A NOVEL STOCHASTIC OPTIMIZATION SOFTWARE FOR THE OPTIMAL DESIGN OF CHEMICAL PROCESSES MODELED IN COMMERCIAL SIMULATION SOFTWARE

F. Duanmu, D. N. Chia and E. Sorensen[1]

Department of Chemical Engineering, University College London, Torrington Place, London WC1E 7JE, UK

Abstract

The design and optimization of a chemical process is often done using commercial software. Current commercial simulation software either only have Non-linear Programming (NLP) optimization functionalities, or the built-in Mixed Integer Non-linear Programming (MINLP) optimizer cannot efficiently handle complex designs, even though the optimization of a chemical process is typically a highly non-convex MINLP problem. Therefore, in this work, a novel stochastic optimization software – StOp – is presented. StOp has a simple user interface and can communicate with commercial simulation software in solving the optimization using stochastic methods coded within StOP (currently Genetic Algorithm, Particle Swarm Optimization, Simulated Annealing, and Fast and Elitist Nondominated Sorting Genetic Algorithm), with the process model expressed in the commercial software. StOp also has key functional features such as parallel computing, dynamic bounds, a timeout function, and saving good solutions, which are customized for the optimization of chemical processes. The software is illustrated by considering the optimization of a three distillation column superstructure.

Keywords

Stochastic optimization, Software, Chemical process

Introduction

Commercial simulation software such as Aspen Plus (Aspen Technology Inc., 2017) and gPROMS Process (Process Systems Enterprise, 2021) are widely used for designing chemical processes by using built-in libraries/packages, for instance reactors and distillation column models. The drag-and-drop functions in these tools are straightforward and easy to use when designing a process, as it does not require the user to code any equations, and the "canvas" (the area where the units are "dropped" onto) gives a direct visualization of the process that is being designed. In addition to simulation, optimization is another important and essential part of the design process. A proper optimization may provide an optimal design with less capital expenditure, less energy consumption, and less space/unit requirements, which leads to a more financially and environmentally sustainable process. For chemical processes, the optimization is usually a highly non-convex Mixed Integer Non-linear Programming (MINLP) problem. The objective function typically includes complex cost equations, which often introduces discontinuities into the model. For this reason, an optimization can easily drop into a local optimum. The built-in optimizer in Aspen Plus is a Non-linear Programming (NLP) optimizer based on Sequential Quadratic Programming (SQP), which is incapable of handling integer variables, and there is currently no built-in MINLP optimizer. Integer variables are, however, very common in a chemical process, *e.g.* the number of stages in a distillation column. Therefore, when simulating in Aspen Plus, an "optimization" study is usually performed instead through sensitivity analysis, which by its nature is inferior to rigorous optimization as it cannot solve the optimization task simultaneously, and the solution will therefore not be truly optimal. Although gPROMS has a built-in deterministic MINLP optimizer, called Outer Approximation / Equation Relaxation / Augmented Penalty (OAERAP), it does not handle complex processes well as it requires quite careful determination of the initial values and bounds of the optimized variables, as well as considerable experience of how to set up the optimization step by step, *e.g.* the optimization variables may need to be added into the optimization task one by one to increase the rate and success of convergence (Chia et al.,

[1] Corresponding author. Email: e.sorensen@ucl.ac.uk.

2021).

To obtain an optimal design, one obvious option is therefore to develop and apply an external optimization tool, for instance a stochastic method. The user can then optimize the chemical process, which can still be constructed within a simulation software, with the help of for example an Application Programming Interface (API) for Aspen Plus or a Foreign Process Interface (FPI) for gPROMS, by doing the optimization externally to the simulation software. However, coupling the external stochastic optimizer libraries/packages with the simulation software requires the user to have in-depth computational knowledge of the ways in which the simulation software and the external optimizer are constructed, how the tools communicate with each other, and how they handle the input and output data (*e.g.* gPROMS requires the input data to be constructed into a 1-D array, and distillation column stages need to be in the form of an array of Special Ordered Set of type 1 (SOS1)). Moreover, significant experience may also be required to apply some of the more advanced functions in the optimization, for example, to use parallel computing or to apply a timeout function on a thread to prevent an "infinitely" long simulation or optimization. Although optimization by combining an external optimizer with a process model developed in a commercial software may be highly successful, the limitations mentioned above may nevertheless discourage a user from attempting an external optimization.

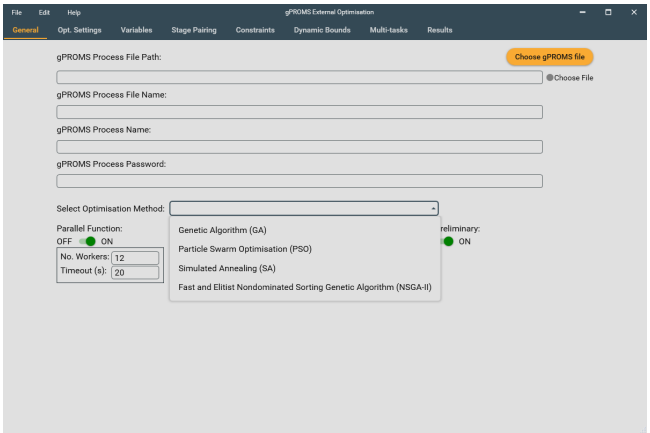

Figure 1: Screenshot of the "general" tab of the Graphical User Interface (GUI) of StOp for use with gPROMS Process.

In an effort to remove some of the barriers to using external optimization methods with commercial simulation software, in this work, an easy-to-use software called StOp (the screenshot of the "general" tab is shown in Figure 1), based on coupling a stochastic optimizer with commercial simulation software, is developed and discussed. To illustrate the performance of the software, a case study of superstructure optimization of distillation sequences is performed and compared with the built-in OAERAP MINLP optimizer in gPROMS.

Methodology

Our stochastic optimization software, StOp, allows process models developed in commercial simulation software to be optimized using external stochastic optimization methods. The tool is developed using C# within the Windows Presentation Foundation (WPF) framework. All the stochastic optimization methods are coded by the authors without the use of any library/packages. So far, StOp can only be operated on Windows, but future work may consider making StOp cross-platform so that it can also work on, for example, Linux. The current version is compatible only with gPROMS but it is the intention to make StOp also compatible with Aspen Plus. The reason to start with gPROMS is that the communication protocol used by gPROMS, called the foreign process interface (FPI), is fully customizable by the user, making FPI very easier to be tailored for different external software.

The external optimizer may not be the only software external to the simulation software. A foreign object (FO) allows the user to call an external software to perform other functions, such as extensive logical expressions or other calculations. For example, gPROMS can be connected to a FO called Multiflash (KBC Advanced Technologies, 2015) for physical properties calculations. The users can also define their own FOs using C, C++, or FORTRAN. In this work, C++ is used to construct the necessary FOs. It should be noted that FO is not compulsory in order to use StOp, but rather is a tool to help with the simulations.

Together with gO:Run (Process Systems Enterprise, 2022a), which is an execution-only engine developed for gPROMS, StOp can easily call and execute the simulation within gPROMS in a very efficient way, also if this simulation is linked to one or more FOs. In the following subsections, the main details of the software will be discussed.

Working with gPROMS

Figure 2 shows the components and communication route of StOp with gPROMS, and the programming languages used to develop them. First of all, the chemical process model should be constructed within a gPROMS product (*e.g.* gPROMS Process (Process Systems Enterprise, 2021) or gPROMS Modelbuilder (Process Systems Enterprise, 2022b). A good initial design is important as it impacts on the success, speed, and quality of the optimization convergence.

The foreign process (FP) shown in Figure 2 is used to communicate between the gPROMS simulations and the optimization tool StOp (*i.e.* data exchange). A Foreign Process Interface (FPI) (*i.e.* a dynamic link library (DLL) file) is developed and compiled using C++. The FPI required by StOp is compiled as part of the software, and therefore the user does not need to worry about developing an FPI of their own. It should be noted that the user is, however, required to have licenses for both the relevant gPROMS product and for gO:Run, and must place the FPI (developed for StOp) in the required path at "`%GPROMSHOME%\fpi`".

In addition to being an easy-to-use stochastic optimization software, the key functional features of StOp include embedded parallel computing, dynamic bounds, a timeout function, and the capability of saving good interim solutions. As other existing stochastic optimization libraries/packages are usually for generic use, these key features are not commonly embedded. One aim of developing StOp is to pro-

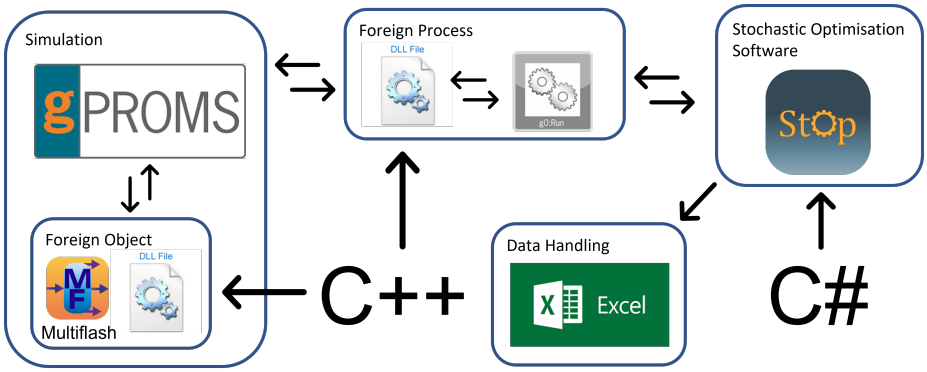

Figure 2: Communication flow between the developed stochastic optimization software (StOp) with the simulation software (here gPROMS). The dynamic link library (DLL) files are developed in C++ while StOp is developed in C#.

vide an easy-to-use stochastic optimization software tailored specifically for chemical processes, hence these features are therefore embedded into StOp. The rational behind their selection, and the importance of these key features for optimization of chemical processes, will be discussed in the next subsections.

Optimization Methods

The current version of StOp (V1.0.0) offers three single objective stochastic optimization methods: genetic algorithm (GA), particle swarm optimization (PSO), and simulated annealing (SA), as well as a multi-objective optimization method called the Fast and Elitist Nondominated Sorting Genetic Algorithm (NSGA-II).

The details of how GA and PSO has been applied are described in our previous work (Chia et al., 2021; Duanmu et al., 2022a). For GA, there are two available methods for parents selection including a well-known binary tournament selection (Back et al., 2000) and a modified Rank selection method described in (Chia et al., 2021) where the top ranked chromosomes are selected without introducing the probability of selection of each chromosomes. The modified rank selection has been found to be effective (Chia et al., 2021), especially for complex processes (*i.e.* for which simulations easily fail to converge) as it can prevent the lost of desired designs and reduces the number of infeasible simulations in each generation. The discrete crossover (Umbarkar and Sheth, 2015) and uniform mutation (Soni and Kumar, 2014) operators are available in StOp. For PSO, as outlined in Engelbrecht (2007), the inertia is dynamically chosen in each iteration using the random adjustments method, and the cognitive (resp. social) acceleration coefficient is linearly decreased (resp. increased). The random forth method (Gandomi and Kashani, 2018) is used for the boundary handling. In both GA and PSO, the constraint handling method proposed by Deb (2000) is utilized to avoid guessing an R value. Both GA and PSO algorithms applied in StOp have been found to be effective in the optimization of different chemical processes (Chia and Sorensen, 2022; Duanmu and Sorensen, 2022).

For SA, different cooling schemes are available including linear cooling (Engelbrecht, 2007), exponential cooling (Engelbrecht, 2007), fast annealing (Ingber, 1989), and Boltzmann annealing (Ingber, 1989). The random candidate generation method is currently applied (Johnson et al., 1989). For the constrain handling, an user-defined value is added to the fitness as fitness penalty for non-desired designs.

For multi-objective optimization, NSGA-II as proposed by Deb et al. (2002) is considered, and the binary tournament selection (Back et al., 2000), discrete crossover (Umbarkar and Sheth, 2015), and uniform mutation (Soni and Kumar, 2014) are applied as well. To validate the NSGA-II method, a constrained CONSTR problem (Deb et al., 2002) is used and the optimization results are shown in Figure 3. The clear Pareto front, and well spread search space, shows the excellent performance of the multi-objective optimization method applied within StOp. Although not shown in this work, the Pareto front obtained from StOp is the same as the Pareto front obtained from Deb et al. (2002).

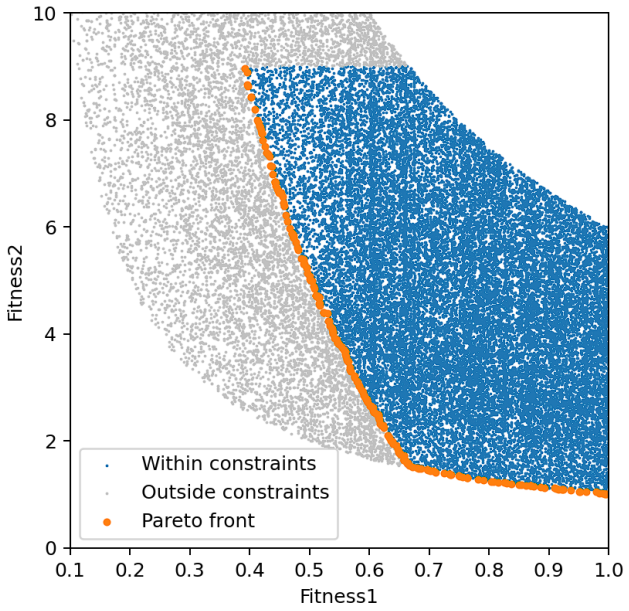

Figure 3: Results obtained from the multi-objective Fast and Elitist Nondominated Sorting Genetic Algorithm (NSGA-II) for the constrained problem, CONSTR, in Deb et al. (2002).

Key Features of StOp

One of the most important features of StOp is the **built-in parallel computing function** (see Figure 1). It is widely agreed that the main limitation of stochastic optimization is the long optimization time. However, this can be improved by applying parallel computing for the simulations in each generation/iteration. (It should be noted that the parallel computing function is not currently available as part of the standard SA implementation in StOp, but a modified SA with parallel computing will be developed in the future.) gPROMS Process operates on logical processors, and StOp requires a few threads for communication between the GUI and the external software as well as for the handling of parallel computing. In practice, with StOp, parallel computing can use a maximum of about 80% of the total number of logical processors in a desktop for optimization (*e.g.* for AMD Ryzen 9 3900x 12-Core 24-Processor CPU, 18 processors can be allocated for parallel computing). The tasks are dynamically allocated to available processors to make the best use of parallel computing as different tasks may require different simulation times.

Another key feature introduced in StOp is the **use of dynamic bounds**. Unlike classic mathematical optimization problems, the optimization variables of a chemical process are often related to each other. For example, for a distillation column, a feed location can never be larger than the total number of stages. However, unless the lower and upper bounds of the feed location and the total number of stages are mutually exclusive, during the course of the optimization (*e.g.* mutation in GA), there is a possibility that the feed location may be set as larger than the total number of stages during optimization. In StOp, a dynamic bound for the feed location can be applied to set the upper bound of the feed location to be equal to the value of the total number of stages in each task, thereby preventing many unrealistic and infeasible simulation steps. For such cases, the dynamic bound feature is found to be extremely useful.

The **timeout function** is essential to stop any "infinitely" long simulations (or any simulation that runs for a significantly longer time than other simulations). For example, in gPROMS, for a complex model, some poor input values (*e.g.* a small number of stages combined with a small reflux ratio for a difficult separation in a distillation column) may lead to a very long simulation time and may eventually end up with a failed simulation as the purity specification(s) cannot be met. Through experience, a user may be able to identify that after a certain amount of time, the simulation will almost certainly fail. In this case, waiting for the simulation to run for a very long time, knowing that the simulation will fail, is clearly not time efficient. Therefore, a timeout function is introduced in StOp so that a simulation can be stopped after a defined time set by the user. This feature is also applicable when parallel computing is utilized.

Another useful feature is to **save good solutions**, where all the feasible simulations from the previous iteration are saved when this function is activated in StOp. Then, in the current iteration, if the same design is found as in previous iterations, the fitness and constraint values can be assigned without performing the simulation, which saves computation time. This is particularly helpful for GA as a percentage of the best chromosomes (*i.e.* the elite parents) are selected and kept, and these chromosomes are repeated in the next iteration(s) until a better chromosome overwrites it. It should be noted that the time needed to "compare and search" the chromosomes in the current iteration with the "saved chromosomes" pool from the previous iteration increases significantly as the "saved chromosomes" pool increases with increased iterations. Therefore, in StOp, after trading off the time saved from repeated simulation and the time needed for "compare and search", the chromosomes yielding feasible simulation are saved only for one iteration, *i.e.* the "saved chromosomes" pool is "emptied" every iteration after "compare and search" is carried out.

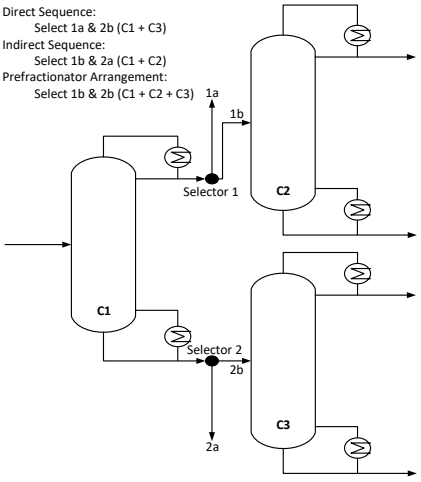

Figure 4: Schematic of the three columns superstructure consisting of the direct sequence, indirect sequence, and prefractionator arrangement.

Case Study

To illustrate the performance of StOp, a case study is performed using the genetic algorithm (GA) and particle swarm optimization (PSO) methods combined with gPROMS Process and Multiflash. The case study is the optimization of a three column superstructure (which includes a direct sequence, indirect sequence, and a prefractionator arrangement) for the separation of a ternary non-azeotropic mixture. This example will demonstrate the ability of StOp to perform superstructure optimization and to illustrate the use of dynamic bounds. Parallel computing (with 40 processors for optimization) is utilized to speed up the optimization using a dual Intel Xeon Gold 6226R CPU with 16 Cores 2.90GHz (in total 64 logical processors) and 192 GB Memory with a speed of 3200 MHz. It should be noted that several optimizations have been carried out but only a single set of results is reported here.

The flowsheet of the three column superstructure is shown in Figure 4 which includes the direct sequence, indirect sequence, and a prefractiontaor arrangement. The separation task considered is the separation of a benzene/toluene/o-xylene (BTX) mixture to obtain at least

Table 1: Comparison of the optimization results for a three column superstructure (direct sequence, indirect sequence, prefractionator arrangement) obtained from the deterministic optimizer built-in within gPROMS (OAERAP) and from StOp using genetic algorithm (GA) and particle swarm optimization (PSO).

Item	**OAERAP**			**GA**	**PSO**	**Unit**
Final design	Direct *	Indirect *	Prefrac. *	Direct	Direct	-
Column 1						
Total stages	30	29	29	35	37	-
Feed stage	15	17	15	17	17	-
Distillate	333.83	334.73	445.97	332.88	335.02	$kmol\,h^{-1}$
Reflux ratio	1.75	0.71	0.50	1.71	1.67	$mol\,mol^{-1}$
Column 2 (top)						
Total stages	-	31	28	-	-	-
Feed stage	-	16	15	-	-	-
Distillate	-	334.65	334.35	-	-	$kmol\,h^{-1}$
Reflux ratio	-	1.45	0.87	-	-	$mol\,mol^{-1}$
Column 3 (bottom)						
Total stages	34	-	31	32	34	-
Feed stage	19	-	16	18	18	-
Bottom	335.56	-	335.04	335.28	334.40	$kmol\,h^{-1}$
Reflux ratio	1.43	-	1.90	1.45	1.41	$mol\,mol^{-1}$
Fitness and Time						
TAC	8.07	9.09	9.08	8.06	8.01	$M\$\,y^{-1}$
CPU time (par.) †	-	-	-	743	826	s
CPU time (unpar.)	75	88	153	10613	9994	s

* Structure fixed

† 40 processors used for parallel computing

99 $mol\%$ of the three components in the product streams. The feed is an equimolar, saturated liquid of BTX at 1 atm supplied at a flow rate of 100 $kmol\,h^{-1}$. UNIQUAC is used for liquid activity coefficient calculations via Multiflash while the ideal gas law is assumed for vapor. An optimization based shortcut method (Duanmu et al., 2022b) is applied to ensure a good initial guess for the design.

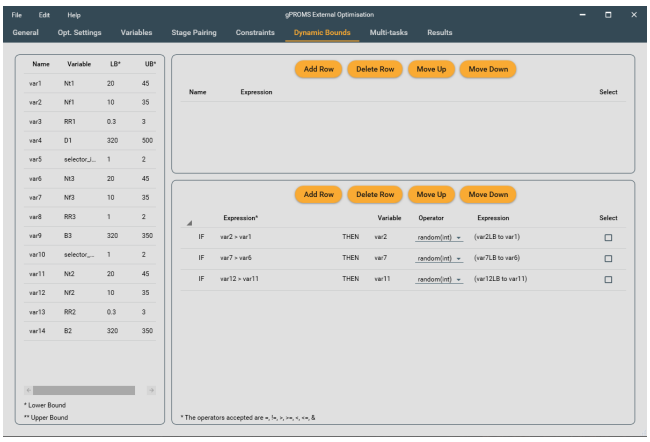

Figure 5: Screenshot of the "dynamic bounds" tab of the Graphical User Interface (GUI) of StOp.

It should be noted that, in addition to parallel computing, the dynamic bound function is also applied (shown in Figure 5) to ensure that the feed location is always equal or smaller than the corresponding total number of stages, and equal or bigger than its own lower bound (*e.g.* if var2 (feed location of column 1 – Nf1) is larger than var1 (total stage of column 1 – Nt1), then a random integer between the lower bound of var2 and var1 will be chosen to replace the initial var2). Moreover, the timeout function, set at 20 seconds (most simulations take about 2 seconds), is also applied to ensure the optimization is not "stuck" on infeasible designs. Although the improvements made by these two functions cannot be quantified as the improvement is strongly dependent on the specific chemical process design, from the authors' experience, these functions greatly reduced the number of infeasible designs and help the optimization to converge faster.

The three column superstructure in Figure 4 is optimized in StOp with both GA and PSO, and then the optimal results of the superstructure are compared to the individual optimizations (for direct sequence, indirect sequence, and prefractionator arrangement, respectively) carried out entirely in gPROMS using OAERAP. The objective function is to minimize the total annualized cost (TAC). The sizing and costs of the equipment can be found in Seider et al. (2016) and Sinnott and Towler (2020), respectively, while the utility costs are taken from Turton et al. (2012). It should be noted that superstructure optimization using OAERAP is not included in the table because OAERAP so often fails to give feasible results. In addition, for OAERAP to work on the superstructure, an initial value that is very close to the optimal design (*e.g.* results from GA) is needed, and this will make the comparison unfair as OAERAP will then inevitably result in a design close to the initial design in a very short time.

Table 1 shows the values for all optimization variables ex-

cept selectors. The individual optimization using OAERAP and the superstructure optimization using GA and PSO all show that the direct sequence is the best design with very close TAC and similar column designs. Taking a closer look at the CPU time (note that parallel computing cannot be applied for OAERAP), the total CPU time used by OAERAP is 316s. Although this CPU time is much shorter than those reported for GA or PSO, OAERAP requires many manual interventions as explained by Chia et al. (2021) and the time for these are not included. Also, with the increasing complexity of the superstructure (i.e. more selectors involved) it becomes very difficult to manually massage the optimization of every possible design, rendering optimization using OAERAP practically impossible.

The parallel computing for GA and PSO are 13 and 11 times faster, respectively, with 40 processors for 70 tasks in each iteration. The low parallel efficiency is due to the feature of the applied FPI in gPROMS and gO:Run. For a specific processor in parallel computing, if the previous simulation succeeds, gO:Run will wait for the next input. However, if an infeasible simulation occurs, gO:Run will be disconnected from the server and must be restarted by StOp for that specific processor. This process takes some time. For simple steady state optimizations, the time taken for a single simulation is short, thus the time penalty for restarting gO:Run caused by an infeasible simulation is relatively large, which leads to a reduced parallel efficiency if a few infeasible simulations happen for each iteration in the optimization. To seek a better parallel efficiency, a moderate number of processors is suggested for simple steady state optimizations. On the other hand, to achieve maximum optimization speed, a large number of processors is recommended.

Conclusion

In this work, a stochastic optimization software named StOp is presented, which can serve as an external optimization tool for chemical processes developed in commercial simulating software. The current version supports genetic algorithm, particle swarm optimization, simulated annealing, and a multi-objective optimization genetic algorithm (NSGA-II). The tool has several important built-in functions such as parallel computing, use of dynamic bounds, a timeout function, and can save good solutions which can be reused as part of the optimization. StOp is illustrated for superstructure optimization of distillation column sequences, in which the optimization is a Mixed Integer Non-linear Programming (MINLP) problem. The results show that StOp has excellent performance in finding a good optimal design within a short time using parallel computing. In future work, StOp will support more stochastic optimization methods, more options in each stochastic optimization (*e.g.* different mutation methods), combining the stochastic optimization with deterministic optimization to form a combined optimization method, and supporting other commercial simulation software (*e.g.* Aspen Plus). The tool is intended to be available as open access.

Acknowledgments

The authors wish to thank Dian Rui Chia for their invaluable help with the development of this software, especially for the development of the GUI and the parallel computing function.

References

Aspen Technology Inc. (2017). Aspen Plus V10.

Back, T., D. B. Fogel, and Z. Michalewicz (2000). *Evolutionary Computation 1: Basic Algorithms and Operators* (1 ed.). Taylor & Francis.

Chia, D. N., F. Duanmu, and E. Sorensen (2021). Optimal Design of Distillation Columns Using a Combined Optimisation Approach. In M. Turkay and R. Gani (Eds.), *31st European Symposium on Computer Aided Process Engineering*, pp. 153–158. Elsevier B.V.

Chia, D. N. and E. Sorensen (2022). Optimal Design of Hybrid Distillation/Pervaporation Processes. In Y. Yamashita and M. Kano (Eds.), *Proceedings of the 14th International Symposium on Process Systems Engineering - PSE2021+*, pp. 313–318. Elsevier B.V.

Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering 186*(2-4), 311–338.

Deb, K., A. Pratap, S. Agarwal, and T. Meyarivan (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation 6*(2), 182–197.

Duanmu, F., D. N. Chia, and E. Sorensen (2022a). A Combined Particle Swarm Optimization and Outer Approximation Optimization Strategy for the Optimal Design of Distillation Systems. In Y. Yamashita and M. Kano (Eds.), *Proceedings of the 14th International Symposium on Process Systems Engineering - PSE2021+*, pp. 1315–1320. Elsevier B.V.

Duanmu, F., D. N. Chia, and E. Sorensen (2022b). A shortcut design method for complex distillation structures. *Chemical Engineering Research and Design 180*, 346–368.

Duanmu, F. and E. Sorensen (2022). Optimal Design of Heat Integrated Reduced Vapor Transfer Dividing Wall COlumns. In Y. Yamashita and M. Kano (Eds.), *Proceedings of the 14th International Symposium on Process Systems Engineering - PSE2021+*, pp. 175–180. Elsevier B.V.

Engelbrecht, A. (2007). *Computational intelligence: An introduction* (2 ed.). John Wiley & Sons, Ltd.

Gandomi, A. H. and A. R. Kashani (2018). Probabilistic evolutionary bound constraint handling for particle swarm optimization. *Operational Research 18*(3), 801–823.

Ingber, L. (1989). Very fast simulated re-annealing. *Mathematical and Computer Modelling 12*(8), 967–973.

Johnson, D. S., R. Aragon, and L. A. Mcgeoch (1989). Optimization Annealing : an Experimental Part 1 , Graph Partitioning Evaluation. *Operations Research 37*(6), 865–892.

KBC Advanced Technologies (2015). Multiflash version 6.1.

Process Systems Enterprise (2021). gPROMS Process version 2.2.

Process Systems Enterprise (2022a). gO:Run.

Process Systems Enterprise (2022b). gPROMS ModelBuilder version 7.1.

Seider, W. D., D. R. Lewin, J. D. Seader, S. Widagdo, R. Gani, and K. M. Ng (2016). *Product and Process Design Principles: Synthesis, Analysis and Evaluation* (4 ed.). Wiley.

Sinnott, R. and G. Towler (2020). *Chemical Engineering Design* (6 ed.). Elsevier.

Soni, N. and T. Kumar (2014). Study of Various Mutation Operators in Genetic Algorithms. *International Journal of Computer Science and Information Technologies (IJCSIT) 5*(3), 4519–4521.

Turton, R., R. Bailie, W. Whiting, J. Shaeiwitz, and D. Bhattacharyya (2012). *Analysis, synthesis, and design of chemical processes* (4 ed.). Pearson.

Umbarkar, A. and P. Sheth (2015). CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW. *ICTACT Journal on Soft Computing 06*(01), 1083–1092.