Iterative Process Design with Surrogate-Assisted Global Flowsheet Optimization

Tim Janus* and Sebastian Engell

DOI: 10.1002/cite.202100095

Flowsheet optimization is an important part of process design where commercial process simulators are widely used, due to their extensive library of models and ease of use. However, the application of a framework for global flowsheet optimization upon them is computationally expensive. Based on machine learning methods, we added mechanisms for rejection and generation of candidates to a framework for global flowsheet optimization. These extensions halve the amount of time needed for optimization such that the integration of the framework in a workflow for iterative process design becomes applicable.

Keywords: Deep learning, Machine learning, Process design, Process optimization, Process synthesis

Received: June 01, 2021; *revised*: September 3, 2021; *accepted*: October 15, 2021

1 Introduction

An interdisciplinary group of experts using steady-state process simulations, heuristic rules and expert knowledge usually performs the design and optimization of chemical plants. The goal is to find feasible and economically promising process configurations. Fig. 1 shows the different phases during the development of a chemical process until the construction or modification of the plant. The decisions in the early phase are very important as the economics of the process are mostly fixed in this phase. The efficiency in the early phase can be improved by a systematic computer-aided global search, which, in comparison to interactive simulation studies and manual search, can explore a broader range of options and determine the best parameters more accurately. Our proposed framework is applicable as soon as early models for unit operations that are linked to possible process designs in a flowsheet are available. Besides that, the framework can also be used for legacy flowsheets of existing plants. In this case, a modification or adaption of the plant can be investigated in a systematic computer-aided global optimization to support the decisions of the experts.

Steady-state process simulation is a mature field with several commercial products in the market. Asprion et al. [1] give an overview of in-house and commercial process simulators used in the process industry. Bröcker et al. [2] point out that in many areas the companies in the process industry depend on a single software provider for process simulation and that there is a high demand for a better integration of process simulators in the complete process lifecycle. Therefore, process simulators of the next generation must better support interoperability, such that the management of models and the transfer of knowledge between process models of different phases of the lifecycle becomes easier. Especially commercial process simulators lack openness, and their interoperability is limited. Additionally, the integration of new models, e.g., data-based models, comes with a high amount of manual effort and the application of external solvers is difficult, as model equations and derivatives are inaccessible.

The available techniques for steady-state process simulations can be divided into the sequential-modular and the equation-oriented simultaneous approaches. Both are explained well in Biegler et al. [3]. From the perspective of optimization, it is important to distinguish between derivative-based and derivative-free methods, where algorithms of the former class are more efficient as they can incorporate more information. Therefore, optimizations based upon process simulators that provide the equations and derivatives of the models are more efficient in general. However, most commercial flowsheet simulation tools do not provide this information. Nonetheless, experts in industry widely use these simulators due to their ease of use, their extensive library of models and thermodynamic properties and the possibility to reuse available unit models and the accumulated experience with these. If it is intended to use such flowsheet simulators for flowsheet optimization, derivative-free optimization techniques [4] must be used, and the simulator then acts as an "oracle" for the optimizer. Based upon the explored options, the optimizer proposes design param-

Tim Janus, Prof. Dr.-Ing. Sebastian Engell
tim.janus@tu-dortmund.de
TU Dortmund University, Process Dynamics and Operations Group, Emil-Figge-Straße 70, 44137 Dortmund, Germany.

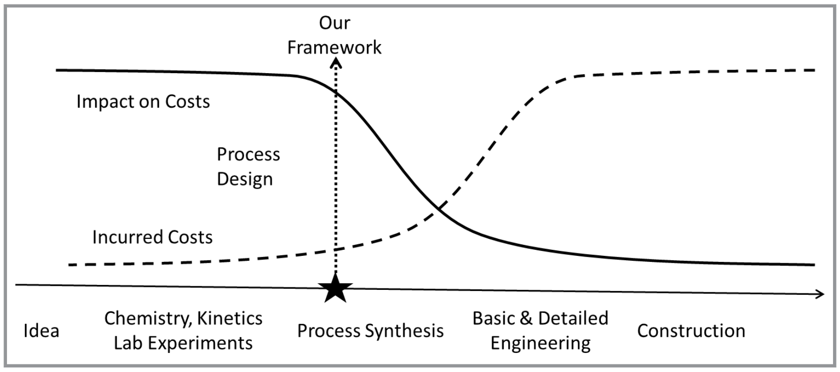

Figure 1. The proposed framework in the context of the phases of process design. The solid line shows the possible impact on costs and the dashed line shows the incurred costs in different phases.

eters and the flowsheet simulator returns the values of the stream and equipment variables, which are used to calculate the cost and the constraint functions. Although parts of the cost functions can be encoded using standard libraries, e.g., the utility and investment costs of unit operations, the cost function and the constraint functions can be very specific and strongly depend on the underlying case study. Some simulation environments, e.g., the well-known simulator Aspen Plus also provide integrated local optimization methods, but still cannot perform a global flowsheet optimization. A major issue when applying flowsheet simulators to evaluate process designs is the significant failure rate of the simulations. If the values of the input variables are admissible, the rate of failure depends on the initial values, on the tolerances and on the maximum number of iterations. When derivative-free optimization methods are used, there can be up to 50 % failing simulations for which only very limited information on the reasons for failure is provided, and the computation time is wasted.

The process design environment couples an evolutionary algorithm to a process simulator that provides only limited information. In this work, the commercial process simulator Aspen Plus is used. The optimization framework builds upon earlier work on a memetic algorithm for the design of integrated process units [5] and it supports several derivative-free optimization methods as local solvers [6] and different initialization strategies of the process simulator [7]. Also, an extension for multi-objective optimization [8] was developed. Janus et al. [9] found that the use of derivative-free optimization methods for local refinement is not worth the effort in comparison to only use an evolution strategy during the course of an optimization. They also integrated an interface to internal local optimizers of the process simulators and used it for constraint satisfaction. Although this approach performs better than derivative-free alternatives, it has technical limitations that depend on the vendor, e.g., arbitrary cost functions cannot be formulated. To the best knowledge of the authors, no in-house nor commercial process simulator currently provides robust global optimization. The open challenge for a global optimization based on calls of commercial process simulators are the high computation times due to large numbers of simulations.

Under the name of surrogate-assisted optimization, methods from machine- and deep learning have been applied to enhance the convergence of derivative-free optimization approaches [10–12].

In this contribution, we extend a global flowsheet optimization framework that uses external process simulators and was designed with the workflow during process design in mind by mechanisms to reduce the number of calls to the simulator. In Sect. 2 we introduce a case study, in Sect. 3 we present the framework and propose a workflow to explain how the frameworks interfaces with a team of experts and give some technical details of the interfaces between the framework and third-party programs. Then in Sect. 4 we explain the machine learning (ML) methods that we used to speed up the optimization process [13]. Afterwards, we investigate how these changes improve the performance of the optimization and show that the turnaround time between an engineer and the framework is significantly reduced. Finally, in Sect. 6, we discuss how parts of this framework may be used in future platforms for process simulation and give an outlook to possible future work.

2 Case Study: Hydroformylation of 1-Dodence to Tridecanal

In this contribution, the case study of the homogenously catalyzed hydroformylation of 1-dodecene in a thermomorphic solvent system (TMS) is investigated. Tridecanal is a long-chained aldehyde that is used in perfume production. To the authors knowledge there is no efficient industrial process for the production of tridecanal. The well-known Ruhrchemie-Rhone Poulenc process [14] is an efficient process for the hydroformylation of short chained olefins but it is not applicable to longer olefins. The TMS process considered here has been investigated in the Collaborative Research Center DFG Transregio SFB 63 "Integrated chemical processes in liquid multi-phase systems" InPROMPT. The process produces *n*-tridecanal by hydroformylation of a mixture of the reactants 1-dodecene and the solvents dimethylformamide (DMF) and decane. The homogenous rhodium catalyst used in the reaction is very expensive, therefore it has to be recovered from the product stream. The price increased by 355 % from May 01, 2020 to May 01, 2021, to nearly 30 000 US $ per ounce[1]. The phase behavior

1) Prices accessed on May 28, 2021, from https://tradingeconomics.com/commodity/rhodium

of the mixture of DMF and decane is temperature-dependent and the temperature of the mixture is changed to switch between a homogenous mixture in the reactor and a mixture with two liquid phases in the decanter [15] where the polar phase that contains the rhodium-based catalyst is recycled to the reactor.

Fig. 2 illustrates the flowsheet, whereby a bold font highlights the DOFs (degrees of freedom). The feed stream consists of the reactant 1-dodecene and the solvents DMF and decane. This stream is heated and pressurized to create a homogenous mixture. This feed stream, carbon monoxide and hydrogen are fed into the reactor. In addition to the main reaction of 1-dodecene to *n*-tridecanal, four side reactions occur in the reactor [16]. Then, a cascade of two heat exchangers cools down the mixture. Hex 2-2, uses cooling water to pre-cool the mixture to 30 °C and Hex 2-1 uses ammonia as coolant to achieve temperatures below 5 °C. Afterwards, a liquid-liquid separator splits the mixture into two phases. The polar DMF-rich phase contains the catalyst and is recycled back into the reactor. The decane-rich phase contains the remaining reactant, the product and the by-products, that are fed into the distillation column for purification. The top stream of the column is recycled back to the reactor and a product-rich liquid stream is obtained at the bottom of the column. The bottom stream must have a purity of 99 mol %.

$$f(x) = \frac{C_{MAT} + C_{INVEST} + C_{OP}}{8000\ \dot{m}_{PRODUCT}} + p(v) \tag{1}$$

$$p(v) = \begin{cases} 0, & v = 0 \\ 4000 + v \cdot 10000, & v > 0 \end{cases} \tag{2}$$

The applied cost function, see Eq. (1), is an indicator of the production cost per ton of product and it consists of costs for the feed, depreciation of the investment, and operational costs as described by Turton et al. [17]. A penalty term is applied if a constraint, e.g., the requested product purity, is violated. The linear function $p(v)$ in Eq. (2) depends on the constraint violation v and consists of a constant that is significantly higher than the costs of a good process alternative and a coefficient that ensures that the cost increase for purity violations are in the order of magnitude of the function *f*(*x*). Most process simulators offer options to resolve a constraint violation, e.g., by design specifications or by using internal solvers. Exploiting these options can lead to significant performance improvements [9], but they are vendor-specific. Our goal here is to improve the handling of design constraints independent from a specific simulator so that the framework can be used with any flowsheet simulation software.

The DOFs can be divided into two groups. Three DOFs represent decisions at design time and seven DOFs represent decisions that are controllable during the operation of the plant. The design-time decisions are the number of theoretical stages, the feed stage of the column and the residence time of the reactor, which defines the required volume of the reactor. The remaining DOFs can be adapted to a certain degree during the operation. These include the pressure and the temperature of the reactor content, the temperature of the decanter, the distillate to feed ratio and the reflux ratio of the column, and the flows of DMF and decane in the feed stream.

3 Iterative Process Design and Systematic Global Flowsheet Optimization

A methodology is proposed that couples iterative process design with a framework for computer-aided global flowsheet optimization. Fig. 3a shows this framework. The upper part represents the inputs of the framework, where a library of cost functions is provided. The two boxes at the bottom show two third party programs, an external process simulator that provides limited information (black or gray box) and a machine learning (ML) toolbox. In this work, Aspen Plus and MATLAB was used as process simulator and as machine learning toolbox, respectively.

In Fig. 3a, the block in the middle contains the three main modules of the framework. Based on the inputs (values of the DOF, cost function) the framework collects further information from the process simulator. The information consists of the units for all variables that are used in the output report. The framework checks the consistency of the optimization problem that is described in a text-based XML file, i.e., if the mapping of degrees of freedom and inputs of the cost function to variables of the process simulation is valid. Then the framework also checks if the cost function is solvable, which means that at least one sequence of calculations exists such that all terms of the cost function can be evaluated. The cost functions in our library can be used with any set of units that can be used in an Aspen Plus flowsheet. The

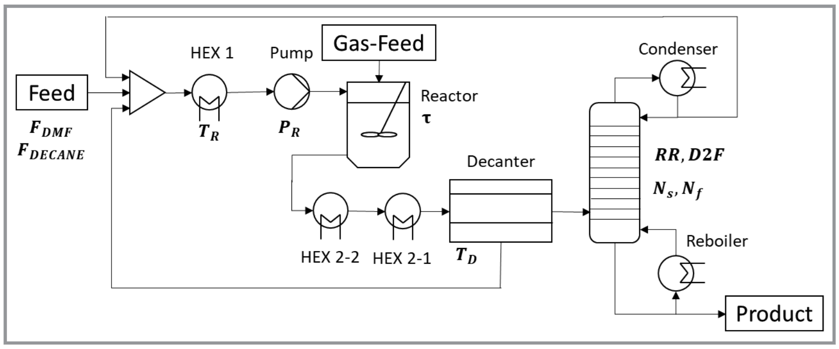

Figure 2. Flowsheet of the hydroformylation of 1-dodecene case study (TMS) with three major unit operations and two recycle streams.

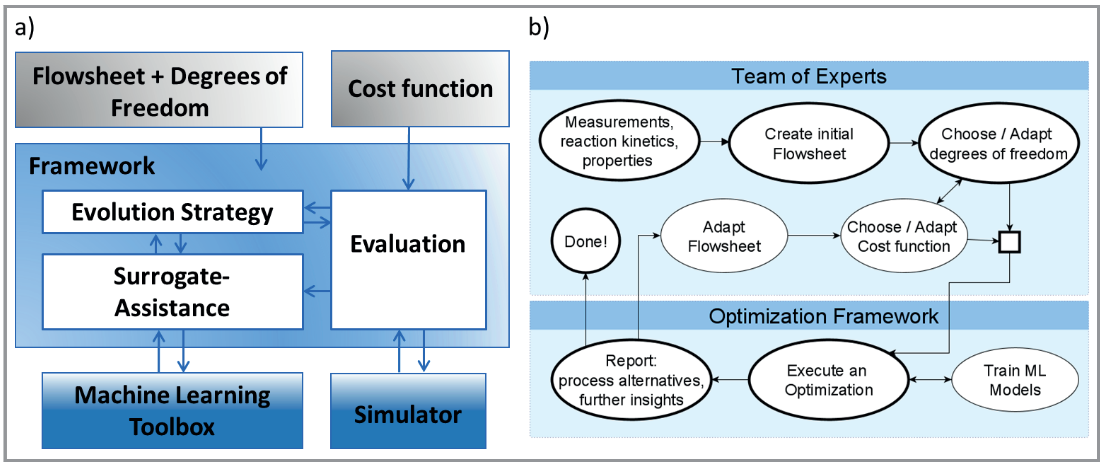

Figure 3. a) Overview of the main components of the optimization framework. b) Diagram of the workflow for iterative process design with systematic global flowsheet optimization.

framework uses two sets of units for the process-related variables and switches between them automatically, such that the cost function library can implicitly interoperate with flowsheets that use different magnitudes and even sets of units from those used in the cost function.

3.1 Evolution Strategy

The global optimization is controlled by an evolution strategy (ES) [18] that initializes a predefined number of μ process alternatives with random uniformly distributed degrees of freedom (DOFs). These process alternatives, called individuals, form a population. To distinguish between different points in time during the optimization, the term generation is used, whereby population is a synonym for the current generation. In the main loop, the ES generates new process alternatives (offspring) based on a random pair of process alternatives from the population. Two operators, recombination and mutation, modify the DOFs and afterwards the offspring are evaluated, i.e., they receive a fitness score that reflects the constraint satisfaction and the value of the cost function. After the ES has generated a predefined number λ of offspring, the members of the next generation are selected by using the best alternatives from the union of the new population and those former designs that are not older than κ generations. The ES repeats these steps and searches for the most economic process alternative until a predefined stop criterion, e.g., a specific number of process simulations or of generations without progress, is reached.

An evaluation of a process alternative consists of three steps. First, the framework prepares the process simulator by setting the values of the degrees of freedom and optionally by initializing the temperatures and compositions of streams, temperature profiles of unit operations or other variables of the simulation [7]. In this work, the default initialization strategy of Aspen Plus is used without estimates for any stream or unit operation. Then the framework simulates the flowsheet, but depending on the initialization, this simulation may not converge. If a simulation does not converge, the candidate process receives the highest possible costs, and no further process information is stored. If the simulation does converge, the framework uses the simulation output to calculate the cost function and stores the entire information for further usage in the machine learning toolbox.

3.2 Workflow for Iterative Process Design

Fig. 3b illustrates the workflow when applying the proposed optimization framework. The bold circles represent required steps in the workflow. After the reaction kinetics and thermodynamic properties have been determined, the team of experts designs an initial flowsheet and then they select promising DOFs and start the optimization that generates many data points. When the optimization finishes, a set of good process alternatives and further information are accessible, e.g., which unit operation produces a simulation error how often. This knowledge helps to understand the operating window of the process. Based on the set of good process alternatives, the team decides if they start another iteration and which adaptation they will apply. Hereby, the team may adapt the flowsheet, the cost function or other parts of the model. They may also revise the selection of the DOFs or enhance the detail of the process model by introducing a more accurate model of a specific process element.

3.3 Interfaces between the Optimization Framework and Third-Party Programs

The IT infrastructure is an important aspect when several components of software need to interplay, as the optimization framework, a process simulator, and a machine learning toolbox. Machine learning tasks, i.e., the training of surrogate models, and the individual simulations of candidates that are proposed by the population-based optimization algorithm are highly parallelizable and therefore benefit from distributed computing. The ML toolbox also benefits from computing machines that use a modern GPU as the training on them is much faster than on a CPU. The interfaces defined for the communication between the framework and the third-party programs are implemented using gRPC, a modern open source high performance remote procedure call (RPC) framework. gRPC can run in any environment, i.e., it can be used to connect computers in a small network, e.g., some unused workstations at a company, but it can also be used in a larger environment, e.g., data centers.

The most important functions of the optimization framework to communicate with the process simulator are shown in Tab. 1. The transfer, simulate and collect functions represent a typical IPO (input, process and output) structure. It also supports many auxiliary functions, beside load and save.

4 Improved Convergence by Surrogate-Assisted Optimization

This chapter explains how surrogate models can shorten the computation time needed for global flowsheet optimization. Generally, a surrogate model approximates a more complex model and has the purpose to reduce either the computation time or the memory usage of a complex model. For global flowsheet optimization, most of the CPU time (approx. 99 %) is spent in the simulation of the process simulator. Therefore, surrogate models are applied to help reducing the number of required simulations. A reduction of the number of simulations can be achieved in two ways, by reducing the number of non-promising simulations or by proposing good candidates. Therefore, a candidate rejection (CR) mechanism is realized as well as a candidate generation (CG) mechanism that applies derivative-based optimization to surrogate models that are trained in the course of the optimization.

The training data for the surrogate models is generated on the fly. After the ES called the simulator n_0 times, a first training is triggered using the collected data from these simulations. The parameter n_0 is case study dependent but as a rule of thumb several hundred simulations are needed before an initial training. From there on, the surrogate models assist the optimization, which speeds-up the remaining optimization. To make use of the generated information, the surrogate models are retrained every n_{it} simulation, such that the surrogate model becomes more accurate in the region that was recently searched by the evolution strategy and that probably contains the optimum.

4.1 Candidate Rejection

To reduce the number of non-promising simulations, a classification was realized that predicts whether a simulation is promising, i.e., the simulation converges and the purity of the product stream is high. Two neural networks are trained. The inputs of both are the DOFs of the optimization. One network P_c predicts whether the simulation will not converge and returns a value between 0 and 1. This return value indicates how sure the neural network is that the simulation will not converge.

The second network P_p predicts the product purity of tridecanal in the product stream. To handle the uncertainty in the predictions of the neural network, the statistical error of the prediction for the feasible simulations of the training set is considered. In this work the mean absolute error is used with respect to the feasible solutions in the training set as shown in Eq. (3).

$$\begin{aligned} & mae_{feas} = \frac{1}{N_{feas}} \sum \left(\left| Y_{feas} - P_{feas} \right| \right) \\ & Y_{feas} = \{ \forall y_i \in Y | y_i \rangle 0.99 \},\ P_{feas} = \{ \forall p_i \in P | y_i \rangle 0.99 \}, \\ & i = 1 \ldots N_{feas} \end{aligned} \tag{3}$$

In Eq. (3), the set Y represents the ground truth and the set P represents the predictions given by the trained network P_p on the training data. Infeasible data points are neglected and a boundary, as shown in Eq. (4), is calculated and takes the mean absolute error and the standard deviation of the absolute prediction error into account to decide which process candidates are filtered.

$$\begin{aligned} & P_p(x) \leq 0.99 - \\ & \left(mae_{feas} + 1.96 \left(\sqrt{\frac{1}{N_{feas}} \sum \left(|Y_i - P_i| - mae_{feas} \right)^2} \right) \right) = b \end{aligned} \tag{4}$$

Table 1. Functions for the communication between the optimization framework and Aspen Plus.

Function	Arguments	Description
transfer	List of Key-Value Pairs	Transfers values of variables identified with keys to Aspen Plus.
simulate	No	Starts an Aspen Plus Simulation and returns the simulation outcome.
collect	List of Keys	Collects values of variables identified with keys from Aspen Plus.
load	Filename	Loads the Aspen Plus flowsheet stored in the file given as an argument.
save	Filename	Saves the Aspen Plus flowsheet stored in the file given as an argument.

In Eq. (4), $P_p(x)$ represents the prediction of the purity by the surrogate model, where x are the inputs that are used to simulate the process candidate. The value b marks the boundary, i.e., every prediction of a purity greater b indicates a promising candidate. This boundary is dynamic and recalculated after each training.

A candidate is rejected if either the predicted product purity is below the dynamic boundary $P_p(x) < b$ or if the first network predicts a non-converging simulation $P_p(x) < 0.5$. This default value can be tuned for a more or less aggressive rejection behavior.

4.2 Candidate Generation

The candidate generation mechanism should guide the search in a way that the purity constraint is satisfied by a more efficient optimization based upon the previously described neural networks. In many cases, the optimum of a chemical process with respect to the cost function is located at boundaries, e.g., on the boundary for the product purity. Therefore, candidates are generated that achieve an estimated purity of 99 % by solving the optimization problem formulated in Eq. (5)

$$\min_x \left|0.99 - P_p(x)\right| \quad s.t. \quad P_c(x) > 0.5 \tag{5}$$

If the Aspen simulation of a process model has converged, its inputs x are used as the starting point of a local optimization method that solves Eq. (5). In this work, an interior point method implemented in MATLAB is applied to solve the optimization problems. For the TMS case study, the computation time to solve this optimization problem is negligible in comparison to the time that an Aspen Plus simulation needs to converge. Afterwards, a simulation validates the proposed design x, and if the simulation converges, a process alternative that achieves a purity close to the boundary is obtained.

5 Results

This section investigates the performance of four variants of the optimization method. The reference variant (REF) applies an evolution strategy (ES) without support based upon surrogate models. The variants with only candidate rejection (CR) and candidate generation (CG) and as the fourth variant the combination of both extensions (CR+CG) are compared. The optimization was repeated ten times for each variant to reduce stochastic effects. In the following, the optimizations with respect to the minimum cost indicator over the number of required simulations is compared, as the training time of the neural networks is negligible for the case study.

The 40 optimizations were performed on different computers but as an estimate, for a medium desktop computer (Intel I5 4th generation, 16 GB Ram) without parallelization, 1500 simulations are reached after 2 h and 3000 simulations after 4 h.

Fig. 4 and Fig. 5 show convergence plots. The plots are divided into regions by gray dotted lines. The leftmost region in Fig. 4 is called data collecting region, it comprises $n_0 = 500$ simulations. In this region, the variants do not differ from each other, as no surrogates are trained. The differences of the average cost indicators in the left region

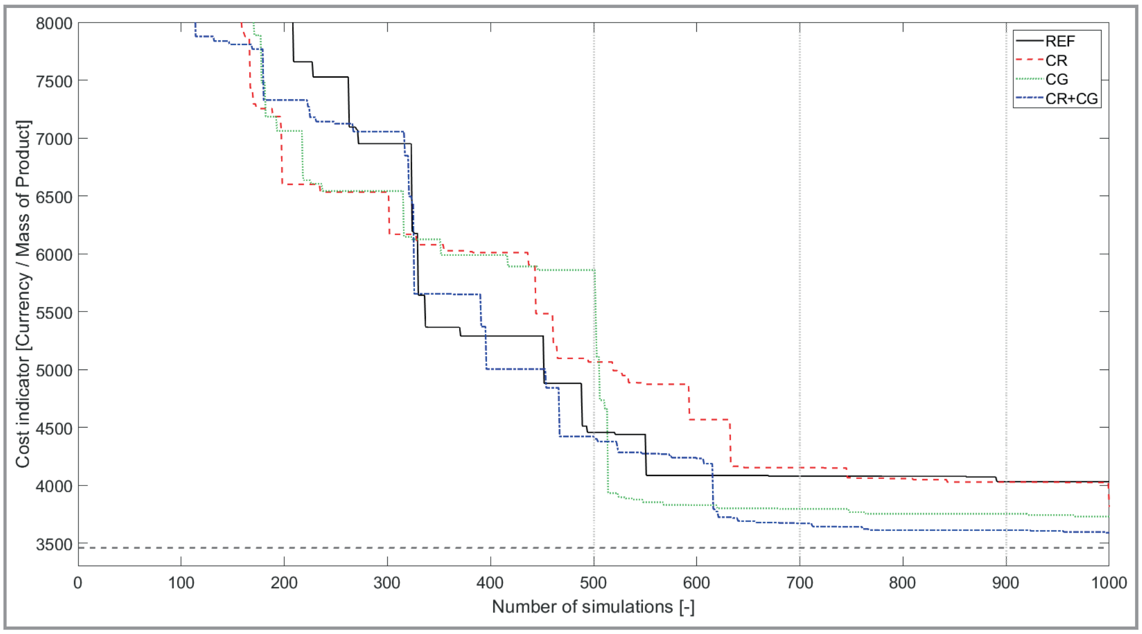

Figure 4. Convergence plot showing the average cost indicators of the first 1000 simulations for the variants REF (black), CR (red), CG (green) and both CR+CG (blue) based on ten optimization runs.

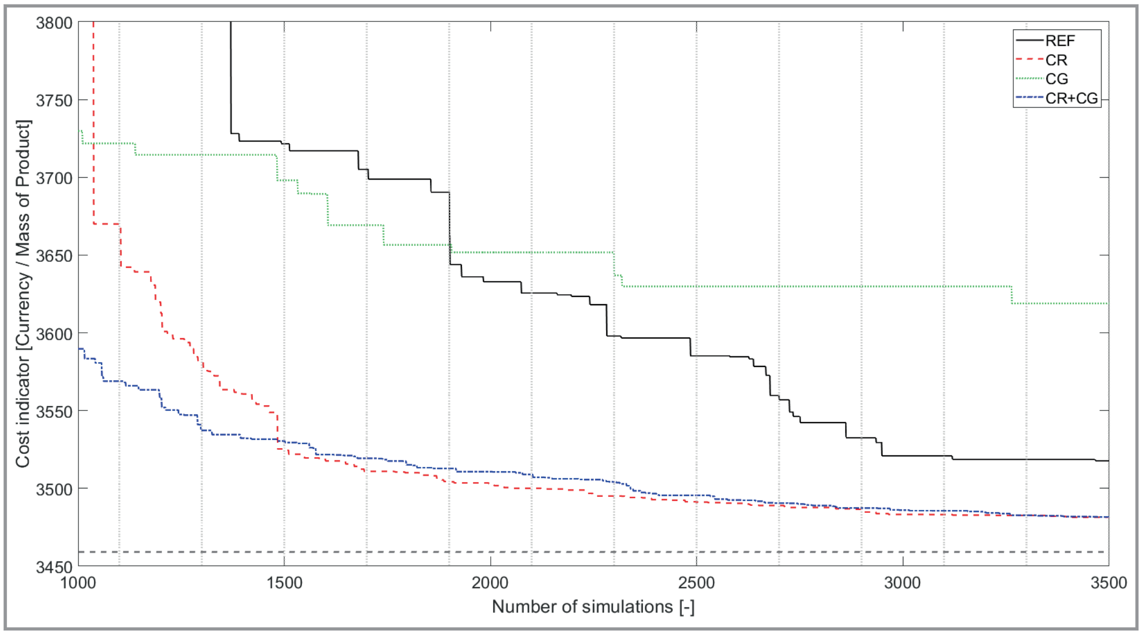

Figure 5. Convergence plot showing the average cost indicators of the simulations 1001–3500 for the variants REF (black), CR (red), CG (green) and both CR+CG (blue) based on ten optimization runs.

of Fig. 4 are due to the stochastic nature of the optimization method and caused by the variation of the ten independent optimizations for each variant in their early phases. The leftmost gray dotted line indicates when the first surrogate models are trained. Each following region has a width of n_{it} = 200 simulations and the line indicates when a retraining of the surrogate models with new data occurs. Both, n_0 and n_{it}, were selected on the basis of preliminary studies of the TMS case study. The REF variant needs 2726 simulations and the best variant CR+CG needs 1244 simulations to reach a value of the cost function of 3550 which is 2.6 % larger than the best known cost, i.e., the CR+CG variant needs less than half of the simulations than REF to converge to a point close to the optimum.

Fig. 6 shows the variance of the best-found process alternatives between the repeats of the optimizations starting at

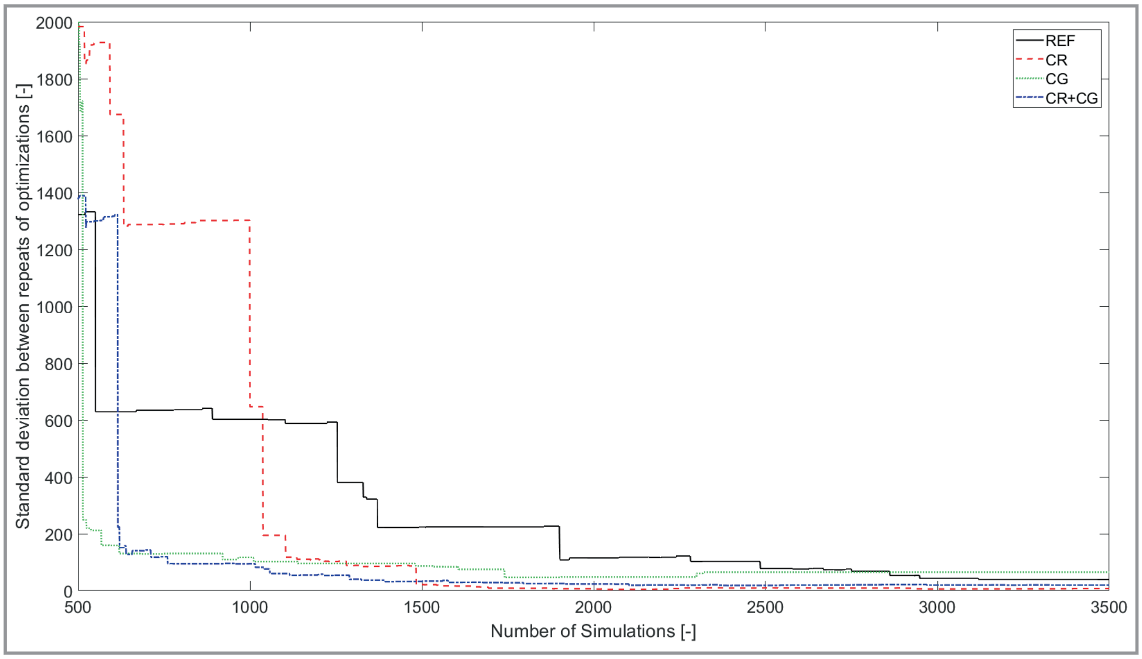

Figure 6. Plot showing the standard deviation between the repeats of the optimizations for the different variants REF (black), CR (red), CG (green) and both CR+CG (blue) based on ten optimization runs.

500 simulations, as before this point the variance is too high. Both CG and CR+CG reduce the variance very fast because the goal of the candidate generation is the satisfaction of the purity constraint and therefore it helps to get rid of the penalty term in the cost function. Neither REF nor CG alone reach the low variance of CR and CR+CG after 3000 simulations. At 3000 simulations, REF has a lower variance as CG probably because the runs of the CG variant get stuck in local optima.

A Wilcoxon signed rank test [19] has been performed for the variants at different numbers of simulations. It confirmed that there is a significant improvement of the CR+CG variant over REF starting at 700 simulations.

Tab. 2 summarizes the performance of the ten optimizations per variant and shows that CR and CR+CG converge more robustly than REF and CG, as the standard deviations are much lower. As indicated by the costs, CG is not capable to converge to the best-known solution even after 12 000 simulations. Tab. 2 shows the average percentages of non-converging, infeasible and successful and feasible simulations for the different variants. The main benefit is the reduced average cost of the CR and CR+CG variants even after 1500 simulations. With this, the turnaround time between a team of experts and the framework is significantly reduced if just 1500 instead of 3000 simulations are used as termination criterion for an optimization.

Although a significant reduction of non-converging simulations can be observed after 12 000 simulations for the CR and CR+CG variants, at the beginning the number of non-converging simulations is worse than REF and slowly reduces in the first 3000 simulations. CR consists of two parts, a classification of non-converging simulations and a rule that permits only promising process candidates for simulation. The latter may reject a process candidate that is an infeasible solution and hereby indirectly raises the ratio of non-converging simulations. CG may propose process alternatives that are in unexplored regions such that the classifier lacks the data for accurate predictions leading to further non-converging simulations. At first sight, the worse behavior of the CG variant, which is stuck in local optima even after 12 000 simulations, is surprising. However, without a rejection of non-converging simulations the interaction of an ES and the candidate generation may lead to parameters of the recombination operation of the ES that produce offspring process candidates that do not converge more often and are not filtered in the CG variant. Moreover, the CG will disturb the self-adaptation and by this the balance of exploration and exploitation of the ES.

6 Conclusion and Outlook

The main drawback of global flowsheet optimization by coupling an external optimizer to a flowsheet simulation tool are the long optimization times, which lead to long turnaround times, i.e., the experts have to wait long until they get the results of the optimization. We extended a framework for global flowsheet optimization based upon a commercial process simulator (Aspen Plus) by machine learning methods and thereby reduced the optimization times significantly, so that the framework is better suited to be applied in a workflow of iterative process design. Towards this goal, a candidate rejection and a candidate generation mechanism were added to the framework. Both

Table 2. Summary of results and optimization outcomes after 1500, 3000 and 12 000 simulations. These numbers are based upon the results of ten independent optimizations for each variant of the algorithm. Best, average and worst costs of all ten optimizations. Average percentages of non-converged, infeasible, successful, and feasible simulations.

Test variant / category	Simulations	Costs			Standard deviation	Average percentage [%] of simulations		
		best	average	worst		non-converged	infeasible	successful and feasible
REF	1500	3485	3722	4283	223.25	32	67	1
CR	1500	3501	3524	3556	24.03	35	57	8
CG	1500	3558	3678	3849	97.82	47	43	10
CR+CG	1500	3489	3530	3593	32.47	42	41	17
REF	3000	3474	3521	3604	42.69	33	65	2
CR	3000	3478	3485	3496	7.73	27	51	22
CG	3000	3529	3616	3722	61.05	48	39	13
CR+CG	3000	3467	3486	3523	19.59	34	32	33
REF	12 000	3469	3484	3512	15.56	25	51	24
CR	12 000	3458	3466	3474	6.38	13	50	37
CG	12 000	3529	3554	3598	19.61	46	36	18
CR+CG	12 000	3438	3463	3480	10.55	17	25	58

extensions employ surrogate models that are trained during the runtime of the optimization. The proposed algorithm performs significantly better than an evolution strategy alone and could reduce the turnaround time by over 50 %. Additionally, a workflow for iterative process design that incorporates this framework for global flowsheet simulation has been introduced.

Fig. 7 shows the inspiring vision of future process simulation by the German chemical process industry [2]. For the box "user and third-party models", the framework described here provides an extension with respect to process models, as it integrates a modular library of custom cost functions. It supports interoperability as it uses two sets of units and is capable to switch between them automatically and therefore supports optimization with flowsheets created with arbitrary units. The framework realizes an efficient interface to control the process simulator. This interface is based upon the IPO (input, process and output) principle. A ML toolbox is integrated that is capable to trigger a retraining for surrogate models. Both interfaces are implemented with gRPC, a high-performance remote procedure call framework, such that the training of data driven models and multiple parallel process simulations can be decoupled and distributed over several computing machines or in the cloud.

The interplay between CR, CG and the ES raises interesting questions about the balance of exploration and exploitation. When an individual is improved by the candidate generation mechanism then all the properties that control the progress of the ES [18] are lost and therefore its self-adaptation is disturbed. From a scientific perspective, a more thorough study of these effects is of high interest. The use of multiple populations, each with a different goal, and diversity control [20] are other interesting directions for further research.

The investigated case study has only ten degrees of freedom while in industrial applications typically 30 or more DOFs are present [12], so further improvements of the efficiency of the strategy are needed. A possible solution for the scale-up challenge is a divide and conquer approach that leads to dimensionality reduction, such that the framework trains a surrogate model for each unit operation individually. Similar modularization techniques were proposed by Caballero et al. [21] and Quirante et al. [22].

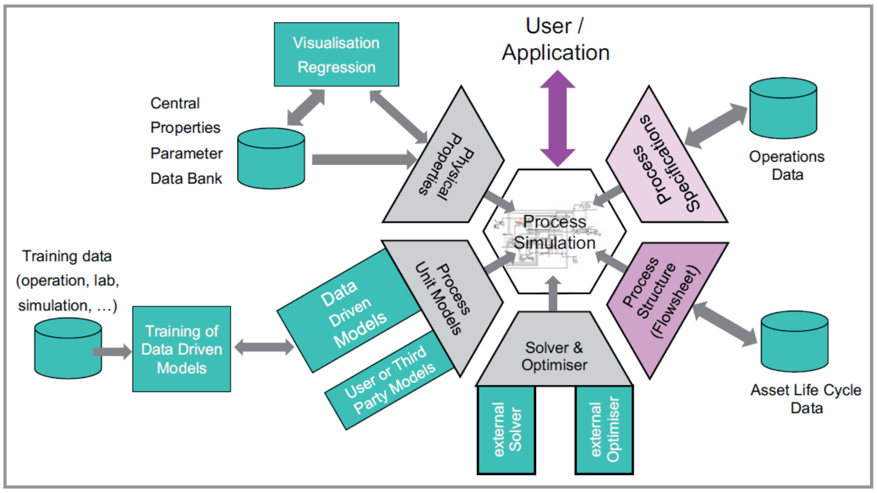

Figure 7. Structure of the future process simulation proposed in [2].

Gefördert durch die Deutsche Forschungsgemeinschaft (DFG) – TRR 63 "Integrierte chemische Prozesse in flüssigen Mehrphasensystemen" (Teilprojekt D1) – 56091768. Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – TRR 63 "Integrated Chemical Processes in Liquid Multiphase Systems" (subprojects D1) – 56091768. The authors want to thank Anne Lübbers, Alexander Böhmer and Felix Riedl for their contributions through their Master thesis work. We also thank Sabine Barkmann and Maximilian Cegla for the collaboration within the project SUPEROPT. Open access funding enabled and organized by Projekt DEAL.

References

[1] N. Asprion, M. Bortz, *Chem. Ing. Tech.* **2018**, *90 (11)*, 1727–1738. DOI: https://doi.org/10.1002/cite.201800051

[2] S. Bröcker, R. Benfer, M. Bortz, S. Engell, C. Knösche, A. Kröner, *Process Simulation – Fit for the Future?*, Position Paper, DECHEMA e.V., Frankfurt **2021**.

[3] L. T. Biegler, I. E. Grossmann, A. W. Westerberg, *Systematic Methods for Chemical Process Design*, Springer-Verlag, New York **1997**.

[4] L. Miguel, R. Nikolaos, *J. Global Optim.* **2013**, *56*, 1247–1293. DOI: https://doi.org/10.1007/s10898-012-9951-y

[5] M. Urselmann, S. Engell, *Comput. Chem. Eng.* **2015**, *72*, 87–108. DOI: https://doi.org/10.1016/j.compchemeng.2014.08.006

[6] M. Urselmann, et al., in *GECCO '16: Proceedings of the Genetic and Evolutionary Computation Conference 2016*, Association for Computing Machinery, New York **2016**. DOI: https://doi.org/10.1145/2908812.2908874

[7] T. Janus, C. Foussette, M. Urselmann, S. Tlatlik, A. Gottschalk, M. Emmerich, T. Bäck, S. Engell, *Chem. Ing. Tech.* **2017**, *89 (5)*, 655–664. DOI: https://doi.org/10.1002/cite.201600179

[8] M. Urselmann, T. Janus, C. Foussette, S. Tlatlik, A. Gottschalk, M. T. M. M. Emmerich, T. Bäck, S. Engell, *Comput.-Aided Chem.*

Eng. **2016**, *38*, 187–192. DOI: https://doi.org/10.1016/B978-0-444-63428-3.50036-9

[9] T. Janus, M. Cegla, S. Barkmann, S. Engell, *Comput.-Aided Chem. Eng.* **2019**, *46*, 469–474.

[10] R. T. Haftka, D. Villanueva, A. Chaudhuri, *Struct. Multidiscip. Optim.* **2016**, *54 (1)*, 3–13. DOI: https://doi.org/10.1007/s00158-016-1432-3

[11] A. Bhosekar, M. Ierapetritou, *Comput. Chem. Eng.* **2017**, *108*, 250–267. DOI: https://doi.org/10.1016/j.compchemeng.2017.09.017

[12] K. McBride, K. Sundmacher, *Chem. Ing. Tech.* **2019**, *91 (3)*, 228–239. DOI: https://doi.org/10.1002/cite.201800091

[13] T. Janus, A. Luubbers, S. Engell, in *2020 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, Piscataway, NJ **2020**. DOI: https://doi.org/10.1109/CEC48606.2020.9185781

[14] W. A. Herrmann, C. W. Kohlpaintner, *Angew. Chem.* **1993**, *105 (11)*, 1588–1609. DOI: https://doi.org/10.1002/ange.19931051102

[15] V. A. Merchan, G. Wozny, *Ind. Eng. Chem. Res.* **2016**, *55 (1)*, 293–310. DOI: https://doi.org/10.1021/acs.iecr.5b03328

[16] G. Kiedorf, D. M. Hoang, A. Müller, A. Jörke, J. Markert, H. Arellano-Garcia, A. Seidel-Morgenstern, C. Hamel, *Chem. Eng. Sci.* **2014**, *115*, 31–48. DOI: https://doi.org/10.1016/j.ces.2013.06.027

[17] R. Turton, R. C. Bailie, W. B. Whiting, J. A. Shaeiwitz, D. Bhattacharyya, *Analysis, Synthesis and Design of Chemical Processes*, Pearson Education, London **2008**.

[18] H.-G. Beyer, H.-P. Schwefel, *Nat. Comput.* **2002**, *1 (1)*, 3–52. DOI: https://doi.org/10.1023/A:1015059928466

[19] K. Krishnamoorthy, *Handbook of Statistical Distributions with Applications*, Chapman and Hall/CRC, London **2020**, 339–342. DOI: https://doi.org/10.1201/9781420011371-34

[20] M. Crepinsek, S. H. Liu, M. Mernik, *ACM Comput. Surv.* **2013**, *45 (3)*, 1–33. DOI: https://doi.org/10.1145/2480741.2480752

[21] J. A. Caballero, I. E. Grossmann, *AIChE J.* **2008**, *54 (10)*, 2633–2650. DOI: https://doi.org/10.1002/aic.11579

[22] N. Quirante, J. A. Caballero, *Comput. Chem. Eng.* **2016**, *92*, 143–162. DOI: https://doi.org/10.1016/j.compchemeng.2016.04.039