Globally Optimized Production by Co-operating Production Agents Based on Bellmans Principle

Norbert Link
Karlsruhe University of Applied Sciences
Karlsruhe, Germany
e-mail: norbert.link@hs-karlsruhe.de

***Abstract*— The production of items is usually separated into a sequence of processing steps from raw materials to the finished product. Each of the processing steps is executed by dedicated machines where the output of one machine is the input of the next machine. The total effort of all processes can be drastically reduced and the resulting quality of the end product be maximized by exploiting the mutual dependencies of the individual process steps. The concepts of task-driven, intelligent production agents are extended to account for this global optimization task, maintaining the autonomous decision of the individual agent about the optimal process parameters. This can be reached by supplying the local production agent with information about the effect of some of its output on the efforts of the subsequent processes and with information about the actual input to be processed. When the process agent knows the efforts related to its own parameters required to transform the input into some output states, the overall effort can be minimized. Stochastic process influences turn the optimization into a Markov decision process where Bellmans equation can be applied to yield on average the best total result at lowest effort. The encountered exponential complexity when solving Bellmans equation via Dynamic Programming is relieved by Approximate Dynamic Programming. By looking upon one single process, as a process chain with discrete, repetitive steps with different process parameter values, the same optimization concept can be applied to control the individual process. Agents using this optimization scheme require special capabilities: output state estimation, state transformation function representation, Bellman optimization and assessment function representation (assigning effort to process output). The concepts, the architecture, the required components and the methods will be presented in this paper.**

Keywords-manufacturing; agent systems; process chain; optimization; control; Markov decision process; Bellmans principle.

I. INTRODUCTION

Optimization of production processes can be performed on different levels: (a) supply chain, (b) workflow, (c) machine. The traditional way of looking at optimization assumes perfectly specified input/output relations on all levels. Deviations are considered as failures with eventual fall-back strategies to react on. In this sense only the most efficient supply chain/workflow has to be found and the best parameter values for the machine settings have to be investigated. The first two optimization tasks are of the discrete type and subject to intense research which has found its way to corresponding software products controlling supply chains and workflows.

On the machine level, a continuous optimization task in the parameter space is encountered, where an optimum point maximizing a certain objective function (of quality, etc.) or minimizing a certain cost function (of energy, wear, time, etc.) has to be found. Usually, these functions are not given as analytic functions of the process parameters which would make optimization an easy task.

One of the highest remaining potentials, but with also the highest challenges (even scientifically), is to account for the mutual influence of the processes in subsequent processes of a chain. The interrelation between processes is due to the fact that parts, with the same specification, may differ in aspects which are relevant for subsequent processing.

Example: Different milling parameters can produce the same geometry but different surface layer properties (hardness, stresses, grain properties, etc.). Later heat treatment may then affect the geometry differently or subsequent surface processing will need different effort.

In order to achieve an overall optimum over a whole process chain, these relations have to be accounted for. This requires that a process gets the information about its actual input. This information will be generated by the preceding process from its process data by means of a quality model. With this information, the parameters have to be set in a way that the cost related with them plus the expected cost of the subsequent processes, related to the output, minus the price of the final product are minimized. There exist scientific approaches to address this challenge, but they are of high computational complexity and need the full set of information, as mentioned above.

An individual process which needs to compensate fluctuations during processing can be considered as temporal sequence of processing steps where after each step the parameter values for the next step are decided. The time steps can be looked upon like processes in a chain, and the approaches for process chains can be applied for the optimal control of a single process as well.

The paper is organized as follows: The basic concept of an agent-based global optimization is developed in Section II, where also the required information and the additional components of self-optimizing agents are identified. Section III discusses the knowledge extraction and representation methods, which are called “Optimizations models” and Section IV presents the concept and methods of a chain-

 ISBN: 978-1-61208-437-4

optimizing closed-loop controller. Conclusions about the presented concept are drawn in Section V.

II. Basic Concept and Components

The core requirement to control individual processes in a way, that the result is also optimized for the subsequent processes, is to make the local optimization account for the subsequent processes. This can be achieved in the following way: if the process knows about the different efforts, which are caused by its different end states (results), then it has to consider the sum of its own effort to reach a certain end state plus the efforts of this end state for the subsequent processes. To be more precise, it is the effort or cost expectation value of the subsequent processes related to the end state. Now it needs to find the parameter values which yield the lowest total cost. Figure 1 shows the associated information flow and required data with the example of a three-step process chain, where a product is made from steel sheets by heat treatment and deep drawing of parts, which are welded together in the final step.

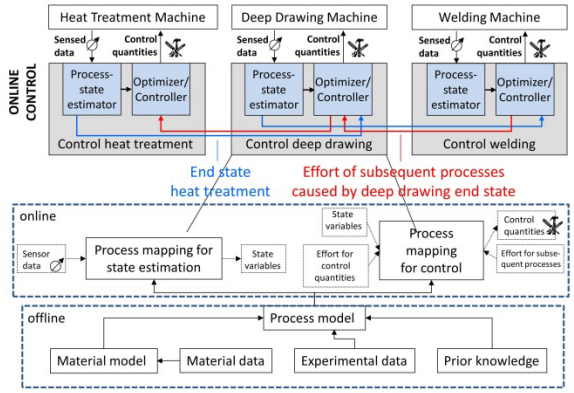

Figure 1. Chain optimization work flow and required components

The information about the „subsequent“ cost is propagated from the process next downstream the process chain (red arrow in Figure 1). To decide upon the control quantity or process parameter values, the information about the initial state is required as well. This is the final state of the preceding process and the information is propagated from there (blue arrows in Figure 1). The process state has therefore to be derived by the preceding process from the data from machine-attached sensors. This transformation from sensor data values to state variable values is represented by the blue box „process state estimator“ in Figure 1, which is using a **quality model** to specify the transformation. The optimizer/controller uses a **state transition model (or process model)** which relates the process state at a future time, to the process state at present under certain values of the control quantities. By virtue of this model, it can find the control quantities minimizing the total cost. This is depicted in the „online“ box in Figure 1, summarizing the activities which must be performed during processing. These activities are controlled by **models** (representing the sensor data – state relation and the state transition) which are fed into the „process mapping for state estimation“ and in the „process mapping for control“ respectively. These models must be executable in process real time, which means that they must be computationally simple. These models represent the dedicated process knowledge and have to be created in an off-line process. Usually, there are no simple analytical models for production processes, which mean that the models have to be formed from formalized prior expert knowledge and from experimental data (either real or simulated process executions with determinations of the process states). For simulation experiments, material models and data are required.

When all the models are available for the processes of a process chain, the path (sequence of parameter values for the process steps) with minimum total cost can be found. The painful way would be to consider all combinations of all process parameters of all processes and to search the combination with the lowest cost (full search). Fortunately this is not necessary thanks to Bellmans theorem [5] which allows to propagate the cost upstream from the end, and to set up the „subsequent cost function“ for each process in the chain.

In order to enable the downstream optimization, the process machines are extended accordingly with dedicated controls as shown in Figure 1. Such a control is an integrated component of some general production agent (for Instance NETDEV [1]) shell which has to acquire the necessary information and to perform the necessary optimizing control activities.

A process chain is assumed to be a Markov process [8] where each step is transforming the input state of the (semi-finished) product into some output state, and where the final product is the result of a sequence of such otherwise independent transformations.

Describing the

- input state via a vector $\vec{x}$ of state quantities, the
- output state by vector $\vec{y}$ of state quantities and the
- process as a general transformation operator $\overset{T}{\rightarrow}$,

we can write a process step as

$$\vec{x}_{i-1} \overset{T_i}{\rightarrow} \vec{y}_i,$$

where a process chain is then written as

$$\vec{x}_0 \overset{T_1}{\rightarrow} \vec{y}_1 = \vec{x}_1 \overset{T_2}{\rightarrow} \vec{y}_2 = \vec{x}_2 \ldots = \vec{x}_{N-1} \overset{T_N}{\rightarrow} \vec{y}_N.$$

If the control is supposed to perform an optimizing process $\overset{T_{i,opt}}{\longrightarrow}$, it will adjust the process parameters (or control quantities) $\vec{u}_i$ in a way $\vec{u}_{i,opt}$, that the transformation will produce the lowest effort (cost) related to the parameters $J_{i,loc}(\vec{u}_i)$ necessary to transform $\vec{x}_{i-1}$ into $\vec{y}_i$ and lowest effort (cost) for all subsequent downstream processes related to the produced output $\vec{y}_i$, namely $J_{i,sub}(\vec{y}_i)$.

In order to do so properly, it requires the following information:

- input state $\vec{x}_{i-1}$

- process model describing the transformation $\overset{T}{\rightarrow}$, e.g., in functional form $\vec{y}_i = \vec{T}(\vec{x}_{i-1}, \vec{u}_i)$
- cost associated with the process parameters $J_{i,loc}(\vec{u}_i)$
- cost of subsequent processes associated with output state $J_{i,sub}(\vec{y}_i)$

The input state information is supplied by the previous process which uses the sensor and process data to estimate its output state. Each NETDEV control component has therefor to generate this estimate for the next subsequent process. The estimation is transforming the observable sensor and process data $\vec{s}_i$ into an estimated output state $\vec{y}_{i,est}$.

This "measurement" transformation $\overset{M}{\rightarrow}$ is represented by a second process model, the "inverse measurement model", which could again be described in functional form $\vec{y}_{i,est} = \vec{M}(\vec{x}_{i-1}, \vec{s}_i)$.

The optimizing process $\xrightarrow{T_{i,opt}}$ consists of minimization of the total cost (local plus subsequent) with respect to the process parameters.

$$\vec{u}_{i,opt} = \text{argmin}_{\vec{u}_i}\{J_{i,loc}(\vec{u}_i) + J_{i,sub}(\vec{y}_i) | \vec{y}_i = \vec{T}(\vec{x}_{i-1}, \vec{u}_i)\} \quad (1)$$

The optimization therefore requires the supply of the respective information which is generated via dedicated models. This is depicted in Figure 2.

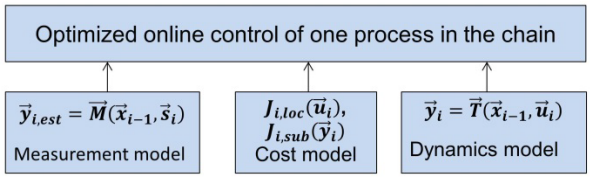

Figure 2. Information, models required for process-chain otpimizing controls

During on-line optimization operation, these models are fed with the actual data (sensor, control quantities or process parameters) from which they derive the actual state, the expected output state and the related cost information, which is used by the optimization algorithm (solving Bellmans equation) in order to derive the optimal process parameters.

III. METHODS FOR DERIVING THE OPTIMIZATION MODELS

As discussed so far the models are transformation functions of different kind:

1. The inverse measurement model which relates a process state to sensed process data and recorded process parameters,
2. The process dynamics model which relates a next state to the present state under the action of dedicated process parameter values,
3. The cost local cost model which relates effort (cost) to dedicated values of the process parameters and
4. The subsequent cost model which relates expected cost to dedicated final process states.

The requirements on the transformation models are:

a. Sufficient precision to enable the optimization algorithms to yield results close to the theoretical optimum and to let them converge there robustly and quickly.
b. Low computational complexity to allow information delivery at times defined by the process real time requirements.

Requirement b prohibits quite often the direct usage of numerical process simulation or the direct solution of first principal equations during processing time, since in almost all cases both approaches are computationally extremely complex or analytically too difficult for a real process.

This is the motivation, why the usage of numerically highly efficient regression models which are formed from process data is proposed. The split of the overall solution in the two steps of model creation and model application allows specifying low-complexity, real-time suited on-line models, which are efficiently executed during processing. The latter are representatives of a restricted class of the process (defined by the process capabilities of the device), encapsulating the respective process knowledge. The functional form of these on-line models is determined in a much more time-consuming off-line step in advance. Real-time properties can be achieved this way for an optimizing control by representing the transformations in a simple, explicit form which is derived from the existing process knowledge.

A generic method to set up these on-line models is to derive them from process data which are collected to form a representative sample of the input and output quantities of the respective transformations. A regression analysis is then applied to the sample data yielding an estimation of the desired transformation function. For this purpose, an Ansatz function with a set of parameters is defined where the parameters are fitted by means of the sample data set via minimization of a deviation cost function. The Ansatz function can be selected as a set of non-linear base functions (such as polynomial, logistic or Gaussian) using standard methods of (preferentially robust) optimization (M-estimators or RANSAC) or as being composed of a set of pre-defined mathematical symbols where the formula structure and the parameters are optimized via genetic or evolutionary algorithms [7], forming the so-called symbolic regression.

The data sampling can be either performed via laboratory experiments or by numerical experiments if such simulation models are available for the process under consideration. Also, a combination of both can be used, where the majority of samples are created in the computer

via simulation which is again calibrated by laboratory experiments to ensure correct simulation.

The model formation, as discussed so far, complements the required methods. The according extension of the diagram of Figure 2 is shown in Figure 3.

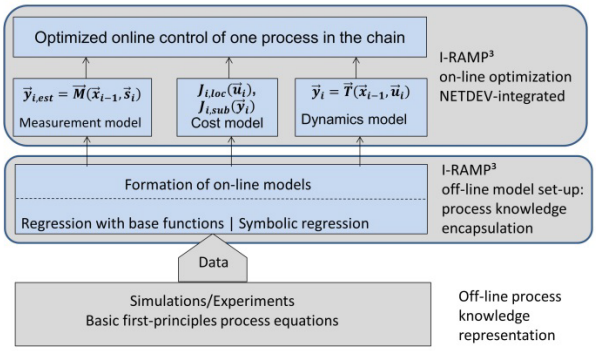

Figure 3. Generic model creation based on representative process data samples

The above discussed model generating components extend the production agent concepts, but not their internal structure.

IV. CONCEPT OF A CHAIN OPTIMIZING CLOSED-LOOP CONTROLLER

We have introduced methods to derive necessary models in a generic way by representing them as transformation functions which are derived via regression from data generated either by numerical or real experiments or from first principles equations. We can generate the necessary input for optimization this way.

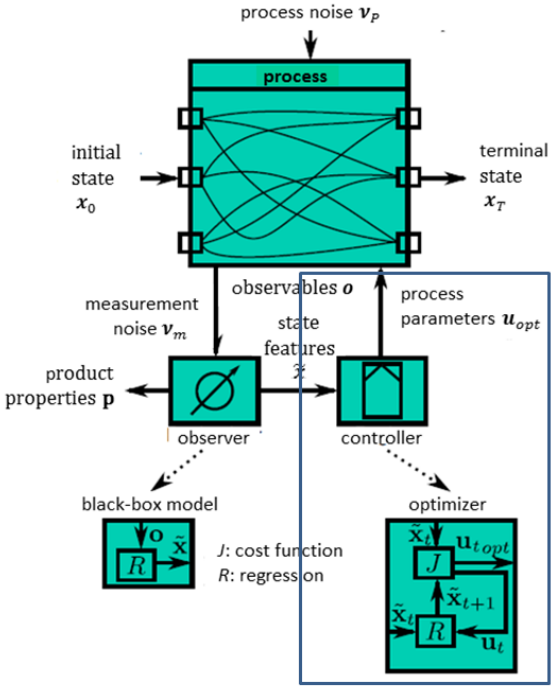

Figure 4. Closed-loop control istantiated by an optimizer

Still a method to find the optimum of the cost function (1) is missing. Figure 4 gives an overview of the local agent context of the closed-loop process control which is instantiated as an optimizer of (1).

The discrete time sequence of a single process or a chain of subsequent processes was introduced in the generic chain optimization concept as a Markov process which is optimized using the Bellman principle [6].

Figure 5 shows a time discrete Markov process where the output $\boldsymbol{x}_i$ of one process i step (green box) is the input of the next process step $i+1$. The output $\boldsymbol{x}_{i+1}$ is the result of the process i (under parameter values $\boldsymbol{u}_i$). The local cost J_{Di} is associated with the deployed parameter values $\boldsymbol{u}_i$. From the subsequent process a cost function $\tilde{J}_{i+1}$ is supplied which associates cost with output $\boldsymbol{x}_{i+1}$. This cost function is the expectation value of the cost, which output $\boldsymbol{x}_{i+1}$ is likely to produce with all subsequent processes minus the price (negative cost) of the expected final state. Since the output $\boldsymbol{x}_{i+1}$ is a result of the input and the process parameter values $\boldsymbol{u}_i$, $\tilde{J}_{i+1}$ is depending indirectly on $\boldsymbol{u}_i$ as well.

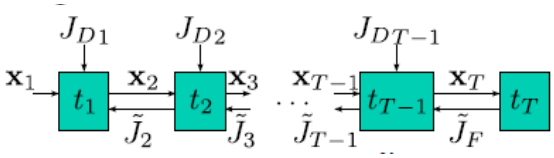

Figure 5. Markov process and Bellman principle

The best parameter values are therefore the ones minimizing the sum of $\tilde{J}_{i+1}$ and J_{Di} which is expressed in

$$u_{t\,opt}(\boldsymbol{x}_t) = \underset{u_t \in U_t}{argmin}\left\{J_{Dt}(\boldsymbol{x}_t, u_t, \boldsymbol{x}_{t+1}) + \left\langle \tilde{J}_{t+1}(\boldsymbol{x}_{t+1})\right\rangle\right\} \tag{2}$$

Where $\langle\ \ \rangle$ denotes the statistical expectation value.

The stochastic influence of disturbances on the process is reflected by the fact, that an output state $\boldsymbol{x}_{t+1}$ is reached from an input $\boldsymbol{x}_t$ state under given process parameter values $\boldsymbol{u}_t$ only with a certain probability $P(x_{t+1}|x_t, u_t)$. Then

$$\left\langle J_{t+1}(x_{t+1})\right\rangle = \sum_{x'_{t+1} \in X_{t+1}} P(\boldsymbol{x}'_{t+1}|\boldsymbol{x}_{t,}, u_t)\, J_{t+1}(\boldsymbol{x}'_{t+1}) \tag{3}$$

With

$$\sum_{x'_{t+1} \in X_{t+1}} P(x'_{t+1}|x_{t,}, u_t) = 1$$

The expectation values can be calculated in a back-propagation way from the final process where the cost function of the final state is given by the price achieved with the end quality. Given the probabilities of arriving at an end state (with given cost) from a certain initial state, with all possible parameters (and associated cost), will result in an expectation value of the cost associated with the initial state. This way the expected cost can be calculated for all input

states which are the output states of the preceding process. The procedure can now be repeated for the process before the last process and so on, until the initial process is reached and the output states have assigned expected cost values for all processes in the chain. The procedure described so far, yields only such cost values for discrete states. These can be stored in a look-up table which can be used to yield a cost value for a state under consideration via assigning the cost value of the Nearest Neighbor (NN) or via interpolation with the nearest neighbors [9]. A complete function for $\langle J_{t+1}(x_{t+1})\rangle$ can be achieved for all processes if regression methods are employed as discussed in the preceding section.

This is the backward Dynamical Programming (DP) algorithm of which the pseudo-code is given in Figure 6.

```
Initialize J_T(x_T)   ∀x_T ∈ X_T
for t = T − 1 : 1
  for all x_t ∈ X_t
    Initialize J_topt(x_t)
    for all u_t ∈ U_t
      J_tmp = C_Dt(x_t, u_t, x_t+1)
            + Σ_{x'_t+1 ∈ X_t+1} P(x'_t+1 | x_t, u_t) J̃_t+1(x'_t+1)
      if (J_tmp < J_topt(x_t))
        J_topt(x_t) = J_tmp
      end if
    end for
  end for
end for
```

Figure 6. Backward DP algorithm for a finite time horizon

The following models are involved in the optimization

- State transition model $\boldsymbol{x_{t+1}} = \boldsymbol{f_t}(\boldsymbol{x_t}, \boldsymbol{u_t}, \boldsymbol{\nu_{p_t}})$ in the deterministic case with added noise $\boldsymbol{\nu_{p_t}}$ or $P(x_{t+1}|x_t, u_t)$ in the stochastic case, from data via regression
- Cost model $J_t(\boldsymbol{x_t})$
- Control law $\boldsymbol{u_t} = \boldsymbol{\pi_t}(\boldsymbol{x_t})$
- Noise model from assumptions or experimental measurements
  - Normally distributed (μ, Σ)
  - Additively superposed with state

Cost considered

- J_{D_t}: Local cost (due to process efforts), derived from vendor knowledge
- J_F: Final (negative) cost (price achieved with end quality), derived from sales knowledge
- $\tilde{J}_{t+1}$ or $\langle J_{t+1}(x_{t+1})\rangle$: Cost of subsequent process steps due to current state, calculated via backward calculation and regression or explored during processing

The proposed method suffers from the curse of dimensionality [6], when longer process chains and high dimensional state spaces are involved which prevents real-time control. The state space dimensionality can be drastically reduced by means of dimension reduction methods such as “Principal Function Approximators” (a kind of non-linear partial least squares method) [4] which already yields an optimally reduced state representation, derived from the observed process values. This approach belongs to the class of Approximate Dynamic Programming (ADP) [9]. If only a few process steps are involved, backward DP approach can be applied as was shown with a two stage process in [3].

Other approaches, also allowing the treatment of more complex processes, such as “Approximate forward Dynamic Programming” [2][11] have been investigated in [10].

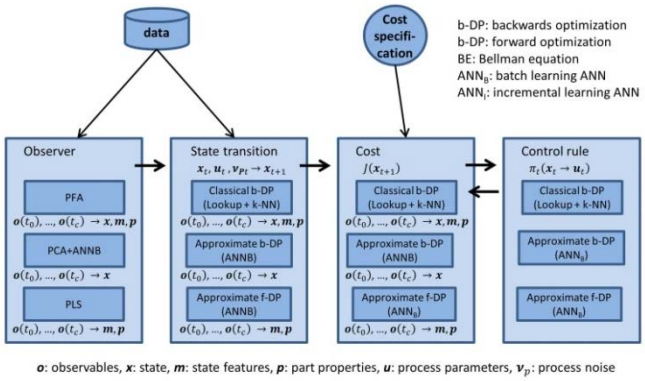

Figure 7. Generic model of a downstream chain optimization control

Combining the observer methods with the different DP approaches finally forms the generic model of downstream process chain optimization (Figure 7).

V. CONCLUSION

A concept has been presented, allowing a process-dedicated production agent to set its process parameter values in a way that a process chain, which it is actually part of, will be (on average) globally optimized. Central optimizer components for each dedicated process chain are no longer required. This makes production more flexible and efficient, since the agent concept is maintained and the efforts are distributed optimally among the process chain members. In order to enable the agents for global optimization, they have to be supplied with extra information about the actual input they have to work on and about the efforts they produce by their outputs with the subsequent processes. This cost function has to be set up by a higher-level component from all downstream process information and is in the responsibility of the process chain agent. A generic method for this purpose has been presented, completing the set of methods contained in the extra components of chain-optimizing methods.

ACKNOWLEDGMENT

The author wants to thank especially Melanie Senn for evaluating, developing and promoting the ADP methods during her work in our research group. The research leading to these results has received funding from the [European

Union] Seventh Framework Programme ([FP7/2007-2013]) under grant agreement no. [314329]."

REFERENCES

[1] G. Gonçalves, J. Reis, R. Pinto, M. Alves, and J. Correia, "A step forward on Intelligent Factories: A Smart Sensor-oriented approach," Emerging Technology and Factory Automation (ETFA), 2014 IEEE (pp. 1-8), 2014.

[2] J. H. Lee, "Model predictive control and dynamic programming," 11th International Conference on Control, Automation and Systems, pp. 1807–1809, 2011.

[3] M. Senn, J. Schäfer, J. Pollak and N. Link, "A system-oriented approach for the optimal control of process chains under stochastic influences," AIP Conference Proceedings 1389, pp. 419–422, 2011.

[4] M. Senn and N. Link, "A universal model for hidden state observation in adaptive process controls," International Journal on Advances in Intelligent Systems 4 (3–4), pp. 245–255, 2012.

[5] R. E. Bellman, "Dynamic Programming," Courier Dover Publications, Mineola, New York, USA, 2003.

[6] W. B. Powell, "Approximate Dynamic Programming: Solving the Curses of Dimensionality," 2nd ed., Wiley, Hoboken, New Jersey, USA, 2011.

[7] M. Davarynejad, J. van Ast, J. L. M. Vrancken and J. van den Berg, "Evolutionary value function approximation," 2011 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), pp. 151–155, 2011.

[8] N. E. Pratikakis, M. J. Realff and J. H. Lee, "Strategic capacity decision-making in a stochastic manufacturing environment using real-time approximate dynamic programming," Naval Research Logistics 57 (3), pp. 211–224, 2010.

[9] J. H. Lee and W. Wong, "Approximate dynamic programming approach for process control," Journal of Process Control 20, pp. 1038–1048, 2010.

[10] M. Senn, N. Link, J. Pollak, and J. H. Lee, "Reducing the computational effort of optimal process controllers for continuous state spaces by using incremental learning and post-decision state formulations," Journal of Process Control, 24(3), pp.133-143, 2014.

.

 ISBN: 978-1-61208-437-4