Ab-initio process synthesis using evolutionary programming

Thibaut Neveux

Ab-initio process synthesis using evolutionary programming

Thibaut Neveux[a,*]

[a]*Laboratoire Réactions et Génie des Procédés (LRGP), UMR 7274 CNRS - Université de Lorraine, Nancy F-54001, France*

Abstract

Process synthesis methods enable the determination of unit operations and their interconnection into a process flowsheet, with associated design and operating parameters, and responding to given objectives. Modern methods are optimization-based, using for example Mixed Integer Non-Linear Programming (MINLP) formulation to optimize a process superstructure. Finding an adequate definition of the search space is a non-trivial problem in such approaches, especially when the number of possible combinations is high due to the process complexity, and is mostly driven by expertise (e.g. heuristics). Consequently, an inductive bias is intrinsically introduced due to restriction of a limited search space, such as the choice of a superstructure representing a limited set of process alternatives.

In this work, an evolutionary method is proposed to generate several process architectures based on a set of available unit operations (and associated models) as elementary building blocks. The procedure is here called *ab-initio* process synthesis since it does not require any pre-defined process structure. The developed method relies on the use of an Evolutionary Programming (EP), mimicking natural evolution at species-level, for the automatic construction of a process by using mutation operators to choose, assemble and connect elementary building blocks (i.e. unit operations). A Non-Linear Programming (NLP) is used for process evaluation, by simultaneously solving balances and optimizing process degrees of freedom.

The method is implemented in a newly developed tool called *PSEvo* (*Process Synthesis by Evolution*). An application to a typical reaction-separation problem is presented, using various problem definitions and evolution control parameters, which demonstrates the method capability to generate optimal processes. The possible uses and the challenges of *ab-initio* process synthesis are finally discussed.

Keywords: process systems engineering, process synthesis, evolutionary programming, optimization, reaction-separation

1. Introduction and motivation

Process synthesis is the procedure of process generation used to select a set of unit operations (transformations of mass and energy), interconnected in a network (process flowsheet) –together with associated design and operating parameters– and optimizing given objectives that could be economic, environmental and/or societal. Such a procedure can be performed by applying various systematic techniques, referred as Process Systems Engineering (PSE) or (global) Process Intensification (PI) methods [1, 2]. Possible techniques for process synthesis include heuristics (especially using experience and expertise) and decomposition-based [3, 4, 5] on one hand, and optimization-based [6, 7] methods on the other hand. For details on process synthesis methods, see dedicated books and review articles [3, 7, 8, 9, 10, 11, 12], only a short review is provided here to highlight the synthesis challenge addressed in this paper.

Decomposition approaches are able to propose efficient processes, but do not guarantee an optimal process since the interactions between different levels of decomposition (e.g. reaction, separation, heat

*Corresponding author

Email address: thibaut.neveux@univ-lorraine.fr (Thibaut Neveux)

Table 1: Brief overview of process synthesis approaches

Class of approaches	**Process synthesis**	**Methods**	**Ref.**
Decomposition	Heuristics, expertise and experience	Hierarchical	[3]
		Onion models	[4]
Optimization	Superstructure based on process units	MILP, MINLP	[11, 7]
		GDP	[14, 15]
	Superstructure based on phenomena or tasks	MINLP	[24, 25]
	Ab-initio, no superstructure is assumed	EP+NLP	this study

integration) are not considered [13]. Optimization-based methods allow to rigorously model and search for an optimal solution within a defined search space, optimizing the selected performance objectives. They rely mainly on mathematical formulations such as generalized disjunctive programming (GDP)[14, 15], or mixed integer non-linear programming (MINLP) [7, 11], which can be solved by a large panel of techniques [11, 15, 16, 17, 18, 19]. Noticeably, other approaches focusing mainly on Process Intensification explore beyond the concept of unit operations and optimize a process at the phenomenological or functional level rather than at the equipment level [20, 21, 22, 23, 24, 25] For a complete overview of optimization-based process synthesis, see for example the recent review of Chen and Grossman [26].

Whatever the mathematical approach in Process Systems Engineering, the first step is to postulate a set of alternatives represented by a process superstructure, including process unit operations of interest and foreseen interconnections [11], Chen and Grossman point out that "limitations here arise in our ability to define an appropriate search space"[26]. This initial step could be a difficult task when the number of possible combinations is high due to the process complexity. Counter-intuitive, yet relevant, solutions could also be missed during the process synthesis procedure.

The challenge addressed in this paper is the possibility of performing optimization-based process synthesis without postulating any superstructure (see the Table 1). The ambition is to avoid the subsequent bias due to the restriction of the space search to a set of possibilities. This approach is called here *ab-initio process synthesis* as a parallel from other fields such as optic [27] where similar design problems occur; the *ab-initio* term is also used in chemical engineering for thermodynamic properties calculations [28]. This approach could also be referred as "superstructure-free" in building design[29].

This paper presents a new generic method for process synthesis that does not require the preliminary definition of any set of alternatives, hence mimicking the whole mechanism of process creation and evaluation. The method is presented in section 2; it relies on the combined use of an Evolutionary Programming (EP) algorithm for automatic construction of process flowsheet, and of a Non-Linear Programming (NLP) algorithm for process evaluation. This hybrid optimization strategy works without any superstructure, given a set of available unit operations as elementary blocks assembled through mutation operators by the EP, and an objective function optimized by the NLP. The method is illustrated in section 3 for a reaction-separation-recycle system; the benzene chlorination process from Kokossis and Floudas [30] is chosen to confront the method presented in this paper to their MINLP optimization of a superstructure. Finally, the possible uses and the challenges of *ab-initio* process synthesis are discussed in section 4.

2. *Ab-initio* process synthesis method

2.1. Method overview

The goal of the presented method is to achieve process synthesis without assuming any superstructure, using a set of available unit operations as elementary blocks to build a process flowsheet. The method was developed by trying to transpose the remarkable work of Lipson and Pollack [31] on robotic design in chemical engineering domain. They *ab-initio* designed robotic lifeforms by evolution from basic building blocks (bars, joints, actuators, artificial neurons) and using a physical simulator for the evaluation of robot fitness (locomotive ability) [31]. In this work, the philosophy is kept whereas the specifics are adapted to the chemical engineering field; instead of bars connected by joints to describe a robot, a process flowsheet is described by unit operations connected by streams.

The process synthesis method is schematized in Figure 1, and is implemented in a newly developed tool called *PSEvo (Process Synthesis by Evolution)*. The method works on two levels combining the use of an Evolutionary Programming (EP) algorithm for automatic construction of process flowsheet, and of a Non-Linear Programming (NLP) algorithm for process evaluation. The EP is applied on the upper level as a construction tool by choosing, assembling and connecting unit operations into a process flowsheet (see section 2.2). Each process is then evaluated by the NLP on the lower level by solving heat and mass balances (see section 2.3) and optimizing an objective function with respect to the process degrees of freedom (e.g. design specifications, operating parameters). The way to define a new process synthesis problem (i.e. application specifics) is described in §2.4, and method implementation in §2.5.

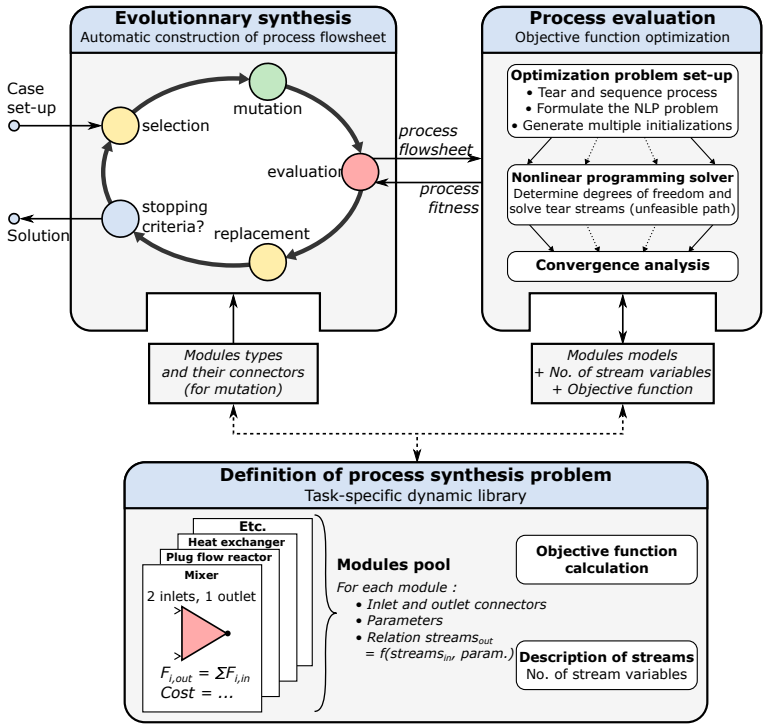

Figure 1: Overview of PSEvo architecture, see sections 2.2 and 2.3 for details respectively on Evolutionary synthesis and process evaluation procedures

2.2. Evolutionary synthesis

The evolution procedure is responsible for the construction of a process flowsheet given elementary blocks (unit operations), an Evolutionary Programming (EP) method is here employed to do so. Evolutionary Programming [32] is part of a wider category of nature-inspired methods called Evolution Algorithm (EA) [33, 34], which includes: genetic algorithms (GA), evolution strategy (ES), evolutionary programming (EP), and genetic programming (GP) methods. EP differs from other EA methods in that mutation is the only evolution driver. The crossover operators used in other EAs are not used in Evolution Programming, new candidate solutions are generated by exclusively applying mutation operators on existing candidate solutions. Metaphorically speaking, Evolutionary Programming mimics natural selection by focusing on the level of the species (phenotype) and not the level of single individuals (genotype) [35].

In chemical engineering, the use of Evolution Algorithms for process synthesis is common [36, 18, 16, 37], mostly by using Genetic Algorithms, and used to optimize a superstructure with the intrinsic bias already mentioned, limiting the search space to a set of alternatives. In this work, by using an Evolutionary Programming method to build a process from elementary blocks (unit operations), the search space is

virtually infinite and not restrained by a fixed-size representation. So far, processes have been generated with up to 100 building blocks on dummy cases, such as maximizing the efficiency without limitations on the number of unit operations. For real cases, limitations appear because the number of building blocks is penalized, usually in the form of a capital cost for additional blocks or increased network complexity. These sorts of *parsimony principle* arising from the definition of the Process Synthesis drivers is discussed on section 4.2.

The general evolution procedure is pictured in Figure 1 ('Evolutionary Synthesis' part), allowing to generate and propose flowsheet architecture by itself. The EP algorithm is summarized as follows:

1. *Initialization*
   a) Assign evolution parameters (default or user-supplied): mutation rate (α), number of individuals (N_{ind}), mutation operators probabilities (p_{add}, p_{remove}, p_{permut}), selection and replacement methods
   b) Determine the initial flowsheet (default or user-supplied) and evaluate its fitness (process evaluation)
2. *Main loop*
   a) Pick individuals in the population (selection method)
   b) Mutate them using mutation operators
   c) Evaluate them (process evaluation)
   d) Pick individuals in the population (replacement method)
   e) Replace these individuals by the mutants
   f) Increase the generation counter and display current best flowsheet and fitness progress
   g) If a stopping criterion is reached, exit main loop
3. *Results*
   Save all data and generate the report with evolution steps, phylogenetic tree and best individuals

The initial population is typically a simple 'empty' flowsheet (a feed connected to an outlet), but could also be any flowsheet defined by the user if required. The stopping criterion can be a maximum number of generations and/or a number of consecutive generations without improvement of the objective function value. Usual selection and replacement methods can be: random uniform, linear or exponential rankings (probability of choice is linearly or exponentially based on process rank), or fitness-proportionate (probability of choice is based on process fitness). The impact on the choice of selection and replacement methods on the convergence is discussed in the case study. Mutation operators are detailed in the next paragraph , and process evaluation (fitness calculation) in §2.3.

2.2.1. Mutation operators

Since the evolution is only driven by mutation, the key aspect is the definition of the mutation operators, so that every process can be obtained from scratch by the application of a sequence of mutation operators. Given a set of elementary building blocks (types of unit operations), mutation operators must be able to assemble them into a flowsheet, while remaining as simple as possible to minimize the inductive bias associated to their definition.

Three elementary mutation operators are used in this work: addition of a block (type of unit operation) between two existing blocks, removal of an existing block, permutation of two streams. See the Figure 2 for details and example.

Using these three simple building rules, any process flowsheet can be created by exploring the possible sets of unit operations (addition and removal operators) and their interconnections (permutation operator). During the mutation step of the evolutionary synthesis, one or several mutation operators are applied with respect to predefined probabilities (see default parameters in §2.5), at least one mutation needs to be applied.

2.2.2. Connectivity rules

During the mutation step, unconnected inlets and outlets can appear. This type of situation occurs if a unit with a different number of inlets and outlets is added or removed. It is particularly illustrated on Figure 2 with the addition of a mixer -a block with two inlets and one outlet- leading an unconnected inlet. Such structures are inconsistent since each connector (inlets and outlets of every block) needs to be linked to another block, which is here circumvented by connectivity rules.

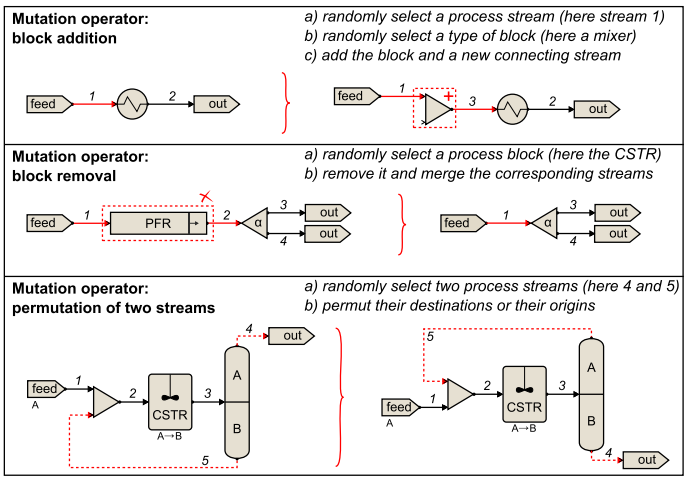

Figure 2: The three elementary mutation operators used during evolution for the construction of a process flowsheet

Firstly, it is checked if other non-linked connectors exist on another unit operation. If this is the case, a new stream is added to connect them. By security, the outlet of a unit operation can not be linked to one of its inlet, to avoid auto-loops. If several non-linked connectors are available, one of them is randomly selected. Secondly, non-linked connectors could remain. If a block has non-linked outlet connector, it is connected to an additional 'process out' block. Similarly, if a block has a non-linked inlet connector, it is connected to an additional 'Null feed stream' block, i.e. a virtual feed with null stream. A cleanup is also performed to remove useless structures such as a 'Null feed stream' block connected to a 'process out' block.

The first step enables to connect as many unit operations as possible, before adding additional unit operations in the second step to ensure complete process connectivity. These connectivity rules are applied at the end of the mutation step, after the application of mutation operators, and ensure that the process is valid from the connectivity point of view and is suitable for process evaluation.

2.2.3. Process representation (phenotype)

Whereas the evolutionary programming algorithm is generic and resembles to the one of Lipson and Pollack[31], the mode of candidate representation is domain specific and should be able to reproduce the candidate phenotype. Here, a process flowsheet is represented by the following elements: N_u the number of unit operations; N_s the number of streams connecting the units; IM the incidence matrix (size $N_s \times N_u$) detailing the connectivity between units and streams; T a string vector (size N_u) of unit types; and M a logical vector (size N_u), 'true' meaning that the unit is mandatory.

The number and types (e.g. Feed, Reactor) of unit operations are defined by N_u and T, their interconnections into a network by N_s and IM. The M logical vector indicates if the unit operation is mandatory in the process flowsheet and cannot be removed by the evolutionary synthesis procedure, this vector is particularly useful if the user wishes to impose certain blocks in the process, such as at least a reactant feed and a process outlet. The unit can yet be placed in a different order in the process, particularly by the application of the permutation operator.

During evolution, mutation operators act by simply modifying the process phenotype. For example, a block addition or removal affects the sizes and content of vectors and incidence matrix; stream permutation switches elements in the incidence matrix. The connectivity algorithm also affects the units and/or the unit operations. An example of process representation (phenotype) before and after the application of a mutation operator and the connectivity algorithm is provided in supporting information.

Within this framework, process feeds and outlets must be considered as unit operations, their impact on the process performance (such as reactant consumption, product sales, energy) is taken into account through calculated parameters of each unit operation.

2.3. Process evaluation

The role of the process evaluation is to get the flowsheet representation (phenotype) from the evolutionary synthesis procedure, and compute the fitness function (see Figure 1). To do so, the process balances (heat and mass, including possible recycles) need to be solved in order to calculate and optimize the objective function (process fitness). The corresponding mathematical problem is automatically constructed from the process phenotype, and is formulated as a classic Non-Linear Programming (NLP) optimization problem:

$$\begin{cases} \max & F_{obj}(\mathbf{z}) & & & \\ \text{s.t.} & \mathbf{g}(\mathbf{z}) & \leq & \mathbf{0} & \\ & \mathbf{h}(\mathbf{z}) & = & \mathbf{0} & \\ & \mathbf{t}(\mathbf{z}) & = & \mathbf{s} - \bar{\mathbf{s}}(\mathbf{z}) & = \mathbf{0} \\ & \mathbf{z} & \in & [\mathbf{z}_l, \mathbf{z}_u] & \end{cases} \tag{1}$$

with

- $\mathbf{z}$ the set of optimization variables $\mathbf{z}^T = [\mathbf{x}^T, \mathbf{s}^T]$
- $\mathbf{x}$ the set of decision variables (process degrees of freedom, for example the design variables of unit operations such as the volume of a reactor)
- $\mathbf{s}$ the set of (guessed) tear variables. $\mathbf{g}$ and $\mathbf{h}$ constraints are (optional) process design inequality and equality constraints, such as limitations on temperature or pressure, minimal production, desired purity and so on
- $\mathbf{t}(\mathbf{z})$ the set of tear equations ensuring that the recycle equations are satisfied, i.e. the difference between $\mathbf{s}$ and the calculated tear stream variables $\bar{\mathbf{s}}(\mathbf{z})$ from the current guess $\mathbf{s}$ and decision variables $\mathbf{x}$. $\bar{\mathbf{s}}$ is obtained by sequentially calling the unit operations models.

In case of non-convergence of heat and mass balances, i.e. the $\mathbf{t}(\mathbf{z})$ tear equations are not equal to zero, the process is considered non-viable and an arbitrary large negative value (or positive in case of minimization) is affected to F_{obj}. The tearing algorithm of Roach [38] is used here to select a set of tear streams, and hence to determine the calling sequence.

An *unfeasible path* approach [39] is used here since the flowsheet convergence is dealt during the optimization. Such an optimization problem can be solved by a NLP code, for example by using a Sequential Quadratic Programming (such as SLSQP [40] or NLPQLP [41]) or an interior-point method such as IPOPT [42]. Since the NLP codes are gradient-based method, a typical *multistart* approach is used to limit the risk of finding a local minimum. A set of different initialization of $\boldsymbol{z}$ is randomly generated in the domain, and the NLP is launched for each of them. The parametric domain is defined by the user, which provides lower and upper values for each optimization parameter (and optionally an initial guess, which is used in addition to the random sampling). The number of sampling points is either fixed by the user, or taken arbitrarily equal to the number of unit operations in the currently evaluated process. In the end, the best solution respecting all constraints is kept So far, it has been observed that the same optimum is found with the various initialization when convergence is reached; this option is still kept for future investigation.

2.4. Definition of process synthesis problem

For each process synthesis problem, i.e. for a new application, the user needs to define the following task-specific elements:

- A pool of unit operations, defining for each one:
  - the name of unit operation type (e.g. heat exchanger, reactor, reactant feed),
  - the number of inlet and outlet streams (connectors),
  - the inlet parameters (i.e. unit operation degrees of freedom such as design specifications),
  - the calculated parameters (e.g. capital cost, energy),
  - the relation giving the outlet streams and calculated parameters as a function of inlet parameters and inlet streams (i.e. the unit operation model),
  - the potential constraints to be respected (e.g. maximum temperature, product purity in the outlet).

- A description of stream: number of variables used to represent a stream (e.g. temperature, pressure, flowrate, composition)
- An objective function calculation (e.g. total energy consumption, process annualized cost), computed from the calculated parameters of the different unit operations

The evolutionary synthesis procedure only uses the name of unit operation type, to distinguish the different types of module, and the number of inlet and outlet streams, to connect them into a flowsheet. The process evaluation procedure uses all the elements to define and solve the optimization problem.

In the end, the approach adopted here by field transfer from robotic to chemical engineering, has a similar philosophy than the one developed by Voll et al. [43] on energy supplied systems and extended by Wang et al. [44] on thermal power plants. In their work, they used six "knowledge-integrated mutation operators" that are application-specific, and the process needs to be hierarchically decomposed into meta, function and technology levels, meaning that mutation operators and process representation needs to be conceptualized for each new synthesis problem. The approach presented here is believed to be more generic since the process description and mutation operators do not depend on the studied system. For a new synthesis problem, the user only needs to define the elementary blocks (pool of unit operations), stream description, and objective function.

2.5. Method implementation: PSEvo

The presented method is implemented in a newly developed tool called *PSEvo (Process Synthesis by Evolution)*, coded in Fortran 2008 in an object-oriented way. External libraries are used for specific tasks: the SLSQP optimizer[45] as NLP solver, Graphviz[46] and Gnuplot[47] for on-the-fly plotting of process flowsheets and objective function during evolution. Detailed information is generated in a synthetic HTML report to examine the evolution results.

Each new process synthesis problem needs to be compiled into a dynamic library (see Figure 1) using a given template, so that the user does only have to focus on definition of unit operations and objective function. An example of the required data is provided in the case study and in the appendix.

By default, the evolution control parameters are the following: N_{ind}=50 individuals (i.e. population size); a mutation rate of 50% (α); mutation operators probabilities of 5% for module addition and removal (p_{add} and p_{remove}), 20% for streams permutation (p_{permut}); random selection (before mutation); replacement based on rank (after mutation); 100 successive iterations without objective function improvement as stopping criterion. These parameters are discussed in section 3.4.

3. Case study with a reaction-separation-recycle system

The search of an optimal design for reaction-separation-recycle systems can be seen as global (process scale) intensification strategy based on flowsheet architecture improvement [2], and has been dealt in many papers [e.g. 30, 48, 49, 50, 23]. This makes reaction-separation-recycle systems a relevant example to validate the developed method and its implementation.

The benzene chlorination process from Kokossis and Floudas [30] is chosen as a case study. The idea is to confront their optimization of a given superstructure to the *ab-initio* process synthesis presented in this work.

The studied reaction is the benzene (A) chlorination to produce monochlorobenzene (B), with dichlorobenzene (C) as a waste product. The mechanisms are simplified into two first-order and irreversible reactions (A $\rightarrow$ B $\rightarrow$ C)[30]. Reactions occur either in Continuous Stirred-Tank Reactors (CSTR) or Plug-Flow Reactors (PFR). The objective being to produce pure B, separators are required. The volatility ranking is A>B>C, ideal distillation columns (100% purity) are used with the following possible separations: A/BC, AB/C, A/B, B/C.

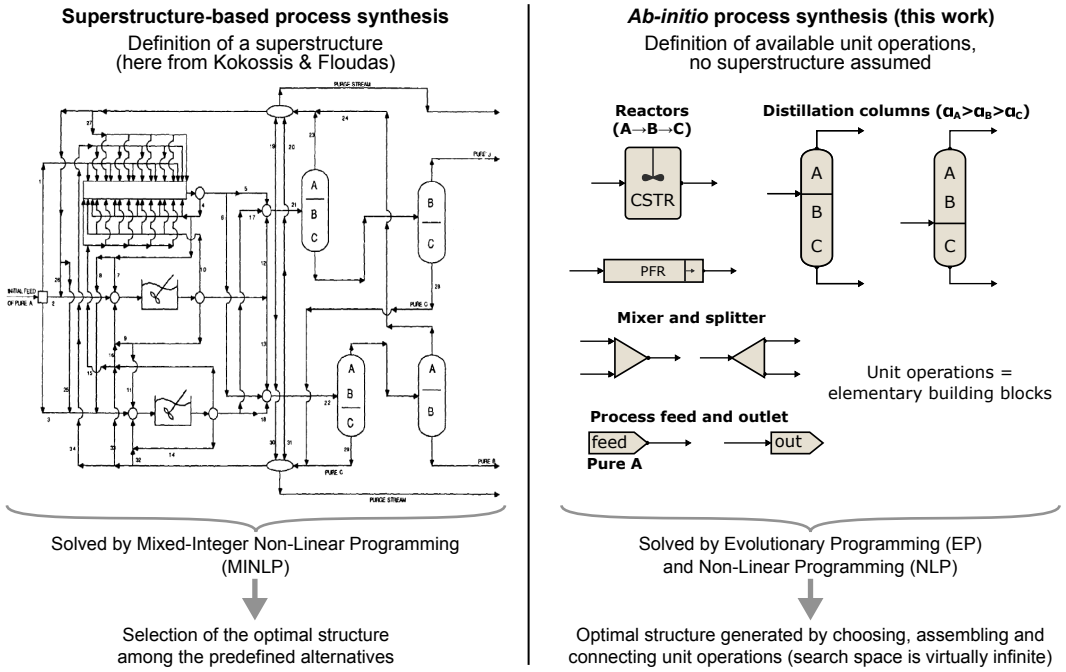

Figure 3: Differences in problem definition for the benzene chlorination case study. Left) one of the superstructure of Kokossis and Floudas [30] (reproduced with permission of Elsevier), a set of alternatives is defined and a MINLP method selects the optimal flow path and associated degrees of freedom. Right) this work: definition of the *ab-initio* process synthesis. No superstructure is assumed, the EP constructs the flowsheet from the available unit operations and the NLP determine the optimal degrees of freedom, i.e. the best objective function obtainable for a given flowsheet.

3.1. Problem definition

Kokossis and Floudas [30] defined several superstructures to cover as many process alternatives as possible, combining the use of reactors (CSTR and PFR), ideal distillation columns (A/BC, AB/C, A/B, B/C), mixers and splitters; cf. one of their superstructure on Figure 3 (left).

In this work, the problem definition is quite different. Instead of defining a set of alternatives represented by a process superstructure, only the available unit operations are defined. The evolutionary Synthesis procedure picks into this pool of unit operations to build any process flowsheet; cf. the Figure 3 (right). An 'empty' flowsheet, i.e. a 'Feed' unit (flowrate of pure A) connected to a 'Process Out' unit, is used as an initial population. The objective here is to maximize the annualized profit, which includes capital cost of unit operations, reactant costs, energy costs and product sales. Two cases are investigated to assess the sensitivity of the optimal process to the objective function definition:

Case A. Maximize profit for a production of 50 kmol/hr of B, same capital cost for PFR and CSTR.

Case B. Maximize profit for a production of 50 kmol/hr of B, higher capital cost for PFR than for CSTR

Case A is slightly similar to case 2 in [30] by using the same capital cost for PFR and CSTR, except that the objective function is to maximize profit instead of minimizing the annual cost. Case B is identical to case 4(a) in [30]. Detailed equations for unit operations and objective functions are reported in Appendix. The default evolution parameters of PSEvo are used (see §2.5).

3.2. Process synthesis results

The optimal processes found by PSEvo are displayed on Figure 4 for cases A and B, with associated degrees of freedom (i.e. reactor volumes and reactant feed flowrate).

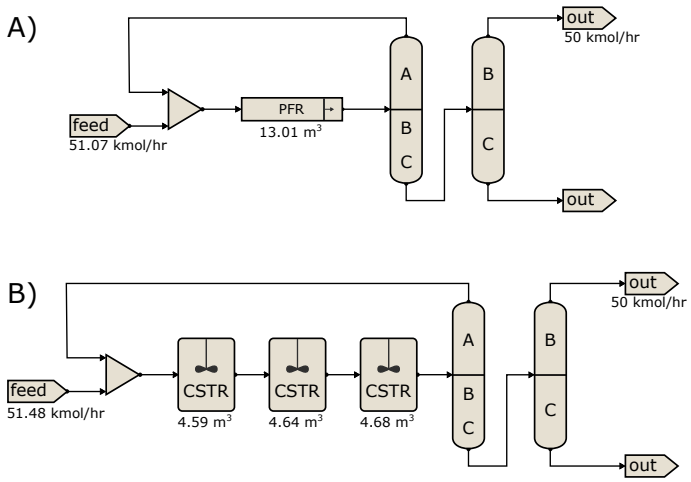

Figure 4: Process synthesis results for cases A (same cost for CSTR and PFR) and B (higher capital cost for PFR)

In both cases, the optimized configuration is based on a reaction part (CSTR in series or PFR), a first distillation column A/BC, recycling of unreacted A using a mixer before the reaction part, and a second distillation column B/C to separate the product B from the waste product C. The only difference is that one PFR of 13.01 m^3 is used for case A, whereas three CSTR in series for a total volume of 13.91 m^3 are used for case B (N.B. the three volumes are slightly different due to the evolution of molar volume with composition. If the molar volume would have been constant, the three volumes would be equal to maximize the reaction conversion). Both feed flowrates are also similar: 51.07 kmol/hr for case A and 51.48 kmol/hr for case B.

Not surprisingly, the use of a PFR instead of CSTR in series is chosen if they have the same capital cost (case A), due to the higher conversion achieved with PFR for the same reactive volume. For case B, 3 CSTR in series are used to achieve a similar conversion but at a lower cost. During evolution, processes with up to 5 CSTR in series were produced but not retained by the algorithm since they exhibit a lower profit. Regarding the separation part, the use of A/BC separation before B/C separation is also logical. Separating and recycling A before purification of B leads to a lower flowrate entering the second distillation column, hence a lower cost. Noticeably, both these structures were also obtained by Kokosssis and Floudas based on the superstructure optimization.

3.3. Evolution path and convergence

To analyse how processes evolve to produce an optimal solution, the objective function (annual profit) is plotted with generation on Figure 5 for Case B. The dots represents the objective function of the entire population, and the red line the evolution of the best achieved objective function with generations. Objective function rises by steps until convergence, process structures corresponding to notable evolution steps are also displayed on Figure 5. Processes evolved from the initial 'void' structure (a) to the final process (f). Successively, the algorithm added a PFR and a first distillation column (b), then replaced the PFR by a CSTR and added a second distillation column (c), positive profit is then obtained by using 3 CSTR in series (d). On step (e), a mixer is added to recycle unreacted A before the third reactor, leading to a substantial gain in profit. Finally, the A recycle is positioned before the first reactor (f, optimal process). It should be noted that processes (a) to (f) do not necessarily correspond to a mutation from one to another. "Good" processes could emerge by modification of an already interesting process (incremental improvement) or of a non-profitable one.

Due to the stochastic nature of the procedure, several optimizations should be systematically launched to check that the optimal solution is independent of the optimization run. The Figure 6 presents the optimization progress with four different random seeds, using the same problem definition. For the four runs,

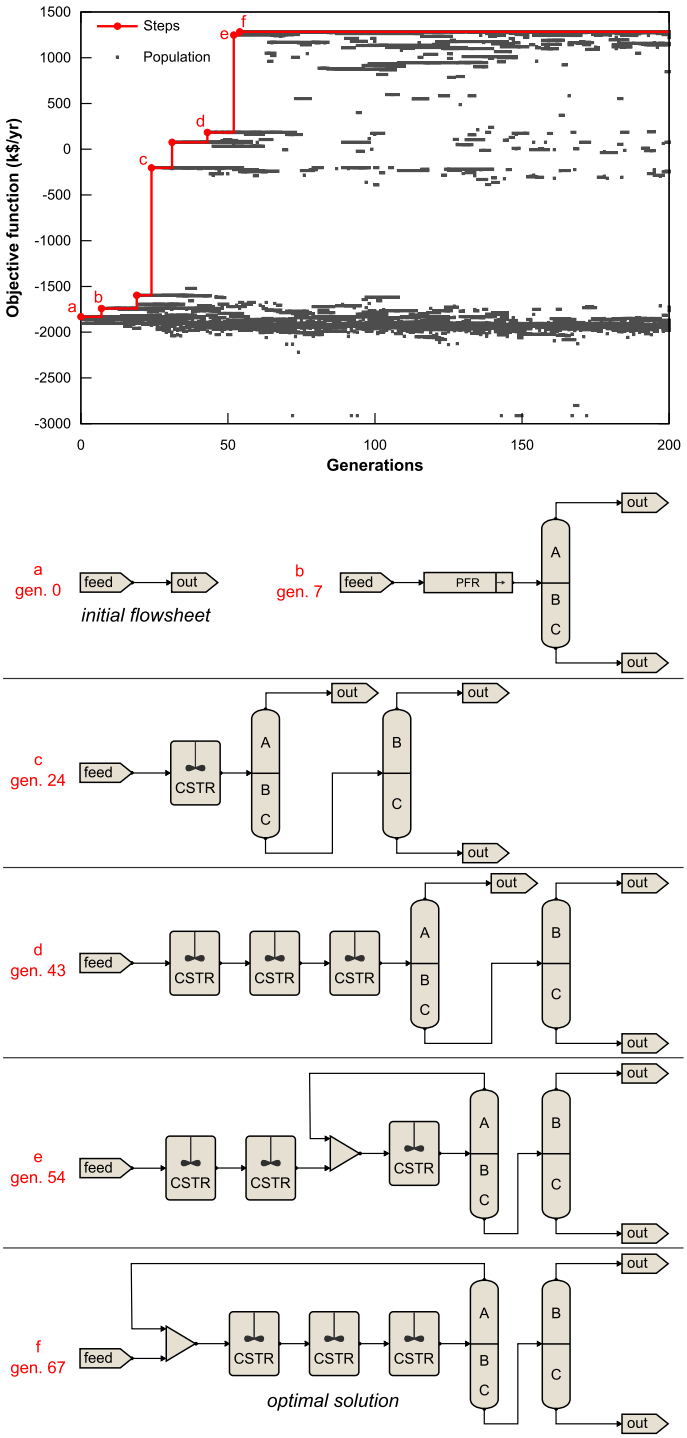

Figure 5: Example of optimization progress for the benzene chlorination case, and process flowsheets of notable evolution steps (case B)

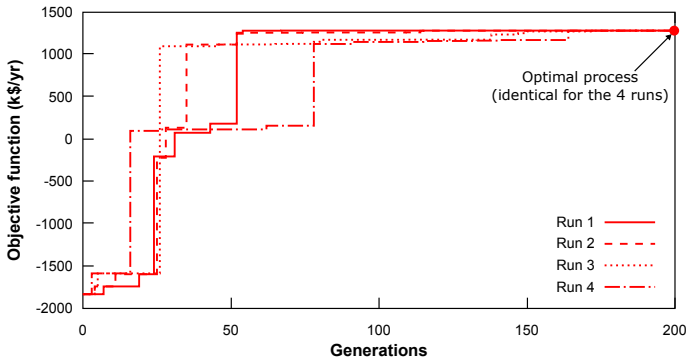

Figure 6: Optimization progress of four runs with different random seeds (case B)

the same optimal process is found, but with distinctive evolution paths. Different applications of mutation operators can indeed lead to the same process structure, this non-exclusivity in flowsheet construction is illustrated on the Figure 6.

3.4. Sensitivity to evolution control parameters

The evolutionary programming algorithm requires the specification of various evolution parameters, whose sensitivity is assessed here.

Selection and replacement methods

Since evolution is a random process driven by evolutionary pressure, at least one random method must be kept, either for selection or for replacement. Three cases are considered:

1. Selection: random. Replacement: linear ranking
2. Selection; linear ranking. Replacement: random
3. Selection: random. Replacement: random

The third case (random-random) is basically a random search within the search space, both other cases apply a selective pressure (linear ranking) during selection or replacement. An example of optimization progress for three cases is presented in Figure 7, with a high number of generations (up to 1000). For each case, 10 runs are performed with different random seeds, the obtained median and interquartile range is plotted with generations. Both methods applying a selective pressure converged in a limited number of generations (around 50-100 generations for case 1, and 150-200 for case 2), whereas the optimal solution is not found in 1000 generations for pure random search (case 3), illustrating the need for a selective pressure for evolutionary algorithms.

In addition to convergence speed, the population diversity during evolution can be analysed. It is expected that a higher diversity would lead to a higher chance of finding the optimal solution at long-term. To compare diversities, the Shannon entropy [51] is used in this work, which is an extension of the entropy concept in information theory, and is particularly used in biology to quantify biodiversity. For a given population, Shannon entropy H is defined as:

$$H = -\sum_i p_i \cdot \log_2(p_i) \tag{2}$$

where i are the different phenotypes observed in the population, and p_i is the probability of observing the phenotype i among the population (i.e. the fraction of population corresponding to the same phenotype). Two individuals belong to the same phenotype if their flowsheet and objective functions are identical. The comparison of flowsheets can be performed by checking if the directed graphs associated to the process structure are isomorphic (i.e. they show exactly the same flowsheet / process structure).

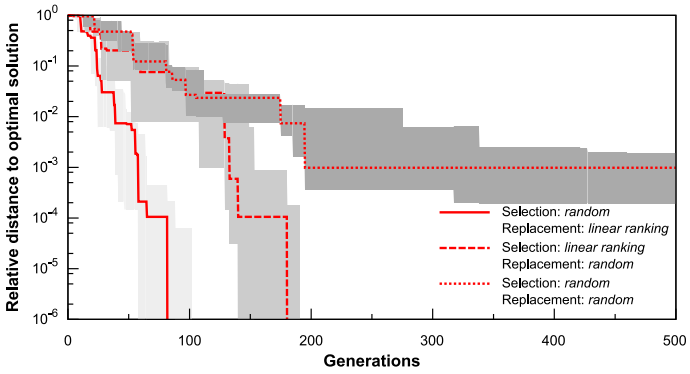

Figure 7: Optimization progress for several selection-replacement methods (median in lines, interquartile ranges in gray). See the corresponding phylogenetic trees and Shannon entropy on Figure 8. The relative distance to the optimal solution is equal to 1 at initialization (annual profit of the initial population, -1831 k$/yr) and to 0 when convergence is reached (annual profit of the optimal solution, 1283 k$/yr).

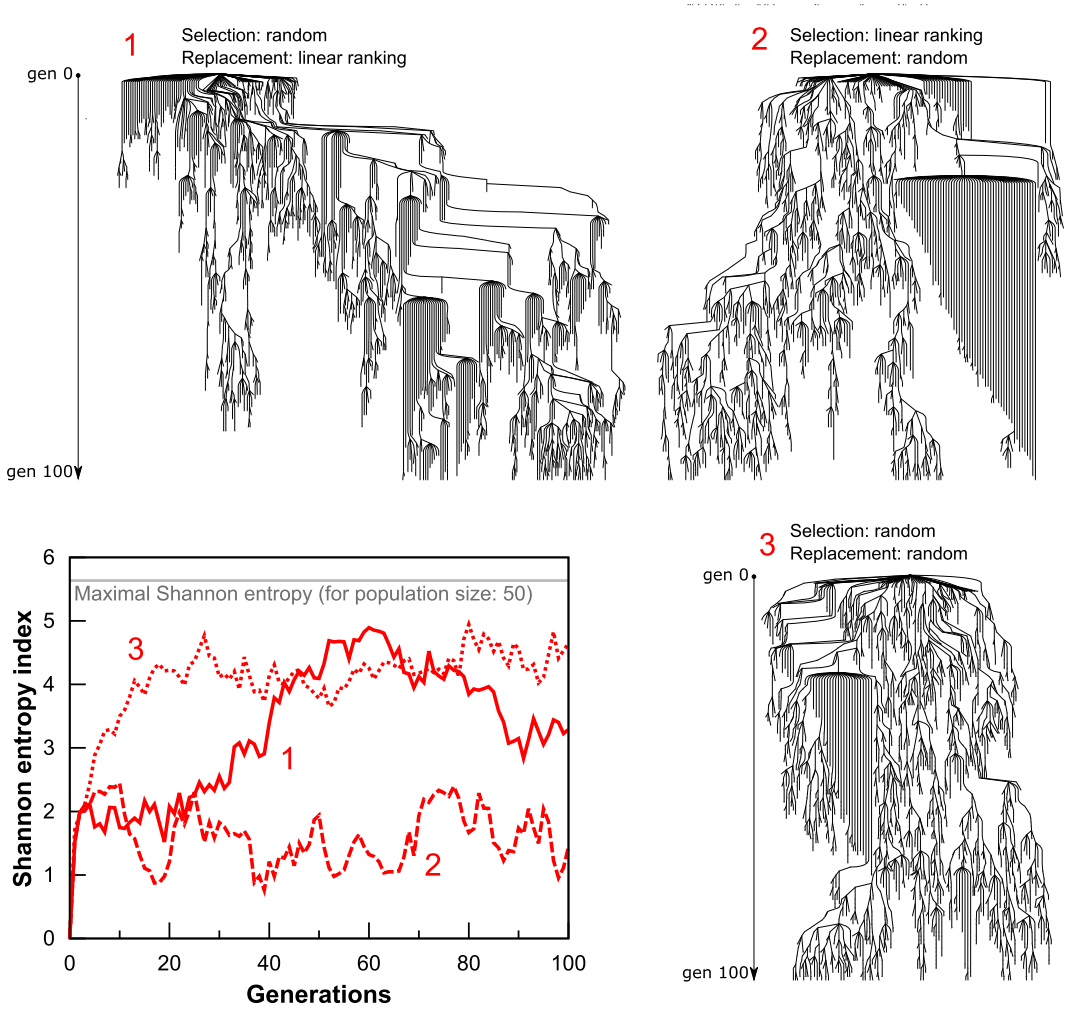

Figure 8: Example of phylogenetic trees and Shannon entropy for several selection-replacement methods during the first 100 generations. The vertical axis of phylogenetic trees represents generations, starting with the initial flowsheet down to the 100^{th} generation, nodes represent individuals and edges ancestral relationships. The corresponding convergences are plotted on Figure 7

The higher the entropy, the higher the biodiversity among the population is. A diverse population has a better long-term viability, meaning a higher chance of finding the optimal solution. A null entropy means that the entire population is uniform with only one phenotype, it only appears during initialization. The maximal entropy is achieved if all individuals are unique in the population. If the number of individuals per generation is 50, the maximal entropy of the system is $log_2(50) \approx 5.64$ ($p_i = 1/50$, $\forall i \in [1, 50]$).

Complementary to Shannon entropy, the phylogenetic trees provide visual information on diversity, representing the ancestral relationships with generations. An example of the evolution of Shannon entropy with generations is presented in Figure 8 for the three cases (same random seed), as well as the phylogenetic trees, for the first 100 generations.

No specific pattern appears for the phylogenetic tree of case 3, since it corresponds to a random exploration. For other cases, branches exist in the trees due to selective pressure. A massive extension even appears during the first generations for case 1 (random selection, linear ranking for replacement). Quantitatively, the Shannon entropy for the random exploration (case 3) quickly increases with generations, and stabilizes close to the maximal entropy of 5.64. For case 2, the Shannon entropy also stabilizes after a few generations, but at a lower Shannon entropy value (around 1-2). Yet, the optimal solution is found in less than 200 generations (cf. Figure 7). Interestingly, the case 1 leads to high Shannon entropy, also close to the maximal entropy, reaches a maximum around 60 generations and then decreases. The loss of diversity corresponds to the branch extinction appearing in the phylogenetic tree, and to the generation where an optimal solution is found. After the maximum in entropy, the population evolves with a diversity decrease as more individuals isomorphic to the optimal solution are generated.

From these results, random selection and linear ranking for replacement appear to be the most efficient combination since it leads to a faster convergence fur 10 runs (cf. Figure 7), while leading to a high population diversity (cf. Figure 8). Hence, a higher chance of finding the optimal solution is expected at long-term.

Population size and mutation rate

The sensitivity to mutation rate (α) and population size (N_{ind}, number of individuals per generation) is assessed. More than 200 optimizations were performed to cover the parameters ranges, the number of process evaluation required to find the optimal solution is reported on Figure 9.

The mutation rate represents the fraction of the population that is replaced at each generation. As seen on Figure 9 (top), too low or too high mutation rates lead to a large number of process evaluations. The population size manifests a similar behaviour at high values of number of individuals per generation. A mutation rate between 0.3 and 0.6, and a population size between 30 and 50 seems to be a good compromise of evolution within a generation. The mutation probabilities (p_{add}, p_{remove}, p_{permut}) were also assessed but no clear trend is observed while variating the mutation probabilities between 0.05 and 0.2.

Choice of evolution parameters

This sensitivity analysis highlights the importance of parameters choices, since the convergence speed can triple depending on the tuning. Other selection/replacement methods could be investigated (e.g. fitness-proportionate, exponential ranking, tournament, age-based), in combination with all other evolution control parameters (population size, mutation rate, probabilities) to cover the entire parameters space, which represents a high number of possibilities. For further tuning, the Shannon entropy could be used to obtain information on the parameters relevance, for example by using the REVAC method [52].

3.5. Case study conclusion

The objective of the case study was to validate the proposed *ab-initio* process synthesis method on a well-known problem. The same structures have been obtained, without assuming neither the number and types of unit operations nor their interconnections (e.g. reactors before separators, presence of recycle). The proof of concept of the newly proposed process synthesis strategy is therefore considered as validated. The adequacy in terms of solution obtained from two different approaches is remarkable and should be stressed. No previous study addressed, to our knowledge, such a critical comparison. Yet, no new solutions were

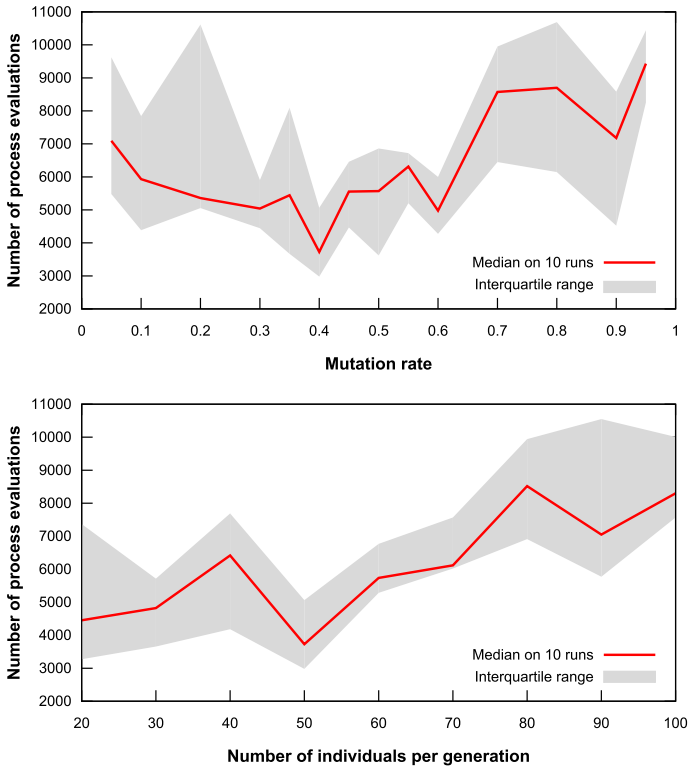

Figure 9: Sensitivity to evolution control parameter - median and interquartile range of the number of function evaluations required to find the optimal solution as function of the mutation rate (top) and the number of individuals per generation (bottom) using 10 runs with different random seeds

found compared to the reference paper used for the case study [30], probably because their superstructure already encompassed a large number of possibilities, using 32 binary variables (~ 4 billions possibilities).

In this work, the method generally requires fewer than 5000 evaluations to converge (for a number of combinations superior to 10^9), which is performed with a CPU time around 2 minutes on a single processor (no parallelization performed yet). This low CPU-time is due to the very simple numerical models used for process units (see the Appendix, no thermodynamic calculations is performed and process units models are mainly composed of analytical expressions) and is not expected to be encountered in real-case applications. The percentage of non-convergence (due to initial values or incoherent structures) observed for the various runs varies between 2 an 20% with an average of 8% on 100 runs.

It should be reminded that this case study remains quite simplified. A higher number of degrees of freedom would be available if non-ideal separations, columns with staged withdrawals and feed, as well as Cl_2 injection and HCl stripping were integrated into the PSE problem, possibly leading to more complex structures.

4. Discussions and perspectives

4.1. On the elementary building blocks

The elementary building blocks include numerical models to calculate outlet streams and parameters from inlet streams and parameters, requiring a certain effort depending on the model complexity. The important question in such process synthesis problem is: where to put the effort?

Types of models

The process designer should put importance into the blocks definition according to the treated problem. Simplified functional or physical models provide relatively few information but are easier to develop and numerically more robust. On the contrary, rigorous models provide detailed information such as a precise link between design and performance, but could be difficult to solve in some cases (internal convergence procedure), hence penalizing the optimization. To circumvent (numerical) complexity, one could use simplified physical models (such as ideal conditions instead of rigorous thermodynamic models, or effective diffusivities instead of Maxwell-Stefan equation for multicomponent mass transfer), but at the –potentially huge– cost of losing accuracy, or missing physical limitations (such as azeotrope in distillation, or diffusion barrier in multicomponent transfer). Depending on the case, a simplified models may not be suited.

The use of surrogate models fitted from a rigorous model may be a good compromise. It requires an additional effort to define a proper numerical design of experiments covering the entire search space (i.e. unit inlets and degrees of freedom) and to perform the model reduction. A large variety of surrogate models can be used (neural networks, polynomial chaos, response surface, kriging etc.). Such models allow to benefit from the information of a complete physical model with the numerical robustness of a simplified one. It should also be noted that this solution allows to use already available models (e.g. former models, using commercial simulator to perform the design of experiments) without requiring informatics coupling between codes.

Internal or external integration

In the case of complex units, numerous degrees of freedom peculiar to the nature of the unit needs to be determined by optimization, including integer values. For example, distillation columns and more specifically heat integrated distillation columns (HIDiC), optimal design requires to determine both continuous (e.g. heat duties, flowrates) and discrete variables (number of stages, withdrawals and injection stages). To deal with such systems, the question is to choose between resolving *internally* or *externally* the case of integer decision variables.

If the complexity is dealt *internally*, a MINLP problem needs to be solved within the unit operation model, while keeping the NLP for convergence and optimization of macroscopic continuous variables at the process level. Such an approach may be long and tedious since the MINLP sub-problem needs to be solve a large number of times, but the overall Process Synthesis problem remains as simple as for a single distillation column. If the complexity is dealt *externally*, it can be directly integrated in the synthesis problem by decomposing the distillation columns into other elementary building blocks (e.g. theoretical stage / transfer unit, mixers, splitters, reboiler, condenser). This approach complexify the Process Synthesis problem but synergies may be obtained. Intensified units such as reactive distillation could emerge if we consider reaction and separations as elementary building blocks. In the presented method, both approaches would be possible, depending only on the choice of available building blocks by the user.

Beyond unit operations as building blocks

The conceptual method lies on the arrangements of elementary 'black-boxes' building blocks. These blocks could therefore represent more than just unit operations. For instance, one could extend the method using building blocks as phenomenological modules, or an entire existing process (e.g. for retrofitting study), or even a task to be performed (for scheduling optimization). Using building blocks not based on process units have already been explored in literature focused on Process Intensification but few in Process Systems Engineering studies. Such building blocks could for example be based on general heat/mass module [20], elementary process functions [22] or directly involved phenomena [23, 24, 25]. No further development would be required to the method, the limitations coming essentially from the process designer creativity.

4.2. On the synthesis drivers

As in all process synthesis methods, the choice of synthesis drivers is crucial to define appropriate constraints and objective function. Such drivers should reflect the designer's criteria, and commonly need to integrate several design and performance considerations. Some criteria can be integrated as optimization

constraints (e.g. security, material temperature limits) while others can be used to define the objective function (e.g. energy, economics, environment).

A key aspect is the introduction of a parsimony principle into the problem formulation to limit the number of equipment and process complexity as much as possible (i.e. when not necessary). This aspect is exacerbated in case of *ab-initio* process synthesis since the algorithm can generate very complex structures. For the case study, an economic objective function was used. It introduced a sort of parsimony principle: each equipment comes at a certain cost, unit operations inducing no performance improvement are therefore dismissed.

A possibility would be the use of RAM (Reliability, Availability, Maintainability) analysis, allowing to convert process complexity into a loss of operability (e.g. a decreased in annual operation time). Such analysis, if integrated into the objective function, would quantify if an increase in process complexity would imply a sufficient gain compared to the loss of operability.

It should also be noted that working with economic functions comes with certain uncertainties due to the process maturity (Technology Readiness Level from 1 up to 9 such as lab-, pilot and industrial scales) and/or the effort put on the evaluation (e.g. conceptual study, basic design, detailed engineering). These uncertainties needs to be properly evaluated [53], especially when comparing several process alternatives.

4.3. On the uses of ab-initio *process synthesis*

The interest and drawbacks of the proposed approach in comparison with other process synthesis methods are yet to be determined. Global optimization methods used for process synthesis, such as MINLP formulation, provide the optimal solution in a restricted search space; whereas *ab-initio* approaches intrinsically cannot guarantee the convergence since they freely explore the structure possibilities. The use of evolutionary programming algorithms for process synthesis could show interest to circumvent the bias appearing in new conceptual studies and other problems with a very large number of possible combinations.

In approaches using total connectivity based superstructure, it is possible to be as exhaustive as possible in the interconnections of process units by allowing free connections between the chosen units. However, the same intrinsic bias exists in these approaches since an appropriate set of unit operations needs to be postulated. To increase the chance of finding an optimal solution, the number of unit operations can be increased, potentially leading to technical limitations (e.g. convergence issues, computational time) due to the large size of the system. Compared to the presented approaches, it may not be differences between the results in practice (which is yet to be tested). Still, one potential interest of *ab-initio* approaches is that the initial time spent to define the problem may be reduced in the proposed approach since the 'process designer' only needs to define the available building blocks (e.g. process unit operation, process function) so that it may focus on the process problem definition, rather trying to include as many alternatives as possible in a mathematical formulation. Further works will try to compare both approaches on a same problem.

Both types of approaches could also be very complementary and used sequentially. An *ab-initio* approach could be used initially to explore the infinite search space using the defined unit operations; then, the structural results of the best individuals could be analysed to define a superstructure; eventually, a superstructure-based approach would find the optimal solution within this restricted design space.

5. Conclusion

A new process synthesis approach is proposed which frees from the definition of a set of structure alternatives such as a process superstructure. From a set of unit operations and an objective function defined by the user -i.e. the process designer-, the method systematically constructs process flowsheet structures using an Evolutionary Programming (EP) algorithm and evaluates them using a Non-Linear Programming (NLP) algorithm. The method's implementation, called PSEvo, has been tested on a reaction-separation-problem. Obtained optimal solutions are in compliance with literature [30], which validates the proof-of-concept.

Future works will include:

- Comparison between several Process Synthesis approaches on the same case study using rigorous models for process units. Candidate problems include separation processes and power cycles in order to evaluate the method performance with multicomponent and highly integrated processes.
- Comparison of different strategies for the definition of integrated process units, to define the best options between integrating the unit complexity within the process unit model and decomposing the unit into elementary functions.
- Definition of appropriate synthesis driver (objective functions and constraints) to consider both process complexity and economic uncertainties during the synthesis procedure.

.

.

.

Acknowledgements

The author addresses its acknowledgements to Jean-Marc Commenge, Laurent Falk and Eric Favre of the CNRS-LRGP laboratory for their valuable comments and inputs to this work.

Nomenclature

C_{tot}	molar density (kmol/m^3)
F_i, F_{tot}	partial and total molar flowrate (kmol/hr)
F_{obj}	objective function
$\mathbf{g}$	inequality constraints
$\mathbf{h}$	equality constraints
H	Shannon entropy index
k	kinetic constant (s^{-1})
p_i	probability of observing an individual (process) among the population
P	purchase of reactants ($/hr)
r	split ratio
$\mathbf{s}$	(guessed) tear stream variables
$\bar{\mathbf{s}}$	calculated tear stream variables
S	product sales of B ($/hr)
$\mathbf{t}$	tear equations
V	reactor volume (m^3)
$\mathbf{x}$	decision variables in the process
$\mathbf{z}$, $\mathbf{z}_l, \mathbf{z}_u$	optimization variables, lower/upper limits
Greek letters	
α	mutation rate
Sub- and superscripts	
add	addition of a unit operation (mutation operator)
in	inlet of a unit operation
ind	individual
out	outlet of a unit operation
permut	permutation of two streams (mutation operator)
remove	removal of a unit operation (mutation operator)
top	top of distillation column

bottom bottom of distillation column

Abbreviations

CAPEX	Capital expenditure ($)
CSTR	Continuous Stirred-Tank Reactors
EA	Evolutionary Algorithm
EP	Evolutionary Programming
MINLP	Mixed Integer NonLinear Programming
NLP	NonLinear Programming
OPEX	Operational expenditure ($/yr)
PFR	Plug Flow Reactor
PI	Process Intensification
PSE	Process Systems Engineering
PSEvo	Process Synthesis by Evolution

References

[1] I. Grossmann, A. W. Westerberg, Research challenges in process systems engineering, AIChE Journal 46(9) (2000) 1700–1704.

[2] J.-F. Portha, L. Falk, J.-M. Commenge, Local and global process intensification, Chemical Engineering & Processing: Process Intensification 84 (2014) 1–13.

[3] J. M. Douglas, Conceptual design of chemical processes, John Wiley & Sons, Ltd., 1988.

[4] R. Smith, B. Linnhoff, The design of separators in the context of overall processes, Chemical Engineering Research and Design 66 (1988) 195–228.

[5] W. D. Seider, J. D. Seader, D. R. Lewin, S. Widagdo, Product and Process Design Principles: Synthesis, Analysis and Design, John Wiley & Sons, Ltd., 2008.

[6] C. A. Floudas, C. E. Gounaris, A review of recent advances in global optimization, Journal of Global Optimization 45 (2009) 3–38.

[7] L. T. Biegler, Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, MOS–SIAM Series on Optimization, 2010.

[8] N. Nishida, G. Stephanopoulos, A. W. Westerberg, A review of process synthesis, AIChE Journal 27 (1981) 321–351.

[9] I. E. Grossmann, M. M. Daichendt, New trends in optimization–based approaches to process synthesis, Computers & Chemical Engineering 20 (1996) 665–683.

[10] L. T. Biegler, I. E. Grossmann, A. W. Westerberg, Systematic Methods of Chemical Process Design, Prentice Hall, 1997.

[11] C. A. Floudas, Nonlinear and Mixed–Integer Optimization. Fundamentals and Applications, Oxford University Press, 1998.

[12] A. Westerberg, A retrospective on design and process synthesis., Computers & Chemical Engineering 28 (2004) 447–458.

[13] S. Cremaschi, A perspective on process synthesis: Challenges and prospects, Computers & Chemical Engineering 81 (2015) 130–137.

[14] M. Barttfeld, P. Aguirre, I. Grossmann, A decomposition method for synthesizing complex column configurations using tray-by-tray gdp models, Computers & Chemical Engineering 28 (11) (2004) 2165–2188.

[15] A. W. Dowling, L. T. Biegler, A framework for efficient large scale equation–oriented flowsheet optimization, Computers & Chemical Engineering 72 (2015) 3–20.

[16] M. Skiborowski, M. Rautenberg, W. Marquardt, A hybrid evolutionary–deterministic optimization approach for conceptual design, Industrial & Engineering Chemistry Research 54 (2015) 10054–10072.

[17] C. Adjiman, S. Dallwig, C. Floudas, A. Neumaier, A global optimization method, αbb, for general twice–differentiable constrained NLPs – I. theoretical advances, Computers & Chemical Engineering 22 (1998) 1137–1158.

[18] R. B. Boozarjomehry, A. P. Laleh, W. Y. Svrcek, Automatic design of conventional distillation column sequence by genetic algorithm, The Canadian Journal of Chemical Engineering 87 (3).

[19] I. E. Grossmann, F. Trespalacios, Systematic modeling of discrete–continuous optimization models through generalized disjunctive programming, AIChE Journal 59 (2013) 3276–3295.

[20] K. P. Papalexandri, E. N. Pistikopoulos, Generalized modular representation framework for process synthesis, AIChE Journal 42 (4) (1996) 1010–1032.

[21] P. Lutze, D. K. Babi, J. M. Woodley, R. Gani, Phenomena based methodology for process synthesis incorporating process intensification, Industrial & Engineering Chemistry Research 52 (2013) 7127–7144.

[22] A. Peschel, H. Freund, K. Sundmacher, Methodology for the design of optimal chemical reactors based on the concept of elementary process functions, Industrial & Engineering Chemistry Research 49 (21) (2010) 10535–10548.

[23] L. A. Zivkovic, N. M. Nikacevic, A method for reactor synthesis based on process intensification principles and optimization of superstructure consisting of phenomenological modules, Chemical Engineering Research and Design 113 (2016) 189–205.

[24] S. E. Demirel, J. Li, M. F. Hasan, Systematic process intensification using building blocks, Computers & Chemical Engineering 105 (2017) 2 – 38, process Intensification.

[25] H. Kuhlmann, M. Skiborowski, Optimization-based approach to process synthesis for process intensification: General approach and application to ethanol dehydration, Industrial & Engineering Chemistry Research 56 (45) (2017) 13461–13481.
[26] Q. Chen, I. Grossmann, Recent developments and challenges in optimization-based process synthesis, Annual Review of Chemical and Biomolecular Engineering 8 (1) (2017) 249–283.
[27] S. Pal, L. Hazra, Ab initio synthesis of linearly compensated zoom lenses by evolutionary programming, Applied optics 50 (10) (2011) 1434–1441.
[28] V. van Speybroeck, R. Gani, R. J. Meier, The calculation of thermodynamic properties of molecules, Chemical Society Reviews 39 (2010) 1764–1779.
[29] S. Boonstra, K. V. D. Blom, H. Hofmeyer, R. Amor, M. T. M. Emmerich, Super–structure and super–structure free design search space representations for a building spatial design in multi–disciplinary building optimisation, 23rd International Workshop of the European Group for Intelligent Computing in Engineering, EG–ICE 2016 87 (2016) 477–492.
[30] A. C. Kokossis, C. A. Floudas, Synthesis of isothermal reactor–separator–recycle systems, Chemical Engineering Science 46 (5) (1991) 1361 – 1383.
[31] H. Lipson, J. B. Pollack, Automatic design and manufacture of robotic lifeforms, Nature 406 (2000) 974–978.
[32] D. B. Fogel, An Overview of Evolutionary Programming, Springer New York, 1999.
[33] A. Eiben, J. E. Smith, Introduction to Evolutionary Computing, Natural Computing Series, Springer, 2003.
[34] X. Yu, M. Gen, Introduction to Evolutionary Algorithms, Springer, 2010.
[35] L. J. Fogel, A. J. Owens, M. J. Walsh, Artificial Intelligence through Simulated Evolution, John Wiley, New York, USA, 1966.
[36] C. Koch, F. Cziesla, G. Tsatsaronis, Optimization of combined cycle power plants using evolutionary algorithms, Chemical Engineering and Processing: Process Intensification 46 (2007) 1151–1159.
[37] T. Keller, B. Dreisewerd, A. Gorak, Reactive distillation for multiple–reaction systems: Optimisation study using an evolutionary algorithm, Chemical and Process Engineering 34 (2013) 17–38.
[38] J. R. Roach, B. K. O'Neill, D. A. Hocking, A new synthetic method for stream tearing in process systems analysis, Chemical Engineering Communications 161 (1997) 1–14.
[39] D. Wolbert, X. Joulia, B. Koehret, L. T. Biegler, Flowsheet optimization and optimal sensitivity analysis using analytical derivatives, Computers & Chemical Engineering 18 (1994) 1083–1095.
[40] D. Kraft, Algorithm 733: TOMP–Fortran modules for optimal control calculations, ACM Transactions on Mathematical Software 20 (3) (1994) 262–281.
[41] K. Schittkowski, A Fortran implementation of a sequential quadratic programming algorithm with distributed and non-monotone line search - user's guide, version 3.0, Tech. rep., Report, Department of Computer Science, University of Bayreuth (2010).
[42] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Mathematical Programming 106 (1) (2006) 25–57.
[43] P. Voll, M. Lampe, G. Wrobel, A. Bardow, Superstructure–free synthesis and optimization of distributed industrial energy supply systems, Energy 45 (2012) 424–435.
[44] L. Wang, P. Voll, M. Lampe, Y. Yang, A. Bardow, Superstructure–free synthesis and optimization of thermal power plants, Energy 91 (2015) 700–711.
[45] J. Williams, SLSQP - Modern Fortran Edition of the SLSQP Optimizer, `https://github.com/jacobwilliams/slsqp` (October 2016).
[46] E. R. Gansner, S. C. North, An open graph visualization system and its applications to software engineering, Software - Practice and Experience 30 (11) (2000) 1203–1233.
[47] T. Williams, C. Kelley, Gnuplot 5.0: an interactive plotting program, `http://gnuplot.sourceforge.net/` (October 2016).
[48] A. C. Kokossis, C. A. Floudas, Optimization of complex reactor networks – I. isothermal operation, Chemical Engineering Science 45 (1990) 595–614.
[49] W. R. Esposito, C. A. Floudas, Deterministic global optimization in isothermal reactor network synthesis, Journal of Global Optimization 22 (2002) 59–95.
[50] H. Soltani, S. Shafiei, Adiabatic reactor network synthesis using coupled genetic algorithm with quasi linear programming method, Chemical Engineering Science 137 (2015) 601 – 612.
[51] C. Shannon, A mathematical theory of communication, Bell system technical journal 27 (1948) 379–423, 623–656.
[52] S. K. Smit, A. E. Eiben, Using Entropy for Parameter Analysis of Evolutionary Algorithms, Springer Berlin Heidelberg, 2010, pp. 287–310.
[53] T. Neveux, O. Authier, M. Kanniche, F. Siros, Uncertainties in techno-economic evaluation of innovative processes - effect of technology maturity and preparation effort, in: Nancy, France - Presentation at the 30th SFGP Congress, July 2017. doi:10.13140/RG.2.2.22632.16644.

Supporting information

An example of process representation (phenotype) before and after the application of a mutation operator and the connectivity algorithm is provided.

	Stream connectivity	Inlet parameter	Outlet streams	Outlet parameters
Feed	0 inlet, 1 outlet	$F_A \in [50-100]$ kmol/hr	$F_{A,out} = F_A$ $F_{B,out} = 0$ $F_{C,out} = 0$	$P = (19.88 + 27.87)F_A$
Out	1 inlet, 0 outlet	none	none	$F_{prod,B} = \begin{cases} 0 & \text{if } x_B < 1 \\ F_{B,in} & \text{if } x_B = 1 \end{cases}$
Mixer	2 inlets, 1 outlet	none	$F_{i,out} = F_{i,in1} + F_{i,in2}$ $i = \{A, B, C\}$	$CAPEX_{mix} = 500\$$
Splitter	1 inlet, 2 outlets	$r \in [0.05 - 0.95]$ (split ratio)	$F_{i,out\,1} = r \cdot F_{i,in}$ $F_{i,out\,2} = (1-r) \cdot F_{i,in}$ $i = \{A, B, C\}$	$CAPEX_{split} = 500\$$
CSTR	1 inlet, 1 outlet	$V \in [0.01 - 50]$ m^3	See eqs. 6 and 7	$CAPEX_{CSTR}$ (see eqs. 8 and 9)
PFR	1 inlet, 1 outlet	$V \in [0.01 - 50]$ m^3	See eqs. 10	$CAPEX_{PFR}$ (see eqs. 11 and 12)
Dist$_{A/BC}$	1 inlet, 2 outlets	none	$F^{top}_{A,out} = F_{A,in}$; $F^{bottom}_{A,out} = 0$ $F^{top}_{B,out} = 0$; $F^{bottom}_{B,out} = F_{B,in}$ $F^{top}_{C,out} = 0$; $F^{bottom}_{A,out} = F_{C,in}$	$CAPEX_{A/BC}$ (see eqs. 13) $OPEX_{A/BC}$ (see eqs. 14)
Dist$_{AB/C}$	1 inlet, 2 outlets	none	$F^{top}_{A,out} = F_{A,in}$; $F^{bottom}_{A,out} = 0$ $F^{top}_{B,out} = F_{B,in}$; $F^{bottom}_{B,out} = 0$ $F^{top}_{C,out} = 0$; $F^{bottom}_{A,out} = F_{C,in}$	$CAPEX_{AB/C}$ (see eqs. 16) $OPEX_{AB/C}$ (see eqs. 17)

Table 2: Definition of unit operations for the case study

Appendix: expressions for the case study

Stream description

3 stream variables: F_A, F_B, F_C (partial molar flowrates, in kmol/hr)

Objective function

The objective function is the annualized profit ($/yr):

$$F_{obj} = 720 \cdot (S - P) - \left(\frac{CAPEX}{2.5} + 0.52 \cdot OPEX \right) \tag{3}$$

with:

- S is the sales of B ($/hr), - see below (4).
- P is the purchase of reactants ($/hr) - see 'Feed' unit operation in Table 2.
- $CAPEX$ is the process capital cost ($) - sum of the CAPEX of all unit operations, $CAPEX = \sum_i CAPEX_i$ with $i \in \{CSTR, PFR, Mixer, Splitter, Dist_{A/BC}, Dist_{AB/C}\}$.
- $OPEX$ is the process operating cost ($/yr) - sum of the OPEX of all unit operations, $OPEX = \sum_i OPEX_i$ with $i \in \{Dist_{A/BC}, Dist_{AB/C}\}$.
- 2.5 is the payout time (yr),
- 0.52 is the income tax rate.
- 720 is the annual operation time (hr/yr). The authors [30] did not report the operation time but it was calculated from one of their example. They obtained a process with a profit of 1224 k$/yr ($F_{obj}$), an annual cost of 321 k$/yr ($\frac{CAPEX}{2.5} + 0.52 OPEX$), a production of 50 kmol/hr ($S = 2481$ $/hr), and a fresh feed stream of 51.97 kmol/hr ($P = 4634$ $/hr). The operation time is therefore 720 hr/yr.

The product sales S are calculated considering the production target of 50 kmol/hr of pure B (92.67 \$/mol). The purity and productivity targets are integrated into the objective function through the product sales, avoiding to define an additional process constraint:

$$\begin{aligned} S &= \begin{cases} 92.67 \cdot 50 & \text{if } F_{prod,tot} \geq 50 \\ 0 & \text{else} \end{cases} \\ &\simeq 92.67 \cdot 50 \cdot H_{smooth}(F_{prod,tot} - 50) \end{aligned} \tag{4}$$

To ensure continuity, a smooth approximation of the Heaviside function is used: $H_{smooth}(x) = \frac{1}{1+\exp(-kx)}$ (with a chosen steepness $k = 5000$ here).

The overall production F_{prod} of pure B (in kmol/hr) is computed from the 'Out' units:

$$F_{prod,tot} = \sum_{Out} F_{prod,B,i} = \sum_{Out} H_{smooth}(x_{B,i} - 1) \tag{5}$$

Available unit operations

The unit operations are detailed in Table 2 and below.

Feed

The 'Feed' has an optimizable flowrate, which is adjusted by the NLP to satisfy the production target of 50 kmol/hr (see §4). The outlet parameter is the purchase of reactants P (in \$/hr), which includes benzene (A, cost of 27.87 \$/kmol) and chlorine (cost of 19.88 \$/kmol) in equimolar quantities: $P = (19.88+27.87)F_A$.

Out

The 'Out' unit is the process outlet. Its calculated parameter is the produced flowrate, which is null if the B purity x_B ($x_B = F_B/(F_A + F_B + F_C)$) is lower than 1.

Mixer and splitter

Unlike in [30], a capital cost of 500\$ is considered for mixers and splitters. This value is low compared to capital cost of other unit operations, it is introduced to affect a penalty (even low) to each block.

CSTR

The CSTR outlet streams are calculated by the following equations:

$$F_{i,out} = F_{tot} \cdot x_{i,out} \quad i = \{A, B, C\} \tag{6}$$

with $F_{tot} = F_{A,in} + F_{B,in} + F_{C,in}$ the total molar flowrate (invariant during reaction), and $x_{i,out}$ the molar composition at the outlet, calculated by mass balance:

$$\begin{aligned} x_{A,out} &= \frac{x_{A,in} \cdot F_{tot}}{F_{tot} + V \cdot k_1 \cdot C_{tot}} \\ x_{B,out} &= \frac{F_{tot} \cdot x_{B,in} + V \cdot x_{A,out} \cdot k_1 \cdot C_{tot}}{F_{tot} + V \cdot k_2 \cdot C_{tot}} \\ x_{C,out} &= \frac{F_{tot} \cdot x_{C,in} + V \cdot x_{B,out} \cdot k_2 \cdot C_{tot}}{F_{tot}} \end{aligned} \tag{7}$$

with $k_1 = 0.412\,\mathrm{s}^{-1}$, $k_2 = 0.055\,\mathrm{s}^{-1}$, and the molar density $C_{tot} = 11.22 \cdot x_A + 9.86 \cdot x_B + 8.84 \cdot x_C$ (kmol/m^3). Since the outlet composition is unknown, a first estimate of C_{tot} is calculated using inlet composition, then an iterative procedure is adopted until convergence.

The CSTR CAPEX is calculated respectively for Case A (same cost for PFR and CSTR) and for Case B (higher cost for PFR) from:

$$CAPEX_{CSTR,\,\text{Case A}} = 12760.43 + 14059.78 \cdot V_{\text{CSTR}} \tag{8}$$

$$CAPEX_{CSTR,\,\text{Case B}} = 25795.0 + 8178.0 \cdot V_{\text{CSTR}} \tag{9}$$

PFR

The PFR outlet streams are calculated by integrating the following ordinary differential equations:

$$\begin{aligned}\frac{dF_A}{dV} &= -k_1 \cdot C_{tot} \cdot \frac{F_A}{F_{tot}} \\ \frac{dF_B}{dV} &= k_1 \cdot C_{tot} \cdot \frac{F_A}{F_{tot}} + k_2 \cdot C_{tot} \cdot \frac{F_B}{F_{tot}} \\ \frac{dF_C}{dV} &= k_2 \cdot C_{tot} \cdot \frac{F_B}{F_{tot}}\end{aligned} \tag{10}$$

with the initial conditions: $F_i(V = 0) = F_{i,in} \quad i = \{A, B, C\}$. The molar density is recalculated at each integration step.

The PFR CAPEX is calculated respectively for Case A (same cost for PFR and CSTR) and for Case B (higher cost for PFR) from:

$$CAPEX_{PFR,\,\text{Case A}} = 12760.43 + 14059.78 \cdot V_{\text{PFR}} \tag{11}$$

$$CAPEX_{PFR,\,\text{Case B}} = 2984.938 + 49332.715 \cdot V_{\text{PFR}} \tag{12}$$

Distillation columns

In [30], four different columns were defined to distinguish ternary (A/BC, AB/C) and binary separations (A/B, B/C). In this work, only two are used (A/BC, AB/C), the different equations between ternary and binary separations are chosen according to feed composition $x_{i,in} = F_{i,in}/(F_{A,in} + F_{B,in} + F_{C,in})$.

$\text{Dist}_{\text{A/BC}}$. If the feed does not contain C ($x_{C,in} = 0$), the column is equivalent to a binary A/B distillation (column 4 of [30]). If not, it is a A/BC distillation (column 1 of [30]). CAPEX and OPEX are calculated accordingly:

$$CAPEX_{A/BC} = \begin{cases} 132718.16 - F_{tot,in} \cdot \\ (369.05 \cdot x_{A,in} - 1113.86 \cdot x_{B,in}) & \text{(if A/B)} \\ \\ 86844 - F_{tot,in} \cdot 1136 \cdot x_{A,in} & \text{(if A/BC)} \end{cases} \tag{13}$$

$$OPEX_{A/BC} = (21.67 + 4.65) \cdot F_{tot,in} \cdot q_{A/BC} \tag{14}$$

$$\text{with } q_{A/BC} = \begin{cases} 10.70 + 28.41 \cdot x_{A,in} & \text{(if A/B)} \\ 3.0 + 36.1 \cdot x_{A,in} - 7.7 \cdot x_{B,in} & \text{(if A/BC)} \end{cases} \tag{15}$$

$\text{Dist}_{\text{AB/C}}$. If the feed does not contain A ($x_{A,in} = 0$), the column is equivalent to a binary B/C distillation (column 3 of [30]). If not, it is a AB/C distillation (column 3 of [30]). CAPEX and OPEX are calculated accordingly:

$$CAPEX_{AB/C} = \begin{cases} 25000 + F_{tot,in} \cdot \\ \left(6985 \cdot x_{B,in} - 3870 \cdot x_{B,in}^2\right) & \text{(if B/C)} \\ \\ 211547 - F_{tot,in} \cdot \\ (1010.0 \cdot x_{A,in} - 479.0 \cdot x_{B,in}) & \text{(if AB/C)} \end{cases} \tag{16}$$

$$OPEX_{AB/C} = (21.67 + 4.65) \cdot F_{tot,in} \cdot q_{AB/C}$$

$$\text{with } q_{AB/C} = \begin{cases} 10.70 + 28.41 \cdot x_{A,in} & \text{(if B/C)} \\ 3.0 + 36.1 \cdot x_{A,in} - 7.7 \cdot x_{B,in} & \text{(if AB/C)} \end{cases} \tag{17}$$