Korean J. Chem. Eng., **25**(3), 402-408 (2008)

Synthesis of nonsharp distillation sequences *via* genetic programming

Xiao-Hong Wang[†], **Yang-Dong Hu* and Yu-Gang Li***

College of Chemistry and Chemical Engineering, Ocean University of China, QingDao, 266100, Shandong, China
*QingDao University of Science and Technology, QingDao, 266042, Shandong, China

(*Received 2 April 2007 • accepted 23 August 2007*)

Abstract−This paper addresses the application of Genetic Programming (GP) to the synthesis of multicomponent product nonsharp distillation sequences. Combined with the domain knowledge of chemical engineering, some evolutionary factors are improved, and a set of special encoding method and solving strategy is proposed to deal with this kind of problem. The system structural variable is optimized by GP and the continuous variable is optimized by the simulated annealing algorithm simultaneously. Because GP has an automatic searching function, the optimal solution can be found including distillation, splitting, blending and bypassing operations automatically without any superstructures of nonsharp distillation sequences. Three illustrative examples are presented to demonstrate the effective computational strategies.

Key words: Non-sharp Distillation Sequences, Genetic Programming, Synthesis

INTRODUCTION

The synthesis of optimal distillation sequences with pure products has been studied very deeply over many years. Segovia-Hernandez et al. [1] gave a comparative analysis of the control properties for the Petlyuk column, and two alternate structures with unidirectional interconnecting flows of the vapor or liquid interconnecting streams were presented, which could provide better theoretical controllability properties than the Petlyuk system. Kim et al. [2] devised an industrial scale hexane process for the implementation of a fully thermally coupled distillation column (FTCDC), and a semi-rigorous material balance and Peng-Robinson equilibrium relation were utilized in the structural design, which was good for a system of many components found from actual field applications. Computer modeling and some comparative works were performed to obtain highly pure dimethyl sulfoxide (DMSO) which was used for fiber spinning solvent for two different distillation sequences by Cho and Kim [3]. The importance of the rigorous rate based model that was the non-equilibrium approach was demonstrated for a typical extractive distillation process in a Glitsch V-1 valve tray column by Pradhan and Kannan [4].

But according to the actual demand of industrial production, it is more significative for separations of a single or several multicomponent feeds into several specified multicomponent products using distillation, splitting, blending and bypassing operations simultaneously. Nath [5] and Muraki [6] studied the synthesis of nonsharp distillation sequences with one feed and multiple multicomponent products by using the material allocation diagram (MAD). Liu et al. [7] improved the individual material allocation diagram (IMAD) to solve these kinds of problems with multi-feed. Bamopoulos et al. [8] proposed a two-stage approach based on the component recovery matrix(R-matrix)which is an algebraic extension of MAD, but only limited stream splitting is permitted in this study. Cheng and Liu [9] devised some techniques for the systematic synthesis of initial sequences for nonsharp separations with heuristics. Liu and Xu [10] improved the separation product matrix transforming method to solve the non-sharp distillation process with multi-feed; then, the two-stage procedure for the synthesis of nonsharp distillation flow was brought forward [11]. Hu et al. [12] gave an ordered heuristics method for the synthesis of nonsharp distillation sequences with simple, sharp distillation columns. Wehe and Westerberg [13] described an algorithmic procedure for the synthesis of sharp distillation sequences with bypass. Aggarwal and Floudas [14] addressed the process including heat integrated nonsharp columns with MINLP method.

In this paper, a new stochastic optimization algorithm-Genetic Programming (GP) [15] is proposed for the synthesis of multicomponent product nonsharp distillation sequences. GP can express the complex net structure of this kind of problem with its hierarchy and can determine the feasible solving area automatically without giving any superstructures. So, the global optimum can be searched automatically in the genetic evolutionary process. Because other algorithms do not possess the above features, a set of solution strategies based on GP is suggested here.

GP is widely used in many fields as an evolutionary-computational method. An improved GP to facilitate the generation of steady-state nonlinear empirical models for process analysis and optimization was proposed by Grosman et al. [16], which could adjust the complexity of the required model to accurately predict the true process behavior. Madar et al. [17] gave a method that used GP to generate nonlinear input-output models of dynamical systems that were represented in a tree structure, and this method resulted in more robust and interpretable models. Venkatraman et al. [18] used GP in quantitative structure activity relationship (QSAR) analysis that required an objective variable relevance analysis step for producing robust classifiers with low complexity and good predictive accuracy. Hinchliffe et al. [19] adopted multi-objective GP to evolve dynamic process models. A modelling of hot yield stress curves that were difficult to describe math for carbon silicon steel by GP

[†]To whom correspondence should be addressed.
E-mail: wxhlee@sohu.com

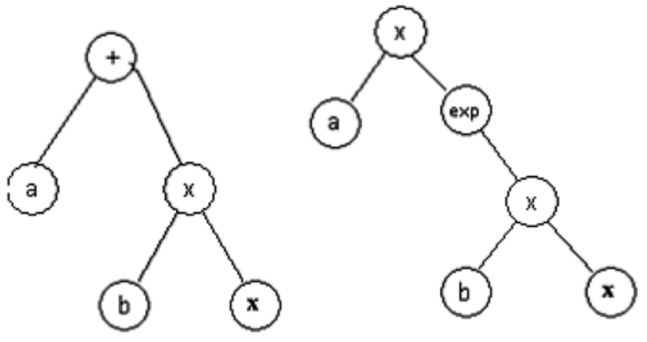

Fig. 1. Dendriform structure of expressions.

was carried out by Kovacic et al. [20].

The principle and operations of GP are very similar to the genetic algorithm (GA), which starts its calculation from initialized populations and generates new populations by using a variety of evolution operators to achieve the optimal solution according to the biological principles of natural selection. The difference between the two methods in nature is that GA uses the length-fixed character string to represent problems, while GP adopts the dendriform code to express complex structure directly. So, the evolutionary operations of GP can also proceed on its dendriform code directly. To give an example that is illustrated in Fig. 1, the following two mathematical expressions can be represented with particular dendriform structures of GP: (1) $y=a+b\times x$ (2) $y=a\times\exp(b\times x)$.

GP code is a kind of dendriform non-linear data structure, not a string or array structure. According to the concept of data structure, GP code is a tree composed by some nodes and branches. The evolution operations of GP can proceed in the dendriform structure of GP code directly, which includes the operations of reproduction, crossover and mutation for a single node or a whole branch. When GP code is used to represent the complex distillation sequence, the branches of GP dendriform structure can be defined as streams, and the nodes can be defined as unit operations (for example column, splitter and blender). So, the combination of all nodes and branches for GP code can be used to express the actual nonsharp distillation structure directly. We can know from the above description that although GP is a domain independent optimization algorithm, it needs domain knowledge to instruct the definition for its tree-like code. Meanwhile, evolution operations need to proceed in the relative professional domains to assure the effectiveness and accuracy of the algorithm. So, combined with a professional knowledge of chemical engineering separation, the definition for GP tree-like code and the evolutionary operations is specially restricted in this work in order to get an efficient optimization method for nonsharp distillation system synthesis.

PROBLEM STATEMENT

The problem to be studied in this paper can be described as follows. A single feed stream of N-component mixture is given with the known conditions (i.e., composition, flow rate, temperature and pressure). The problem is to synthesize an optimal separation flow that can separate the multicomponent mixture into the specified multicomponent product streams. The configurations can involve series and/or parallel arrangements of distillation columns, as well as stream splitting, blending and bypassing. To be convenient for expression and calculation, the following assumptions are introduced in this work:

(1) Each distillation column performs a “sharp” separation.

(2) The maximum number of separators is $N-1$.

(3) Components in any stream are ranked according to their relative volatility order. To give an example, a stream containing N components is arranged as $H_1, H_2, \ldots, H_N$, where H_1 is the most volatile (lightest) component, H_2 is the next lightest, and so forth down to the heaviest of all the components H_N.

(4) Heat integration among the distillation columns is not permitted.

(5) The separation flow is optimized in order to minimize the following simple nonlinear objective function: $\sum_{i=1}^{n-1}(L_iK_i)^{0.6}$ [10], where L_i and K_i are, respectively, the separation mass load and the difficulty degree of the *i*th component separation, where $K_i=1/\ln\alpha$.

GP-BASED ENCODING STRATEGY

1. Nonsharp Distillation Sequence Encoding

In order to express the positions and the connection relationships of distillation, splitting, blending and bypassing operations in GP tree-like code, three kinds of nodes (column node, splitter node and blender node) are defined in this paper. Some relevant properties are set for these nodes to represent their features, respectively, and the definition can be stated as follows:

1-1. Design for Column Node

Because the information for the stream cut (separation) position (i.e., assignation for the light and heavy key component) is a very important property for column node that determines the separation mission of this distillation column, we use the corresponding ordinal number of the light key component to represent it. For example, integer one denotes the light key component (i.e., No 1 component) and the heavy key component is the adjacent one to the light key component for a sharp separation column. It requires $N-1$ columns to separate an N-component mixture when only conventional columns with sharp separation are adopted. So, it corresponds to $N-1$ different integers to represent cut property, and the relationship between the property and the corresponding separation mission is as follows.

$$H_1\underset{1}{|}H_2\underset{2}{|}H_3\cdots H_i\underset{i}{|}H_{i+1}\cdots H_{N-1}\underset{N-1}{|}H_N \tag{1}$$

Feed composition is other important information for column node. It is represented with two integers in this paper: one of them denotes the lightest component in the feed stream and the other indicates the heaviest component, which both being expressed by the corresponding component’s ordinal number.

It is specified that the information of stream cut position and feed composition is randomly selected for every column when the GP code is produced in this paper; therefore, the randomization of this algorithm can be assured.

1-2. Design for Splitter Node

As everyone knows, a splitter can divide an inlet stream into several outlet streams that have the same compositions and physical properties but different flow rates. Similar to the definition for column node, two kinds of information are needed to express the properties for splitter node, too. First, two integers are used to describe the information of feed composition, and the definition method for the two integers is the same as for the column node that is mentioned above. Further information for splitter node is the definition of the distribution coefficient for every outlet stream that denotes the ratio of an outlet stream flow rate to the inlet stream flow rate. In order to assure the regularity of GP tree-like code, we limit that every splitter can only divide an inlet stream into two outlet streams in this paper. A real number is used to indicate the distribution coefficient of one outlet stream: meanwhile, the distribution coefficient of another outlet stream is confirmed automatically, because the sum of the two distribution coefficients must be 1.

1-3. Design for Blender Node

A blender is very important in such nonsharp separation sequence. If the definition for it is random, that is, every blender is encoded randomly just as the same as that of the other two kinds of nodes, it will appear randomly at any position of GP code. Then, the GP code may become too complex and disordered to possess high validity. Moreover, it will result in the failure to express the mixed operation of any two or multiple feeds into the same blender. In order to express the special netlike structure of a nonsharp separation procedure, a special encoding method for blender node is adopted in this paper, which sets all the blenders of GP code to be arranged in a kind of grid framework. The maximum layer number for the grid framework is usually confirmed according to the scale of every detailed problem. Meanwhile, the maximal number of blenders in every layer for the blender grid framework is defined to the number of products. In order to denote the position of each blender in the framework, two integers are figured as the property of the blender node. One of them denotes the layer number and the other indicates the ordinal number of the blender in this layer. The last layer of blender grid framework is named the terminal products blender. In order to produce the prescribed mixture products, all streams of GP code will be collected to the several terminal products blenders, so the number of terminal products blender must be the same as that of the products.

2. Generating GP Code

The whole GP code is based on the above-mentioned blender grid structure and every blender in it can be as a sub-node from which a sub-tree can be generated via growing downward randomly. Thus, many sub-trees can be formed according to the same principle. When the whole GP code reaches the maximum depth limited by the algorithm, all the branches will be connected with several terminal blender nodes randomly to produce different mixture products and the whole algorithm tree ceases growth. That is, the whole GP tree-like code is formed finally, based on the growth of sub-tree layer upon layer, which can be used to express the complex netlike structure of nonsharp distillation sequence.

The detailed process for generating GP code includes the following steps. First, a column node or a splitter node is selected randomly as GP code tree's root node that expresses the original feed can be separated by a distillation column or be divided by a splitter. If a column node is selected as the root node, which corresponds to a distillation column in the actual flow, there are two branches: a right branch and a left branch for the node. The right branch corresponds to the top stream of the column that includes the components lighter than the light key component, while the left branch corresponds to the bottom stream that includes the components heavier than the heavy key component. If a splitter node is selected as the root node for a GP code, two branches (left branch and right branch) are also generated that correspond to the same compositions and physical properties but different flow rates of the two outlet streams for this splitter, respectively. Once the root node for GP tree is confirmed, no matter if it is a column node or a splitter node, the connection relation for both the left branch and right branch with the following nodes should be expressed. It is specified in the algorithm that the next sub-node can be randomly selected from three types of nodes, including column node, splitter node and blender node. Then, every sub-node connects to the next sub-node produced randomly, that results in each branch of any node extending downwards continuously. The connection proceeds alternately until all branches reach the GP code tree's maximum deepness, which has been defined in this paper; then all branches of the GP code are randomly connected to the different terminal product blenders to produce the specified product streams. Now, the growth of the GP tree stops, which means a whole GP tree-like code has been achieved, just like Fig. 2 (symbol S-splitter node, symbol C-column node, symbol M-blender node).

During the GP code forming process, if any branch of any subnode is confirmed randomly to be connected to a blender node, a pair of property parameters of this blender node can be used to indicate the position of this blender node. That is, which layer and which column among the blender grid framework the blender is located in. It means that any node of any sub-tree in GP code has the possibility of connecting to any blender node in the blender grid structure. Thus, any blender has the possibility to mix multiple streams from different sub-trees. Then, this blender node will be as a sub-root node and a corresponding sub-tree downwards grows randomly based on it.

The coding method can be used to express the bypass operation successfully. When a splitter node is confirmed as the root node of a GP tree-like code randomly and a terminal products blender of the GP code is just selected by one of the two branches of this splitter

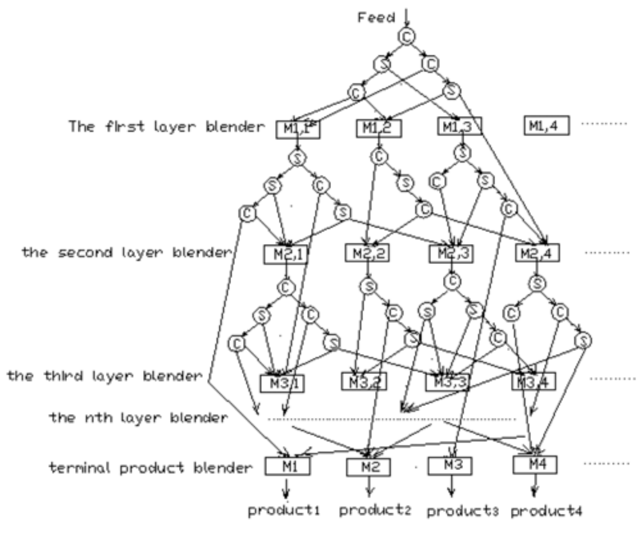

Fig. 2. Netlike structure of GP code.

node directly, it means that a part of the original feed can be sent to the products blender directly without being separated by a distillation column. This connecting way can be used to express the bypass operation for the original feed stream. Meanwhile, for any sub-node located anywhere in the whole GP tree-like code, when one branch of it is connected to the terminal products blender directly, it also means that a part of inlet stream for this node can be sent to the terminal products blender directly without being separated by a distillation column, and the bypassing operation for this middle stream can be realized.

3. GP Evolutionary Operations Based on Chemical Engineering Domain Knowledge

In a similar way to GA, GP also needs the operations of reproduction, crossover and mutation to achieve the evolutionary optimization process.

3-1. Reproduction Operation

Reproduction operation is used to regenerate the optimal members from the population of one generation to the next generation and eliminate the bad members. The athletics selection method [15] of GP does not need to evaluate the fitness of every member in one generation, which obviously speeds up the evolutionary velocity of the population. So, this method is adopted to execute the reproduction operation in this paper.

3-2. Crossover Operation

Crossover operation is used to exchange the part genes of two members of GP code to produce two new members and it is divided into two classes in this work, i.e., the crossover of a node and the crossover of a node property.

When the crossover of a node proceeds, according to the special feature of multicomponent product nonsharp distillation sequence, it is specified in this work that two column nodes can exchange to each other or two splitter nodes can exchange to each other as well as a column node and a splitter node can exchange to each other too, but a column node or a splitter node cannot be crossed to a blender node, and two blender nodes cannot be crossed to each other also. If two allowed nodes would be crossed, in order to guarantee the validity of this operation, it is specified that the two nodes must have the same feed composition. The judgment method is to compare whether the feed composition information contained in the two nodes is the same or not. When the crossover of a node proceeds, the contents of the crossover operation include not only the node itself but also all properties of the node. Meanwhile, both the node itself and the two branches of this node partake in the crossover operation. Here, not every blender node can partake in the crossover operation in order to protect the blender grid frame.

When the crossover operation is aimed at a node property, it is defined that only corresponding properties of the two homogeneous nodes are crossed while the node itself remains changeless.

Sameness: every blender node cannot partake in the crossover of node property in order to protect the blender grid frame.

To ensure the randomization of this algorithm, stochastic selection is adopted in this paper to determine the types of the crossover operation (i.e., the crossover of a node or the crossover of a node property).

3-3. Mutation Operation

According to the characteristic of the nonsharp distillation process, it is specified in this work that the column node itself and the splitter node itself can participate in the mutation operation and they can mutate to each other too. When the mutation operation is for column node or splitter node itself, it is limited in that the mutation operation cannot change the feed information for the node but can do mutation for the information of the stream cut.

The mutation operation method for a column node or splitter node is that the left and right sub-nodes of the selected node are deleted first. Then, the information of the stream cut is randomly changed again in the range of the feed information included in this node. Lastly, the new left and right branches of this node are generated according to the new information of stream cut that can assure the mutated code being valid. Here, the new information of stream cut for column node is to assign the light and heavy key components randomly again, and for the splitter node it is to generate the distribution coefficient randomly again. When the column node and the splitter node mutate to each other, the feed information for every node must be kept fixed first and the new left and right branches will be produced according to the new stream cut information (or distribution coefficient) of this new node.

Sameness, the types of mutation operation (i.e., the mutation operation for homogeneous node or the mutation operation for different type node) is randomly specified here in order to assure the diversity of GP code.

To sum up, GP tree-like code can be used to represent every individual of the population directly and the evolution operations can proceed in the code forthwith, not like the GA using binary character to represent separation flow. So, the improved GP-based algorithm can be used to describe various netlike structures of the nonsharp separation process directly and may discover more unimaginable flow structures.

4. Calculation Steps for the GP-Based Synthesis Algorithm

The GP-based synthesis algorithm includes the following steps:

(1) Generate initial population randomly as described in 4.2. Because too many nodes and blender layers can lead to the GP code being excessively complex, and even possibly lead to the failure of the calculation of this algorithm fail, the maximal depth of the GP algorithm tree is set to 7. As a result, the sum of total nodes is limited in order to assure the success of this algorithm. Meanwhile, the maximal blender layer number is set to 5, which includes the last layer of the terminal product blender. On the other hand, the population size is set to P and the maximal evolutionary generation is set to W, where P and W will be assigned according to every actual problem.

(2) The simulated annealing (SA) algorithm [21] is a well established technique for optimization problems that cannot be represented by simple and explicit functions. Typically, the SA algorithm simulates the physical process of annealing, i.e., melting a solid by increasing its temperature, followed by slow cooling and crystallization to a minimum free energy state. SA may be viewed as a randomization device that allows some wrong-way movements during the course of the optimization, through an adaptive acceptance/rejection criterion.

In this paper, the SA algorithm is used to optimize the continuous variable, the distribution coefficient of every splitter for every GP code, while the GP algorithm is used to optimize discrete variables, the separation flow structure. First, we give an initial value of the distribution coefficient for every splitter; then, the optimal solution can be found generally by properly adjusting the change

Table 1. Data for Example 1

Flow rate (kmol·h^{-1})	A Propane	B Isobutane	C n-Butane	D Isopentane	E n-Pentane	Quantity
Feed	45.36	136.08	226.8	181.44	317.52	907.2
Product1	20	100	100	150	100	470
Product2	25.36	36.08	126.8	31.44	217.52	437.2
Separation	C_1	C_2	C_3	C_4		
(T=350 K) K=1/lnα	1.19	3.66	1.26	4.72		

step size and iteration number for the value of the distribution coefficient.

The GP-based strategy improved in this paper can deal with both discrete and continuous variables simultaneously.

(3) Translate every GP code into its corresponding nonsharp distillation flow and get the separation mass load for every distillation column node. Then, the fitness of every GP code can be calculated with the formula: $\sum_{i=1}^{n-1}(L_iK_i)^{0.6}$ given in section 3.

(4) The operations of reproduction, crossover and mutation are performed as described in 4.3 where the reproduction rate is set to 15%, the crossover rate is set to 50% and the mutation rate is set to 8%.

(5) Steps (2) to (4) are repeated until the fitness does not change in the last two generations or the number of the maximal generation is reached, which means the algorithm should be terminated.

EXAMPLES

The calculation software for the GP-based synthesis algorithm is programmed by using C++ language, and the effectiveness of the algorithm is validated by the following three typical nonsharp distillation problems on a Pentium 4/3.0 G computer. The data are given in Tables 1-3 and the synthesis is to find the optimal separation flow that can minimize a simple nonlinear objective function: $\sum_{i=1}^{n-1}(L_iK_i)^{0.6}$ given in section 3.

Table 2. Data for Example 2

Flow rate (kmol·h^{-1})	A	B	C	D	Quantity
Feed	10	32	20	48	110
Product1	2	3.2	2	4.8	12
Product2	1	6.4	0	4.8	12.2
Product3	5	6.4	2	19.2	32.6
Product4	0	9.6	10	14.4	34
Product5	2	6.4	6	4.8	19.2
Separation	C_1	C_2	C_3		
K=1/lnα	1.2	1.5	3.0		

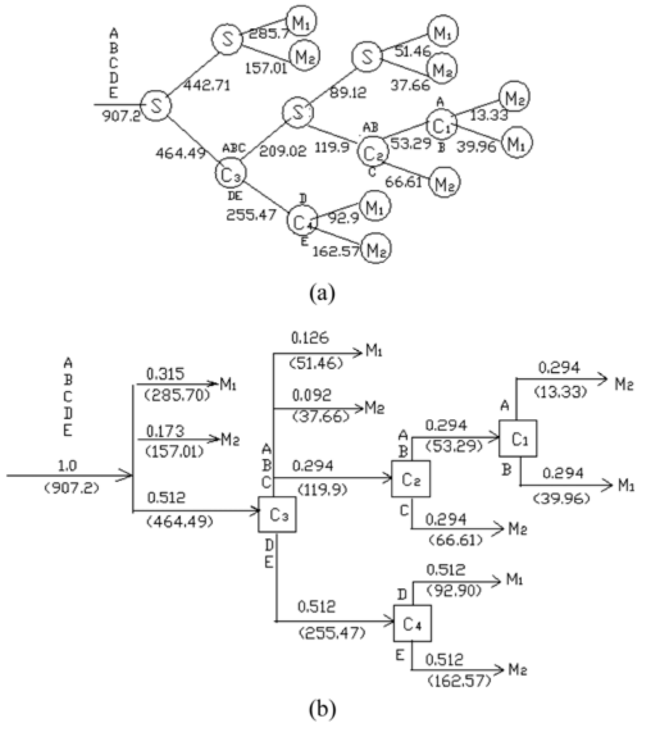

Fig. 3. (a) GP code for Example 1 (numbers are stream flow rates); (b) Configuration for Example 1 (Fractions are relative flow rates; numbers in brackets are stream flow rates).

Table 3. Data for Example 3

Flow rate (kmol·h^{-1})	A Propane	B Isobutane	C n-Butane	D Isopentane	E n-Pentane	F n-Hexane	G n-Heptane	Quantity
Feed	15	20	25	35	30	45	30	200
Product1	7	4	10	15	12	20	15	83
Product2	3	8	7	10	9.4	15	10	62.4
Product3	5	8	8	10	8.6	10	5	54.6
Separation	C_1	C_2	C_3	C_4	C_5	C_6		
(T=350 K) K=1/lnα	1.19	3.66	1.26	4.72	1.04	1.08		

Example 1 is the second example of Liu and Xu [10]. The population size for GP is set to 50, while the maximal evolutionary generation is set to 40. The GP code for the optimal solution obtained in this work is illustrated in Fig. 3(a) and the corresponding optimal flow configuration is expressed in Fig. 3(b), which is different from the one of Liu and Xu [10]. In the optimal flow obtained by us, all middle blenders are cancelled by the GP-based algorithm and the M_i expresses the *i*th product terminal blender in Fig. 3(a). From Fig. 3(b), we know that the parallel C_2 and C_4 separations are executed simultaneously after C_3 separation, and the separation mass load L of C_1, C_2, C_3 and C_4 severally is 53.29, 119.9, 464.49 and 255.47, respectively. From Table 1, we know that the separation difficulty degree K between AB, BC, CD and DE components, respectively, is 1.19, 3.66, 1.26 and 4.72. So, the objective function of this work is calculated by the formula $\sum_{i=1}^{4}(L_iK_i)^{0.6}=166.9$ and that of Liu and Xu [10] is 172.9. It is evident from the values of the objective function that the sequence obtained in this work is better than that of Liu and Xu [10]. The optimal configuration is found at the evolutional generation of 30 and the time for CPU is about 600 s.

Example 2 is the third example of Liu and Xu [22]. The size of the population for GP is set to 60, while the maximal evolutionary generation is set to 50. The GP code for the optimal solution and its corresponding optimal flow configuration obtained in this work are illustrated in Fig. 4(a) and (b) severally. The same as Example 1, all middle blenders are cancelled by the GP-based algorithm for the optimal solution and the M_i expresses the *i*th product terminal blender in Fig. 4(a). The separation sequence is the same as the one of Liu and Xu [22], but they did not give the detailed separation flow. From Fig. 4, we know that the separation mass load L of C_1, C_2 and C_3 severally is 16.8, 77 and 27.2, respectively. From Table 2, we know that the separation difficulty degree K between AB, BC and CD components, respectively, is 1.2, 1.5 and 3.0. So, the objective function of the optimal separation configuration obtained in this work is calculated by the formula $\sum_{i=1}^{3}(L_iK_i)^{0.6}=37.4$; the optimal solution is found at the 40th GP evolutional generation, and the time for CPU is about 900 s.

Example 3 is a seven-component three-product nonsharp separation problem that is made by us. The size of the population for GP is set to 80, while the maximal evolutionary generation is set to 60. The GP code for the optimal solution and its corresponding optimal flow are explained in Fig. 5(a) and (b). Sameness, no middle blenders are found and the M_i expresses the *i*th product terminal blender in Fig. 5(a). It is difficult to find such a complex flow by the algorithm of experiential rules. However, the GP-based synthe-

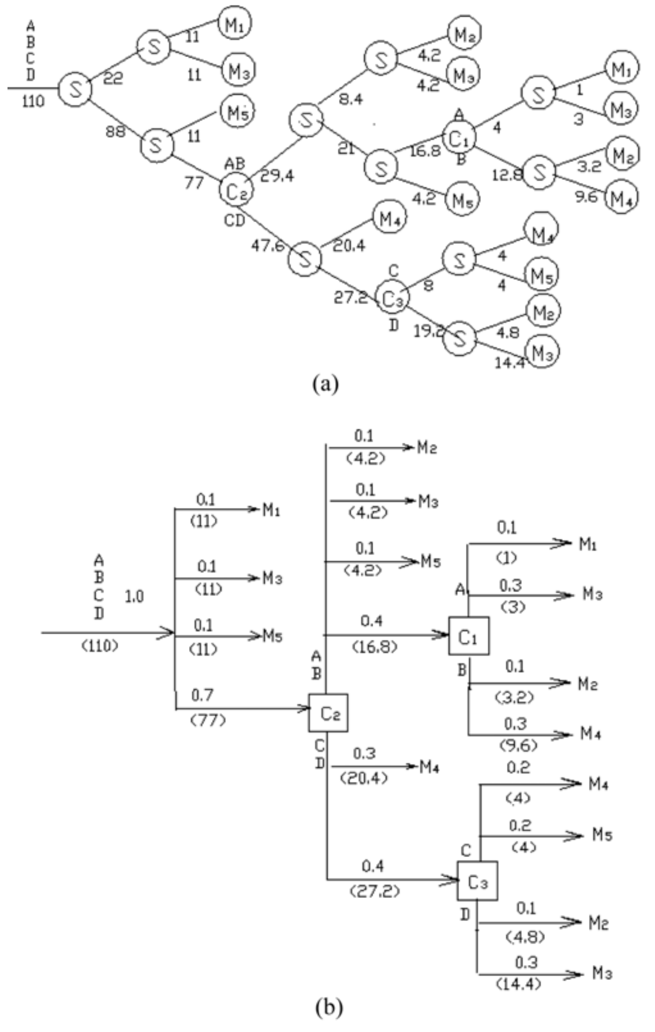

Fig. 4. (a) GP code for Example 2 (numbers are stream flow rates); (b) Configuration for Example 2 (Fractions are relative flow rates; numbers in brackets are stream flow rates).

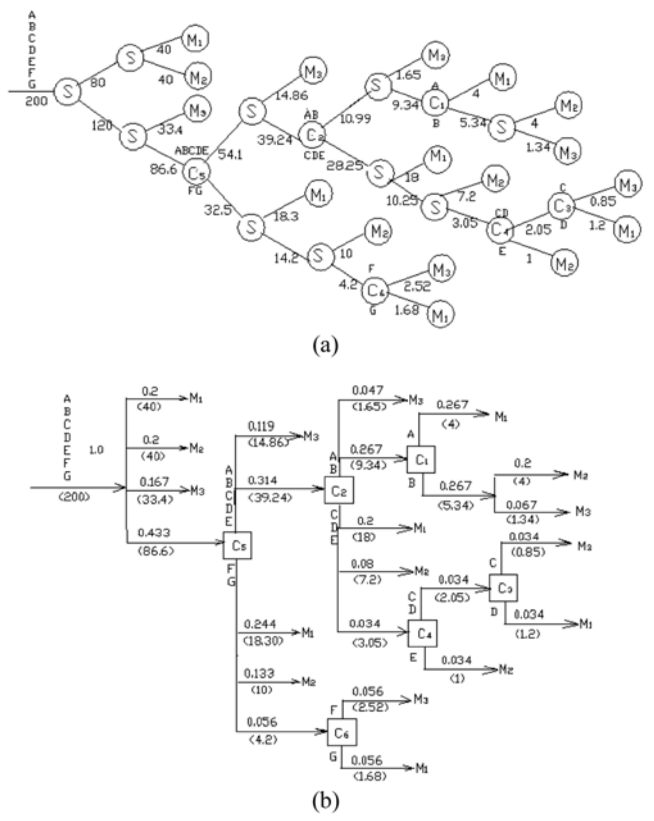

Fig. 5. (a) GP code for Example 2 (numbers are stream flow rates); (b) Configuration for Example 3. Fractions are relative flow rates; numbers in brackets are stream flow rates.

sis algorithm proposed in this work can find the optimal solution at the 45th GP evolutional generation within about 1,200 seconds without giving any superstructure.

From Fig. 5, we know that the separation mass load L of C_1, C_2, C_3, C_4, C_5 and C_6 severally is 9.34, 39.24, 2.05, 3.05, 86.6 and 4.2, respectively. From Table 3, we know that the separation difficulty degree K between AB, BC, CD, DE, EF and FG components, respectively, is 1.19, 3.66, 1.26, 4.72, 1.04 and 1.08. So, the objective function of the optimal separation configuration determined in this work is calculated by the formula $\sum_{i=1}^{6}(L_iK_i)^{0.6}=48$.

CONCLUSION

A new stochastic optimization algorithm, Genetic Programming (GP), is proposed for the synthesis of multicomponent product nonsharp distillation sequences. Based on the special blender grid framework, a complex nonsharp distillation netlike structure is devised and can be expressed directly by using GP's special hierarchical configuration. The synthesis of such separation sequences is to minimize a simple nonlinear objective function that is mentioned above. In order to assure the proceeding validity of this algorithm, several evolutionary factors are improved according to the domain knowledge of chemical engineering. While GP is used to optimize the structural variable, SA algorithm is used to optimize the continuous variable simultaneously. It is shown by the computation results that the proposed GP-based synthesis method can automatically solve the optimization synthesis problem of nonsharp distillation sequence including distillation, splitting, blending and bypassing operations quickly and effectively.

NOMENCLATURE

H_i	: the *i*th component
K_i	: difficulty degree of the *i*th component separation
L_i	: separation mass load of the *i*th separation column
N	: component number for mixture
P	: size of the population for GP
W	: maximal number of evolutionary generation for GP
α	: relatively volatilization degree

REFERENCES

1. J. G. Segovia-Hernandez, A. Bonilla-Petriciolet and L. I. Salcedo-Estrada, *Korean J. Chem. Eng.*, **23**, 689 (2006).
2. Y. H. Kim, K. S. Hwang and M. Nakaiwa, *Korean J. Chem. Eng.*, **21**, 1098 (2004).
3. J. Cho and D. M. Kim, *Korean J. Chem. Eng.*, **24**, 438 (2007).
4. S. Pradhan and A. Kannan, *Korean J. Chem. Eng.*, **22**, 441 (2005).
5. R. Nath, *Studies in the synthesis of separation process*, Ph.D. dissertation, University of Houston, TX (1977).
6. M. Muraki and T. Hayakawa, *Chem. Engng. Sci.*, **43**, 259 (1988).
7. Z.-Y. Liu, Z.-S. Guo, et al., *Journal of Chemical Industry and Engineering (China)*, **45**, 321 (1994).
8. G. Bamopoulos, R. Nath, et al., *AIChE J.*, **34**, 763 (1988).
9. S. H. Cheng and Y. A. Liu, *Ind. Engng. Chem. Res.*, **27**, 2304 (1988).
10. Z.-Y. Liu and X.-E. Xu, *Chem. Engng. Sci.*, **50**, 1997 (1995).
11. Z.-Y. Liu, L.-N. Hu, et al., *Journal of Chemical Engineering of Chinese Universities*, **13**, 66 (1999).
12. Z. Hu, B. Chen and X. He, *Comput. Chem. Eng.*, **17**, 379 (1993).
13. R. R. Wehe and A. W. Westerberg, *Comput. Chem. Eng.*, **11**, 619 (1987).
14. A. Aggarwal and C. A. Floudas, *Comput. Chem. Eng.*, **16**, 89 (1992).
15. J. R. Koza, *Genetic programming: On the programming of computer by means of natural selection*, Cambridge: The MIT Press (1992).
16. B. Grosman and D. R. Lewin, *Comput. Chem. Eng.*, **28**, 2779 (2004).
17. J. Madar, J. Abonyi and F. Szeifert, *Ind. Engng. Chem. Res.*, **44**, 3178 (2005).
18. V. Venkatraman, A. R. Dalby and Z. R. Yang, *Journal of Chemical Information and Computer Sciences*, **44**, 1686 (2004).
19. M. P. Hinchliffe and M. J. Willis, *Comput. Chem. Eng.*, **27**, 1841 (2003).
20. M. Kovacic, M. Brezocnik and R. Turk, *Materials and Manufacturing Processes*, **20**, 543 (2005).1
21. S. Kirkpatrick, D. D. Gelatt and M. P. Vecchi, *Science*, **220**, 671 (1983).
22. Z. Y. Liu and X. E. Xu, *Chem. Engng. Res. Des.*, **73**(Part A), 13 (1995).