REAL-TIME OPTIMIZATION STRATEGIES USING SURROGATE OPTIMIZERS

Dinesh Krishnamoorthy* and Sigurd Skogestad
Department of Chemical Engineering
Norwegian University of Science and Technology (NTNU), Trondheim, Norway

Abstract Overview

This extended abstract focuses on addressing the computational challenges associated with solving numerical optimization problems for online process optimization. In order to address this challenge, we propose to use a "surrogate optimizer" to approximate the online optimization problem using artificial neural networks. We propose three different surrogate optimization structures and demonstrate the performance using a CSTR case example.

Keywords

Neural Network, Real time optimization, surrogate optimization, explicit optimization

Introduction

Real-time optimization (RTO) is traditionally based on rigorous steady-state process models that are used by a numerical optimization solver to compute the optimal steady-state inputs and setpoints. The optimization problem needs to be re-solved every time a disturbance occurs. This step is also known as "data reconciliation". A schematic representation of the RTO structure is shown in Fig.1. Since steady-state process models are used, it is necessary to wait, so the plant has settled to a new steady-state before updating the model parameters and estimating the disturbances. It was noted by Darby et al. (2011) that the steady-state wait time is one of the fundamental limitations of the traditional RTO approach.

Among others, one of the challenges with online process optimization is the computational issues including numerical robustness. Despite having a large potential for applying optimization-based approaches, solving numerical optimization problems can be computationally expensive, even with today's computing power. For example, in many large-scale problems, solvers may take a long time to converge to the optimal solution or, in some cases, may even fail to converge to an optimal solution. Addressing the computational bottleneck is therefore an important aspect of addressing the computational issues of solving online optimization problems.

In order to avoid solving numerical optimization problem online, we aim to approximate computationally intensive optimization problems using machine-learning algorithms. Instead of developing surrogate models that will be used in the optimizer, we propose to build "surrogate optimizers" or "AI optimizers" that approximate the numerical optimization solvers. In this extended abstract, we present three different kinds of surrogate optimizers, as described below.

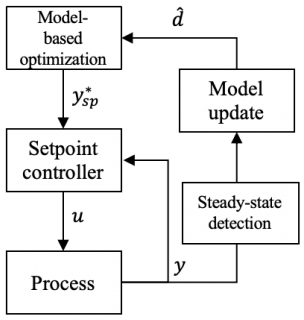

Figure 1. Traditional RTO structure

* To whom all correspondence should be addressed

Open-loop surrogate network based on measured disturbances:

In this approach, the main idea is to solve the numerical optimization problems offline and use the trained network online to drive the process to its optimal operation. By doing so, we effectively move the computationally intensive optimization problems offline and the trained network approximates the numerical optimization solver and acts as the surrogate for online optimization.

In other words, we train a neural network to map the relation between disturbances d to the optimal setpoints, such that given the current disturbance, the trained network provides the corresponding optimal solution as output as shown in Fig.2. This structure approximates the model-based optimization block from Fig.1. This is analogous to multi-parametric programming, where the models are used offline to generate an optimal solution space as a function of the disturbances/parameters (Pistikopulous, 2009). Here, instead of using a pre-computed solution space using first-principle models, we use a neural network to map the relation between the various disturbances to the optimal setpoints.

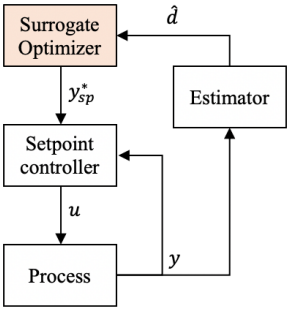

Figure 2. Open-loop surrogate optimization structure using neural network that maps measured disturbances to optimal setpoint

Closed-loop surrogate network based on available measurements

Alternatively, models can also be trained to directly map the relation between the available measurements to the optimal setpoints in the same fashion as training a model from the disturbances to the optimal setpoints as shown in Fig.3. In this structure, the neural network approximates both the model update and the optimization block from Fig.1. This approach would be preferred over the open-loop approach in Fig.2, since this structure does not require the disturbances to be measured, which can be advantageous in many systems where disturbance measurements are not available.

Gradient-based approach

Another approach is to train a network that directly maps the measurements to the steady-state cost gradient. This approach does not require solving numerical optimization problems either offline or online. The process measurements y along with the steady state gradient data evaluated offline are used to train a neural network to map the local steady-state gradient as a function of the current measurements.

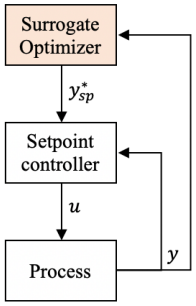

Figure 3. Closed-loop surrogate optimization structure using neural network that maps measurements to optimal setpoint

The trained model is then used online to estimate the steady-state cost gradient using real-time measurements. Optimal operation can then be achieved by simply controlling the estimated gradients to a constant setpoint of zero using simple feedback controllers, thereby achieving the necessary condition of optimality. A schematic representation of the proposed surrogate optimization approach is shown in Fig.4.

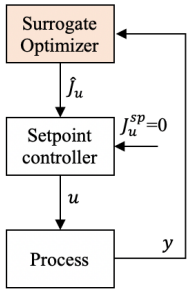

Figure 4. Gradient-based surrogate optimization structure using a neural network to estimate the steady-state cost gradient.

Simulation results

The effectiveness of the proposed surrogate optimization approach in this section was tested using a CSTR case example used in (Economou et al., 1986) and (Krishnamoorthy et al., 2019), where the optimization problem is given as,

$$\min_{T_i} J = -[2.009C_B - (1.657 \times 10^{-3}(T_i - 410))^2]$$

$$s.t.$$

$$\begin{aligned} \frac{dC_A}{dt} &= \frac{1}{\tau}(C_{A,i} - C_A) - r \\ \frac{dC_B}{dt} &= \frac{1}{\tau}(C_{B,i} - C_B) + r \\ \frac{dT}{dt} &= \frac{1}{\tau}(T_i - T) + \frac{-\Delta H_{rx}}{\rho C_p} r \end{aligned} \tag{1}$$

The optimization problem was solved offline for various realizations of the disturbances to generate the training data. As a proof of concept, a shallow neural network with 10 neurons was trained and validated using this data to approximate the optimization problem from the disturbance to the optimal setpoints. The trained neural network was then used online to optimize the process using the structure in Fig.2. The process was simulated with disturbances in the feed concentrations as shown in Fig.5. Note that in this simulation, we assume that the disturbances are measured. The cost function using the proposed approach was compared with the ideal steady-state optimal cost (solid black lines) and is shown in Fig. 5 (solid red lines).

We then use the training data to train and validate a neural network from the available measurements to the optimal setpoints to build a closed-loop surrogate network using available measurements as shown in Fig.3. The process was simulated for the same disturbances and in this case, we assume that the disturbance measurements are not available. The simulation results using this approach is also shown in Fig.5 (dashed blue lines) to compare the open-loop and closed-loop surrogate approaches and we see that the system is able to reach the optimum despite the unmeasured disturbances.

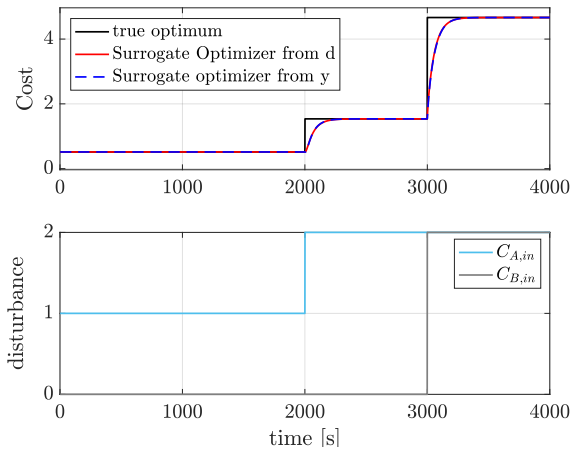

Figure 5. Cost obtained using the surrogate optimizer structure from Fig.2 (solid red) and Fig.3 (dashed blue) compared with the ideal optimal cost (solid black)

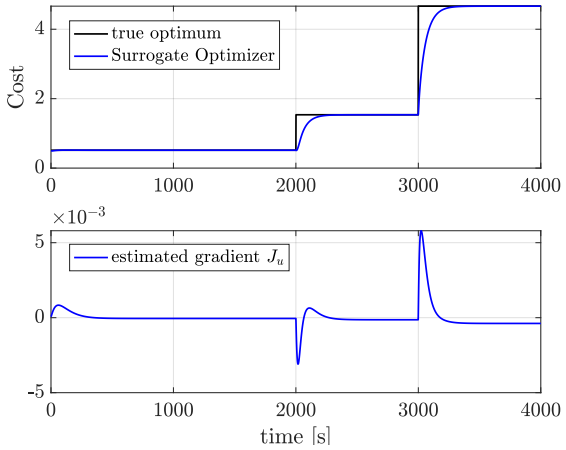

Figure 6. Cost obtained using the gradient-based surrogate optimizer structure from Fig.4 (solid blue) compared with the ideal optimal cost (solid black)

The gradient-based surrogate optimization approach was also tested on the same CSTR case example. The process was simulated to generate measurement data for different disturbance realizations. The measurement data along with the corresponding steady-state cost gradient evaluated offline was used to train and validate a neural network with 10 neurons. The trained model was then used to estimate the steady-state gradient, which was controlled to a constant setpoint of zero using a PI controller. The cost function along with the estimated gradient for the same disturbance as in the previous case is shown in Fig.6 (solid blue lines) along with the ideal steady-state cost (solid black lines).

Acknowledgments

The authors gratefully acknowledge the financial support from SUBPRO, which is financed by the Research Council of Norway, major industry partners and NTNU.

References

Pistikopoulos, E. N. Perspectives in multiparametric programming and explicit model predictive control. AIChE Journal 2009, 55 (8), 1918-1925.

Krishnamoorthy, D., Jahanshahi, E. and Skogestad, S., 2018. Feedback Real-Time Optimization Strategy Using a Novel Steady-state Gradient Estimate and Transient Measurements. Ind. & Eng. Chem. Res., 58(1), pp.207-216.

Economou, C. G.; Morari, M.; Palsson, B. O. Internal model control: Extension to nonlinear system. Industrial & Engineering Chemistry Process Design and Development 1986, 25, 403–411.