AIChE

Pattern Recognition in Chemical Process Flowsheets

Tong Zhang and Nikolaos V. Sahinidis
Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

Jeffrey J. Siirola
Dept. of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213

Dept. of Chemical Engineering, Purdue University, West Lafayette, IN 47907

DOI 10.1002/aic.16443
Published online November 22, 2018 in Wiley Online Library (wileyonlinelibrary.com)

By recognizing similarities in flowsheets, engineers can understand ways in which to improve the design and efficiency of chemical processes. However, there is no prior literature on how to compare flowsheets and mine them for common patterns. To fill this gap, we propose the first systematic methodology to mine patterns in chemical process flowsheets. The proposed methodology consists of three major steps, each of which has a polynomial time complexity. We apply our methodology to several case studies that involve comparisons of up to 18 different flowsheets. The patterns identified by our methodology are consistent with engineering practice and heuristic rules in the process synthesis literature. © 2018 American Institute of Chemical Engineers *AIChE J*, 65: 592–603, 2019
Keywords: flowsheet comparison, process pattern, heuristic rule, sequence alignment

Introduction

A chemical flowsheet is a diagram that contains a wealth of information to describe a chemical process. Such a diagram displays the process unit layout needed to achieve the conversion of raw materials to final chemical products. In practice, engineers generate process flowsheets by accounting for many factors such as operating conditions, environmental impact, and economics.

This article is based on the premise that process designers can learn from existing flowsheet structures to improve process design and operations. Analyzing a chemical manufacturing process flowsheet is analogous to understanding a sentence. We comprehend a sentence by reading words sequentially. In the same way, we analyze a chemical flowsheet by tracking material flows in a specific order. Raw materials are converted into final chemical products through sequences of unit operations. While sentences make sense in the space of words, chemical flowsheets likewise are characterized by individual process units. Flowsheets share process patterns that result either from economic considerations or material properties. These patterns are aggregations of process units that perform particular functions in chemical processes. Such patterns, analogous to phrases, can be assembled and utilized to engineer improved process designs. Furthermore, identified patterns may aid the development of new heuristic design rules. However, manually comparing flowsheets and identifying process patterns is time consuming and requires a significant amount of domain-specific knowledge.

There is no literature on chemical process pattern recognition. In this article, we develop the first systematic methodology to identify patterns in chemical process flowsheets. We fully automate the proposed methodology, which has three steps. The first step is to generate a graph for a given flowsheet. Chemical process flowsheets are converted into directed colored graphs, where nodes correspond to process units in the original flowsheets. The arcs in the graph represent material flows to and from process units. In the second step, flowsheet graphs are converted to SFILES strings.[1] To preserve unit connection information of the flowsheets, the strings are generated through traversing the graphs from the raw material nodes with depth-first search. The final step compares generated flowsheet strings and identifies process patterns with sequence alignment algorithms. The proposed methodology is efficient; the SFILES string generation step and the pattern identification step both have a polynomial time complexity. Furthermore, our methodology requires no domain-specific knowledge.

The proposed methodology is applied and demonstrated in three different settings. First, we compare two methanol process flowsheets. Second, we compare a methanol process flowsheet with an ethylene oxide process flowsheet. Finally, we apply this technique to 18 chemical process flowsheets to identify general process patterns. Among the identified process patterns, some of the patterns represent intuitive approaches to process design. Other patterns correspond to heuristic rules in the process synthesis literature.[2,3]

The remainder of this article is organized as follows. In the next section, we review the literature on process flowsheet representations, including the SFILES notation. Then, we detail the proposed methodology in three subsections: flowsheet

This contribution was identified by Ana Torres (Universidad de la Republica Uruguay) as the Best Presentation in the session "Process Design" of the 2017 AIChE Annual Meeting in Minneapolis.

Correspondence concerning this article should be addressed to N. V. Sahinidis at sahinidis@cmu.edu.

graph representation, flowsheet string notation generation, and sequence alignment algorithms. Next, we detail the aforementioned applications of the proposed methodology. Finally, we provide conclusions from this work and describe potential extensions in the last section.

Background

Flowsheet representation

Flowsheets can be understood as networks composed of material streams and equipment. They are often used in the context of engineering and design applications. The three most commonly used graph representations of chemical processes are block flow diagrams, process flow diagrams, and piping and instrumentation diagrams. In an effort to design an optimal flowsheet, many representations of flowsheets have been developed and implemented, as shown in Figure 1. The two most frequently used representations are state task network (STN) and state equipment network (SEN). STN was introduced by.[4] To avoid the ambiguities caused by general recipe networks, the authors introduce two types of nodes: the *state* nodes representing materials and the *task* nodes denoting the process operations. The *state* nodes are connected to *task* nodes with directed edges in the network. Another complementary representation is SEN, where instead of using *task* nodes, *equipment* nodes are defined to denote the devices that execute given tasks. While tasks are predefined and must be assigned to certain equipment in STN, equipment in SEN is provided and awaits the assigned tasks. Both representations can be used to formulate optimization models that perform process synthesis.[5]

Another representation similar to STN is the process graph (P-graph). Friedler et al.[6] proposed this concept in the early 1990s based on graph theory and combinatorial techniques. By exploiting the unique process structure features, this graph representation can be used to solve general process synthesis problems.[7] Friedler et al. also proved that any feasible process structure can be uniquely represented by a directed bipartite graph.

R-graph is another graphical representation proposed by.[8] It is a one-task–one-equipment graph in which the nodes are the input and output ports of the process units. As shown in Figure 1, the directed edge always begins at an output port node of a unit and ends at an input port node of a unit.

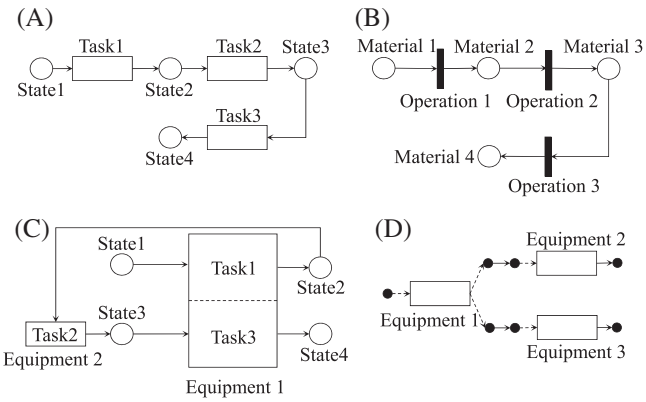
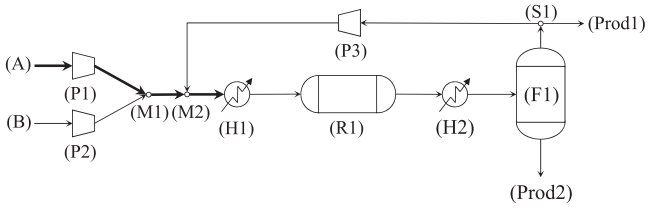


Figure 1. Superstructure representations: (a) STN; (b) P-graph; (c) SEN; (d) R-graph.



|(A)(P1)(M1)<2(M2)<1(H1)(R1)(H2)[(F1)(Prod2)]

(F1)[(S1)(P3)(M2)1](S1)(Prod1)|(B)(P2)(M1)2

Figure 2. Process flowsheet and corresponding SFILES string.

Simplified flowsheet input line entry system (SFILES)

Anterraches[9] developed a simple notation system for efficient storage of structural information of process flowsheets called simplified flowsheet input line entry system (SFILES). This string notation system is similar to simplified molecular input line entry system (SMILES), which is largely used for describing the structure of molecules using ASCII strings.

In our implementation, each element in parenthesis represents a process unit in the flowsheet. For example in Figure 2, the reactor is denoted as (R1). Two consecutive process units imply a connection from the first unit to the second unit. Unlike a molecular bond in which the order of atoms does not matter, the connection between two process units is specifically oriented. Therefore, the order of process units is maintained in SFILES to preserve the information about the flow direction. The SFILES string is read from left to right. The left square bracket in SFILES denotes the start of a branch, while the right bracket terminates the branch. Thus, the SFILES substring (R1)(H2)[(F1) in Figure 2 is interpreted in the following sequence: the material goes through the reactor (R1), heat exchanger (H2), and flash (F1), then leaves the flash drum through one of the outlet streams. Finally, as Figure 2 indicates, multiple lines may be used to completely represent a SFILES string.

Pattern Mining Methodology

The proposed process pattern mining methodology contains three major steps. The first step is to develop a graph representation for general process flowsheets. Nodes in this graph representation are assigned attributes that model unit operation peculiarities and important characteristics. In the second step, a depth-first search is applied, converting the graph representation of the flowsheet to a SFILES string. Finally, SFILES strings are mined for process patterns with sequence alignment algorithms for text comparison. We will now discuss each of these three steps in more detail.

Flowsheet directed graph representation

In the directed graph representation of a process system, process units and connections are denoted as the arcs and nodes of the graph. This graph can be used for representing and analyzing a process system. Several other graph representations were developed for solving process synthesis problems.[10] However, instead of defining two different types of nodes like STN and SEN, we introduce nodes in our graph that represent the raw materials, products, and process units in flowsheets. Figure 3 shows a simplified process flow diagram for methanol production using hydrogen and carbon dioxide as raw materials. This flowsheet can be converted into the

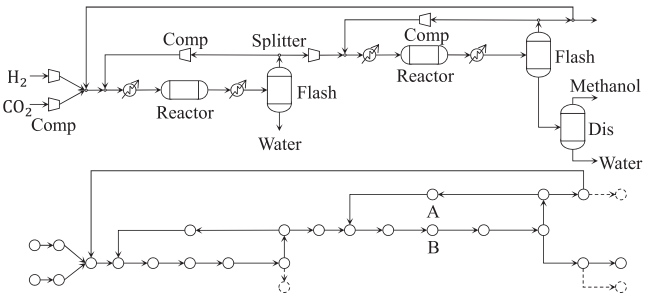


Figure 3. Flowsheet for methanol production process.

directed graph seen in Figure 3 in which nodes represent raw materials, products, and process units. Nodes outlined by dotted lines represent the purge and water streams. They can be eliminated from the graph if they are not critical to the process.

It is insufficient to use a simple directed graph to represent a general process flowsheet. Each node that corresponds to a specific process unit must maintain its original functional information in our graph representation. For example, in Figure 3, nodes A and B each have one input and one output stream. Both nodes preserve the same connection property in the directed graph but represent different process units in the original flowsheet. Therefore, we label (color) the nodes according to their functionalities in our directed graph representation. As shown in Figure 4, each node in our graph representation is assigned an attribute that corresponds to a particular type of unit operation.

The proposed node-labeling technique resolves the problem of identifying nodes by their functions. In the same way, arcs in the graph can be distinguished by their functional characteristics. This is especially important for arcs that are connected to heat exchangers. Each heat exchanger has two output streams. The material in the cold input stream must leave the heat exchanger through the arc that represents the cold outlet stream.

Using node and arc labeling, we convert a flowsheet to a colored weighted directed graph for process system structural analysis.

SFILES string generation

To automate the SFILES string generation of a flowsheet, we start traversing the graph from one of the raw materials and concatenate the name of each visited node. For example, raw material (A) flows through a compressor (P1), two mixers (M1) and (M2), and a heat exchanger (H1) sequentially, as highlighted in Figure 2. The corresponding SFILES substring is |(A)(M1)(M2)(h_ex). We use a vertical bar to indicate that we start from a raw material (e.g., |(A) in Figure 2). Every time we move to a node that was previously visited and used in the SFILES string, we append a number to the current generated string and insert a symbol "<#" at the position where the node was used. The less-than symbol is used to specify the

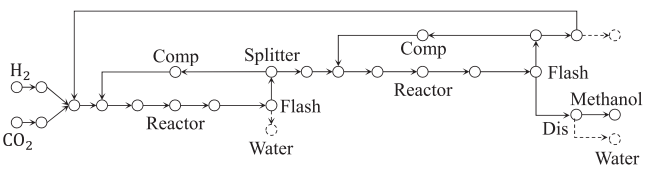


Figure 4. Flowsheet for methanol production process and corresponding directed graph representation.

flow direction. The number ("#") counts the times that we arrive at a node that has been visited.

Canonical SFILES String Notation Generation. For a given flowsheet, the SFILES string can be automatically generated with a graph traversal from the raw material. Generating the SFILES string of a flowsheet is challenging as there are multiple ways to construct the SFILES string of a given flowsheet. By following various sequences through the units in the flowsheet, different SFILES strings can be written. The Morgan algorithm[11] is a well-known and widely used method for determining the unique order of the atoms in molecules. In this article, we implement the same concept to help us generate a unique SFILES string for a given flowsheet. For molecules, other methods for determining the canonical ordering[12,13] are typically used instead of the Morgan algorithm to deal with symmetries. Although SFILES is similar to SMILES, unlike molecules, chemical process flowsheets rarely have topological symmetries. Additionally, since our order is based on domain-knowledge (labels), it eliminates symmetries, thus making it unnecessary to utilize these algorithms.

SFILES String Canonicalization. The Morgan algorithm systematically assigns a unique number to each process unit in a flowsheet based on its connectivity. Thus, the numbering is dependent only on intrinsic properties of the flowsheet. Before we introduce the algorithm, one key concept must be illustrated: connectivity value. Connectivity value is defined as the number of nodes connected to a specific vertex. If a vertex has three connections, its connectivity value is, by definition, three. The first step of the Morgan algorithm is to number each process unit in our graph representation by its connectivity value. For example, in Figure 5a, each node representing a process unit is labeled by its connectivity value in the first step. Then, the connectivity value of a process unit is updated by the sum of the connectivity values of its neighbors (Figure 5b). We repeat this step iteratively until the number of different connectivity values is maximized. However, finding the maximum number of different connectivity values is difficult. In our implementation, we terminate the calculation if the number of different connectivity values does not increase after 10 iterations. For example, Figure 5d shows that the number of different connectivity values of this graph stays at 10 after several iterations. Thus, we terminate the calculation. After the calculation, we update the labels of nodes with their final connectivity values in the graph representation. When we traverse the graph to generate the SFILES string, we visit the next node and select from the unvisited successors with the smallest connectivity value (Figure 5d).

SFILES String Generation. The breadth-first search and the depth-first search are two basic algorithms for exploring a graph. In this work, we traverse the graph with the depth-first search algorithm that has the complexity $\mathcal{O}(|V|+|E|)$, where $|V|$ is the number of nodes and $|E|$ is the number of arcs in the graph.

The canonical SFILES string generation is composed of the following steps. First, assign an attribute for each vertex in the graph according to its final connectivity value calculated using the Morgan algorithm. Next, beginning at a node that represents raw material, traverse the graph by the depth-first search. At the same time, append the name of visited nodes to the generated string. The SFILES string of the given flowsheet is generated after the graph traversal is completed.

SFILES String for Pattern Mining. Recall that the objective of this study is to develop a technology capable of identifying process patterns in flowsheets. The name tagged to the

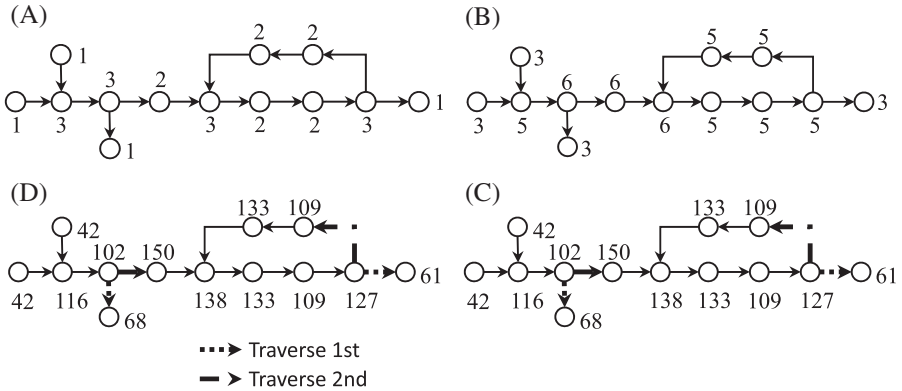


Figure 5. Labeling the nodes with connectivity values.

process unit in the graph representation should not influence the result. For example, a reactor named as (R1) and a reactor named as (R2) should be matched, since both of them represent a reactor. Thus, it is crucial for us to generate SFILES strings according to the type of unit operation a node belongs. In Table 1, we define a set of conventional process units used in this study. These process units represent frequently used unit operations in process synthesis. The SFILES string used for pattern mining of the flowsheet in Figure 2 is the following sequence:

|(raw)(pp)(m)<2(m)<1(h_ex)(r)(h_ex)[(f)(prod)](f)[(s)(pp)(m)1](s)(prod)|(raw)(pp)(m)2

Specifically, the symbol (A) in the original SFILES string signifies a node that represents the raw material. By referring to the process unit attributes in Table 1, we should use (raw) in the SFILES string for process pattern mining. Thus, in the final SFILES string, we replace the raw material (A) by (raw).

Sequence alignment

String comparison has been studied in the field of computational linguistics[14] and has found many applications in bioinformatics.[15] As SFILES strings preserve the connection information of process units, we can identify process patterns by comparing two flowsheets in string format with sequence alignment algorithms. Other methods[16,17] cannot be used directly to identify flowsheet patterns because we compare two directed multigraphs for pattern recognition. Many sequence alignment methods[15,18] based on dynamic programming have been proven to be highly practical and popular over the last few decades. The Needleman-Wunsch algorithm, for instance, is a global alignment technique that attempts to align every element in a given sequence to an element in another sequence. The Smith-Waterman alignment algorithm is a local alignment method that aligns arbitrary length segments or substrings of each sequence. An important characteristic of alignment algorithms is whether or not they utilize a penalty for the unaligned portion of the sequence. The local algorithm lets the alignment start at any arbitrary position in the sequence. In this case, two empty spaces are introduced to the second sequence and the aligned subsequence starts at the third place. Both alignment algorithms have polynomial time complexity.

Table 1. Process Unit Attribute Table

Process Unit	Attribute	Process Unit	Attribute
Mixer	(m)	Splitter	(s)
Compressor	(comp)	Pump	(pp)
Valve	(v)	Heat exchanger	(h_ex)
Reactor	(r)	Flash	(f)
Distillation column	(dis)	Absorption column	(ab)
Extraction column	(ex)	Stripping column	(str)
Crystallization	(cry)	Evaporation	(ev)
Refrigeration	(re)	Storage	(t)
Raw material	(raw)	Product	(prod)

Smith-Waterman Local Alignment Algorithm. As a local alignment algorithm, the Smith-Waterman algorithm[18] finds the longest similar subsequence of two sequences. The optimal aligned subsequences are identified by maximizing the sum of the scores of aligned element pairs, while minimizing the total cost of mismatched pairs. The key word "local" means that we choose arbitrary length segments of each sequence to be matched in this algorithm. Additionally, the alignment begins at an arbitrary position in the sequences.

The Smith-Waterman alignment algorithm is a dynamic programming algorithm. At each recursion step, we add new matched element pairs to the aligned subsequences from the last step. The added element pair is determined by the calculation of the scoring function (also known as recursion function). We recursively find the best-aligned element pair, and the longest aligned subsequence corresponds to the one with the largest alignment score.

Consider the case of matching a m-character sequence x with a n-character sequence y. Three elementary string operations are allowed in the Smith-Waterman algorithm: inserting a character, deleting a character, and substituting a character. These string operations correspond to three element pairs in Table 2. For element pair (x_i, y_j), either a match score is associated if two elements are matched or a mismatch score is associated representing the cost of substitution. The match function $S_{i,j}$ returns the alignment cost of aligning pair (x_i, y_j). If x_i is matched with y_j, function $S_{i,j}$ returns a positive match score; otherwise, $S_{i,j}$ returns a negative mismatch score. For the pairs $(x_i, -)$ and $(-, y_j)$, a gap penalty represents the cost of inserting a null character "–." The inserted null character represents a missing character or a deletion of the character in the original sequence.

Suppose, we are comparing two subsequences: $x_1 \ldots x_i$ from sequence x and $y_1 \ldots y_j$ from sequence y. The subsequences $x_1 \ldots x_{i-1}$ and $y_1 \ldots y_{j-1}$ are already best aligned from the previous recursion step. To find the optimal alignment result of this recursion step, we calculate the alignment score by considering four different cases: x_i is aligned with y_j, x_i is not aligned with y_j, x_i is matched with a null character "–" inserted in y, and y_j is matched with a null character inserted in x. The first two cases

Table 2. Element Pairs and String Operations

Element Pair	Match	Substitution (Mismatch)	Insertion	Deletion	Alignment Cost
(x_i, y_j)	✓	✓			$S_{i,j}$
$(x_i, -)$			✓	✓	Gap
$(-, y_j)$			✓	✓	Gap

can be interpreted as matching element pair (x_i, y_j), while the last two cases correspond to matching $(x_i, -)$ and $(-, y_j)$, respectively. The maximal alignment score is determined by adding the alignment score of $(x_1 \ldots x_{i-1},\ y_1 \ldots y_{j-1})$ to the cost of matching for each of the four cases separately and then picking the largest outcome.

All-Local Alignment Algorithm. The Smith-Waterman local alignment algorithm identifies the longest aligned subsequence. However, it cannot align multiple smaller subsequences that represent certain process patterns. Therefore, we introduce the all-local alignment algorithm to help us find multiple local aligned subsequences beyond a certain length. This algorithm also runs in $\mathcal{O}(mn)$ time.

The all-local alignment algorithm is similar to the Smith-Waterman alignment algorithm but does not allow the insertion of null characters in sequences, forcing aligned subsequences to be strictly matched. In this algorithm, we create a scoring matrix of size $m \times n$ in which we arrange our sequences x and y to perform the alignment. Every entry in this scoring matrix is initialized at zero. In the second step, the all-local alignment algorithm uses the following scoring function

$$M(i,j) = \begin{cases} 0, & \text{if } x_i \neq y_j \\ \max\{S(i,j),\ M(i-1,\ j-1) + S(i,j)\}, & \text{if } x_i = y_j \end{cases} \quad (1)$$

Here, the alignment value is determined by match and mismatch costs only. If x_i cannot be aligned with y_j, we have a mismatch score of zero at position (i, j). If two characters can be matched $(x_i = y_j)$, we examine the previous characters x_{i-1} and y_{j-1}. If previous characters x_{i-1} and y_{j-1} have already been aligned, the alignment score $M(i, j)$ is the sum of the match score at position $(i–1, j–1)$ and the score of matching pair (x_i, y_j). Otherwise, the subsequence alignment starts from pair (x_i, y_j).

For example, consider two sequences ABBBCDEAC and ABBBC, with the match score of one (e.g., $S(i, j) = 1$). In the first step, our scoring matrix is created and zero-initialized. Then, we apply the scoring function (Function 1) to fill the scoring matrix recursively from the upper left cell to the lower right. The filled scoring matrix is illustrated in Table 3.

To find all locally aligned subsequences with a length greater than the threshold value, we need to complete the traceback step. For example, let us assume that we want to find aligned subsequences with a length equal to or greater than two. There are three cells that have alignment scores that satisfy this criterion. Starting from these three cells, we move diagonally in the scoring matrix until we reach the cell with a value of one. By doing so, we find three different paths through the scoring matrix. One path $(1 \leftarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 5)$ represents subsequence ABBBC. The other two paths $(1 \leftarrow 2)$ represent BB. ABBBC is obviously one of the desired results. Now consider the two subsequences BB. As the paths for subsequences BB are next to the path of ABBBC, these aligned results do not count. This is because subsequences BB appear within the rectangle formed by the path $1 \leftarrow 2 \leftarrow 3 \leftarrow 4 \leftarrow 5$. In other words, subsequences BB are already covered by aligned subsequence ABBBC. Other aligned subsequences (e.g., A or C) are discarded as their length is less than two.

Table 3. All-Local Alignment Scoring Matrix

	A	B	B	B	C	D	E	A	C
A	**1**	0	0	0	0	0	0	**1**	0
B	0	**2**	**1**	1	0	0	0	0	0
B	0	**1**	**3**	**2**	0	0	0	0	0
B	0	1	**2**	**4**	0	0	0	0	0
C	0	0	0	0	**5**	0	0	0	**1**

Case Studies

In this section, the proposed methodology is illustrated with case studies of well-known chemical processes.

Comparison of methanol processes

In the first case study, we compare two methanol production process flowsheets. Methanol is made from a wide variety of feedstocks, making it one of the most flexible chemical commodities and energy sources available today. At present, the most popular process of methanol production uses synthesis gas as the raw material. A gas mixture of hydrogen and carbon monoxide is the basis for production. Another popular production method produces methanol by combining hydrogen and carbon dioxide captured from the atmosphere. Figure 6[19] shows two different process flowsheets without heat integration for methanol production.

To identify the similarities of two flowsheets, we first generate their respective graph representations. Every process unit in the flowsheet will be mapped to a node in the graph with an attribute determined by the functionality of the unit (Table 1). For example, the flash drum f1 corresponds to a vertex in the

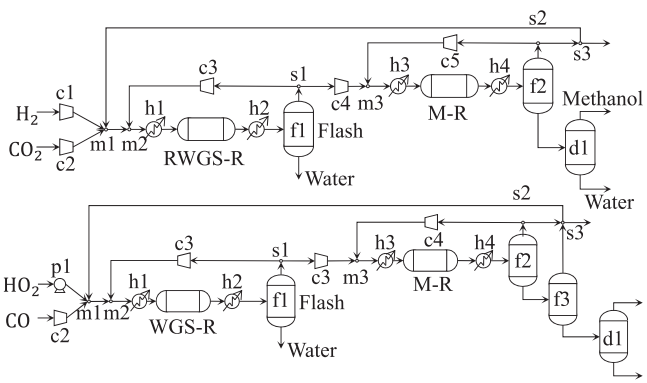


Figure 6. Methanol production process flowsheets. (a) Alignment with −0.5 gap penalty. (b) Alignment by setting gap = − 1, − 1.5, − 2.

graph with an attribute f. Then, for each graph, we generate a simple SFILES string using the names of the nodes. The SFILES string is generated simultaneously as we traverse the graph from the raw material using depth-first search. The SFILES string of the first flowsheet in Figure 6 is shown below

|(raw1)(c1)(m1)<2(m2)<3(h1)(RWGS-R)(h2)[(f1)(w1)]
(f1)[(s1)(c4)(m3)<1(h3)(M-R)(h4)[(f2)[(d1)(product)]
(d1)(w2)](f2)[(s2)(c5)(m3)1](s2)[(s3)(purge)](s3)(m1)2]
(s1)(c3)(m2)3

This SFILES string can be interpreted by the SFILES notation generation rules introduced in the subsection entitled "SFILES string generation." The raw material (raw1) goes through the compressor (c1) and mixer (m1) sequentially, and then proceeds to the next process unit. In this study, we define raw material hydrogen as (raw1). For simplicity, we start the graph traversal from this single source. We can add another source carbon dioxide in our SFILES string to consider multiple sources when we traverse the graph. Similarly, the SFILES string of the second flowsheet is shown below where carbon monoxide is defined as (raw1).

|(raw1)(c1)(m1)<1(m2)<4(h1)(WGS-R)(h2)[(f1)(w1)](f1)[(s1)(c3)
(m3)<2(h3)(M-R)(h4)[(f2)[(f3)[(d1)(product)](d1)(w2)](f3)[(s3)
(purge)](s3)<3(m1)1](f2)[(s2)(c4)(m3)2](s2)(s3)3](s1)(c2)(m2)4

Both SFILES strings are composed of names tagged to nodes in the graph. Using the name of a node will lead to case sensitive problems when performing sequence alignment to identify process patterns. If we change one node's name, the change will lead to different identified pattern results. The sequence alignment algorithms should be applied to SFILES strings according to the functionalities of the nodes. Although process units use different names in the flowsheets, they should be identical if they perform the same unit operation. We need to generate and compare SFILES strings defined by the operations of the nodes when we perform sequence alignment. With operation units defined in Table 1, the first SFILES string for comparing the two flowsheets is

|(raw)(comp)(m)<2(m)<3(h_ex)(r)(h_ex)[(f)(w)](f)[(s)(comp)
(m)<1(h_ex)(r)(h_ex)(f)[(dis)(*prod*)](dis)(w)](f)[(s)(comp)
(m)1](s)[(s)(purge)](s)(m)2](s)(comp)(m)3

The second SFILES string is

|(raw)(comp)(m)<1(m)<4(h_ex)(r)(h_ex)[(f)(w)](f)[(s)(comp)
(m)<2(h_ex)(r)(h_ex)[(f)[(f)[(dis)(prod)](dis)(w)](f)[(s)
(purge)](s)<3(m)1](f)[(s)(comp)(m)2](s)(s)3](s)(comp)(m)4

Smith-Waterman Alignment Results. To identify flowsheet similarities, we compared the SFILES strings using the Smith-Waterman local alignment algorithm. The choice of match/mismatch scores and gap penalty of the algorithm may result in different alignments. In our implementation, the default match and mismatch scores are selected as 2 and − 3, respectively. These scores are used in similar algorithms in the bioinformatics toolkit BLAST. The gap penalty is defined as a constant. We tested a set of gap penalties ($\{-0.5, -1, -1.5, -2\}$) in this study. The alignment results are shown in Figure 7. The first alignment in Figure 7a is obtained by using a gap penalty of −0.5. The remaining gap penalties produced the same alignment that is shown in Figure 7b. To make these alignments more interpretable, we depict matched process units with dotted lines in Figure 8. We highlight the difference between two alignments with blue dotted lines. The black and blue parts together correspond to the alignment in Figure 7a. The black part represents the alignment in Figure 7b. Although using a large gap penalty ($-1, -1.5, -2$) fails to identify the first cycle streams, the major process units are successfully matched with different parameter settings. The objective of the Smith-Waterman

```
(raw) (comp) (m) < 2 (m) < 3 (h_ex) (r ) (h_ex) [ (f) (waste) ] (f) [ (s) (comp) (m) < 1 (h_ex) (r) (h_ex) [ – – (f)
|     |      |   | | |   | | |      |   |      | | |       | | | | |      |   | | |      |   |      |       |
(raw) (comp) (m) < 1 (m) < 4 (h_ex) (r ) (h_ex) [ (f) (waste) ] (f) [ (s) (comp) (m) < 2 (h_ex) (r) (h_ex) [ (f) [ (f)

[ (dis) (prod) ] (dis) (waste) ] – – –  –  – – – – – – (f) [ (s) (comp) (m) 1 ] (s) [ (s) (purge) ] (s) – (m) 2 ] (s) (comp) (m) 3
| |    |      | |   |       |                              | | | |      |   | | |                          | | | |      |   |
[ (dis) (prod) ] (dis) (waste) ] (f) [ (s) (purge) ] (s) < 3 (m) 1 ] (f) [ (s) (comp) (m) 2 ] (s) – –  –  – – (s) – 3 ] (s) (comp) (m) 4
```

(A) Alignment with −0.5 gap penalty

```
(raw) (comp) (m) < 2 (m) < 3 (h_ex) (r ) (h_ex) [ (f) (waste) ] (f) [ (s) (comp) (m) < 1 (h_ex) (r) (h_ex) [ – – (f)
|     |      |   | | |   | | |      |   |      | | |       | | | | |      |   | | |      |   |      |       |
(raw) (comp) (m) < 1 (m) < 4 (h_ex) (r ) (h_ex) [ (f) (waste) ] (f) [ (s) (comp) (m) < 2 (h_ex) (r) (h_ex) [ (f) [ (f)

[ (dis) (prod) ] (dis) (waste) ] (f) [ (s) (comp) (m) 1 ] (s) [ (s) (purge) ] (s) (m) 2 ] (s) (comp) (m) 3
| | |    |   | |      |   | | | |
[ (dis) (prod) ] (dis) (waste) ] (f) [ (s) (purge) ] (s) < 3 (m) 1 ] (f) [ (s) (comp) (m) 2 ] (s) (s) 3 ] (s) (comp) (m) 4
```

(B) Alignment by setting gap $= -1, -1.5, -2$

Figure 7. Pairwise sequence alignment for methanol production process using Smith-Waterman alignment algorithm.

Table 4. All-Local Alignment Results from Comparison of Methanol Flowsheets

SFILES Substring	All-Local Alignment
(s)(comp)(m)	5
(comp)(m) < #	2
(f)[(s)(comp)(m)	2
(h_ex)(r)(h_ex)[(f)	2
(raw)(comp) < #	1
(m)1]	1
(m)2](s)	1
[(s)(purge)](s)	1
(waste)](f)[(s)	1
(waste)](f)[(s)(comp)(m)	1
[(f)[(dis)(prod)](dis)(waste)](f)[(s)	1
(h_ex)(r)(h_ex)[(f)(waste)](f)[(s)(comp)(m) < #	1

algorithm is to maximize the overall alignment score. Therefore, a large gap penalty results in fewer inserted null characters. To identify long aligned subsequences, we use −0.5 as our default gap penalty.

The alignment reveals that the two flowsheets are similar. In Figure 7a, aligned elements in substrings in the top block represent process units in the main material stream. In both methanol flowsheets, raw material moves sequentially through the mixer, heat exchanger, reactor, another heat exchanger and flash drum (... (m)(h_ex)(r)(h_ex)[(f)...). In the bottom block,

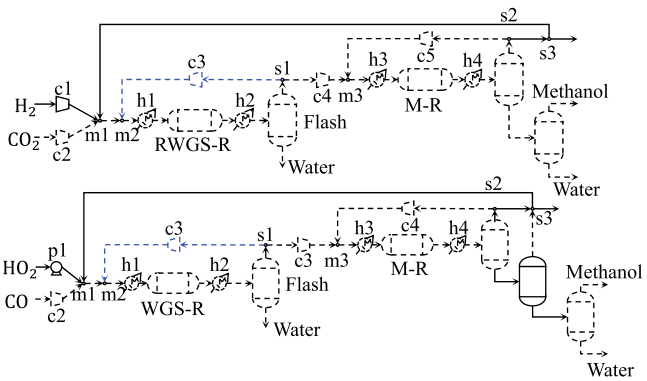


Figure 8. Comparison of methanol flowsheets by the Smith-Waterman alignment algorithm.
[Color figure can be viewed at wileyonlinelibrary.com]

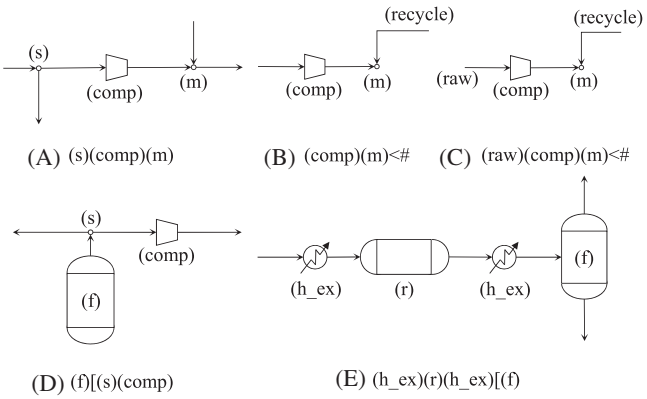


Figure 9. General process patterns in methanol flowsheets found by the all-local alignment algorithm.

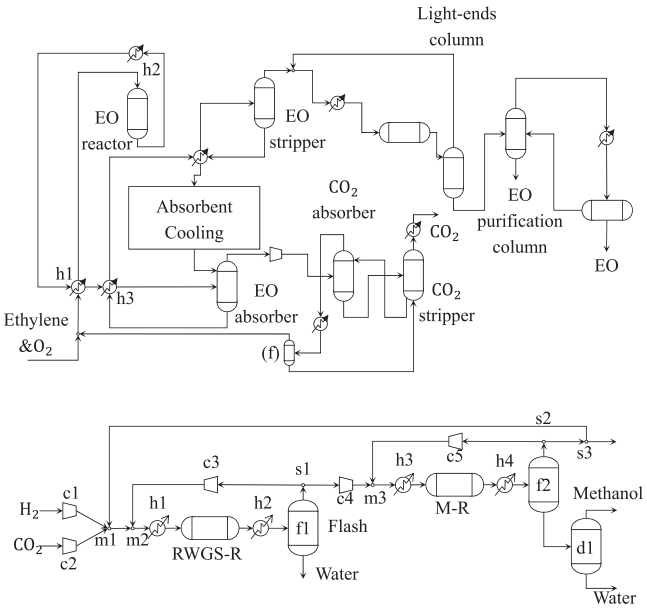


Figure 10. Ethylene oxide production process flowsheet and methanol production process flowsheet.

the aligned subsequences (s)(comp)(m)# represent two recycle streams in the flowsheets.

All-Local Alignment Results. We used the all-local alignment algorithm introduced previously to find multiple strictly aligned subsequences in the SFILES strings. Alignment results are listed in Table 4. In Figure 9, we interpret and show the first five patterns using the process diagrams. For example, the aligned subsequence (s)(comp)(m) indicates that material flows through a splitter (s), a compressor (comp) and a mixer (m) sequentially. The corresponding process pattern thus depicted is shown in Figure 9a. We can easily identify patterns in Figure 9 from the original flowsheets in Figure 6. Aligned subsequences in Table 4 represent common process patterns in the flowsheets, but these substrings are not found by the Smith-Waterman algorithm.

Comparison of methanol process to ethylene oxide process

Our proposed methodology can also be applied to process flowsheets that produce different chemical products. General process patterns are identified by comparing flowsheets for the production of different products. In Figure 10, the first flowsheet produces ethylene oxide using pure oxygen, while the second flowsheet produces methanol.

The ethylene oxide process flowsheet contains more reactors and distillation columns than the methanol process. Additionally, the ethylene oxide process has many recycle streams and a heat integration network built into it. The original and converted SFILES strings of the methanol process have been

Table 5. Smith-Waterman Alignment Results by Comparing Methanol Process with Ethylene Oxide Process

(m)	<	3	(h_ex)
|	|	|	|
(m)	<	7	(h_ex)
(m)	<	1	(h_ex)
|	|	|	|
(m)	<	7	(h_ex)

introduced in the previous section. The original SFILES string of the ethylene process is

|(raw)(m1)<2[(h1)<1(r1)(h2)(h1)1[(h3)<5[(eo_ab)(comp1)[(co_ab)<3 (h4)[(f)(m1)2](f)(co_str)<4(co_ab)3](co_ab)(co_str)4](eo_ab)<9 (h3)5[(h5)<8[(eo_str)(m2)<7(h6)(t1)[(lec)(eo_pur)<6[(t2)(prod)] (t2)(eo_pur)6](lec)(m2)7](eo_str)(h5)8(cool)(eo_ab)9]]]

The corresponding converted SFILES string is

|(raw)(m)<2[(h_ex)<1(r)(h_ex)(h_ex)1[(h_ex)<5[(ab)(comp)[(ab)<3 (h_ex)[(f)(m)2](f)(str)<4(ab)3](ab)(str)4](ab)<9 (h_ex)5[(h_ex)<8[(str)(m)<7(h_ex)(t)[(dis)(dis)<6[(t)(prod)] (t)(dis)6](dis)(m)7](str)(h_ex)8(re)(ab)9]]]

These two flowsheets of the different processes are dissimilar. By comparing the ethylene oxide process flowsheet with the methanol process flowsheet, two matched substring pairs were found with the Smith-Waterman algorithm, as shown in Table 5. Both aligned substring pairs depict the same process patterns in which the material goes through a mixer and is then preheated by a heat exchanger before entering the next process unit. The mixer in this pattern mixes the input material with a recycled material.

By applying the all-local alignment algorithm, two process patterns in the flowsheets were identified (Figure 11). The second pattern is used more frequently in the flowsheets.

The first pattern in Figure 11a is easy to interpret. Before entering a distillation column, materials should either be cooled down to the dew point or preheated to the bubble point, since saturated feed is the most widely used feed condition.

In order for direct oxidation process to produce ethylene oxide, per pass conversion must be low. Recycling unreacted reactants is economically advantageous. Although the conversion is higher for methanol production, it is still viable to save operating costs by recycling critical materials. To ensure the reactants are well mixed, we should consider using mixers in the design. For these reasons, we have the pattern in Figure 11b.

General process patterns

144 Comparisons of 18 Flowsheets. To extract general process patterns, this case study uses pairwise comparison to explore 18 flowsheets covering nine diverse chemical products. These nine products, ranging from organic to inorganic chemicals, are listed in Table 6. Two different flowsheets are

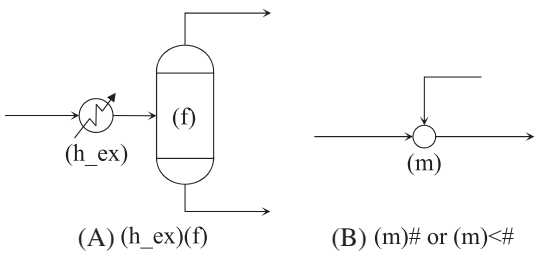


Figure 11. All-local alignment results from comparing methanol process with ethylene oxide process.

Table 6. Chemical Products Used for Case Studies

Product		
Nitrogen acid	Ethylene	Ethylene oxide
Methanol	Propane	Urea
Benzene	Phenol	Cumene

selected for each chemical product. Instead of comparing them directly, we perform cross pairwise comparisons. In 144 cross comparisons, we find 103 physically meaningful matched substring pairs using the Smith-Waterman local alignment algorithm. In Table 7, we list the most frequently aligned subsequences and their frequencies. As the SFILES string describes the connection structure of the flowsheet, the aligned common substrings depict the interconnection patterns of operations. For instance, "(m) < #" means that recycle streams and crude inlet streams are mixed together before being sent to the next process unit in the flowsheet. Other process patterns in Table 7 can be interpreted in a similar manner. These listed patterns consistently appear in case studies that we present in more detail below.

By applying the all-local alignment algorithm, we find 163 aligned substring pairs corresponding to 19 flowsheet patterns. The most frequent patterns are listed in Table 7. According to our observations, the all-local alignment algorithm supplements the results identified by the Smith-Waterman algorithm.

With the Smith-Waterman alignment algorithm and the all-local alignment algorithm, we identified multiple heat transfer patterns for reactions in this study (Figure 12). The identified patterns reflect heuristic rules or engineering practice. The heat transfer patterns in Figures 12c–e reflect that chemical engineers interchange heat in flowsheets in practice.[20] For moderate exothermic or endothermic heats of reaction, the patterns suggest that engineers should consider the use of intercoolers or interheaters with utilities (Figures 12a–c). For reactions that release a large amount of heat, the identified pattern suggests that it is more efficient to use reaction products to preheat raw materials in the process (Figure 12e). In the next section, where we add working attributes to process units, we explicitly show that this pattern is used for material preheating.

Comparisons with Working Attributes. In the previous case studies, process units in the graph representation are assigned function attributes. These function attributes are defined by the names of the unit operations. To further describe process unit functionalities and their working conditions, we add an additional attribute to each process unit in the graph representation. Attributes that describe working

Table 7. General Process Patterns by Smith-Waterman Alignment

SFILES Substring	Smith-Waterman	All-Local Alignment
(m) < #	37	35
(r) < #	11	22
(dis)(dis)	7	40
(f)(dis)	7	8
(dis)(r)	6	7
(h_ex)(f)	3	20
(dis)(dis)(dis)	2	2
(h_ex)(r)	11	–
(h_ex)(r)(h_ex)	5	4
(r)(h_ex) #	–	8
(h_ex) < #(r)(h_ex)	–	3
(h_ex) < #(r)(h_ex) #	14	1

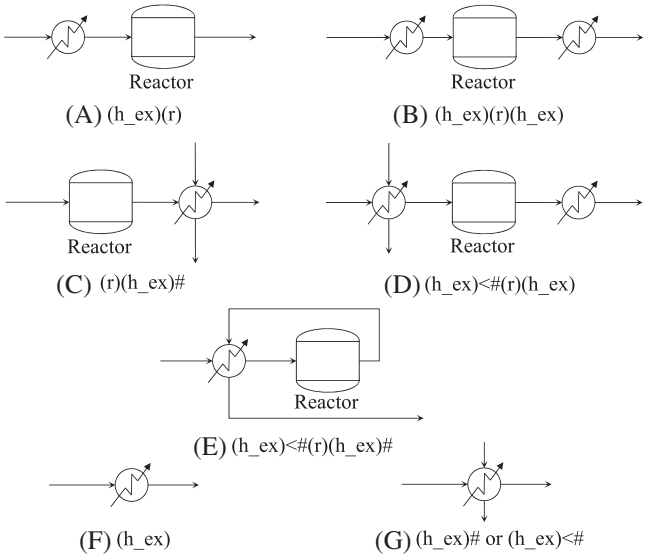


Figure 12. General heat transfer patterns for reactions. (a) First separation classification. (b) First separation classification.

conditions are assigned to process units and defined in the form of verbs in Table 8 (i.e., heat exchangers, which heat process materials are attached with label [heat]). Heat exchangers that cool process materials are assigned the attribute (cool).

In Table 8, the flash and distillation column are given the same attribute (dis). We use (dis) to illustrate that both operating units perform the distillation task. In the same way, we define absorption, extraction, and stripping as (extract). We categorize these three separation tasks as extraction operations to indicate that solvents are introduced into the system to "extract" key components from the mixture (Figure 13a).

Different classification criteria can be applied when we design working attributes. We can also categorize separation tasks by referring to the material phases in equipment. Since stripping, absorption, flash, and distillation are two-phase unit operations (gas–liquid), we group them together as (Dis). However, extraction is a liquid–liquid phase separation task, so we categorize it as (extract) (Figure 13b). The classification is not unique. By exploring different material properties, we can develop different classification criteria to compare and cluster separation tasks.

Table 8. Process Unit Working Condition Attribute Table

Operating Unit	New Attribute
Mixer	(m)
Compressor	(comp)
Valve	(v) (cool)
Reactor	(r)
Distillation column	(dis)
Extraction column	(extract)
Crystallization	(cry)
Refrigeration	(re)
Raw material	(raw)
Splitter	(s)
Pump	(pp)
Heat exchanger	(heat) or (cool)
Flash	(dis)
Absorption column	(extract)
Stripping column	(extract)
Evaporation	(ev)
Storage	(t)
Product	(prod)

Smith-Waterman Alignment Results. Once we add working attributes to each process unit in the flowsheets, we compare the flowsheets by referring to the newly added attributes. Using the Smith-Waterman alignment method, 144 substring pairs were found. Some of the aligned subsequences and their frequencies are listed in Table 9. To better interpret the listed process patterns in the SFILES string format, we denote each process unit by using two attached attributes in parentheses. The first attribute before the slash corresponds to the function attribute defined in Table 1. The second attribute after the slash is the working attribute in Table 8.

According to our observations, it is evident that recycling is the most common process pattern across different process flowsheets. In particular, we found three types of recycling patterns: mixers that use recycled material, reactors that use recycled material, and separators that use recycled material streams. Among these three patterns, mixers using recycled material streams and reactors using recycled material are the two most common structures.

All-Local Alignment Results. The all-local alignment increased the patterns identified by the Smith-Waterman

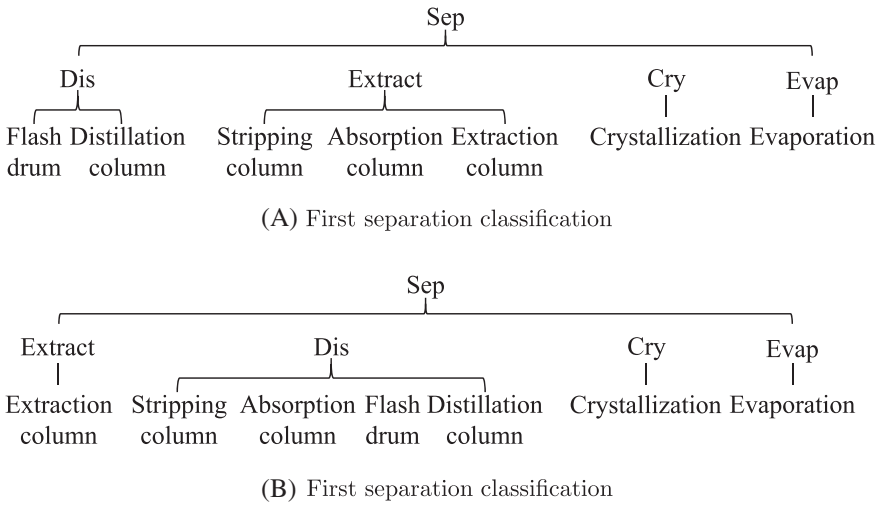


Figure 13. Separation classification.

Table 9. Smith-Waterman Alignment Results by Referring to Working Condition Attributes

SFILES Substring	Frequency
Mixer using recycled material	
(m/m) < #	30
(comp/comp)(m/m) < #	10
(dis/dis)(m/m) #	6
(raw/raw)(m/m) < #	6
(raw/raw)(m/m) < #(pp/pp)[	2
Reactor using recycled material	
(r/r) < #	22
(dis/dis)(r/r) #	6
(h_ex/heat)(r/r) < #	4
(r/r) < # (extract/extract)	2
Separation using recycled material	
(dis/dis) < #(dis/dis)	2
(extract/extract) < #	2
Purify the reactant before reaction	
(dis/dis)(r/r)	14
Heat transfer pattern for reaction	
(raw/raw)(m/m) < #[(h_ex/heat) < #(r/r)(h_ex/cool)	10
(r/r)(h_ex/cool)	10
[(h_ex/heat) < #(r/r)(h_ex/cool)	4
(m/m) < #[(h_ex/heat) < #(r/r)(h_ex/cool)	4
Heat transfer pattern for distillation	
(h_ex/cool)[(f/dis)	12
(h_ex/cool)[(dis/dis)	6
(h_ex/cool)[(f/dis)(dis/dis)	4
Two consecutive distillation tasks	
(dis/dis)(dis/dis)	39
(f/dis)(dis/dis)	23
Three consecutive distillation tasks	
(dis/dis)(dis/dis)(dis/dis)	6
(f/dis)(dis/dis)(dis/dis)	4
(f/dis)(f/dis)(dis/dis)	2
Complex pattern	
(h_ex/heat)(r/r)(h_ex/cool)[(f/dis)(dis/dis)	2
(h_ex/heat)(r/r)(h_ex/cool)[(dis/dis)(dis/dis)	2

algorithm. Using the all-local alignment algorithm, we found 220 matched subsequences with a length greater than three in 144 cross comparisons.

Recycling Patterns. Inserting empty characters is rejected in the all-local alignment algorithm. The all-local alignment algorithm identifies process patterns represented by shorter SFILES subsequences. As a result, we find more recycling patterns when using the all-local alignment algorithm compared to the Smith-Waterman algorithm. The identified process patterns are consistent with previous results in Table 7 which indicate that recycling is the most common pattern across process flowsheets, and (m/m) < # and (r/r) < # are the two most frequent recycling patterns. This outcome reflects the fact that recycling is the most flexible process pattern in flowsheets.

Recycle streams can appear anywhere in a flowsheet, either in a reaction section or a separation section. Most of the reactions in the studied processes are reversible. Reversible reactions cannot proceed to completion. As a result, some reactants will also appear among products. Higher process yields and greater economies will be achieved if we recover some of the unconverted reactants from the reactor effluent and have them undergo further conversion. Thus, reactors using recycled reactants frequently appear in chemical process flowsheets.

In addition to reactors with recycle streams, separation tasks can also make use of recycled materials. In particular, recovering solvents from process streams is economically and ecologically necessary. If a solvent is introduced in the separation system, further separation is required to extract the product from the solvent. Additionally, solvent recovery reduces the need to purchase a new solvent to save money on operating costs. Recycle design is a process step of secondary importance for separations using solvents.

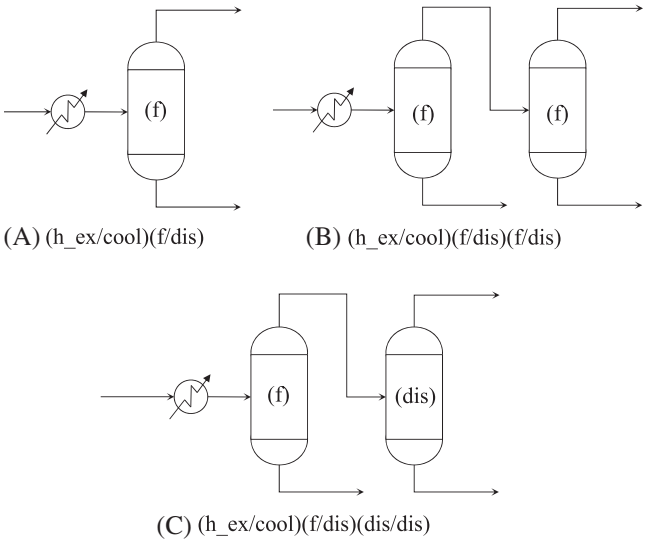


Figure 14. Heat transfer patterns for distillation.

Heating and Cooling. The first heating and cooling pattern we observed relates to the reaction. In the previous case study, we used intercoolers or interheaters. By introducing working attributes in Table 8, the patterns correspond to preheating the raw material before the reaction and cooling the product before the next operation. Reaction temperatures in the studied processes are higher than ambient temperatures. As a result, the raw materials must be preheated before the reaction.

In this case study, we discovered a new process pattern. Chemical products in Table 6 are small molecule compounds. Unreacted raw materials and the postreaction products are at high temperatures and are typically in gas phase. The step after the reaction is the separation task. If we use standard distillation techniques, saturated feed is a favorable feed condition. The mixture from the reactor should be cooled to the dew point before entering the distillation column, especially in cases where the next unit operation is flash. As a result, the flash pattern appears frequently for small molecule compounds. This cooling pattern for the flash task is shown in Figure 14. The cooling patterns of the separation task also help explain our results, in which we have intercoolers after the reactors (Table 10).

Furthermore, our proposed methodology can identify heat transfer patterns of heat exchangers by using utilities

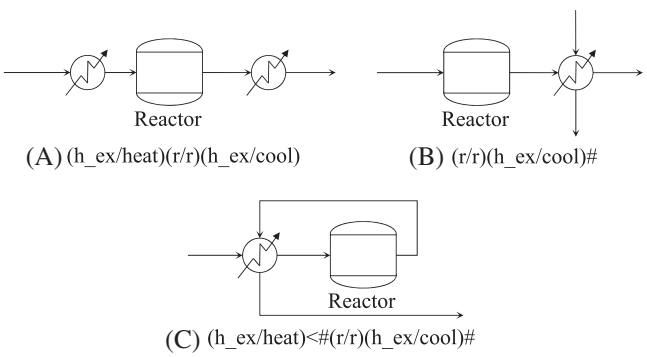


Figure 15. Heat transfer patterns for reaction.

Table 10. Heuristics and Corresponding Aligned Subsequences using All-Local Alignment Algorithm by Referring to the Working Attributes

SFILES Substring	Frequency	Reference
(m/m) < #	19	Use mixer when we have recycle stream (Engineering practice)
(raw/raw)(m/m) < #	9	
(comp/comp) (m/m) < #	5	
(…/dis)(m/m) < #	2	
(m/m) < #(m/m)	1	
(r/r) < #	20	Recycle un-converted reactants (Engineering practice)
(dis/dis)(r/r) #	2	
(h_ex/heat)(r/r) < #	2	
(absorb/extract) < #	3	Recycle material for separation tasks (Engineering practice)
(dis/dis)(dis/dis) < #	3	
(dis/dis) < #	1	
(dis/dis) < # (dis/dis)	1	
(dis/dis)(r/r)	22	Purify the material before reaction (Engineering practice)
(r/r)(h_ex/cool)#	20	Use intercoolers or interheaters for reactions [3, pp. 174–178]
(h_ex/heat)(r/r)(h_ex/cool)	8	
(h_ex/heat) < # (r/r) (h_ex/cool)	6	
(h_ex/cool)(f/dis)	24	Cool down mixture before distillation tasks in processes producing small molecule chemical products (Newly discovered)
(h_ex/cool)(f/dis) (f/dis)	3	
(h_ex/cool)(f/dis) (dis/dis)	1	
(f/dis)(dis/dis)	53	Perform the easiest separation first and delay the complicated separation tasks[21]
(f/dis)(f/dis)(dis/dis)	2	
(f/dis)(dis/dis) (dis/dis)	2	
(dis/dis)(absorb/extract)	1	
(f/dis)(str/extract)	1	
(wash/extract) (dis/dis)	1	

(Figure 15a) or without using utilities (Figures 15b, c). For an exothermic reaction that releases a large amount of reaction heat, using the latent heat of reaction products to preheat the reactant is frequently observed in practice (Figure 15c). This pattern is consistent with the pattern in Figure 12f, and the matched subsequence in Figure 15c justifies that the pattern in Figure 12f is for material preheating.

Distillation Sequence. Process designers are responsible for selecting suitable separation techniques and sequences. Therefore, they should apply different separation tasks to different mixtures according to material properties. This approach results in a variety of distillation sequences.

Table 11. Process Patterns of Distillation Sequence

Pattern	SFILES Substring	Frequency
(…/dis)(…/dis)	(dis/dis)(dis/dis)	120
	(f/dis)(dis/dis)	53
	(f/dis)(f/dis)	10
	(dis/dis)(f/dis)	1
(…/dis)(…/dis)(…/dis)	(dis/dis)(dis/dis)(dis/dis)	10
	(f/dis)(f/dis)(dis/dis)	2
	(f/dis)(dis/dis)(dis/dis)	2
	(f/dis)(dis/dis)(f/dis)	1
	(dis/dis)(f/dis)(dis/dis)	1

For two consecutive distillation tasks, we identified four different combinations: (f)(f), (dis)(dis), (f)(dis), and (dis)(f). For three consecutive distillation tasks, we identified five different process patterns that are listed in Table 11. The diversity of distillation sequences indicates that distillation sequence design is not a trivial task in chemical process design. Factors such as mixture properties and quantities will influence the structure, and the result is case sensitive. Thus, designers should exercise caution when extending existing separation sequences to their designs.

If we take these patterns a step further, we find that structures in which easier separations precede complicated separations frequently appear among the identified distillation sequences. Pattern (f/dis)(dis/dis) appears 53 times while (dis/dis)(f/dis) occurs only once. This observation reflects a heuristic rule for designing separation processes that we discuss next.

Separation Heuristic Rule. In our case study, several connection patterns in Table 10 reflect a top-ranked general separation heuristic rule recommended by [2, pp. 400–401] for determining distillation sequences. This heuristic rule suggests performing the easiest separation task first and delaying the complicated separation tasks. The identified patterns justify the implementation of this general heuristic rule in different processes of real world designs. Therefore, when designing separation tasks, it is recommended to start at the easiest separation before applying complicated techniques.

Conclusions

A chemical process flowsheet is a set of process units linked together to convert raw materials to commercial chemical products. For a given process flowsheet, the type of unit operations and their sequence are determined. By analyzing operation sequences, we can identify process patterns that correspond to preferred engineering feasibility and economics. In this work, we developed the first systematic methodology for mining process patterns. The proposed flowsheet pattern mining methodology contains three automated steps: (1) graph generation for process flowsheets; (2) SFILES string generation; and (3) process pattern identification.

We applied our methodology to many chemical process flowsheets to mine process patterns. We first compared flowsheets that produced the same chemical product with two sequence alignment algorithms. We assessed the similarity between two flowsheets and discovered structure features that characterize the processes. Our methodology was subsequently illustrated through the comparison of flowsheets that represent the production of different chemical products. In a more extensive case study, we identified many frequent structure patterns hidden in 18 flowsheets with 144 cross comparisons. In our results, some process patterns reflect the heuristic rules defined

in the literature, while other patterns reflect chemical engineering practice. We further extended our methodology by introducing working attributes. Additionally, we discovered a heat exchange flash pattern in the processes that produce small molecule chemical products. The aforementioned case studies reveal that our proposed methodology allows significant flexibility for customization.

Potential extensions of our work may focus on three aspects: (1) adding user-defined rules during the SFILES generation step to take practical factors into consideration, (2) implementing multiple sequence alignment algorithms to explore common patterns among flowsheets simultaneously, and (3) extending Table 1 by further classifying unit operations.

Literature Cited

1. Tula AK, Eden M, Gani R. Process synthesis, design and analysis using a process-group contribution method. *Comput Chem Eng.* 2015;81:245-259.
2. Biegler L, Grossmann I, Westerberg A. *Systematic methods for chemical process design*. Old Tappan, NJ: Prentice Hall; 1997.
3. Seider WD, Seader JD, Lewin DR. *Product & process design principles: Synthesis, analysis and evaluation*. Hoboken, NJ: Wiley; 2009.
4. Kondili E, Pantelides CC, Sargent RWH. A general algorithm for short-term scheduling of batch operations–I. MILP formulation. *Comput Chem Eng.* 1993;17:211-227.
5. Yeomans H, Grossmann IE. A systematic modeling framework of superstructure optimization in process synthesis. *Comput Chem Eng.* 1999;23:709-731.
6. Friedler F, Tarjan K, Huang YW, Fan LT. Combinatorial algorithms for process synthesis. *Comput Chem Eng.* 1992;16:S313-S320.
7. Szlama A, Heckl I, Cabezas H. Optimal design of renewable energy systems with flexible inputs and outputs using the P-graph framework. *AIChE J.* 2015;62:1143-1153.
8. Farkas T, Lelkes Z. Process flowsheet superstructures: structural multiplicity and redundancy: part I: basic GDP and MINLP representations. *Comput Chem Eng.* 2005;29:2180-2197.
9. d'Anterroches, L. Group contribution based process flowsheet synthesis, design and modelling, Ph.D. thesis. Technical University of Denmark, 2006.
10. Wu WZ, Henao CA, Maravelias CT. A superstructure representation, generation and modeling framework for chemical process synthesis. *AIChE J.* 2016;62:3199-3214.
11. Morgan HL. The generation of a unique machine description for chemical structures–A technique developed at chemical abstracts service. *J Chem Doc.* 1965;5:107-113.
12. Weininger D, Weininger A, Weininger JL. SMILES. 2. Algorithm for generation of unique SMILES notation. *J Chem Inf Comput Sci.* 1989; 29:97-101.
13. Schneider N, Sayle RA, Landrum GA. Get your atoms in order–an open-source implementation of a novel and robust molecular canonicalization algorithm. *J Chem Inf Model.* 2015;55:2111-2120.
14. Levenshtein VI. *Binary codes capable of correcting deletions, insertions, and reversals*. Vol 10; 1966:707-710.
15. Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol.* 1970;48:443-453.
16. Ullmann JR. An algorithm for subgraph isomorphism. *J ACM.* 1976; 23:31-42.
17. Xie W, Sahinidis NV. A reduction-based exact algorithm for the contact map overlap problem. *J Comput Biol.* 2007;14:637-654.
18. Smith TF, Waterman MS. Identification of common molecular subsequences. *J Mol Biol.* 1981;147:195-197.
19. Kim J, Johnson TA, Miller JE, Stechel EB, Maravelias CT. Fuel production from CO_2 using solar-thermal energy: system level analysis. *Energ Environ Sci.* 2012;5:8417-8429.
20. Umeda T, Harada T, Shiroko K. A thermodynamic approach to the synthesis of heat integration systems in chemical processes. *Comput Chem Eng.* 1979;3:273-282.
21. Seader JD, Westerberg AW. A combined heuristic and evolutionary strategy for synthesis of simple separation sequences. *AIChE J.* 1977; 23:951-954.

Manuscript received Sep. 20, 2018.