*Engineering Optimization*
Vol. 00, No. 00, Month 200x, 1–21

## RESEARCH ARTICLE

# A rewriting grammar for heat exchanger network structure evolution with stream splitting

Eric S. Fraga
Centre for Process Systems Engineering
Department of Chemical Engineering
University College London (UCL)
Torrington Place
London WC1E 7JE
United Kingdom
telephone: +44 20 7679 3817
fax: +44 20 7383 2348
email: e.fraga@ucl.ac.uk

(*Received 00 Month 200x; final version received 00 Month 200x*)

The design of cost optimal heat exchanger networks is a difficult optimisation problem due both to the nonlinear models required and also the combinatorial size of the search space. When stream splitting is considered, the combinatorial aspects make the problem even harder. This paper describes the implementation of a two level evolutionary algorithm based on a string rewriting grammar for the evolution of the heat exchanger network structure. A biological analogue of genotypes and phenotypes is used to describe structures and specific solutions respectively. The top level algorithm evolves structures while the lower level optimises specific structures. The result is a hybrid optimisation procedure which can identify the best structures including stream splitting. Case studies from the literature are presented to demonstrate the capabilities of the novel procedure.

**Keywords:** heat exchanger network design; stream splitting; term rewriting; formal grammar; hybrid optimisation

## 1. Introduction

A process plant may have large cooling and heating demands. For instance, a popular technology for separating liquid mixtures is distillation. A distillation unit operates by boiling liquid at the bottom of the unit and condensing vapour at the top. Meeting the heating and cooling requirements in a plant can involve large amounts of utilities, such as steam and cooling water. Besides the obvious economic impact, there is also a significant environmental impact from utility consumption. Therefore, it is beneficial to reduce utility consumption whenever possible.

Utility consumption can be reduced by using excess heat in one part of a process plant to meet the heating requirements elsewhere in the same process plant, subject to the laws of thermodynamics. Using heat in this way is known as *process integration*. Identifying the optimum integration between all the processing units in a process plant, including the design of the necessary equipment for heat transfer, is known as the *heat exchanger network synthesis* (HENS) problem.

The definition of a HENS problem is a set of cold streams, a set of hot streams and the set of utilities available for meeting any cooling and heating demands not satisfied by integration. Mathematically, the aim is to minimise, for instance,

ISSN: 0305-215X print/ISSN 1029-0273 online

DOI: 10.1080/0305215YYxxxxxxx
http://www.informaworld.com

an annualised cost for meeting the heating and cooling requirements of a process plant taking into account not only the utility consumption but also the cost of the necessary exchanger equipment.

As an optimisation problem, all possible integrations must be considered. This is a combinatorial problem and is particularly challenging when we allow for streams to be split so that, for instance, a hot stream may exchange heat with two cold streams in parallel. Previous attempts at solving the full heat exchanger network synthesis problem with stream splitting have been based on the *a priori* definition of a superstructure (Aaltola 2002, Wei *et al.* 2004) or through the use of multi-level encodings in stochastic methods (Pariyani *et al.* 2006, Lewin 1998).

A recent review of the large body of work for heat exchanger network design (Furman and Sahinidis 2002) does indicate the need for methods which can handle stream splitting. This paper, therefore, presents the design and implementation of a hybrid procedure for the generation of heat exchanger networks with stream splitting, allowing for by-passes and parallel heat exchangers. The hybrid procedure consists of a structure generation level, with structures evolved through a formal string rewriting grammar which encapsulates all possible useful structures, and an inner level which solves a nonlinear programme (NLP) for each structure generated to determine the actual exchanges for the given structure.

### 1.1. *Case study 1: 4SP, a simple illustrative problem*

A simple case study illustrates the presentation of the novel evolutionary procedure for heat exchanger network design. The case study appears in the literature as the 4SP problem (Pariyani *et al.* 2006). It consists of two cold streams and two hot streams, as shown in Table 1. The capital cost of heat exchangers for integrated exchanges and for cooling with utility is $6250 + 83.26A$ and the capital cost for exchanges with hot utility is $6250 + 99.91A$, where $A$ is the area for the actual exchange of heat in $m^2$. Although, in this example, these cost models are relatively simple, typically the cost models are a function of a non-integral power of the area and can be described more generally by

$$c = \alpha + \beta A^{\gamma}$$

where $\alpha \geq 0$ denotes a fixed cost and $\gamma \in (0, 1]$, with $\gamma$ often given a value close to $\frac{2}{3}$.

The area of an exchanger is estimated by

$$A = \frac{Q}{U \, \mathrm{LMTD}}$$

where $Q$ is the amount of heat to exchange in kilowatts (kW), $U$ is the effective heat transfer coefficient ($\frac{kW}{m^2 \, K}$), and LMTD is the *log*-mean temperature difference of the two streams involved, an estimate of the driving force:

$$\mathrm{LMTD} = \frac{\Delta T_{\mathrm{in}} - \Delta T_{\mathrm{out}}}{\log \Delta T_{\mathrm{in}} - \log \Delta T_{\mathrm{out}}}$$

where $\Delta T_{\mathrm{in}}$ and $\Delta T_{\mathrm{out}}$ are the temperature differences at the two ends of the exchanger. Even for this simple case study, although the cost correlations as a function of the area are relatively simple, they are non-convex. Also, the area calculations pose problems for optimisers due to the calculation of the driving force.

Table 1. The 4SP case study problem definition.

| Stream | $T_{\text{in}}$ $(K)$ | $T_{\text{out}}$ $(K)$ | $\dot{Q}$ $(kW)$ | $h$ $\left(kW\,K^{-1}m^{-2}\right)$ | Cost $\left(\$\,kW^{-1}\,y^{-1}\right)$ |
|---|---|---|---|---|---|
| Process Streams | | | | | |
| H1 | 443 | 333 | 30 | 1.6 | |
| H2 | 423 | 303 | 15 | 1.6 | |
| C1 | 293 | 408 | 20 | 1.6 | |
| C2 | 353 | 413 | 40 | 1.6 | |
| Utilities | | | | | |
| Steam | 450 | 450 | | 4.8 | 80 |
| Water | 293 | 313 | | 1.6 | 20 |

Other estimates of the driving force are possible (see Chen 1987, for a commonly used approximation) but they are all, nevertheless, nonlinear.

This small case study, although appearing simple, is complex enough to demonstrate the design of the novel method. Larger case studies will be presented in the results section below.

## 2. Structural evolution for heat exchanger network synthesis

The novel method is based on the distinction between genotype and phenotype. The former can be thought of as a plan for an item whereas the latter is the actual item. These terms come from biology, specifically the area of genetics. In optimisation, these terms have been used in evolutionary optimisation to implement methods for the design of new structures (De Jong 2006). We have developed a genotype representation for the structural elements of a HEN, including the identification of integrated exchanges and stream splits. The phenotype corresponding to a genotype is an actual HEN which includes the design specifications for the network once the phenotype has been instantiated and evaluated.

A genotype describes a family of structures. A specific structure, i.e. a heat exchanger network including both utility and integrated heat exchangers, will be created by instantiating a phenotype from the genotype. The genotype is used to identify key properties of a family of structures. These properties include the specification of one half of integrated exchangers and the presence of stream splits. The actual integration pattern is not defined by the genotype but will be the responsibility of the instantiation of the genotype into a phenotype.

### 2.1. *A string rewriting grammar for genotype evolution*

The genotype is represented by the concatenation of a series of strings, one for each stream in the problem definition. A string, in this context, is a sequence of symbols from an alphabet. The strings are manipulated using a set of rules, a grammar, to create new strings. The use of a string rewriting grammar for HENS was motivated by the power of *Lindenmayer systems* (Prusinkiewicz and Lindenmayer 1990), often known as an *L-system*, in the generating of pleasing structures in computer art:

> *The central concept of L-systems is that of rewriting. In general, rewriting is a technique for defining complex objects by successively replacing parts of a simple initial object using a set of* rewriting rules *or* productions. (Prusinkiewicz and Lindenmayer 1990)

We wish to apply this concept of rewriting to streams in heat exchanger network synthesis problems to create potential integration structures.

Rewriting is accomplished through the use of a string rewriting grammar. The grammar defined below is an example of a Semi-Thue system (Book 1985), $(\Sigma, R)$, where $\Sigma$ is the alphabet used to represent the strings in the language and $R$ the

Table 2. The transformation rules, $P$, for the string rewriting grammar for structural evolution of heat exchanger network structures.

| Rule | Target → Replacement | Description |
|---|---|---|
| R1 | `-` → `x-` | Add an exchanger to a cold stream |
| R2 | `-` → `s[x-][x-]m-` | Split a cold stream |
| R3 | `+` → `x+` | Add an exchanger to a hot stream |
| R4 | `+` → `m]x+[]x+[s+` | Split a hot stream |
| R5 | `S` → `S` | A *do-nothing* rule |

rewriting rules. Our grammar can also be described as an *unrestricted grammar* due to there being no distinction between terminal and non-terminal systems. In any case, the notation we will actually use is based on that presented by Prusinkiewicz & Lindemayer (Prusinkiewicz and Lindenmayer 1990) due to the special starting conditions we have.

An L-system is defined by a tuple, $G = \langle V, \omega, P \rangle$, where the alphabet, $V$, is a set of symbols which can be replaced in a string by specific combinations symbols from the same set. If $V^*$ denotes the Kleene closure of $V$, the set of all words over $V$, $P \subset V \times V^*$ is the set of rules specifying the different replacement possibilities. $\omega$ is the initial configuration or set of strings. From a notation point of view, $V$ is the equivalent of $\Sigma$ in a Semi-Thue system and $P$ is $R$. A Semi-Thue system does not have an initial set of strings; instead, as in most formal grammars, it has a starting symbol, $S$. We are using the Lindenmayer notation to have a more general starting point but it is worth emphasising that these systems are all fundamentally the same in terms of string rewriting and we can map from one to the other easily.

For the heat exchanger network synthesis problem, we introduce a new L-system, $G_{\text{HEN}}$, with the following component definitions:

$V$ The alphabet includes symbols to denote the heating and cooling requirements of each stream, using `-` and `+` for parts of the cold and hot streams respectively, and symbols to represent the structural elements: the start, $S$, and end, $E$, of each stream, specific operations on the streams including exchange, `x`, split, `s`, and mix, `m`, and markers to denote the start, `[`, and end, `]`, of split stream segments. The full alphabet, therefore, is

$$V \stackrel{\text{def}}{=} \{-, +, \mathrm{S}, \mathrm{E}, \mathrm{s}, \mathrm{m}, [, ]\}$$

$\omega$ The starting set of symbols is a set of strings, one string for each stream in the network problem definition. Each cold stream is represented initially by the string `S-E` and each hot stream by `E+S`. The hot and cold streams are written in opposite order to indicate the use of counter-current heat exchangers.

$P$ The rules are shown in Table 2. Rules that add an exchanger to a stream only add "half" an exchanger. The matching half is identified during the phenotype instantiation step. Note that the rule for splitting a hot stream creates a structure that is the reverse of that created by a cold stream split rule. Also, the last rule, which effects no transformation on the strings, is included to cater for the multiplicity in the phenotype instantiation, as described in the next section. Although the inclusion of this rule is not necessary, it does simplify the implementation of the evolutionary algorithm.

This L-system is *context free* and *nondeterministic*. These properties motivate the use of a stochastic evolutionary algorithm for identifying possible structures for the HEN problem.

A particular set of strings, using words from $V^*$, will define a genotype for the HENS problem. The genotype describes a structure or configuration. Specifically, it will identify splits in streams and locations for integrated exchangers. As there is no guarantee that the rules for hot and cold streams will be applied in equal numbers, we only assume that one half of each integrated exchanger may be explicitly noted in the genotype. The implications of this assumption will be clearly noted in the next section when the instantiation of the genotype into a phenotype is described.

2.1.1. *Case study 1: Initial genotype.*

The initial genotype for the first case study is a string defined by the concatenation of the set of strings corresponding to the streams in the network problem definition:

$$\overbrace{\texttt{E+S}}^{\text{H1}}\ \overbrace{\texttt{E+S}}^{\text{H2}}\ \underbrace{\texttt{S-E}}_{\text{C1}}\ \underbrace{\texttt{S-E}}_{\text{C2}} \tag{1}$$

This initial genotype defines a non-integrated structure, one in which all heating and cooling requirements met by utilities alone. This special starting point is useful in that the non-integrated structure is often feasible and usually provides an upper bound on the optimum.

## 2.2. *The evolutionary algorithm*

A simple evolutionary algorithm, presented fully in Algorithm 1, has been implemented in Jacaranda (Fraga *et al.* 2000) to explore the space defined by the L-system described above. The algorithm is based on a population of solutions. At any point, a solution is chosen using a fitness directed random selection procedure, the $L$-system is applied to it to create a new solution, and the new solution is inserted in the population. Inserting a new solution is subject to diversity control using string matching so that each genotype is unique within a population. When the population becomes too large, a solution is removed from the population.

There are two numeric parameters for this procedure, the number of generations, $n_g$, to perform and the size of the population, $n_p$. There are also choices in the selection methods for choosing the member of the population to apply a transformation rule on, the actual rule to apply and the target symbol on which the rule is applied. The latter two are done randomly; the first is done either randomly or through a fitness based selection procedure emphasising the most fit solutions in the population. The evaluation of a genotype requires its instantiation as a phenotype and this is described in the next section. The small number of tunable parameters makes this algorithm easy to use.

Although this is a simple evolutionary algorithm, the underlying representation provided by the grammar and its rules could form the basis for a more complex method, based for instance on genetic algorithms or simulated annealing. However, as shall be seen later, even this simple algorithm is able to achieve good results for complex heat exchange problems.

2.2.1. *Case study 1: Example rule application.*

The starting population for this case study consists of the single genotype, shown above. The application of rule R1 ($- \rightarrow$ x$-$) to the first instance of the - token in (1) is shown in Figure 1. The notation, $Rm(n)$, is used to indicate the application of rule $m$ to the $n$-th instance of the target token for that rule. A further iteration

**Algorithm 1** Evolutionary algorithm for structure evolution through application of L-system

**Given:** population size, $n_p$; number of generations, $n_g$; the L-system, $G_{\text{HEN}} = \langle V, \omega, P \rangle$.
**Outputs:** Best solution found.

$p \leftarrow \omega$
**for** $i = 1, \ldots, n_g$ **do**
  Select $g$ from population $p$
  Select rule, $r$, from $P$
  $g \xrightarrow{r} g'$
  Evaluate $g'$ (*Requires instantiation of $g'$ into phenotype, cf. Algorithm 2*)
  **if** $g'$ increases diversity **then** (*Diversity is based on genotype representation*)
    $p \leftarrow p \cup g'$
  **else**
    $\tilde{g} \leftarrow \{g : g \in p$ with same genotype as $g'\}$
    **if** $g'$ is better than $\tilde{g}$ **then**
      $p \leftarrow g' \cup p \backslash \tilde{g}$
    **end if**
  **end if**
  **if** $|p| > n_p$ **then**
    Remove the least fit individual in $p$
  **end if**
**end for**
**return** best solution in $p$

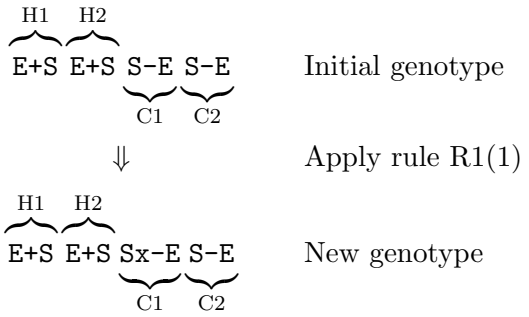



Figure 1. Illustration of the application of a rule to create a new genotype for case study 1.

of the genotype evolution algorithm may apply, for instance, rule R2 to the second suitable token found in the new genotype shown in Figure 1. This is illustrated in Figure 2. This new solution has a split on the second cold stream, C2, resulting in two explicit exchangers. This cold stream has three stream segments (`-`) on which rules can now be applied, allowing integrated exchanges or splitters to be placed on either branch of the split or after the mixer.

### 2.3. *Phenotype instantiation*

A genotype describes the general structure of a heat exchanger network. It explicitly indicates stream splits and locates at least one half of any potential integrated exchangers. Instantiating a genotype into a phenotype to create an actual network requires tying up the integrated exchangers and defining an appropriate embedded

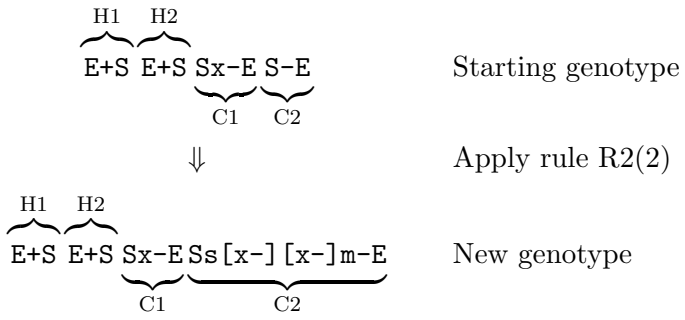



Figure 2. Illustration of the further application of another rule to generate another genotype for case study 1.

**Algorithm 2** Exchanger matching for phenotype instantiation

**Given:** $g$, the genotype to instantiate.
**Outputs:** $p$, a phenotype instantiation of the genotype

$p \leftarrow g$ (*Phenotype is initially the genotype.*)
**while** $\exists$ unassigned explicit exchange in $p$ **do**
  $x_1 \leftarrow$ randomly chosen unassigned exchange in $p$
  $x_2 \leftarrow$ randomly chosen unassigned exchange in $p$ from complementary stream
  **if** $\nexists x_2$ **then**
    $x_2 \leftarrow$ exchange inserted in randomly chosen complementary stream
    **if** $\nexists x_2$ **then** (*Match might not be possible*)
      **return** $\phi$ (*Indicating infeasible instantiation*)
    **end if**
  **end if**
  assign$(x_1, x_2)$
**end while**
**return** $p$

optimisation problem to determine the actual sizes of the exchangers and the split fractions for the split streams.

The first step, Algorithm 2, is to identify, for each half integrated exchanger indicated in the genotype (via the `x` symbol), the other half of this exchanger. The identification procedure first searches for explicit exchanger symbols in the streams of the opposite type. If any are found, a stream is chosen at random from those that have this symbol present. The first symbol in the genotype representation for that stream is used. If no possible explicit matches are found, an implicit exchanger is introduced in a randomly selected stream which provides a feasible match. This is done by the application of the appropriate exchanger introduction rule (either R1 or R3 in Table 2) and then applying the explicit match algorithm again. It should also be noted that the matching algorithm may fail to find a match for an explicit exchanger in the genotype. This will result in an infeasibility when the phenotype is evaluated. Otherwise, all structures generated by the instantiation of a genotype are potentially feasible, depending on actual designs of the splitters and exchangers, as described below in the solution of the embedded nonlinear optimisation problem.

The possibility of any exchange matching a number of other exchanges in complementary streams means that the network defined by the instantiation will typically be one of a number of possible alternatives. A genotype describes a blueprint for an exchange network; different phenotypes may be instantiated from a single genotype (cf. Tufte and Haddow (2004) for a similar scenario applied to the evolution of

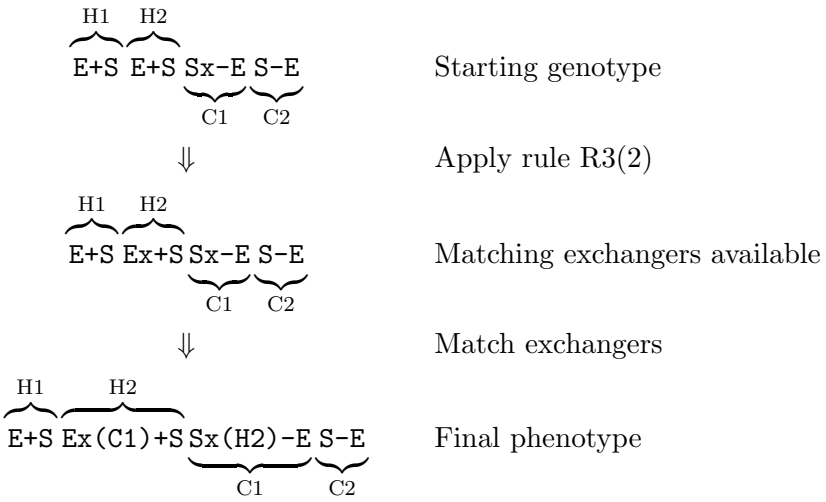



Figure 3. Instantiation of genotype to produce phenotype.

hardware). For this reason, we have defined the *do-nothing* rule, R5. This rule allows the same genotype to be instantiated more than once into a phenotype. As the same genotype may lead to different phenotypes, due to the matching procedure, new solutions based on the same genotype will be inserted into the population, after evaluation, if they lead to better solutions when the phenotype is evaluated.

Once the explicit exchangers in the genotype have all been processed, the phenotype has been instantiated. From this phenotype, a network structure is created which defines a nonlinear programme (NLP). This is an optimisation problem with real valued decision variables. The decision variables are the amounts to exchange, for each match identified in the instantiation, and the split fraction for any stream splits. All these variables are $\in [0, 1]$. For split amounts, the variable indicates the heat to send to (in the case of a hot stream) or to receive from (in the case of a cold stream) one branch based on the cooling or heating requirement in the stream segment at that point. For an exchange, the variable is the amount based on the maximum available. The maximum available is determined automatically depending on the values of the split variables and the total amount available for each stream.

It should be noted that the NLP essentially defines a superstructure. The decision variables, including both the amounts to exchange and the split fractions, may lead to the inactivation of parts of the structure. This inactivation also applies to the utility exchangers defined implicitly for every stream in the problem definition as the utility exchangers will only be part of the solution if they are required. As a result, all structures generated by the instantiation are feasible: a value of 0 for all design variables leads to a network with all exchanges deactivated and is essentially a network which meets all its requirements from utilities. This is, in fact, the starting solution for the solution of the NLP and provides a worst case solution for any optimisation.

#### 2.3.1. *Case study 1: phenotype instantiation.*

Suppose the genotype obtained in Figure 1 were to be instantiated as a phenotype. The matching procedure finds that only C1 has an explicit integrated exchanger half. This symbol is chosen and potential matches are searched for. As no explicit matches are present in any of the hot streams, a hot stream from those

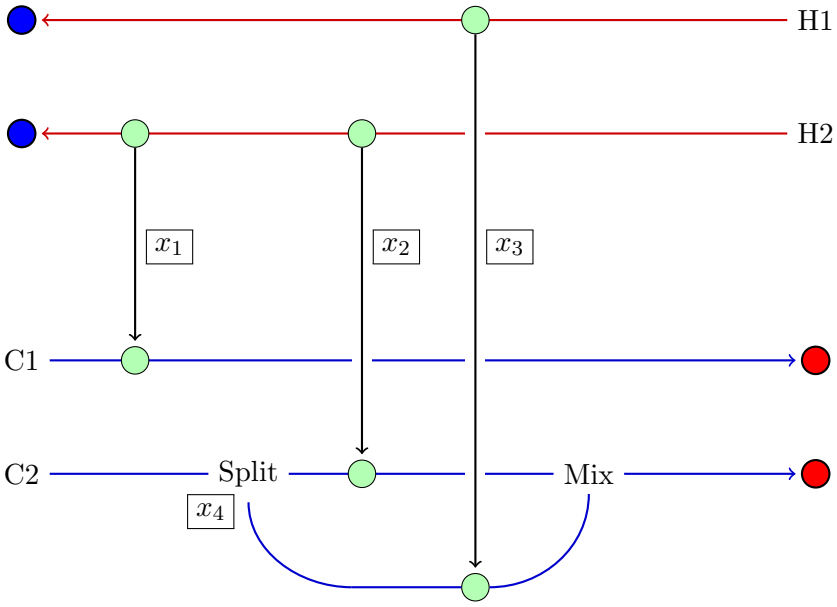
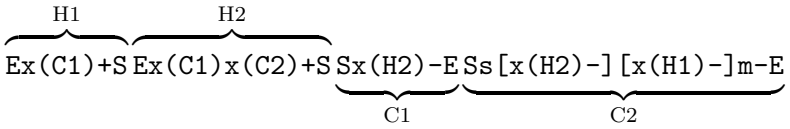



Figure 4. One possible phenotype instantiation for the genotype resulting from the application of rules R1(1) and R2(2) for the simple case study.

that could potentially match with this cold stream is chosen at random. Suppose H2 were chosen. Rule R3 is applied to the stream's genotype to get `Ex+S`. This yields a genotype string which now has an explicit exchanger so a match has been identified. The phenotype representation is almost identical to the genotype representation, differing only in that exchanges are indicated explicitly. The resulting phenotype for this instantiation is shown in Figure 3. In the final string, the phenotype, the `x` tokens have been parametrised with a string which identifies a specific match. For an exchanger on a hot stream, the match is specified by `x(Cn)` where `n` is replaced by the index of the cold stream. For a cold stream, the replacement text is `x(Hm)` where `m` is the hot stream index. The NLP corresponding to this network has only one decision variable, the amount of heat exchanged in the integrated exchanger.

If, instead, the final genotype obtained in Figure 2 were instantiated, one possible result would be that shown in Figure 4, shown graphically as a heat exchanger network structure with possible exchanges as described by the phenotype. Each stream, whether hot or cold, is represented by a line with temperature increasing to the right. Hot streams, therefore, come in from the right and exit left; cold streams are in the opposite direction. In this figure, and in all those that follow, the top lines are the hot streams. Circles represent exchanges of heat, either with utility if unconnected or with a stream of opposing kind if connected by a vertical line to another circle. This figure also shows the associated design variables for the NLP, three of which are the amounts to exchange between the two streams and the fourth, $x_4$, the split ratio. The phenotype for this network structure is



### 2.4. *Phenotype solution*

The phenotype instantiated from the genotype is *evaluated* by solving the NLP defined by the phenotype. This NLP may be solved, in principle, using any of a number of optimisation solvers suitable for nonlinear domain bounded and constrained optimisation problems. The NLP is non-convex and is, therefore, difficult to solve to global optimality. In a previous work (Fraga 2006), a hybrid optimisation procedure was presented for the optimisation of heat exchanger networks of the form generated by the phenotype instantiation. In this work, we use the hybrid procedure, combining a simple genetic algorithm, similar to that used in (Rowe and Fraga 2006) for the post-processing step but with a boundary mutation operator, with the Nelder-Mead simplex method (Nelder and Mead 1965) and the Hooke & Jeeves direct search method (Hooke and Jeeves 1961).

The hybrid optimiser uses, as a starting guess, a solution with all design variables set to 0. This starting guess is always feasible and corresponds to the solution with no heat integration, that is a solution where all heating and cooling requirements are met by external heating and cooling utilities. The hybrid procedure is a simple loop in which the current best solution to the NLP is given to one of the embedded solvers. The solution returned by the embedded solver is then passed to the next in the list, and so on through the list of available solvers. The loop over the embedded solvers is repeated so long as the best solution known continues to improve, subject to a maximum number of iterations (5 by default).

Although there is no guarantee that this embedded optimisation framework will be solved to optimality, this combination of direct search and stochastic methods for the solution of the NLP was found to be effective in practice. Other combinations are, of course, possible and the framework can easily accommodate alternative methods. In any case, the presence of the *do-nothing* rule, R5, gives the evolutionary procedure the chance to evaluate the same phenotype more than once, increasing the chances that the optimal network for a given genotype will be found.

### 2.5. *Summary*

A new rewriting grammar suitable for representing heat exchanger networks has been presented in this section. It is based on the decomposition of the design problem into a genotype, describing an overall blueprint for a heat exchange structure, and a phenotype which represents the actual exchanger network with full design information. This grammar is evolved through the application of a simple algorithm, Algorithm 1. This approach is similar, in concept, to the method proposed by Tufte and Haddow (2004) for the design of electrical circuits. They present a method based on a genotype-phenotype decomposition and argue that this decomposition allows for the generation of complex phenotypes from simpler genotypes. The method for heat exchanger network design exploits this property, as the results below will demonstrate.

The general approach of combining a grammar with an evolutionary algorithm leads to a *developmental system* (Luerssen and Powers 2007). Developmental systems, and more generally genetic programming methods (Riolo *et al.* 2007), have been used widely for the design of electrical circuits (Koza *et al.* 2008), a problem that shares some characteristics with the design of heat exchanger networks. Genetic programming methods are based on a language or grammar, defined by the symbols describing features of a program, which will solve some defined problem, and a set of rules for combining these symbols. Our rewriting grammar provides this aspect. A genetic programming method also requires a fitness measure (the instantiation of the phenotype, Algorithm 2, and the result of the subsequent embedded

optimisation step), the evolutionary procedure (Algorithm 1 and a terminating condition. These have been described fully for the new method.

The system has been implemented in Jacaranda (Fraga *et al.* 2000) using Jacaranda's own interpreted language for the modelling of the heat exchange networks. Jacaranda is written in Java and is available from the author, free for academic users. The combination of an interpreted modelling language on top of Java does not lend itself to computational efficiency and so cpu times are not reported in the results presented in the next section. Jacaranda provides a good environment for prototyping methods in process systems engineering. If computational efficiency is required for technology transfer, the particular methods can be rewritten directly in more efficient languages.

## 3. Results

A number of case studies from the literature has been investigated. The case studies have been selected to provide a thorough range of problems that exercise the new design method. The case studies are the following, indicating the number of hot and cold streams ($n_H \times n_C$) and the best known previously published solution (cost per year):

(1) The 4SP problem from a number of sources including Pariyani *et al.* (2006): $2 \times 2$, $83.5 \times 10^3$.
(2) The 10SP1 problem (Lewin 1998): $5 \times 5$, $43.4 \times 10^3$.
(3) A problem originally (Morton 2002) for investigation of retrofit optimisation: $3 \times 3$, $1.78 \times 10^6$ (Fraga 2006).
(4) Case study A from Lewin (1998): $5 \times 1$, $573 \times 10^3$.
(5) The aromatics problem from Lewin (1998): $4 \times 5$, $2.94 \times 10^6$.

All but one of these problems require splitting of streams to achieve the economically best solution. The exception is the 10SP1 problem. In this case, the best solution known does not split any streams. More detailed descriptions of the results obtained in each case study are presented below.

The configuration of the top level L-system evolutionary procedure was the same for all the case studies. The population size was 100, 5000 iterations were performed and a random selection procedure was used. The selection procedure gave all solutions the same likelihood of being chosen as the target for the L-system application at each iteration.

The underlying NLP solver parameters were also the same in all cases. The same combination of methods was used: a genetic algorithm (Goldberg 1989) and the Hooke and Jeeves (Hooke and Jeeves 1961) and the Nelder-Mead Simplex (Nelder and Mead 1965) direct search methods. The genetic algorithm used a stopping criterion of 100 generations, the population size was 30, the crossover rate was 70%, the mutation rate 10% and a tournament selection with size 2 was used. Default settings were used for the two direct search methods. The Hooke and Jeeves Java code was created from a C language implementation (Johnson 1994) with the aid of the `c2java` tool (Stonis 2007). The Nelder-Mead simplex method from the Mantissa project was used (Maisonobe 2007).

The stochastic nature of the evolutionary procedure, both in the overall procedure and in the phenotype instantiation, means that each attempt at a given problem will potentially identify a different solution as *best*. Therefore, any analysis of the effectiveness of a stochastic optimisation procedure must include a statistical breakdown. Table 3 shows a summary of the performance of the novel method on the case studies presented above. The method works well in all cases, obtaining the

Table 3. Summary of statistical analysis for all the case studies showing the best objective function value (to 3 significant digits) found from 10 attempts, the mean final value obtained from all the attempts and the worst final solution obtained.

| Case | Objective function value ($\times 10^3$ \$ $y^{-1}$) | | | |
|---|---|---|---|---|
| study | Best | Mean | Worst | $\sigma$ |
| 1 | 83.5 | 84.5 | 89.1 | 1.07 |
| 2 | 44.9 | 45.1 | 45.4 | 0.171 |
| 3 | 1620. | 1680. | 1720. | 35. |
| 4 | 573. | 575. | 594. | 6.61 |
| 5 | 2940. | 2960. | 2980. | 13. |

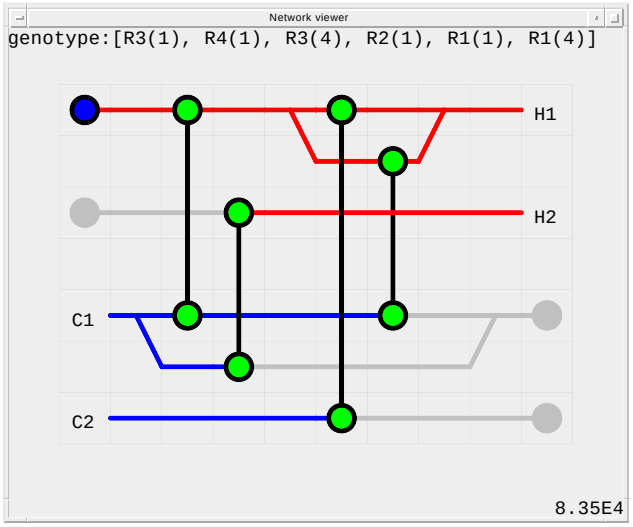



Figure 5. Best solution found for case study 1, the 4SP problem.

best solution known to date (cases 1 and 5), finding a solution close to the best known (case 2) or generating a better solution yet (case studies 3 and 4).

### 3.1. *Case study 1: 4SP*

The best solution obtained for this case study is presented in Figure 5

To get a better measure of the effectiveness of the stochastic method, beyond the statistics presented in Table 3, this problem has been attempted 100 times. Figure 6 shows the evolution of the best solution. The graph shows the best of any of the runs (the lower curve), the worst of any of the runs (the upper curve) and the average, over the 100 runs, of the best solution at each iteration. This figure shows that the method converges quickly but that there is some variation in the final solution obtained.

Figure 7 shows the value of the final solution obtained in each run. The best known solution is obtained approximately 20% of the time. However, although in almost 80% of the cases, the best solution is not obtained, the structure identified in the vast majority of the cases corresponds to that shown in Figure 5. The only exceptions are those solutions which have the value indicated by the top horizontal dashed line. This solution is the best solution achievable without stream splitting. The conclusion is that the procedure presented in this paper is effective at identifying good solution structures, including stream splitting, but that the embedded NLP solution method may not be as effective as we would like.

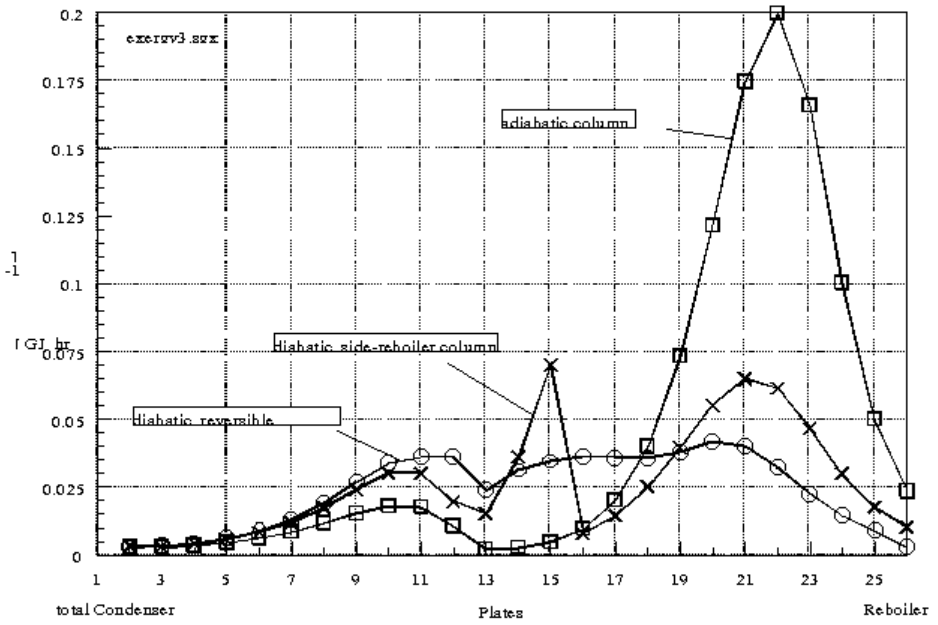



Figure 6. The evolution of the best solution for case study 1, the 4SP case, based on 100 runs of the algorithm.

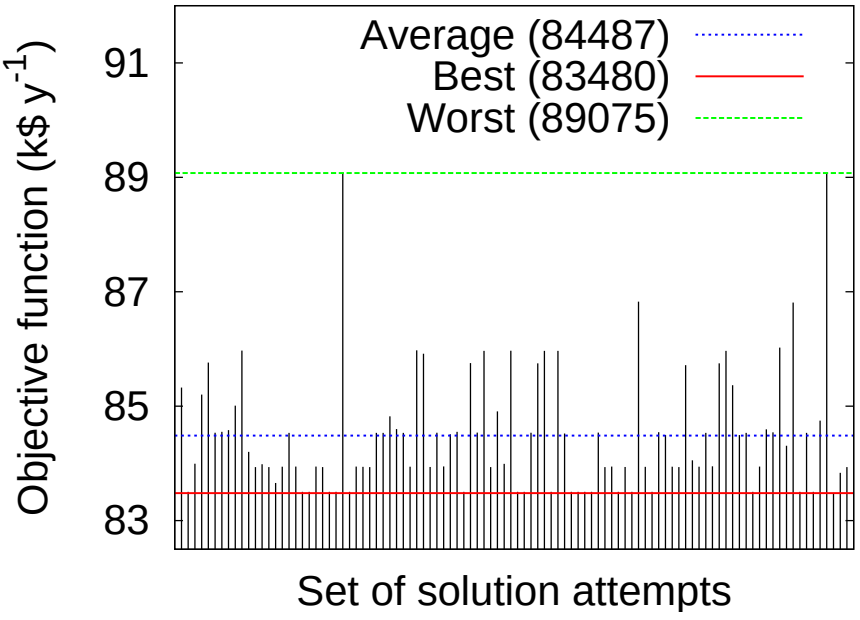



Figure 7. The value of the final solution obtained in each of the 100 runs attempted for case study 1, the 4SP case.

### 3.2. *Case study 2: 10SP1*

The best solution obtained is shown in Figure 8. The solution obtained, with a cost of $44.9 \times 10^3 \, \$ \, y^{-1}$, is not as good as the best reported in the literature, cost $43.4 \times 10^3 \, \$ \, y^{-1}$ (Lewin 1998). Our solution includes a split on stream H3; the best solution involves no stream splitting. It is worth noting, however, that our solution does meet all the requirements of the cold streams so, from a utility point of view, there is no improvement possible.

The emphasis of our novel method is the generation of structures with stream splitting. The fact that the only case study in which the best known solution is not

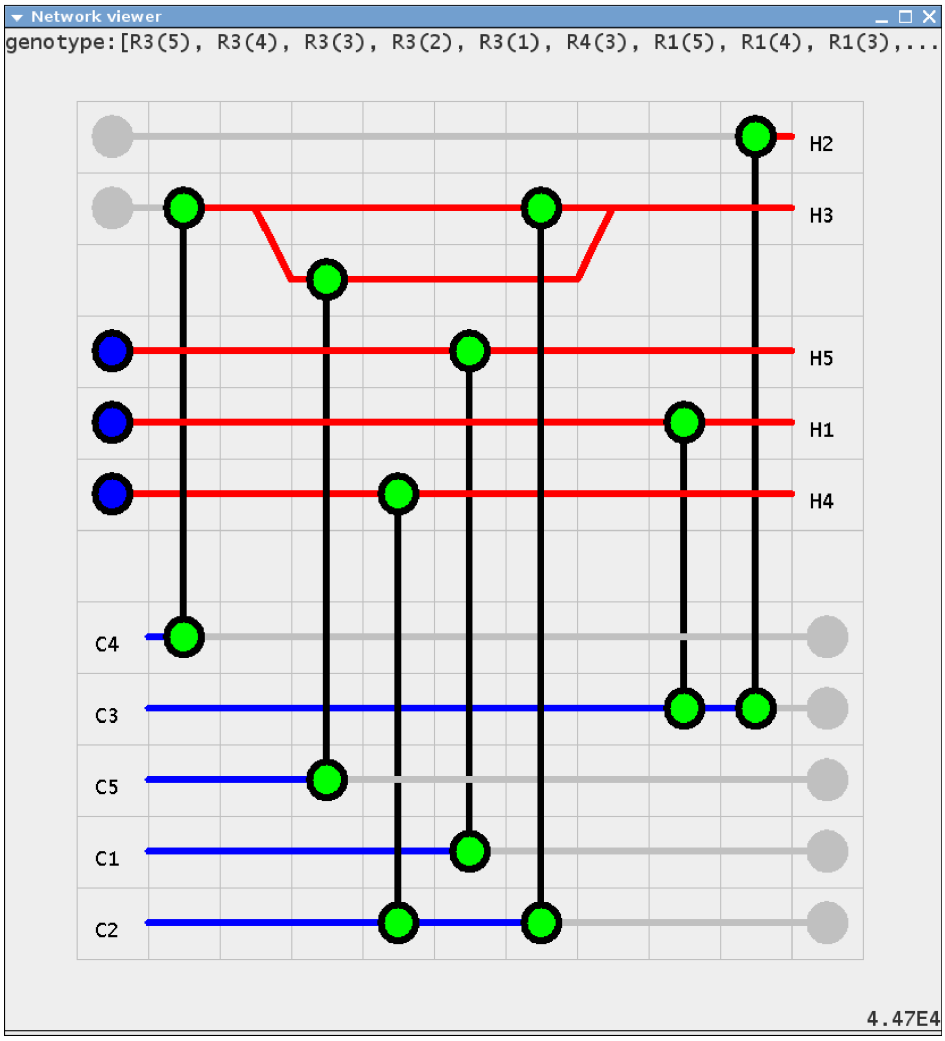



Figure 8. Best solution found for case study 2, the 10SP1 problem (Lewin 1998).

obtained is one in which stream splitting is not required suggests that, in general, a complement of tools may be appropriate for solving heat exchanger network design problems. Other approaches have been shown to be highly effective for problems without stream splitting (Fraga *et al.* 2001).

Alternatively, the current procedure treats all rules equally. The procedure is as likely to introduce an integrated exchanger as it is to split a stream. It may be useful to consider different probabilities for each rule. For some problems, stream splitting may be more important than a large number of exchanges; for other problems, the converse may be true. An implementation of variable probability rule selection is left for future work.

### 3.3. *Case study 3*

This case study is a generalisation of the stream splitting case study presented by Morton (2002). Previous attempts at this problem have considered stream splitting through the use of two stage procedures (Fraga and Rowe 2006, Rowe and Fraga 2006). In the first stage, stochastic methods (simulated annealing (Fraga and Rowe 2006) or an ant colony model (Rowe and Fraga 2006)) are used to determine a reduced superstructure for good solutions. This reduced superstructure is then solved, as an NLP (Fraga 2006). Although these attempts were successful

Table 4. Problem definition for case study 3 (Morton 2002).

| Stream | $T_{in}$ $(^{\circ}C)$ | $T_{out}$ $(^{\circ}C)$ | $\dot{Q}$ $(kW)$ | $h$ $\left(\frac{kW}{K \cdot m^2}\right)$ | $c_u$ $\left(\frac{£}{kW \cdot y}\right)$ |
|---|---|---|---|---|---|
| Process Streams | | | | | |
| H1 | 200 | 40 | 6400 | 0.8 | |
| H2 | 120 | 60 | 600 | 0.8 | |
| H3 | 90 | 50 | 200 | 0.8 | |
| C1 | 25 | 180 | 3100 | 1.6 | |
| C2 | 80 | 210 | 3250 | 1.6 | |
| C3 | 35 | 160 | 2250 | 1.6 | |
| Utilities | | | | | |
| Steam | 220 | 219 | | 1.6 | 700 |
| Water | 30 | 40 | | 0.8 | 60 |

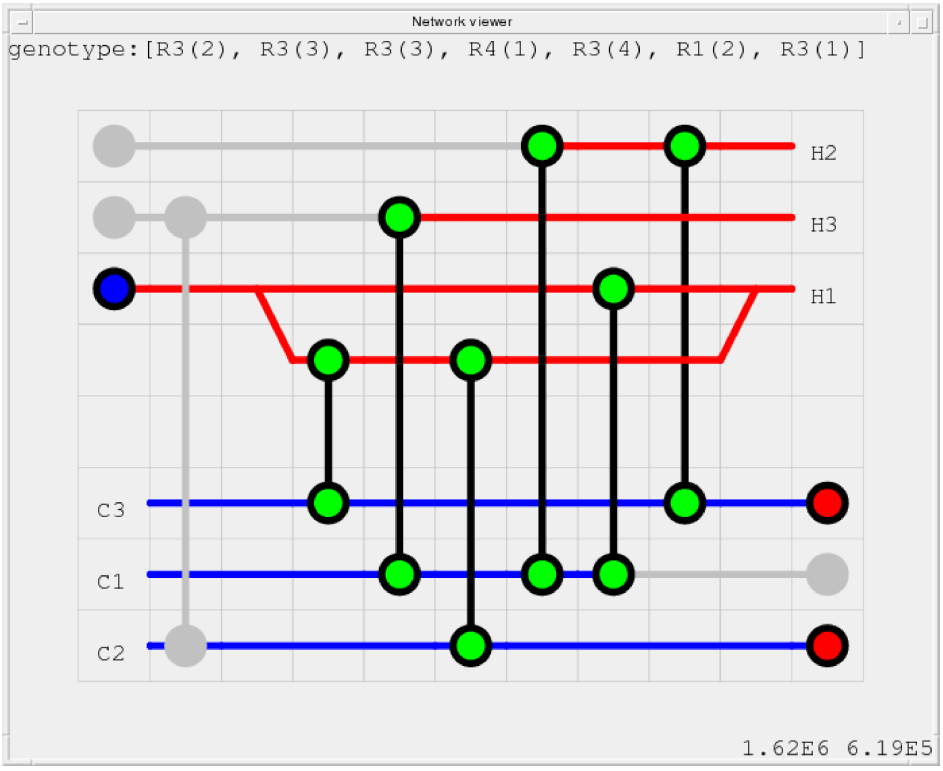



Figure 9. Best solution found for case study 3, the retrofit problem (Morton 2002, Fraga 2006).

at obtaining good solutions, they did not search the space of designs thoroughly and, in the case of the ant colony model approach, relied on interpretation of the visualisation results by the user before proceeding to the second stage. In any case, the best solution obtained previously for this problem has an objective function value of $1.78 \times 10^6 \, \$ \, y^{-1}$ so the new method has improved this by almost 10%.

Not only does the solution obtained have a better objective function value, it is simpler than that presented in earlier work (Fraga 2006), involving only one two-way split instead of the two splits (a two-way split on C1 and a three-way split on H1) in the earlier work. There is one more exchanger but the overall energy recovery has increased by 350 kW. Table 5 presents the design information for the network shown in Figure 9. The design value is the value of the design variable in the NLP for the particular exchange. The single split fraction is 0.45.

Table 5. Design information for best solution obtained for case study 3.

| Hot | Cold | Design value | $\dot{Q}$ (kW) | Area ($m^2$) |
|---|---|---|---|---|
| Integrated exchangers | | | | |
| H1 | C3 | 0.36 | 811 | 191 |
| H3 | C1 | 1.00 | 200 | 10 |
| H1 | C2 | 0.90 | 2558 | 497 |
| H2 | C1 | 0.35 | 209 | 13 |
| H1 | C1 | 1.00 | 2691 | 518 |
| H2 | C3 | 0.65 | 391 | 130 |
| Utility exchangers | | | | |
| Steam | C2 | | 692 | 42 |
| Steam | C3 | | 1048 | 15 |
| H1 | Water | | 340 | 92 |

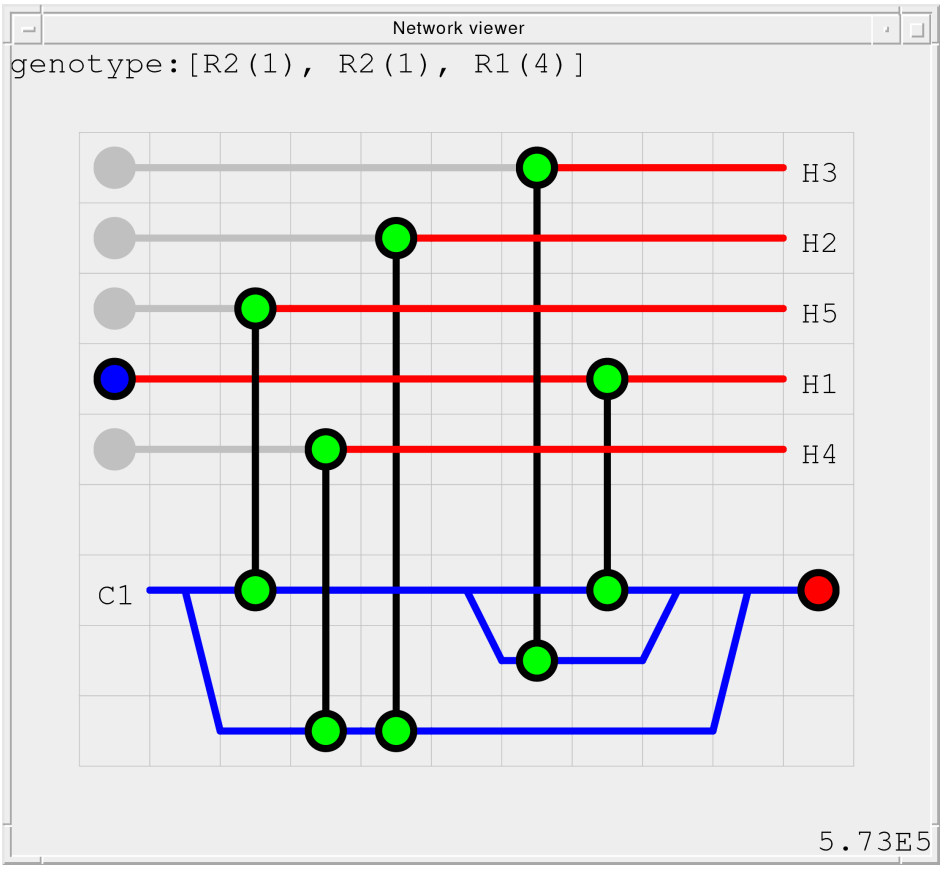



Figure 10. Best solution found for case study 4, case study A from Lewin (Lewin 1998).

### 3.4. *Case study 4*

Case study A from Lewin (1998) consists of 5 hot streams and a single cold stream. Any effective solution will require splitting of the cold stream. The best solution found, with an objective function value of $572,739\,\$\,y^{-1}$ and shown in Figure 10, requires two splits on the cold stream. The cooling requirements of all but one of the hot streams are met. The solution is marginally better than that presented by Lewin (1998) which had an objective function value of $573,205\,\$\,y^{-1}$ and involved two three-way splits.

Table 6. Design information for best solution obtained for case study 4.

| Hot | Cold | Design value | $\dot{Q}$ (kW) | Area ($m^2$) |
|---|---|---|---|---|
| Integrated exchangers | | | | |
| H1 | C1 | 0.84 | 902 | 69 |
| H2 | C1 | 1.00 | 400 | 25 |
| H3 | C1 | 1.00 | 600 | 40 |
| H4 | C1 | 1.00 | 400 | 14 |
| H5 | C1 | 1.00 | 720 | 22 |
| Utility exchangers | | | | |
| Steam | C1 | | 3638 | 32 |
| H1 | Water | | 178 | 7 |

Table 7. Design information for best solution obtained for case study 5, the aromatics plant.

| Hot | Cold | Design value | $\dot{Q}$ (kW) | Area ($m^2$) |
|---|---|---|---|---|
| Integrated exchangers | | | | |
| H2 | C5 | 1.00 | 9600 | 1566 |
| H1 | C3 | 1.00 | 5130 | 1089 |
| H3 | C4 | 1.00 | 6600 | 1886 |
| H4 | C3 | 1.00 | 13420 | 3191 |
| H4 | C2 | 0.66 | 6004 | 1002 |
| H1 | C2 | 1.00 | 3026 | 367 |
| H4 | C1 | 0.24 | 3874 | 900 |
| H1 | C1 | 0.43 | 7008 | 716 |
| H1 | C5 | 0.32 | 7066 | 306 |
| Utility exchangers | | | | |
| Steam | C1 | | 9119 | 1255 |
| Steam | C5 | | 15334 | 1987 |
| H1 | Water | | 6471 | 570 |
| H3 | Water | | 3000 | 451 |
| H4 | Water | | 22702 | 2529 |

The design information for the heat exchangers is presented in Table 6 and the split fractions are 0.46 for the first splitter from the left and 0.73 for the second. Although the amount of heating and cooling through the use of utilities is a little larger than that required by the best solution known before (Lewin 1998), the total exchanger area is $220.4\,m^2$. Again, this area is larger than that quoted by Lewin. The solution, nevertheless, is better overall due to the use of 5 exchangers instead of 6, benefitting from the less than unity power law relationship between area and capital cost.

### 3.5. *Case study 5: the aromatics plant*

The Aromatics Plant case study, described by Lewin (1998), has been studied extensively by many researchers. The best solution by the new method is presented in Figure 11, showing splits on cold stream C3 (split fraction 0.28) and hot stream H4 (split fraction 0.36). The remaining design information is shown in Table 7.

The total energy recovery is 61.7 MW out of the 93.9 MW of cooling and 86.2 MW of heating required. In comparison with the best quoted by Lewin (1998), our so-

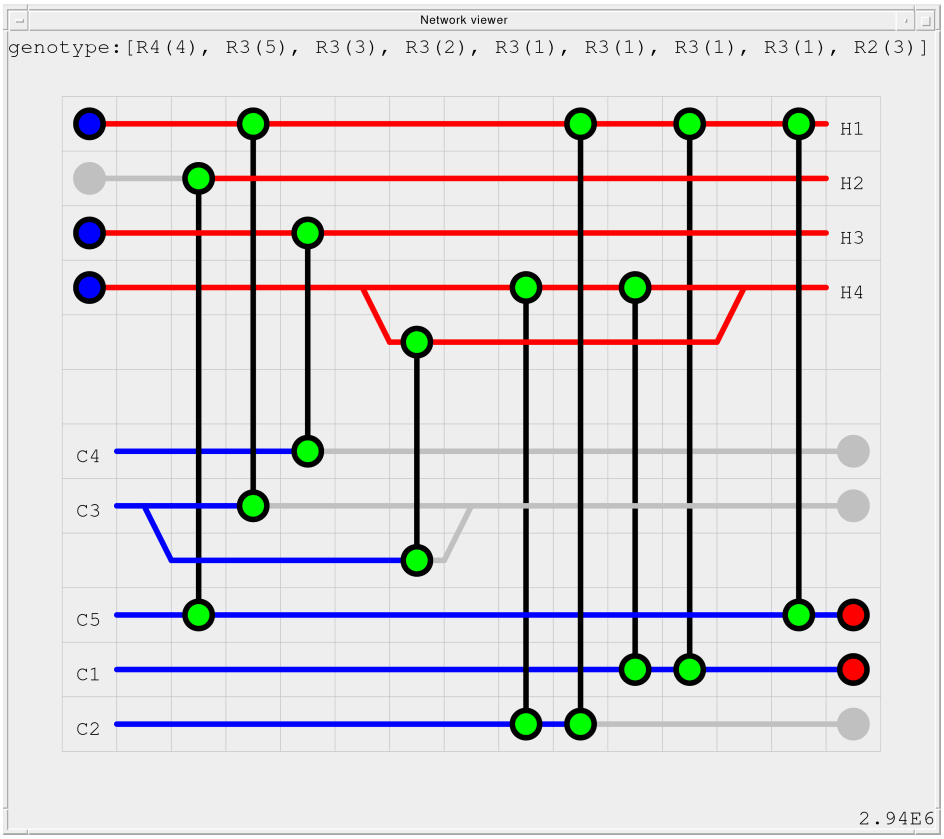



Figure 11. Best solution found for case study 5, the aromatics plant (Lewin 1998).

lution has a total annualised cost of $2\,935 \times 10^6$ versus $2\,936 \times 10^6$, a total exchanger area of $17\,815\,m^2$ versus $17\,050\,m^2$ and heating and cooling requirements $Q_h = 24.45$ MW and $Q_c = 32.17$ MW versus $Q_h = 25.09$ MW and $Q_c = 32.81$ MW. Our solution is marginally better in terms of cost and utility requirements with a slightly larger exchanger area overall. Our solution has two splits as does the solution presented by Lewin although ours are on both cold and hot streams whereas Lewin's involves splits on two hot streams. Finally, Lewin's solution has 12 exchangers versus 14 in our case.

### 3.6. *Discussion*

Five case studies have been presented. As summarised earlier, the new method is able to improve on previous results in two of the cases and it is able to match the best known results in two others. It is unable, however, to match the best result in one case study (case 2) although it comes close. The latter is an example in which stream splitting is not present in the best known solution. The new method is targeting stream splitting; other methods developed over the years are more suitable for the simpler cases involving no stream splitting. The new method, therefore, should be considered as one tool in an engineer's tool set for designing heat exchanger networks, where the engineer will use the most appropriate tool for the particular problem. In this context, the ability of the new method to find good solutions without requiring the engineer to manipulate a large number of method parameters is crucial. An engineer will use tools which can be easily applied to any problem.

## 4. Conclusions

Identifying a structure for process heat integration, allowing for stream splitting, is a challenging optimisation problem. The combination of combinatorial explosion with non-linear process and cost models ensures that the best solutions are difficult to find. This paper has presented a novel approach for structural evolution which searches the combinatorial design space. Using a string rewriting grammar, interesting and feasible structures are generated from the non-integrated starting point.

A simple population-based evolutionary strategy is used to explore the structure space. Each structure is optimised to determine the values of the continuous quantities, specifically the split ratios for stream splits and the amounts of heat to exchange for each possible exchange. This latter optimisation procedure is based on a hybrid method combining both stochastic and direct search methods. The combination of structure evolution with a hybrid NLP solution method results in a robust and effective tool for heat exchanger network design.

Although the approach presented is based on a simple evolutionary algorithm, more robust methods with provable convergence properties may be required for larger heat exchanger network design problems. Experience gained in genetic programming over the past decade will be valuable here (Riolo *et al.* 2007). It would be straightforward to consider both genetic algorithms and simulated annealing as modifications of the top level procedure. There are also possible improvements on the fitness evaluation to include a measure of the essentially unused rules that may exist in a genotype (due to the solution of the NLP having 0 values for exchanges or splits, for instance). Furthermore, the solution of the NLP could be improved, as noted in the discussion of the first case study. More alternative methods in the sequential hybrid procedure used for the NLP solution could be added easily or, more likely, a better overall NLP procedure might be appropriate. Nevertheless, the results indicate that the method presented is effective and successful for the wide range of case studies considered.

As this article was being finalised after revision, a new method for heat exchanger network design with stream splitting has been published (Luo *et al.* 2009). This new method is based on a hybrid procedure, combining a genetic algorithm for structure and parameter evolution and a simulated annealing procedure for tuning the parameters in the exchanger designs. The results obtained improve on those presented in this paper and therefore further motivates our interest in considering more complex evolutionary procedures using the grammar developed for stream splitting in heat exchanger network design.

## References

Aaltola, J., 2002. Simultaneous synthesis of flexible heat exchanger network. *Applied Thermal Engineering*, 22 (8), 907–918.

Book, R.V., 1985. *Thue systems as rewriting systems*. Lecture Notes in Computer Science Vol. 202. Heidelberg: Springer Berlin.

Chen, J.J., 1987. Comments on improvements on a replacement for the logarithmic mean. *Chem. Eng. Sci.*, 42 (10), 2488–2489.

De Jong, K., 2006. Evolutionary design: Lessons from biology. *In*: I.C. Parmee, ed. *Adaptive Computing in Design and Manufacture* The Institute for People-centred Computation (IP-CC), 23–25.

Fraga, E.S., Patel, R., and Rowe, G.W.A., 2001. A visual representation of process

heat exchange as a basis for user interaction and stochastic optimization. *Chem Eng Res Des*, 79 (A7), 765–776.

Fraga, E.S., Steffens, M.A., Bogle, I.D.L., and Hind, A.K., 2000. An object oriented framework for process synthesis and simulation. *In*: M.F. Malone, J.A. Trainham and B. Carnahan, eds. *Foundations of Computer-Aided Process Design*, Vol. 96 of *AIChE Symposium Series*, 446–449.

Fraga, E.S., 2006. Hybrid methods for optimisation. *In*: J. Žilinskas and I.D.L. Bogle, eds. *Computer Aided Methods for Optimal Design and Operations*. World Scientific Publishing Co., 1–14.

Fraga, E.S. and Rowe, G.W.A., 2006. A discrete interactive graphical method for heat exchanger network synthesis. *In*: W. Marquardt and C. Pantelides, eds. *16*th *European Symposium on Computer Aided Process Engineering and 9*th *International Symposium on Process Systems Engineering*, no. 21A in Computer-aided Chemical Engineering Garmisch-Partenkirchen, Germany: Elsevier, 877–882.

Furman, K.C. and Sahinidis, N.V., 2002. A critical review and annotated bibliography for heat exchanger network synthesis in the 20th century. *Ind. Eng. Chem. Res.*, 41, 2335–2370.

Goldberg, D.E., 1989. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

Hooke, R. and Jeeves, T.A., 1961. "Direct search" solution of numerical and statistical problems. *Journal of ACM*, 8, 212–229.

Johnson, M.G., 1994. *Nonlinear optimization using the algorithm of Hooke and Jeeves* [online]. : . `ftp://netlib2.cs.utk.edu/opt/hooke.c` [????].

Koza, J.R., Streeter, M.J., and Keane, M.A., 2008. Routine high-return human-competitive automated problem-solving by means of genetic programming. *Information Sciences*, 178 (23), 4434–4452.

Lewin, D.R., 1998. A generalized method for HEN synthesis using stochastic optimization – II. The synthesis of cost-optimal networks. *Computers chem. Engng*, 22 (10), 1387–1405.

Luerssen, M.H. and Powers, D.M.W., 2007. Evolvability and redundancy in shared grammar evolution. *In*: *2007 IEEE Congress on Evolutionary Computation, vols 1-10, Proceedings*, IEEE Congress on Evolutionary Computation IEEE Congress on Evolutionary Computation, Singapore, SINGAPORE, SEP 25-28, 2007 New York: IEEE, 370–377.

Luo, X., Wen, Q.Y., and Fieg, G., 2009. A hybrid genetic algorithm for synthesis of heat exchanger networks. *Computers & Chemical Engineering*, 33 (6), 1169–1181.

Maisonobe, L., 2007. *Mantissa: Mathematical Algorithms for Numerical Tasks In Space System Applications* [online]. : . `http://www.spaceroots.org/software/mantissa/index.html` [????].

Morton, W., 2002. Optimisation of a heat exchanger network superstructure using nonlinear programming. *Proc. Inst. Mech. Eng. Part E*, 216 (2), 89–104.

Nelder, J.A. and Mead, R., 1965. A simplex method for function minimization. *Computer Journal*, 7 (4), 308–313.

Pariyani, A., Gupta, A., and Ghosh, P., 2006. Design of heat exchanger networks using randomized algorithm. *Computers chem. Engng*, 30, 1046–1053.

Prusinkiewicz, P. and Lindenmayer, A., 1990. *The Algorithmic Beauty of Plants*. Springer-Verlag `http://algorithmicbotany.org/papers/#abop`.

Riolo, R., Soule, T., and Worzel, B., eds. , 2007. *Genetic Programming Theory and Practice IV*. Genetic and Evolutionary Computation Series Vol. 4. New York: Springer 4th Workshop on Genetic Programming, Theory and Practice, Ann

Arbor, MI, May 11-13, 2006.

Rowe, G.W.A. and Fraga, E.S., 2006. Co-operating ant swarm model for heat exchanger network design. *In*: I.C. Parmee, ed. *Adaptive Computing in Design and Manufacture VII* The Institute for People-centred Computation, 29–35.

Stonis, A., 2007. *C to Java converter* [online]. : . `http://www.soften.ktu.lt/~stonis/c2java/index.html` [????].

Tufte, G. and Haddow, P., 2004. Biologically-inspired: A rule-based self-reconfiguration of a virtex chip. *In*: Bubak, M and VanAlbada, GD and Sloot, PMA and Dongarra, JJ, ed. *Computational Science - ICCS 2004, PT 3, Proceedings*, Vol. 3038 of *Lecture Notes In Computer Science* 4th International Conference on Computational Science (ICCS 2004), Cracow, Poland, Jun 06-09, 2004 Berlin: Springer-Verlag Berlin, 1249–1256.

Wei, G.F., Yao, P.J., Luo, X., and Roetzel, W., 2004. Study on multi-stream heat exchanger network synthesis with parallel genetic/simulated annealing algorithm. *Chinese Journal of Chemical Engineering*, 12 (1), 66–77.