph in solid black and dashed gray lines. The other, colored lines correspond to a single tuning experiment of a single expert.

We can see that many of the experts are reaching low objective values earlier than the "BO (mean)," though none of them can reach an objective value lower than 2.35, which is surpassed by BO around the 30^{th} iteration. From the "BO (min/max)" lines, we see that BO can get lucky and sometimes does significantly better than any of our experts, but it is not the case on average. Overall, from Figure 2, we conclude that some humans have additional knowledge or insights that BO does not have, which underpins the possibility of using human experts in a HF-CL setup. As mentioned earlier, how to quantify and rank this additional knowledge is not trivial, though. Will the human who reached the lowest final objective value, provide the best initial experiment design for BO?

In order to evaluate the performance of HF-CL on this data, we executed 20 runs of BO with different random seeds, initialized from data from one of the 14 experts, taking their first $1, 2, 3, \ldots 10$ iterations as initial points for the BO. This results in $20 \cdot 14 \cdot 10 = 2800$ runs of BO. In order to mitigate the randomness of BO, we calculated the mean descent curves within every 20 experiments that use the same number of points from the same expert: the mean of the cumulative minimum objective values per iteration. That left us with $14 \cdot 10 = 140$ mean descent curves.

Now, in order to find an answer to the question: in which of these (human, number of initial samples) combinations did the BO manage to internalize more knowledge, we were to rank the lines based on how good they are, in terms of which one achieves a lower minimum objective value in fewer iterations. One way of ranking them is the lowest value they ever reached [23]; however, that does not take into account how many iterations were needed to reach that value. A quantity that also takes into account how early certain objective values were reached, is the Area Under Curve (AUC). A benefit of using AUC is that, for example, a curve that ends in an objective value of 2.32, but only starts to strongly descend at the 80^{th} iteration, will be ranked lower than another curve that reaches 2.33 at the third iteration, but stays there and does not improve. On the other hand, if we continue to optimize without any of the curves making an improvement, the final minimum objective values will have larger and larger impacts on the AUC, and that might reverse the ranking. However, aside from such special scenarios, AUC still looked to provide a realistic and practical ranking between the curves, thus we calculated it up to the 100^{th} iteration per curve for this comparison.

We have sorted all the descent curves corresponding to one of the 140 HF-CL combinations of (human, number of initial samples) by AUC, and we also added the BO-only case in the ranking. We see that in AUC, actually many of the human-initialized cases are better than BO on its own, i.e., 57 out of 140 cases. We have plotted the five best on Figure 3 A). We can see that, on average, these cases result in better final solutions compared to BO without human initialization, from which we conclude that HF-CL is a feasible way to improve tuning of the robotic use case up to a level that neither humans nor BO on their own are capable.

Furthermore, a particularly interesting case is shown in Figure 3 B). If we look at the full descent curve of the human here, we see that it converged really slowly, and did not achieve an impressive final objective value, but it still provided a very good initial experiment for the BO: using its first 6 iterations in HF-CL it actually did achieve the best results with respect to AUC. This means that we cannot always judge in advance the amount of useful knowledge that an expert brings to the table based on its descent curve.

While it is clear that using the data from the expert in a HF-CL loop is a feasible way of doing commissioning, it is a question if we really need the human in the loop all the time when we are commissioning similar systems with a slightly similar context. The context can have many aspects, though in the robotic VUC, there is a β parameter, which roughly determines the tendency of the slider to rotate rather than translate when pushed. This parameter is also used in the controller of the system. As this is a virtual experiment, we wanted to make it similar to a real-world scenario by introducing a model mismatch between the simulation and the control model, with parameters $\hat{\beta}$ and β, respectively, which are related through the formula

$$\hat{\beta} = \frac{\beta}{\sqrt{\beta_m}}. \tag{2}$$

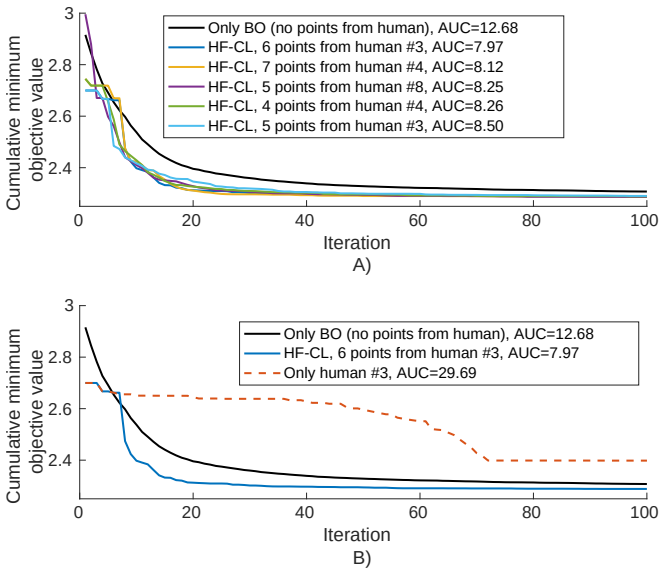


Fig. 3. BO, experts, and expert initialized BO (HF-CL) compared on the robotic use case. A) The five HF-CL cases that are best in terms of AUC. B)-C) Comparison of particular HF-CL cases with the expert whose data was used to initialize them.

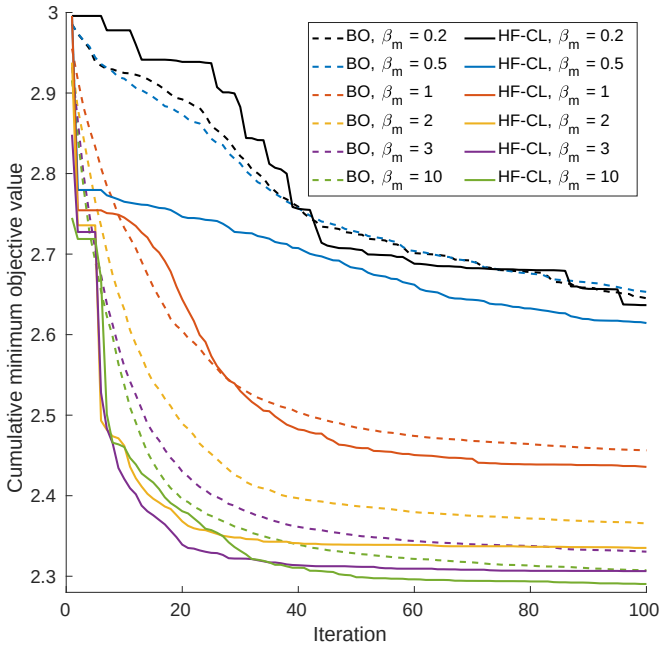


Fig. 4. Evaluating the effect of changing the context on HF-CL: changing the β_m parameter on the robotic VUC, for human #4, using 6 initial data points for HF-CL.

When we recorded the data for HF-CL with the experts tuning the VUC, β_m was set to 10. Does the method still work for different values of β_m, which lead to different trajectories for the same $\boldsymbol{K}$ parameters? In Figure 4 we see that it does: the solid lines (HF-CL) always end up below the dashed lines (Only BO). On the other hand, repeating this for different (humans, number of initial samples) shows that some combinations are quite robust to that, and some others are not. The figure shows the HF-CL case with human #4 and 6 initial points used.

Acknowledging that the HF-CL approach works, we conclude that compared to BO, humans can more easily navigate through the complex landscape of the objective at the beginning of the tuning session. However, ideally, we could capture this structural view of the human to do the same compeletely automatically.

IV. The subgoal approach

As mentioned earlier, our observation is that humans often decompose complex tasks into multiple subgoals that are easier to achieve in the right sequence than the original goal in one shot. The idea of breaking up complex tasks into subgoals is a known approach in the domain of RL. In [27], they use a Bayesian model to tell how pre-defined subtasks are correlated. They embed this model into an RL framework, and this method is then applied to complex robotic tasks such as stacking cubes on each other. In [28], BO is used on top of a subgoal-based RL setting, in order to adjust the locations of the subgoals, the length of each episode, and the number of replications per trial.

While we appreciate the ability of RL to learn controllers for complex tasks, we are looking for simpler methods that can adjust a few parameters in a minimum number of trials. RL approaches usually require many trials (in [27], tens of thousands to millions of evaluations are performed, and in [28] the total number of interactions with the environment is also in the range of 10^5 ... 10^7). These problems are much higher in both dimension (wrt. tuned parameters) and evaluation budget than our target industrial commissioning problems, which are also feasible for human operators to tune manually.

In the end, considering that humans can exploit the structure of the problem to break it up into subgoals, we would like to construct a BO mechanism to imitate such a strategy. There are some problems for which vanilla BO does not do really well; it cannot find an acceptable solution in a feasible number of iterations, but humans do.

A. The "moonshot" projectile trajectory planning virtual use case

We constructed a little game that corresponds to a tuning problem that is hard to tune by BO alone. Imagine that you are throwing a ball from a fixed position, while someone is taking a photo from another fixed position. The task is to throw the ball so that it covers the Moon at the time the photo is taken. There are three settings to tune: the φ angle, the initial velocity v_{initial} with which the ball is thrown, and the timer setting T_{timer} of the camera. This timer starts counting at the moment when the ball is thrown, and takes the picture after waiting the exact amount of T_{timer} time set. As an additional rule, the full trajectory has to be contained within the screen: if it goes outside, the photo is taken at that moment. Similar to the robotic VUC, the shot

trajectory depended in a completely deterministic way on the parameters.

An illustration of the game is shown in Figure 1. We provide the users with three sliders to tune, while they can simultaneously see the trajectory of the ball, along with the final position to hit. The final goal was to decrease the distance between the target and the ball at the time that the photo is taken, as much as possible. An interested reader can try a web version of this game[2]. As a note, the cannon projectile application tackled in [24] is similar, but with only two parameters.

Without seeing the canvas with the trajectory and physically interpreting the results, this tuning problem is difficult. Many of the possible $(\varphi, v_{\text{initial}})$ combinations will just send the ball out at the top or right side of the screen. Even if these two parameters are set in a way that the trajectory passes the target, only about 4% of the available range of T_{timer} will get the ball any closer to the target.

We did some limited preliminary experiments by giving the "moonshot" VUC to three people to tune. These suggested that humans can always solve this game in a few trials. For example, in one experiment, it took 15 trials for the expert to achieve a final distance of 0.026 meters. However, we also applied BO (using [11] with LogEI, 8 initial Sobol-sampled steps) on the same problem, and the results were not at all good: the closest we got in 300 trials was 1.8 meters, with the trajectory endpoint visibly missing the target.

There are several ways of formulating this problem as a goal sequence. Below we show a goal sequence A) that we think coincides with an engineering approach to the problem:

- *Subgoal A1:* slightly overshoot the target with the ball, so that the ball falls out on the bottom of the screen. This is achievable by fixing T_{timer} to 10.0, and playing with φ and v_{initial}.
- *Subgoal A2:* pass through the target with the trajectory, not minding taking the picture at the wrong time. This can be done by further adjusting only v_{init}, and keeping φ and T_{timer} at the previous values.
- *Subgoal A3:* precisely hit the target by adjusting T_{timer} only, and keeping the other two parameters fixed.

All these three subgoals are possible to formulate mathematically with some human effort. For example, for A2, it is the minimum distance between the target $(p_{\text{tx}}, p_{\text{ty}})$ and any point in the trajectory $(\boldsymbol{p}_{\text{rx}}, \boldsymbol{p}_{\text{ry}})$:

$$e_{\text{L}} = \min_i \sqrt{(p_{\text{tx}} - p_{\text{rx},i})^2 + (p_{\text{ty}} - p_{\text{ry},i})^2}. \tag{3}$$

The final goal, as in A3, can be formulated as:

$$e_{\text{t}} = \sqrt{(p_{\text{tx}} - p_{\text{rx,last}})^2 + (p_{\text{ty}} - p_{\text{ry,last}})^2}. \tag{4}$$

We also show a second, similar goal sequence B). This is less sophisticated, but it is simpler in the sense that it only uses components that are already in the final goal: $e_{\text{x}} = |p_{\text{tx}} - p_{\text{rx,last}}|$ and $e_{\text{y}} = |p_{\text{ty}} - p_{\text{ry,last}}|$ are the distance from the goal in the X and Y direction, respectively. This sequence is as follows:

- *Subgoal B1:* minimize e_{x} by tuning all parameters.
- *Subgoal B2:* minimize e_{y} : fix the v_{init} and φ to the best (in e_{x}) found during B1, and tune T_{timer}.

[2]https://rexpek.github.io/moonshot_game_html

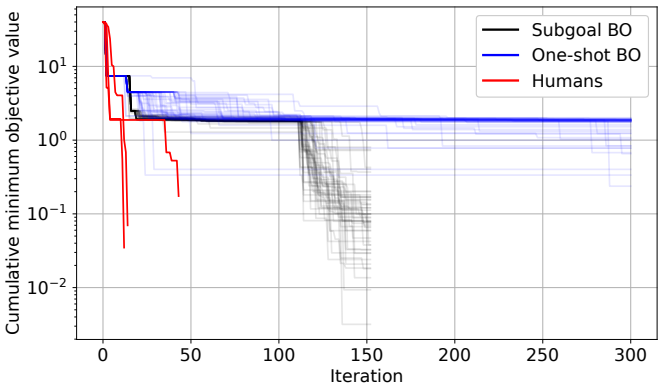


Fig. 5. Comparison of experts, vanilla, and subgoal-based BO on the "moonshot" projectile trajectory planning VUC.

- *Subgoal B3:* minimize e_t : stay within $\pm 20\%$ bounds around the best values found for the φ, v_{init} in B1 based on e_{x}, and T_{timer} in B2 based on $e_{\text{y}} + 0.1 \cdot e_{\text{x}}$.

As a note, in B3 the selection objective is $e_{\text{y}} + 0.1 \cdot e_{\text{x}}$ because there are usually two possible solutions in e_{y} out of which we want to select the one as the initial point for B3 that has not ruined e_{x}.

B. Results on the subgoal approach

We believe that humans, even if not spelled out consciously, apply a similar approach to A1-A3 or B1-B3. While these two subgoal sequences are not completely equivalent, their relation is somewhat similar to that of different basis vectors in a vector space. Now the next question is, does combining this approach with BO work? The answer is yes. We formulated B1-B3 as separate BO problems with no knowledge of the previous trials and ran them for 50 iterations each. The results are shown on Figure 5, which allows us to compare humans, vanilla BO, and subgoal-based BO. It is visible that one-shot BO does not converge to an acceptable solution on average, even in 300 steps, while the subgoal-based BO does. The difference is really striking; the one-shot BO produces solutions that are often closer to the screen edge than to the target.

While one way to see A1-A3 and B1-B3 is as one of the ad-hoc heuristics that are commonly applied in engineering, it is interesting to consider that these can be broken up into certain ingredients of knowledge, which would allow us to formulate a subgoal-based BO approach:

- the mathematical formulation of the subgoal objective,
- the sequence in which the subgoals need to be achieved,
- the subset of parameters that need to be tuned while optimizing for a given subgoal, along with any additional constraints,
- how the previous best solution is selected and transferred to the subsequent subgoal.

Now that we have collected these ingredients, the question is, can we extract any of them automatically from the data, or make it simple for the user to define them? For example, the formulation of subgoals might require someone to have a mathematical background. However, often the main objective contains multiple components, just like e_{X} and e_{Y} already appear inside e_{t}, we can thus readily use these as subgoals. The other points currently need physical insight from the

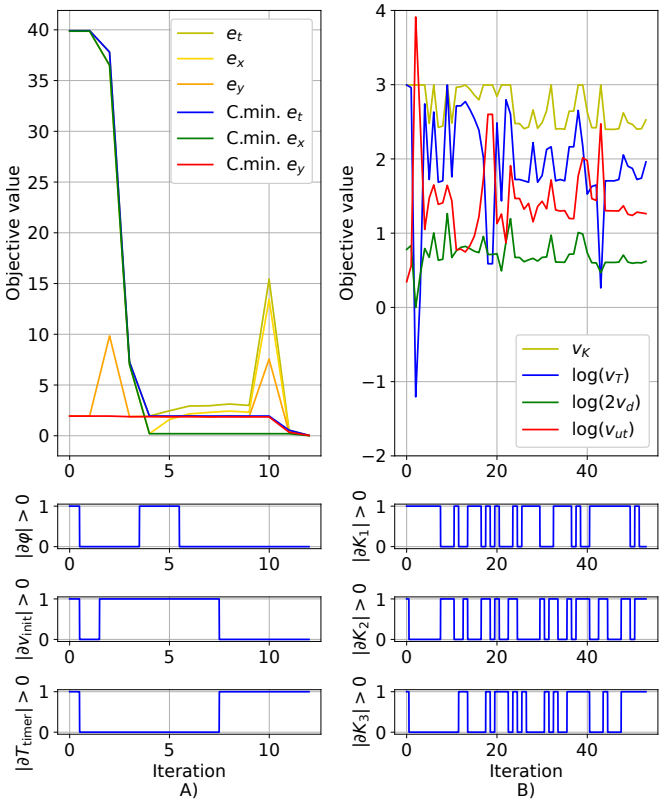


Fig. 6. Example human tuning data decomposed into subgoals, for the "moonshot" (A) and the robotic (B) VUCs. "C.min." stands for cumulative minimum.

human; thus, they form an interesting topic to tackle in future research.

Pushing this direction further, one interesting thing to check is if we can see any obvious patterns of subgoals in the data recorded from human tuning experiments. In order to evaluate these, in Figure 6 we plot the data collected from humans on both the "moonshot" and robotic VUCs. The top plots contain how the values of the subgoals ($e_\mathrm{x}, e_\mathrm{y}$ and $\log(v_\mathrm{T}), \log(2v_\mathrm{d}), \log(v_\mathrm{ut})$) and the final goals ($e_\mathrm{t}$ and v_K) changed along with the iteration number. The bottom plots such as $|\partial T_\mathrm{timer}| > 0$, $|\partial \nu_\mathrm{init}| > 0$, $|\partial \varphi| > 0$ indicate if specific parameters have changed, and if so, at which iteration: a value of 1 means a change in either the positive or negative direction, a value of 0 means it remained the same.

Figure 6 A) shows an expert tuning the "moonshot" VUC. After about 2-3 exploratory moves, the expert starts to orient the ball close to the target in the X direction, and then he changes subgoals to move the ball into the target also in the Y direction, which he achieves in four more iterations. The change happens exactly at iteration 8: prior to that, only φ and v_init are tuned, afterward only T_timer. In this last phase, we see a strong spike in e_x, because it is not the main focus anymore. We see a smaller spike also in e_y, probably because the two subgoals that we have artificially chosen are not exactly aligned with the human thinking; the expert might think more in angular coordinates than horizontal and vertical distances in this game. Nevertheless, this spike is the start of the final descent, successfully achieving the goal in two more iterations.

We were wondering if similar patterns are visible in the robotic VUC data. In Figure 6 B) we show some of the interesting components that the objective value can be decomposed into, similarly to the "moonshot" game. It is certain that aside from aiming for the minimum objective value, the human does a lot of exploration to understand the effect of each parameter. The fact that some components of the final objective are antagonistic to each other, i.e., if one decreases, the other one increases, seems to add some complications to the task. In the end, it seems that it is harder to come up with a conscious human strategy for this compared to the "moonshot" projectile trajectory planning game; nevertheless, we still see some patterns in the data that hint that the experts try to approach it based on subgoals. For example, we think a particular subgoal is being optimized for between iterations 11 and 18, visible in the descent in $\log(v_T)$, though that comes with the increase of $\log(v_{ut})$. Also, in the bottom plots, we see longer episodes of only two parameters being tuned and the third one left intact.

V. Conclusions

Our experiments showed that combining human insight and knowledge with BO for faster and cheaper tuning in industrial commissioning, such as tuning controllers, is an interesting avenue. Out of all the possible directions, we have observed two in detail: a HF-CL and a subgoal-based approach.

We found that the HF-CL approach is transferable to the problem of tuning a cascade controller for a slider-pusher setup. Our results underline that one needs to carefully compare different combinations of experts and the number of initial samples used in order to achieve a low objective value in the smallest number of iterations. Furthermore, our analysis extends that of [23] by reusing the data from the expert in an offline manner if the context is slightly different: we found that this is possible, though the mentioned comparison needs to be done to find the combination most robust to context difference. It is an open question how to tell when to give control from the expert to the computer without this complicated comparison. It would be interesting to evaluate if the human is able to assess when this switch needs to happen.

In the subgoal approach, we have found that imitating human behavior by breaking a problem up into several subgoals with different objectives certainly does help solve the toy problem of the "moonshot" projectile trajectory planning VUC. In future work, it would be interesting to detect the subgoal objectives, and the length and sequence of subgoals, focusing on additional real-world use cases. The separate parameter groups that are being tuned under each subgoal could give a hint for the latter.

Appendix: The cascade controller of the slider-pusher setup

The u_t and u_n control actions (tangential and normal velocities, respectively) are calculated by a cascade controller derived based on the model equations described in [26]. This

controller is defined as

$$\ddot{x}_r = -\frac{2}{K_1}\dot{x}_r + \frac{1}{K_1^2}(x^* - x), \ddot{y}_r = -\frac{2}{K_2}\dot{y}_r + \frac{1}{K_2^2}(y^* - y),$$
$$u_n = \frac{\beta^2 + d^2}{\beta^2}\sqrt{\dot{x}_r^2 + \dot{y}_r^2}, \quad \theta_r = -\arctan\frac{\dot{x}_r}{\dot{y}_r},$$
$$\ddot{\theta}_r = -\frac{2}{K_3}\dot{\theta}_r + \frac{1}{K_3^2}(\theta_r - \theta), \quad d_r = \frac{u_n \pm \sqrt{u_n^2 - 4\dot{\theta}_r^2\beta^2}}{2\dot{\theta}_r},$$
$$\ddot{d}_r = -\frac{2}{K_4}\dot{d}_r + \frac{1}{K_4}(d_r - d), \quad u_t = \dot{d}_r + \alpha\frac{d}{\beta^2 + d^2}u_n,$$

where x^*, y^* are the target positions, $K_{1\ldots3}$ are the parameters tunable by the user, and K_4 is fixed to 0.5 to simplify the task. Furthermore, $[x, y, \theta, d]$ and $[\dot{x}_r, \dot{y}_r, \dot{\theta}_r, \dot{d}_r]$ are the state of the system and the controller, respectively, and β is a contextual parameter that can be calculated from the physical dimensions of the block. Out of the two solutions for d_r, we choose the one that is closer to d in the previous step.

ACKNOWLEDGMENT

This research is supported by the Flanders Make SBO project REXPEK: Reproducing Expert Knowledge.

Declaration of use of AI tools [according to IEEE PSPB Operations Manual 2024, 8.2.1/B/10.]: OpenAI Codex, GPT-3.5, GPT-4, GPT-4o models, up to versions available on November 25, 2024, were used for implementing many parts of the software for VUC GUIs and plots. GPT-4o was used to help with grammar and style improvements for the text. Furthermore, the practical solution of comparing convergence curves with AUC was inspired by GPT-4o. The authors have verified any output from AI for correctness to the best of their knowledge.

REFERENCES

[1] Ikujiro Nonaka. The knowledge-creating company. *Harvard Business Review*, 85(7-8):162–171, 2007.

[2] Tariq Samad, Margret Bauer, Scott Bortoff, Stefano Di Cairano, Lorenzo Fagiano, Peter Fogh Odgaard, R. Russell Rhinehart, Ricardo Sánchez-Peña, Atanas Serbezov, Finn Ankersen, Philippe Goupil, Benyamin Grosman, Marcel Heertjes, Iven Mareels, and Raye Sosseh. Industry engagement with control research: perspective and messages. *Annual Reviews in Control*, 49:1–14, 2020.

[3] Oluwasegun Ayokunle Somefun, Kayode Akingbade, and Folasade Dahunsi. The dilemma of PID tuning. *Annual Reviews in Control*, 52:65–74, 2021.

[4] Sébastien Gros and Mario Zanon. Data-driven economic NMPC using reinforcement learning. *IEEE Transactions on Automatic Control*, 65(2):636–648, 2020.

[5] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[6] Simon Kristoffersson Lind and Johannes Tykesson. Self-optimization of camera hardware, 2021. Master's thesis, Department of Computer Science, Lund University.

[7] Yifan Wu, Aron Walsh, and Alex M. Ganose. Race to the bottom: Bayesian optimisation for chemical problems. *Digital Discovery*, 3(6):1086–1100, 2024.

[8] Jolan Wauters, Joris Degroote, Ivo Couckuyt, and Guillaume Crevecoeur. SAMURAI: A new asynchronous Bayesian optimization technique for optimization-under-uncertainty. *AIAA Journal*, 60(11):6133–6156, November 2022.

[9] Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. Bayesian optimization for adaptive experimental design: A review. *IEEE access*, 8:13937–13948, 2020.

[10] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[11] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G. Wilson, and Eytan Bakshy. BoTorch: A framework for efficient Monte-Carlo Bayesian optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 21524–21538. Curran Associates, Inc., 2020.

[12] Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google Vizier: A service for black-box optimization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 2 of *KDD '17*, page 1487–1495. ACM, August 2017.

[13] João P.L. Coutinho, Lino O. Santos, and Marco S. Reis. Bayesian optimization for automatic tuning of digital multi-loop PID controllers. *Computers and Chemical Engineering*, 173:108211, 2023.

[14] Matthias Neumann-Brosig, Alonso Marco, Dieter Schwarzmann, and Sebastian Trimpe. Data-efficient autotuning with Bayesian optimization: An industrial control study. *IEEE Transactions on Control Systems Technology*, 28(3):730–740, 2020.

[15] Christopher König, Miks Ozols, Anastasia Makarova, Efe C. Balta, Andreas Krause, and Alisa Rupenyan. Safe risk-averse Bayesian optimization for controller tuning. *IEEE Robotics and Automation Letters*, 8(12):8208–8215, December 2023.

[16] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.

[17] Carl Hvarfner, Danny Stoll, Artur Souza, Luigi Nardi, Marius Lindauer, and Frank Hutter. πBO: Augmenting acquisition functions with user beliefs for Bayesian optimization. In *International Conference on Learning Representations*, 2022.

[18] Peng Liu. Human-in-the-loop Bayesian optimization with no-regret guarantees. *SSRN Electronic Journal*, page 18, Oct 2022.

[19] Michael A. Gelbart, Jasper Snoek, and Ryan P. Adams. Bayesian optimization with unknown constraints. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, page 250–259, Arlington, Virginia, USA, 2014. AUAI Press.

[20] Yuki Koyama and Masataka Goto. BO as assistant: Using Bayesian optimization for asynchronously generating design suggestions. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, UIST '22. ACM, October 2022.

[21] Javier González, Zhenwen Dai, Andreas Damianou, and Neil D. Lawrence. Preferential Bayesian optimization. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1282–1291. PMLR, 2017.

[22] Tom Savage and Ehecatl Antonio del Rio Chanona. Human-algorithm collaborative Bayesian optimization for engineering systems. *Computers and Chemical Engineering*, 189:108810, 2024.

[23] Keren J. Kanarik, Wojciech T. Osowiecki, Yu Lu, Dipongkar Talukder, Niklas Roschewsky, Sae Na Park, Mattan Kamon, David M. Fried, and Richard A. Gottscho. Human–machine collaboration for improving semiconductor process development. *Nature*, 616(7958):707–711, March 2023.

[24] Elvis A. Eugene, Kyla D. Jones, Xian Gao, Jialu Wang, and Alexander W. Dowling. Learning and optimization under epistemic uncertainty with Bayesian hybrid models. *Computers & Chemical Engineering*, page 108430, 2023.

[25] Tom Lefebvre, Sander De Witte, Thomas Neve, and Guillaume Crevecoeur. Differential flatness of slider–pusher systems for constrained time optimal collision free path planning. *Journal of Dynamic Systems, Measurement, and Control*, 145(6):061001, 2023.

[26] Neve, Thomas and Lefebvre, Tom and De Witte, Sander and Crevecoeur, Guillaume. Flatness-based MPC using B-splines transcription with application to a pusher-slider system. In *2023 IEEE/ASME Internation conference on Advanced Intelligent Mechatronics (AIM)*, pages 132–137. IEEE, 2023.

[27] Haozhe Ma, Thanh Vinh Vo, and Tze-Yun Leong. Mixed-initiative Bayesian sub-goal optimization in hierarchical reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, Auckland, New Zealand, May 2024.

[28] Yijia Wang, Matthias Poloczek, and Daniel R. Jiang. Dynamic subgoal-based exploration via Bayesian optimization. *Transactions on Machine Learning Research*, 2023.