Synthesis of Heat-Exchanger Network by Simulated Annealing and NLP Procedures

G. Athier, P. Floquet, L. Pibouleau, and S. Domenech
Laboratoire de Génie Chimique–UMR 5503 CNRS, ENSIGC INPT, 31078 Toulouse Cedex 04–France

A two-level procedure for heat-exchanger network synthesis problems, which is related to structural optimization, is carried out by a simulated-annealing procedure. The main control parameters, that is, the initial annealing temperature, the temperature length, and the temperature-decreasing scheme, are extensively studied on the well-known 10SP1 problem, and their values are used for solving a 5SP1 problem with forbidden matches and a larger scale problem, the 20SP1. In all cases, the structural search is initialized with the network involving only utilities. For each generated network, the operating conditions (temperatures and split rates) are optimized by an NLP procedure to minimize the sum of investment and operating costs. Particular attention is given to the size reduction of the NLP problem. The proposed procedure leads to a reduction by a factor of 3 of the problem size in terms of number of variables and constraints. This slave problem is solved by an SQP procedure from the IMSL Library.

Introduction and Previous Works

The problem of heat-exchanger network synthesis (HENS) has received considerable attention over the last twenty years. Several significant reviews have been published on this subject, as for example the papers of Gundersen and Naess (1988) and Jezowski (1994). In these articles a large number of procedures are described, as well as their evolution.

The HENS problem to be addressed in this article was first formulated by Masso and Rudd (1969). At this time the design methods were generally based on heuristic approaches (Masso and Rudd, 1969; Nishida et al., 1971; Ponton and Donaldson, 1974). In order to decrease the exchange area, these authors used a set of heuristics for matching the streams; the rule most commonly implemented was to compulsorily match the hottest stream with a colder one. These methods did not provide the optimal solution; however, they generally gave a good feasible network. Later, Hohmann (1971) and Linnhoff and Flower (1978) suggested that the HENS problem could be tackled by breaking it up into two subproblems: 1. determine the minimum utility requirement, that is, the maximum energy recovery (MER); 2. determine the heat-exchanger matches that result in the minimum number of heat exchangers subject to the MER.

These concepts led to the definition of the pinch method, first introduced by Linnhoff and Hindmarsh (1983), source of many other subsequent publications. This technique first establishes the MER and pinch temperature based on a prescribed minimum temperature approach ΔT_{min}. In this dichotomy, a network characterized by a minimum utility consumption, and consequently by a minimum operating cost, is searched. The minimum number of heat exchangers also ought to decrease the investment cost. In the last decade, this method was extended to design networks of minimum cost by tracking capital costs against energy recovery (Townsend and Linnhoff, 1984). Following the development of computer science both in hardware and software, other methods based on mathematical approaches appeared in the 1980s. Papoulias and Grossmann (1983) formulated the MER problem as a linear programming (LP) problem, and they minimized the number of heat exchangers by means of a mixed-integer linear programming (MILP) code. In this method, which is one of the first attempted for automatically generating the network configuration, the objective function of Linnhoff and Hindmarsh (1983) was used with no cost consideration. Then Floudas et al. (1986) reused the procedure of Papoulias and Grossmann (1983) by adding a third step related to the minimization of investment cost subject to minimum utility cost and fewest number of units, by a nonlinear program (NLP) code. However, this sequential (not global) approach may often lead to suboptimal networks. For bypassing this problem more global procedures have appeared from the end of the 1980s. In this framework, Floudas and Ciric (1989) have developed a mixed-integer nonlinear programming (MINLP)

model to simultaneously optimize the selection of process stream matches and the network topology for a fixed level of energy recovery (HRAT). The formulation was based on the superstructure previously presented by Floudas et al. (1986), which embedded all possible matches. The optimization procedure leads to the set of matches that minimizes the total investment cost of the heat exchangers. This method based on the pinch concept and utility targeting task can be considered as an improvement of the previous one. However, if the definition of the pinch point and the specification of a minimum-temperature approach simplify significantly the problem, the truly minimum cost network can be eliminated from the superstructure. Ciric and Floudas (1991) implemented a MINLP model to simultaneously solve utility consumption levels, process stream matches, and optimal network topology. They assumed, on the one hand, that the minimum cost design will not locate a heat exchanger across the pinch point, and on the other hand, a minimum number of units subject to the MER. Let us note that dual-temperature approaches (Trivedi et al., 1989) allowing for crosspinch exchangers, relax the pinch decomposition, but from a thermodynamical point of view crosspinch exchangers are not efficient (Guiglion et al., 1992). The goal was always to minimize the utility requirements and the exchange area; however, the optimal global cost was not guaranteed. Yee et al. (1990a) and Yee and Grossmann (1990b) proposed global methods based either on NLP or MINLP models using a stagewise representation method (i.e., a superstructure) and several assumptions as isothermal mixers and no pinch concept to solve a global cost or a minimum exchange area problem.

However, in the case of large-scale problems, the implementation of MINLP algorithms can give birth to many computational difficulties. Even fairly simple systems can lead to large-scale combinatorial synthesis problems that cannot be satisfactorily tackled by MINLP procedures, in particular, with regard to CPU times and local optima. Indeed in all published works the largest synthesis problem solved is the 10SP1 network.

Dolan et al. (1989, 1990) proposed a quite different global method, based on the simulated annealing (SA) procedure, for solving the discrete and continuous problems together. However, the SA is not essentially a continuous optimization procedure, and as pointed out by Patel et al. (1991), NLP algorithms are slightly more efficient than SA for solving continuous problems in terms of accuracy and CPU times.

In this article, a new procedure based on a numerical approach is presented for solving HENS problems in terms of investment and operating costs. A mixed algorithm is developed for bypassing the implementation of MINLP procedures that may give rise to difficulties in the case of large-scale problems. The main goal of this study is to tackle industrial-sized HENS problems. Another concern is to reduce structural constraints commonly encountered in classic approaches, as for example the sequential-analysis method and the transshipment model proposed by Papoulias and Grossmann (1983), or the superstructure method implemented by Yee et al. (1990) and Yee and Grossmann (1990) using a stage representation with the assumption of isothermal mixers.

In the proposed solution networks may involve stream splitting, nonisothermal mixing, no bypasses, and multiple matches between two same pairs of streams, possibility of cold-to-cold or hot-to-hot matches in the case of classic forbidden matches. Multiple utility levels and different costing procedures for the heat exchanger can be used in order to take into account different pressure levels or different types of construction materials. The problem is formulated for optimizing a cost-objective function by defining the feasible network structures with an SA (discrete optimization) procedure. For each generated network the continuous operating parameters (split rates and temperatures) are optimized with an NLP package, at a lower level. In fact, the proposed approach is similar to a MINLP method, where the master problem is solved by an SA algorithm instead of a MILP procedure, and so the restrictive assumptions of continuity and convexity can be relaxed. Theoretical properties of the method are discussed, as is its implementation. In the last part, the approach is illustrated by three examples: 10SP1, 5SP1, and 20SP1.

Problem Statement

The HENS problem to be addressed in this article can be stated as follows.

A set of hot NH streams to be cooled and a set of NC cold streams to be heated are given. Fixed heat capacity, flow rates, inlet, and outlet temperatures are specified for these streams. Auxiliary heating and cooling are available from a set of NUH hot utilities and NUC cold utilities.

The goal of the HENS problem is to provide a network that satisfies the specifications at a total minimum cost (investment and operating costs). The basic assumptions used are the following:

- *Possibilities of Stream Splitting.* Stream splitting significantly increases the search space, but it often leads to better heat integration.
- *No Bypass.* This assumption uselessly increases the complexity of the configurations (discrete problem) for few improvements in terms of global cost.
- *Possibility of Forbidden Matches.* From Dolan et al. (1987) and Wiswanathan and Evans (1987) the utility requirement can be reduced in a forbidden-match problem. The method consists of matching one of the streams (hot/cold) of the forbidden match with another same stream (hot/cold) by utilizing the heat content of the second stream; as a result, the utility consumption could be considerably reduced. These authors have shown that in the best case, the minimum utility is the same as for the unrestricted case, and in the worst case, there is no improvement of the utility consumption.
- *Available Different Utility Levels.* For example, fuel or stream at different levels (high, medium, and low pressure) and hot water can be used as heating utilities, while cold water and refrigerants can be used as cooling utilities. Specific data and cost are assigned to each utility.
- *Different Types of Heat Exchangers.* In the published works, all the exchangers of the networks are generally built with the same construction materials, and operated with the same pressure. These assumptions that constitute important restrictions for practical applications are relaxed in this study. Different capital costs for taking into account construction materials and operating pressure levels are considered. For example, a corrosive stream requires special construction materials. Furthermore, a specific cost, depending on the size of

the heat exchanger area, can be affected by a given heat exchanger. From all these considerations the cost evaluation obviously becomes a tedious task.

- *Countercurrent Heat Exchangers.* This type of heat exchanger represents the best solution to decrease the exchange area and consequently the investment cost.
- *Constant Thermodynamic Data.* Heat capacity flow rates and heat-transfer coefficients are assumed to be constant within the temperature interval of each stream. This implies that no phase change is possible.
- *No Pinch Consideration.* Thermodynamic concepts are not used in the proposed method. Indeed, the specification of a minimum-temperature approach may drastically decrease the search space, and consequently leads to only suboptimal solutions.
- *Utilities Located at the End of the Internal Network.* It is a classic hypothesis for avoiding the combinatorial explosion.

General Solution Strategy

Figure 1 represents the classic flow chart for solving MINLP problems by a two-level approach; the fundamental difference between well-established MINLP procedures is that an SA algorithm is implemented to solve the master problem, instead of a MILP procedure that requires strong convexity properties. At the upper level (master problem), a HEN topology is first generated and then iteratively modified by the SA procedure according to structural assumptions that permit the whole search space to be overlaid and to give birth to feasible networks only. At a lower level (slave problem) an NLP package is implemented for optimizing the operating parameters of the network proposed by the master problem, with respect to the global cost (investment and operating costs). The procedure is repeated until the convergence is reached and provides the optimal solution according to the chosen objective function and the convergence criterion. The master and slave problems are detailed in the two following sections.

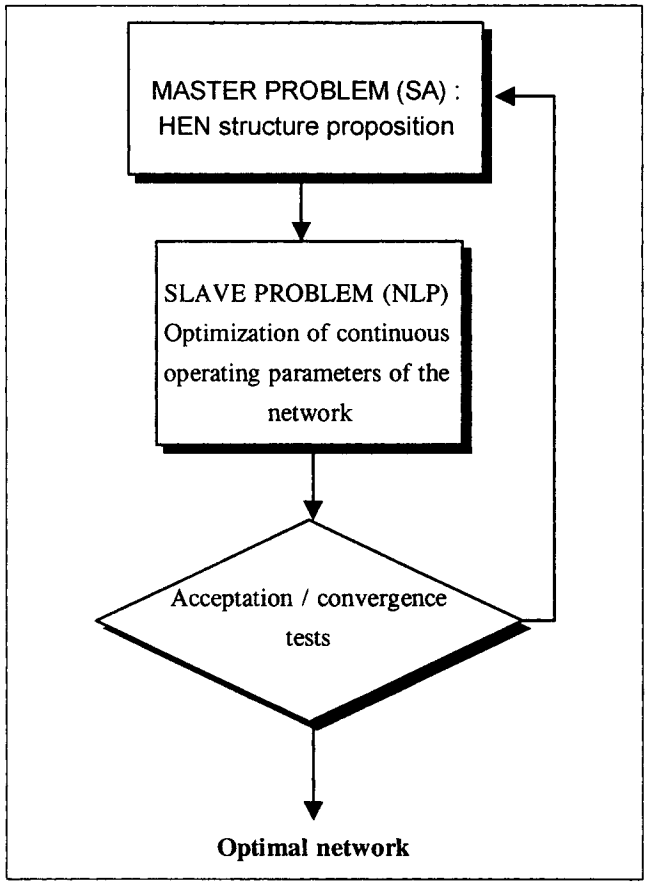

Figure 1. General solution strategy.

Simulated Annealing Procedure

SA is a recently developed multivariable combinatorial optimization technique, based on an analogy with statistical mechanics: the physical annealing of solids. It derives from the Metropolis algorithm (Metropolis et al., 1953), and the theory of Markov chains provides mathematical properties about its asymptotic convergence. So for an infinite time scheduling, the global optimal solution is guaranteed by the SA. In practical applications, a good suboptimal solution can be obtained in a finite time, in a way practically independent of the problem initialization; however, the CPU time obviously depends on the problem size. Due to the concept of unidirectional method implemented in the SA, different runs can lead to different suboptimal solutions that consequently must be tested by a statistical analysis. SA is a direct method (not based on the objective function gradient), and so it can easily handle discontinuities. SA has been successfully implemented for solving a wide class of combinatorial problems, as for example, the classic traveling salesman problem (Kirkpatrick et al., 1983), and in the last ten years, various chemical engineering problems, like HENS (Dolan et al., 1989, 1990), pipeline networks (Dolan et al., 1989), or batch processes (Das et al., 1990). The basic principle of SA, is that in the annealing of solids, the aim is to find some atomic configuration that minimizes internal energy. For a given configuration, a random move is carried out by randomly picking a molecule and moving it in a random direction for a random distance. This structural modification is called a *move*. The new configuration is then accepted or rejected according to an acceptance criterion, based on a Boltzmann function $\exp(-\Delta E/T)$, where ΔE represents the change in energy between two configurations, and T is the system temperature. The move acceptance criterion not only depends on ΔE but also on the temperature, which is gradually reduced during the search; this reduction is called the *annealing schedule*. From Kirkpatrick et al. (1983) a careful annealing schedule leads to the lowest energy state, while rapid cooling generates a higher energy metastable structure.

The SA algorithm for HENS problems, where the energy E represents the cost C, follows:

1. Select an initial feasible HEN.
2. Select initial values for the pseudotemperature *Tsa* and the length of the pseudotemperature *Nsa*.
3. If the termination criterion is not yet reached, then
   3.1 Perform the following loop *Nsa* times:
      3.1.1 Generate a new configuration S' neighbor of S
      optimize the operating variables by a NLP method

3.1.2 Based on the move acceptance criterion, accept or reject the solution (move).

3.2 Set a new value of temperature (as per the annealing schedule).

4. S represents the optimal HEN.

Starting from the initial network, a new configuration is generated by a move. The total cost C of this configuration is obtained from the operating parameters optimization, as described in the NLP section. The new configuration is accepted or rejected according to a probabilistic criterion based on the Boltzmann function

$$p = \exp(-\Delta C/Tsa), \tag{1}$$

where ΔC represents the change in cost between two configurations S and S'. Instead of a true temperature as in thermodynamical systems, a pseudotemperature Tsa that is the control parameter of the algorithm is gradually reduced during the search.

The main advantage of SA is that it can accept uphill moves and consequently escape from local minima. As the acceptance probability increases for ΔC and higher Tsa, SA starts with a high Tsa value in order to accept a large number of uphill moves. Then as Tsa gradually decreases, the number of uphill moves decreases. At the lowest value of Tsa, all the accepted solutions are very close, and SA becomes quite equivalent to a descent method. Let us note that in order to obtain a dimensionless argument for the Boltzmann function, the pseudotemperature Tsa must have the same dimension as the objective function, that is to say, a cost.

Different types of acceptance criteria have been proposed in the literature. The most popular ones are the Metropolis algorithm (Metropolis et al., 1953) and the Glauber procedure (Glauber, 1963). The Metropolis algorithm accepts all moves that decrease the cost ($\Delta C < 0$). Otherwise, the current solution is accepted with a probability p that depends on the cost change and the pseudotemperature Tsa. The acceptance probability of any move can be mathematically expressed as:

$$\begin{cases} \text{for } \Delta C \le 0 & p = 1 \\ \text{for } \Delta C > 0 & p = \exp(-\Delta C/Tsa). \end{cases} \tag{2}$$

The Glauber algorithm does not necessarily accept all downhill moves, any more than it accepts all the uphill moves; the acceptance probability is given by

$$p = \frac{\exp(-\Delta C/Tsa)}{1+\exp(-\Delta C/Tsa)}. \tag{3}$$

For high temperatures, the Metropolis algorithm accepts all the moves with probability 1, while the Glauber procedure accepts all the moves with probability 0.5. As the temperature is reduced, the probability acceptance of downhill moves remains constant at 1 in the Metropolis algorithm, while it increases from 0.5 to 1 in the Glauber procedure. On the other hand, the acceptance probability of an uphill move decreases from 1 to 0 in the Metropolis algorithm, and from 0.5 to 0 in the Glauber procedure. In a general way, both procedures provide good results, with a slight improvement for the Metropolis algorithm, which has been retained in this study.

Concerning the annealing schedule, two basic types are generally used. In the exponential schedule (Aarts and Van Laarhoven, 1985), the pseudotemperature is reduced according to Kirkpatrick's schedule (1983):

$$Tsa^{k+1} = \alpha\, Tsa^{k} \tag{4}$$

where α is a constant lying in the range $]0,1[$, defining the speed of annealing. The smaller the value of α, the faster the annealing. This annealing schedule is therefore determined by the choice of α. Aarts and Van Laarhoven (1985) have proposed another annealing schedule, where the new temperature is defined as

$$Tsa^{k+1} = \frac{Tsa^{k}}{1+\dfrac{Tsa^{k}\cdot\ln(1+\delta)}{3\sigma(Tsa^{k})}}, \tag{5}$$

where $\sigma(Tsa^k)$ represents the standard deviation of the objective function at the temperature Tsa^k and δ is a specified value lying in the range $]0,1[$, which is a measure of the desired closeness to the equilibrium. δ is also called *speed parameter*. The smaller δ is, the greater the desired closeness to equilibrium, and the slower the annealing. The higher the value of the standard deviation of the objective function, the system moves further away from equilibrium and the slower the annealing.

Another important feature of an SA is the choice of the initial pseudotemperature Tsa^0, which is closely linked to the nature of the problem. The pseudotemperature can be obtained by specifying the ratio of generated solutions to be initially accepted. The higher this temperature, the larger the number of moves accepted. In a general way, the initial pseudotemperature is chosen to be about 10 times the maximum value of ΔC between any two successive configurations when all moves are accepted. This corresponds to an initial temperature that gives, respectively, about a 90% and 47.5% for the acceptance probability for the move giving the maximum value of ΔC in the Metropolis and Glauber algorithms. Let us recall that this initial value depends equally on the nature of the problem under consideration, as it is shown in the first example.

For a given pseudotemperature, the algorithm performs a number of *Nsa* moves, also called *length of temperature*. This number is dependent on the nature of the random move, and it is equal to the number of reachable solutions from any given configuration by making a simple random choice (see numerical examples).

Various methods for stopping the optimization search exist. Aarts and Van Laarhoven (1985) have suggested ending the search when the derivative, with respect to the temperature, of the smoothed average objective function at a given temperature level becomes less than a specified value ϵ ($0 < \epsilon \le 1$). The termination test can be expressed as

$$\frac{d\langle C(Tsa)\rangle}{dTsa}\,\frac{Tsa}{\langle C(Tsa^0)\rangle} < \epsilon \tag{6}$$

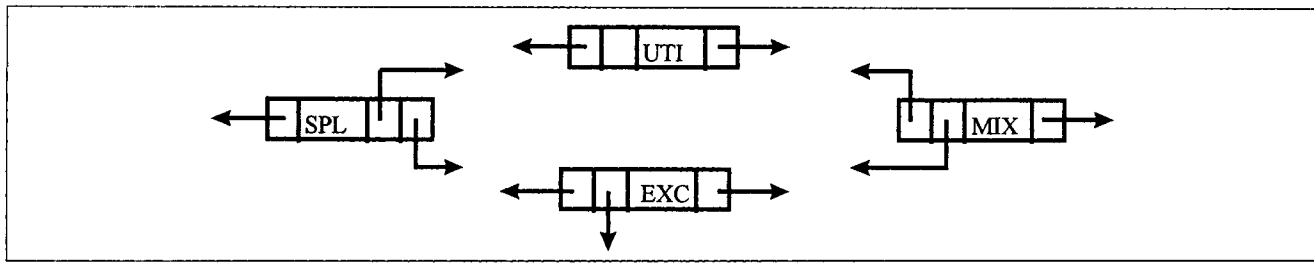

Figure 2. Representation of the four different objects.

where $\langle C(Tsa)\rangle$ is the smoothed average objective function, computed over a fixed number of previous temperature levels. The ratio $Tsa/\langle C(Tsa^0)\rangle$ is used to make this termination condition dimensionless, in order to be compatible with the dimensionless tolerance ϵ. Dolan et al. (1989) have proposed the simplest method: the search ends when the temperature becomes less than unity. Painton and Diwekar (1994) end the search if the objective function cannot be improved after several temperature decreases. Insofar as the last two methods are simple to implement, and as they generally give good results, we have only tested them. Let us note that one of the advantages of an SA is that the convergence does not depend on the initial solution; however, if this initial solution is infeasible, the search space becomes empty.

The way of performing the moves is a basic point of an SA implementation. The efficiency of an SA strongly depends on these moves. Before describing them for the HENS problem, the implementation of the network and the way to modify it must be defined. The HENS is a particular problem in which the number of constraints and variables are not constant during the search. For simulating a HEN configuration, its modification, and evolution, the classic method is to use a superstructure-based representation, with mixed variables and structural assumptions, as in MINLP approaches. The major disadvantages of this method are the introduction of structural inflexibilities (due to structural assumptions) and the great number of useless variables at each iteration.

Another method is implemented in this study; it consists of using an advanced programming language such as C that offers such concepts as dynamic memory allocation, pointers, linked-list representation, and object programming. The use of pointers (addresses) instead of classic numerical data allows us to manage the HEN configuration within the slightest CPU times. For the proposed model, based on the Dolan's concepts, the major feature is the representation of the HEN structure by means of a linked list in which each element is considered an object (physical properties). Each different component represents a specific class: EXC, SPL, MIX, and UTI (see Figure 2), to simulate, respectively, a side of a heat exchanger, a splitter, a mixer, and an auxiliary utility. A splitter is assumed to split a stream in only two parts, so a series of n splitters will split a stream into $n+1$ parts. Connections between each object are made by three pointers (process streams). Two of them are used to link the next and previous object of the network, as shown in Figure 3. The third is used to simulate the second part of a splitter or a mixer, and to branch both sides of an exchanger; it is not used for a utility. Each object is implemented with its associated properties and data (input and output temperature, exchange area, different items for the NLP problem, and all data required for the NLP formulation) called the internal state (see the Appendix).

Other different data structures are used for defining the input/output of the network (I/O array) and the necessary random access (RA array) for the SA procedure. The I/O array is used to provide sequential access to the network and to encapsulate all general data linked with a specified stream (the input and output temperature of the stream, heat capacity, flow rate, etc.). The random access to the object is made by storing pointers of the objects in the RA array. Figure 4, where only the main variables are represented, shows the implementation of the HEN. This object programming approach allows us to randomly pick an object and to carry out some operation on it, without having to search for its location along the linked list by using the I/O array. A last data structure comparable to a database contains all the data required for the problem solution (data for streams, cost functions, etc.). As circumstance requires, this data structure supplies each object, and allows separation of the data; so independent programming modules can be developed. Figure 5 shows the global procedure implementation and the connections

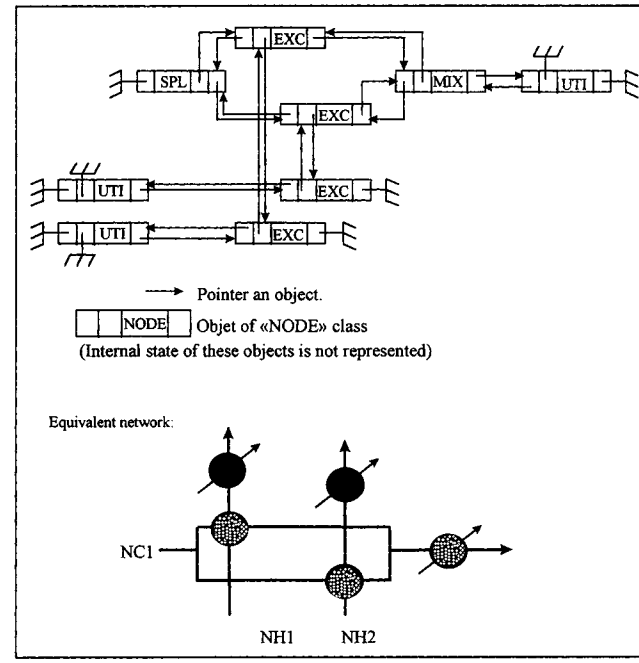

Figure 3. Representation of a part of a HEN configuration.

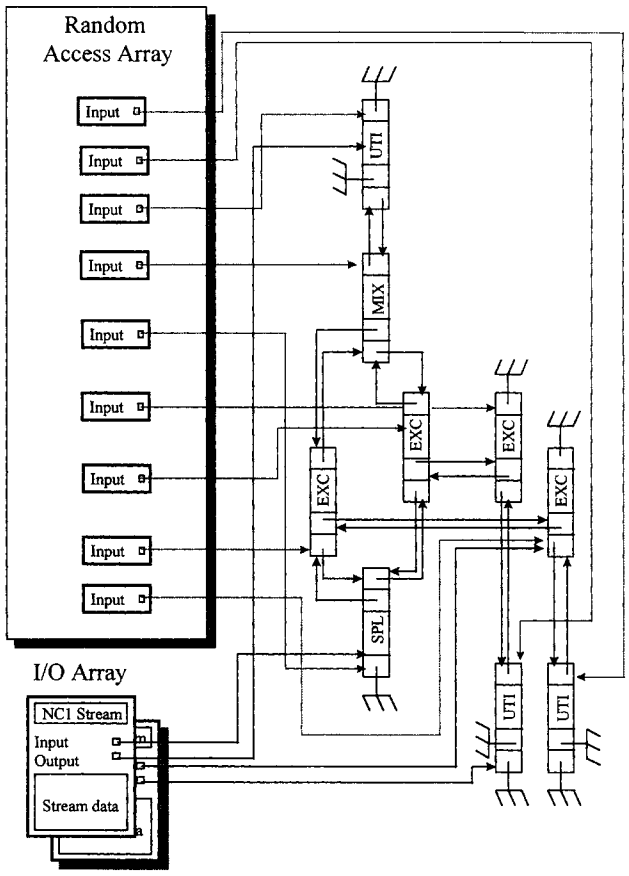

Figure 4. Implementation of the HEN.

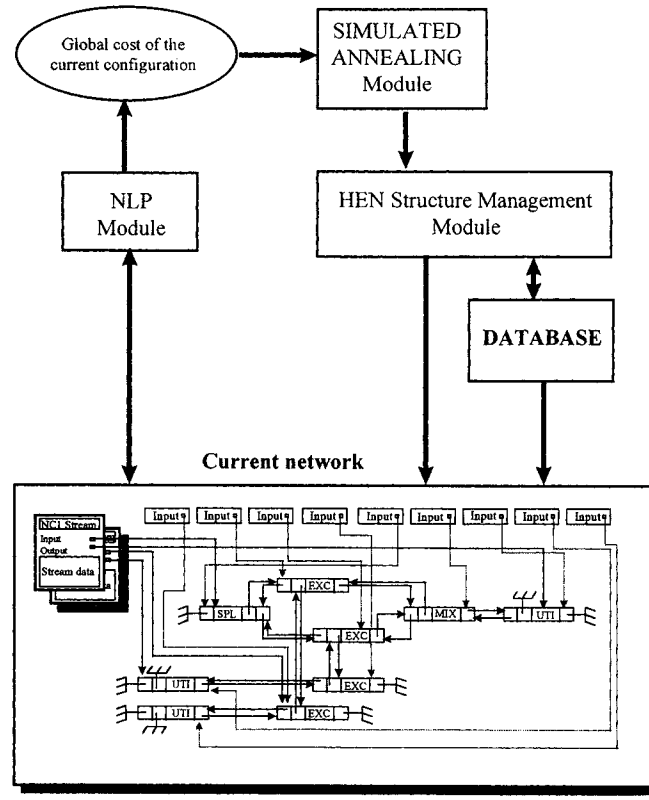

Figure 5. Global procedure implementation.

between each module. To define a configuration, only the required objects are to be stored.

After addressing a given HEN implementation, let us now describe the way to modify it. The moves used in this study allow an easy simulation of the topological changes of a structure while verifying the structural specifications and assumptions. Indeed the procedure only generates feasible HEN, which implies the following points. For a given move from a structure S to a structure S', it is possible to return to the structure S by one or several other moves. A configuration S' can be reached from a configuration S in different ways. Four simple different moves allowing generation of all the possible structures verifying the structural assumptions, have been developed. At each iteration of the SA procedure, one of the following moves is carried out with the same probability.

- *Add a heat exchanger.* This move consists of adding two objects of EXC class at a point randomly selected in the network. If a classic match is forbidden, the set of match possibilities to cold-to-cold and hot-to-hot matches between a forbidden stream and another stream of the same type is increased. Then, the heat-exchanger data required for the NLP optimization are charged in the object (activation of its internal state). If different heat-exchanger costing functions are available, one is randomly chosen.
- *Add a splitter.* This move consists first of adding an object of class SPL and an object of class MIX, at a randomly chosen point in a randomly chosen stream. If the nodes are side by side, then a heat exchanger is added on the newly created branch (note that a heat exchanger is only added on the newly created branch).
- *Remove a heat exchanger.* Two objects of class EXC, which represents the same exchanger, are removed. This move eliminates a split if one of its branches does not involve a heat exchange, (bypass is not allowed).
- *Modify a utility level.* This move consists of modifying the internal state of a randomly selected heating or cooling when several are available.
- *Modify a cost function of a heat exchanger.* When several costing expressions are available, this move also consists of modifying the internal state of a randomly chosen heat exchanger. Let us note that a save/restoration unit that allows a configuration to be saved before a new SA iteration and eventually to restore it if the move is rejected, has been implemented. Further, the initial network only involves utilities in order to guarantee a feasible network, and no other configuration is privileged.

Nonlinear Formulation

Having addressed the problem of generating all the feasible configurations for the HENS, the NLP formulation for the minimization of the global cost of a given configuration is presented in this section. All the information required to derive the NLP problem is contained in the internal state of each component of the network. So these parameters can be directly optimized without using external data. Since the structure is held constant for a given continuous problem, only the existing streams, variables, and constraints in this structure are taken into account in this NLP formulation.

Thus, the convergence rate for large-scale problems can be significantly improved. Before deriving the NLP formulation, the following sets are defined to characterize the topology of the current configuration.

- *Set of streams*

$$SC = \{i/i \in [1, NC]\}, \quad \text{the set of cold streams} \quad (7a)$$

$$SH = \{j/j \in [1, NH]\}, \quad \text{the set of hot streams} \quad (7b)$$

$$SHC = \{k/k \in SH \cup SC\}, \quad \text{the set of all streams.} \quad (7c)$$

For every cold stream $i \in SC$ and hot stream $j \in SH$, the capacity flow rates, and the inlet and outlet temperatures are denoted as FCp_i, T_i^I, T_i^O for the cold stream, and FCp_j, T_j^I, T_j^O for the hot stream.

- *Set of auxiliary cooling and heating*

$$SUC = \{ii/ii \in [1, NUC]\},$$

the set of available auxiliary cooling (8a)

$$SUH = \{jj/jj \in [1, NUH]\},$$

the set of available auxiliary heating (8b)

For every cold utility $ii \in SUC$, the heat capacity and the inlet and outlet temperatures are denoted as Cp_{ii}, T_{ii}^{UHI}, T_{ii}^{UHO}. For every hot utility $jj \in SUH$, represented by a heat exchanger, the latent heat $Hlat_{jj}$ at the temperature T_{jj}^{v} is given. In the case where the hot utility jj is represented by a furnace, the calorific power Pv_{jj} is specified.

- *Set of nodes*

$$SN_k = \{n/n \in (EXC, SPL, MIX, UTI),\ n \in [n^0, n^*]\},$$

the set of nodes of stream k, $k \in SHC$ (9a)

$$SEXC_k = \{e/e = EXC;\ e \in [e^0, e^*]\},$$

the set of exchangers of stream k, $k \in SHC$ (9b)

$$SSPL_k = \{s/s = SPL;\ s \in [s^0, s^*]\},$$

the set of splitters of stream k, $k \in SHC$ (9c)

$$SMIX_k = \{m/m = MIX;\ m \in [m^0, m^*]\},$$

the set of mixers of stream k, $k \in SHC$ (9d)

$$SUTI_i = \{u/u = UTI\},$$

the hot utility jj of cold stream i, $i \in SC$, $jj \in SUH$ (9e)

$$SUTI_j = \{u/u = UTI\},$$

the cold utility ii of hot stream j, $j \in SH$, $ii \in SUC$. (9f)

Note that the utility levels are fixed for a particular configuration.

- *Set of matches*

The available information to derive the NLP comes from the knowledge of topology and involves the following match set. For a given structure, the set of heat exchanges is known:

$$SRC = \{e/e \in SEXC_k,\ k \in SHC\},$$

the set of heat exchanges of the cold side (10a)

$$SRH = \{e'/e' \in SEXC_k,\ k \in SHC\},$$

the set of heat exchanges of the hot side (10b)

$$SR = \{ee'/e \in SEXC_i,\ e' \in SEXC_j/\text{cold side } i$$

heat exchanges with hot side j, $i \in SRC$, $j \in SRH\}$,

the set of all the heat exchanges. (10c)

Note that the cold or hot side of a heat exchanger does not necessarily represent a cold or hot stream. For a cold-to-cold or hot-to-hot match (case of a forbidden match), the cold side (hot side) of a heat exchanger represents a hot stream (cold stream).

Having defined the index sets describing a given configuration, the constraints linking the optimization variables (temperatures t and the split rates y) are in the general case the following:

- *Mass balance for splitters*

$$y_s^I = y_s^O + y_s^{O'} \qquad \forall s \in SSPL_k,\ k \in SHC. \quad (11)$$

- *Mass and heat balances for mixers*

$$y_m^I t_m^I + y_m^\Gamma t_m^\Gamma = y_m^O t_m^O \qquad \forall m \in SMIX_k,\ k \in SHC \quad (12a)$$

$$y_m^I + y_m^\Gamma = y_m^O. \quad (12b)$$

- *Mass and heat balances for heat exchangers*

$$Q_{ee'} = y_e FCp_e (t_e^O - t_e^I) = y_{e'} FCp_{e'} (t_{e'}^I - t_{e'}^O) \quad \forall ee' \in SR. \quad (13)$$

- *Mass and heat balances for hot utilities*

If the hot utility is represented by a heat exchanger, we have:

$$Q_u^{UC} = y_u FCp_u (t_u^O - t_u^I) = w_u Hlat_u \qquad \forall u \in SUTI_i,\ i \in SC. \quad (14a)$$

If the utility is represented by a furnace, we have:

$$Q_u^{UC} = y_u FCp_u (t_u^O - t_u^I) = w_u Pv_u \qquad \forall u \in SUTI_i,\ i \in SC. \quad (14b)$$

- *Mass and heat balances for cold utilities*

$$Q_u^{UH} = y_u FCp_u (t_u^I - t_u^O) = w_u Cp_u (t_u^{UHO} - T_u^{UHI}) \qquad \forall u \in SUTI_j,\ j \in SH \quad (15a)$$

with

$$T_u^{UHI} < t_u^{UHO} \le T_{\max_u}^{UHO} \quad (15b)$$

or

$$t_u^{UHO} = T_u^{UHO}. \quad (15c)$$

Equations 14a, 14b, and 15a are not considered as constraints. They are only used to evaluate the mass flow rate w_u of utility.

- *Minimum temperature approach constraints*

$$t^O_{e'} - t^I_e \geq \Delta T_{\min} \quad \text{and} \quad T^I_{e'} - t^O_e \geq \Delta T_{\min} \qquad \forall ee' \in SR \quad (16a)$$

$$T^V_u - t^I_u \geq \Delta T^{UC}_{\min} \quad \text{and} \quad T^V_u - t^O_u \geq \Delta T^{UC}_{\min} \qquad \forall u \in SUTI_i,\ i \in SC \quad (16b)$$

$$t^I_u - t^{UHO}_u \geq \Delta T^{UH}_{\min} \quad \text{and} \quad t^O_u - T^{UHI}_u \geq \Delta T^{UH}_{\min} \qquad \forall u \in SUTI_j,\ j \in SH. \quad (16c)$$

In order to compare our results with some published works, a constraint on the minimum difference temperature approach has been foreseen (in the literature $\Delta T_{\min}$ is generally fixed at 20°F). Let us note that for a pure synthesis problem, $\Delta T_{\min} = 0$°F is used. In that case, in the solution the minimum temperature approach will be optimal (and computed without reducing the search space *a priori*).

- *Minimum of heat load exchanged*

$$y_n FCp_n (t^O_n - t^I_n) \geq Q_{\min} \qquad \forall n \in \{SEXC_i \cup SUTI_i\},\ i \in SC \quad (17a)$$

$$y_n FCp_n (t^I_n - t^O_n) \geq Q_{\min} \qquad \forall n \in \{SEXC_j \cup SUTI_j\},\ j \in SH. \quad (17b)$$

These constraints have two meanings: on the one hand, they specify that a cold stream is to be heated and a hot stream is to be cooled; on the other hand, the specification of a heat load exchanged indirectly affects the minimum size of heat exchangers and consequently turns down solutions with very few heat exchangers.

- *Constraints on heat exchangers size*

$$y_e FCp_e (t^O_e - t^I_e)/(H_{ee'} LMTD_{ee'}) \geq A \min_{ee'} \qquad \forall ee' \in SR \quad (18a)$$

$$y_{e'} FCp_{e'} (t^I_{e'} - t^O_{e'})/(H_{ee'} LMTD_{ee'}) \leq A \max_{ee'} \qquad \forall ee' \in SR. \quad (18b)$$

$H_{ee'}$ represents the overall heat-transfer coefficient for a given match; $\text{LMTD}_{ee'}$ is the log mean temperature for a given match. These constraints permit us to specify a minimum and maximum size for a heat exchanger, and consequently to manage different cost functions depending on interval size for the area.

- *Specifications of inlet and outlet temperatures and split rates*

$$t^I_{no} = T^{CI}_i \quad \text{and} \quad t^O_{n*} = T^{CO}_i \qquad \forall n \in SN_i,\ i \in SC \quad (19a)$$

$$t^I_{no} = T^{HI}_j \quad \text{and} \quad t^O_{n*} = T^{HO}_j \qquad \forall n \in SN_j,\ j \in SH \quad (19b)$$

$$t^I_n = t^O_{n-1} \quad \text{and} \quad t^O_n = t^I_{n+1} \qquad \forall n \in SN_k,\ k \in SHC \quad (19c)$$

$$y^I_{no} = 1 \quad \text{and} \quad y^O_{n*} = 1 \qquad \forall n \in SK_k,\ k \in SHC \quad (19d)$$

$$y^I_n = y^O_{n-1} \quad \text{and} \quad y^O_n = y^I_{n+1} \qquad \forall n \in SN_k,\ k \in SHC. \quad (19e)$$

- *Specifications of bounding constraints*

$$T^{CI}_i \leq t^I_n \leq T^{CO}_i \quad \text{and} \quad T^{CI}_i \leq t^O_n \leq T^{CO}_i \qquad \forall n \in SN_i,\ i \in SC \quad (20a)$$

$$T^{HO}_j \leq t^I_n \leq T^{HI}_j \quad \text{and} \quad T^{HO}_j \leq t^O_n \leq T^{HI}_j \qquad \forall n \in SN_j,\ j \in SH \quad (20b)$$

$$0 \leq y_n \leq 1 \qquad \forall n \in SN_k,\ k \in SHC. \quad (20c)$$

The areas of heat exchangers can be expressed in terms of known heat loads and temperatures of streams, that is to say:

$$A_{ee'} = Q_{ee'}/(H_{ee'} LMTD_{ee'}) \qquad \forall ee' \in SEXC_k,\ k \in SHC \quad (21a)$$

$$A^{UC}_u = Q^{UC}_u/(H_u LMTD_u) \qquad \forall u \in SUTI_i,\ i \in SC \quad (21b)$$

$$A^{UH}_u = Q^{UH}_u/(H_u LMTD_u) \qquad \forall u \in SUTI_j,\ j \in SH. \quad (21c)$$

The objective function for minimizing the global cost (investment and operating cost) of the HEN is given by

$$\min \Sigma C^{EXC}_{ee'} + \Sigma C^{UC}_u + \Sigma C^{UH}_u \quad (22)$$
$$ee' \in SR;\quad u \in SUTI_i,\ i \in SC;\ u \in SUTI_j,\ j \in SH$$

with

$$C^{EXC}_{ee'} = a_{ee'} + b_{ee'} A^{C'_{ee'}}_{ee'} \quad (23a)$$

$$C^{UC}_u = a_u + b_u A^{UC^{c_u}}_u + U^{UC}_u \text{ for a heat exchanger} \quad (23b)$$

$$C^{UC}_u = a_u + b_u Q^{UC^{c_u}}_u + U^{UC}_u \text{ for a furnace} \quad (23c)$$

$$C^{UH}_u = a_u + b_u A^{UH^{c_u}}_u + U^{UH}_u, \quad (23d)$$

where U^{UC}_i and U^{UH}_j are the annual operating costs of cold and hot utilities; and a, b, and c are costing coefficients. The minimization of the objective function (Eq. 22), subject to the set of constraints (Eqs. 11 to 21), defines a classic nonlinear programming problem, where the variables to be optimized are the temperatures t and the split rates y. Let us note that the heat loads do not appear in the problem to be solved, because they are expressed in terms of temperatures and split rates.

Reducing the NLP Size

For a given configuration the solution of the preceding NLP problem provides the optimal values of variables (temperatures and split rates) with respect to the set of constraints (Eqs. 11–21). Then at the upper level, a new move performed by the SA procedure may modify only a part of the current configuration.. The new generated network is then composed of two parts: one part modified by the move, and eventually another part unchanged by the move, and consequently considered as optimal (this part can be empty). The algorithm implementation was carried out to analyze the structural modifications, and consequently to delete from the current NLP all the streams not affected by the move; on that account, the NLP size may considerably be reduced. An exam-

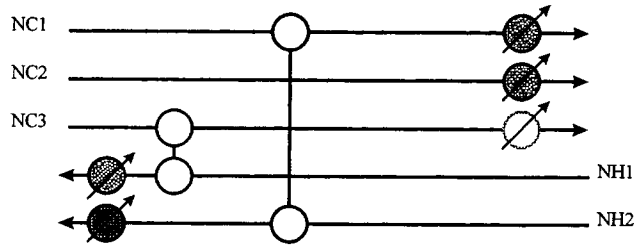

Figure 6. Example of reducing the NLP size.

ple is shown in Figure 6, where the current network, only a heat exchanger, is added between streams NC1 and NH2. The resulting network consists of an unchanged part (NC2, NC3, and NH1) that was previously optimized after the last move, and in a modified part (NC1 and NH2), which is concerned only with the NLP optimization.

Another feature for reducing the NLP size concerns the specifications of inlet and outlet temperatures and split rates. Temperatures are located between each node, split rates are only defined in the case where a split stream is present, and constraints specifying inlet and outlet temperatures and split rates are directly introduced in the problem formulation, insofar as they are equality constraints linking only two variables.

The last point for reducing the NLP size is related to the network feasibility. After each move, the network feasibility is checked by computing all constraints involving constant data and deleting the subset of verified constraints from the problem formulation. The procedure can be summarized as follows:

- Pick only one constraint concerning the minimum heat load exchanged (Eq. 17a or Eq. 17b).
- Delete constraints linked with constant data of the problem when they are verified.
- Suppress utilities when their associated constraints in the minimum temperature approach (Eqs. 16b and 16c) are ignored (with respect to the heat balance on the current structure).
- Pick constraints by the size of a heat exchanger only when they are specified in the problem formulation.

Let us consider, for example, a value of $\Delta T_{\min} = 20°$ and the part of the network represented in Figure 7. The hot stream NH1 and the cold stream NC1 verify the constraints

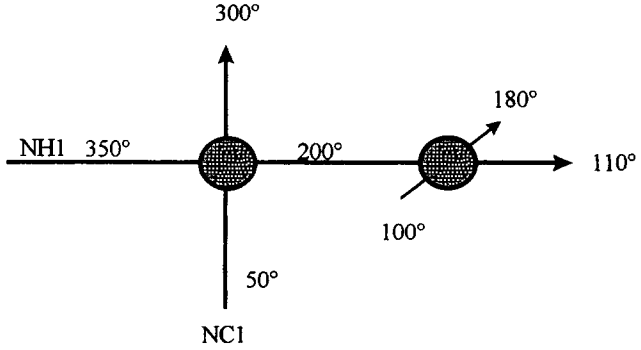

Figure 7. Example of the management of the constraints on minimum-temperature approach.

(Eq. 16a), while the output hot stream NH1 and the cold utility UTI violate the constraints (Eq. 16c). So the problem will be solved by considering that the utility for stream NH1 is not available. In the case where the minimum-temperature-approach constraint (Eq. 16a) cannot be verified (input temperature of the hot stream NH1 minus output temperature of the cold stream NC1 less than $\Delta T_{\min}$), the problem becomes unsolvable. So the move is rejected, by assigning an infinite value to the cost. These considerations lead to a problem of minimum size, whose formulation depends on the current configuration. Due to the form of the objective function, the problem is nonconvex, so a unique optimal solution cannot be guaranteed. Indeed, the solution obtained is only a local minimum, and several optimal solutions, depending on the initialization, can be reached for the same configuration. Fortunately, reducing the problem size makes the initialization phase easier and increases the convergence rate toward a solution. When convergence is obtained, the global cost of the current HEN is given directly by the optimal value of the objective function. However, due to the lack of such mathematical properties as convexity, the NLP problem may fail to converge. In that case, the following procedure has been implemented. Insofar as we have used an infeasible-path method (SQP procedure) to solve the NLP problem, when convergence fails the constraints can be satisfied or not. If they are, the last obtained values are considered to be the solution and the global cost is computed; if they are not, we assign an infinite value to the global cost (i.e., $\Delta C = \infty$). This procedure allows us to carry out a new move by the SA.

Initialization Procedure for the NLP

Several initialization procedures have been tested, but only the best one is reported here. Generally, HENs using heuristic rules are based on the method of Ponton and Donaldson (1974), which consists of matching the hottest hot stream with the highest cold target stream. In the developed method, based on this concept, the initial point favors matches with a high driving force. The initialization procedure can be summarized by the four following steps.

Step 1. For each heat exchanger $ee' \in SR$:
If $t_{e'}^I - t_e^O$ is constant, then $dt_{ee'} = t_{e'}^I - t_e^O - \Delta T_{\min}$.
If $t_{e'}^O - t_e^I$ is constant, then $dt_{ee'} = t_e^O - t_e^I - \Delta T_{\min}$.
Else $dt_{ee'} = (t_{e'}^I - t_e^O + t_{e'}^0 - t_e^I)/2 - \Delta T_{\min}$.

Step 2. For each heat exchanger $ee' \in SR$:

$$Q_{ee'} = \text{Min}\Big(FCp_i(T_i^O - T_i^I)\cdot dt_{ee'}/\Sigma dt_{ee'},\ FCp_j\big(T_j^I - T_j^O\big)\cdot dt_{ee'}/\Sigma dt_{ee'}\Big).$$

Step 3. If splitters are present in the network, the split rates are initialized by using Nishida's theorem (Nishida et al., 1971): "If the allocation of heat capacity flow rate and heat duty is optimal, then a fraction of hot substream, the cold substream, and the heat duty are equal within an exchanger." That is to say, if there is a split on stream i with two heat exchanger $e1$ and $e2$ between the streams i, $j1$, and $j2$ (see Figure 8), we have:

$$\frac{y_1}{y_1 + y_2} = \frac{Q_{e1}}{Q_{e1} + Q_{e2}}.$$

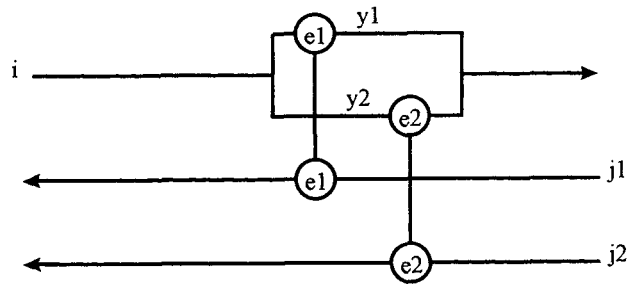

Figure 8. Nishida's theorem.

Step 4. For all matches, the initial temperatures are classically defined by:

$t_e^O = Q_{ee'}/FCp_e + t_e^I$ for the cold side of an exchanger.

$t_{e'}^O = t_{e'}^I - Q_{ee'}/FCp_{e'}$ for the hot side of an exchanger.

It can be noted that the preceding initialization procedure generally produces an initial feasible configuration. However, in a few cases the constraints on the minimum temperature approach may be violated in the initial (infeasible) network. This is not a major inconvenience because in the infeasible path SQP procedure, the search can be initialized with some violated constraints.

SA Implementation

An efficient implementation of the SA procedure strongly depends on some items, like the initial configuration, the initial pseudotemperature Tsa^0, the length of pseudotemperature *Nsa*, and the stopping criteria. We have already mentioned that the initial network comprises only external utilities. Concerning the pseudotemperature Tsa^0, its value generally depends on the average variations of the objective function and also on the acceptance probability of a climbing move at the beginning of the search, which is quite hard to estimate without some statistical information. The average cost change obviously lies between 0 and the global cost of the initial configuration. To select this value Tsa^0, in the first example we have tested several initial pseudotemperatures, each representing a fraction of the initial cost.

The number of moves *Nsa* attempted at each annealing temperature (i.e., the size of the reachable subspace from each annealing temperature) also has to be estimated. For a given network there are $NH \cdot NC$ possible ways of choosing a match between a hot and cold stream. As several matches between two hot and cold streams are permitted, the number of possible matches is always $k \cdot NH \cdot NC$, with $k \geq 1$. In the following example, we have tested $k = 1$ and $k = 2$.

Numerical Examples

Some combinatorial aspects

For HENs problems involving only heat exchangers (i.e., no mixers and splitters), the number of feasible structures is $(NH \cdot NC)!$ (Ponton and Donaldson, 1974). So for the 10SP1 problem the number of structures is $(5.5)! = 25! \approx 1 \cdot 55 \cdot 10^{25}$. For the 20SP1 problem, this combination increases to $(10 \cdot 10)! = 100! \approx 9 \cdot 3^{157}$. As a comparison, the number of grains of sand on the earth is estimated at 10^{74} (Wells, 1987). Furthermore, for networks involving mixers and splitters, the combination obviously grows, that is to say, the previous astronomical values are only lower bounds on the number of feasible networks.

Table 1. Number of Variables and Constraints for Reduced NLP

Problem	No. of Variables	No. of Constraints
5SP1	5	5
10SP1	12	17
20SP1	24	24

Concerning the NLP subproblem dimension, the procedure to reduce the size divides by about 3 the problem size compared to a classic formulation. The mean numbers (these numbers may vary from one given HEN to another) of variables and constraints are reported in Table 1, after the reducing size step, for 5SP1, 10SP1, and 20SP1 problems.

Example 1. The first example to adjust the parameters of the SA studied in detail, is the classic 10SP1 first introduced by Pho and Lapidus (1973). It involves five cold streams and five hot streams, and a cold and a hot utility are also available. Data are taken from Nishida et al. (1977); all results were obtained on an IBM RS6000 computer. Concerning the NLP solution, the SQP package used is the subroutine NOONF of the library IMSL, working in double precision (Schittowski, 1986).

Table 2 gives the results for the annealing schedule of Kirkpatrick (1983), and the results for the annealing schedule of Aarts and van Laarhoven (1985) are reported in Table 3. The initial cost C^0 of the 10SP1 HEN (which comprises only utility) was \$38,606/year. The initial pseudotemperature Tsa^0 is initialized at $C^0/2$, $C^0/4$, $C^0/8$, and $C^0/16$. Values of the reducing factor of temperature α (Eq. 4) and speed parameter δ (Eq. 5) are classic values generally reported in the literature. Two values for the length of temperature, corresponding to $NH \cdot NC$ and $2 \cdot NH \cdot NC$ (that is to say $k = 1$ and 2) are tested. The average cost represents the arithmetical mean of all the results computed, and the second value is the standard deviation. For each set of values α, δ, Tsa^0, and *Nsa*, the best cost reported is the arithmetical mean of near optimal solutions computed over 15 SA runs.

From Tables 2 and 3, several interesting points can be noted. Due to its stochastic nature, the SA procedure leads to different results, but the same minimal solution (cost \$43,856/year), is obtained from the two annealing schedules. This solution, shown in Figure 9, is obtained for Kirkpatrick's schedule with ($\alpha = 0.9$, $Tsa^0 = C^0/8$, $Nsa = 50$), and with (($\delta = 1.0$, $Tsa^0 = C^0/4$, $Nsa = 50$) and ($\delta = 1.0$, $Tsa^0 = C^0/16$, and $Nsa = 50$)) for the Aarts and van Laarhoven's schedule. This solution is \$22 less than the best solution reported in the literature (Floudas and Ciric, 1989; Yee et al., 1990a).

It can also be observed that when the annealing time increases, the quality of the solution is generally improved, because the size of the explored search space increases with the computational time.

Concerning the initial pseudotemperature, values $C^0/2$, $C^0/4$, and $C^0/8$ give similar results. This shows that the value $C^0/8$ is high enough to guarantee an efficient randomization

Table 2. Results for the Annealing Schedule of Kirkpatrick

α	Initial *Tsa* ($/yr)	*Nsa*	Avg. Cost ($/yr)	Avg. CPU Time (s)
0.98	193,031	25	43,955 ± 22	1216
0.98	193,031	50	43,948 ± 12	2167
0.98	96,515	25	43,956 ± 11	1036
0.98	96,515	50	43,974 ± 68	2033
0.98	48,257	25	43,956 ± 11	1011
0.98	48,257	50	43,953 ± 12	1900
0.98	24,128	25	43,960 ± 20	1027
0.98	24,128	50	43,955 ± 11	1816
0.95	193,031	25	44,020 ± 108	471
0.95	193,031	50	43,988 ± 72	927
0.95	96,515	25	44,131 ± 197	432
0.95	96,515	50	43,995 ± 84	857
0.95	48,257	25	43,998 ± 84	565
0.95	48,257	50	43,955 ± 11	683
0.95	24,128	25	44,075 ± 145	400
0.95	24,128	50	44,062 ± 145	800
0.9	193,031	25	44,166 ± 154	263
0.9	193,031	50	44,122 ± 286	456
0.9	96,515	25	44,435 ± 483	284
0.9	96,515	50	44,131 ± 262	456
0.9	48,257	25	44,063 ± 125	212
0.9	48,257	50	43,985 ± 129	471
0.9	24,128	25	44,170 ± 141	205
0.9	24,128	50	44,191 ± 332	575
0.85	193,031	25	44,540 ± 754	179
0.85	193,031	50	44,061 ± 194	298
0.85	96,515	25	44,259 ± 315	219
0.85	96,515	50	44,057 ± 136	297
0.85	48,257	25	45,044 ± 952	231
0.85	48,257	50	44,107.194	258
0.85	24,128	25	44,512 ± 440	130
0.85	24,128	50	44,075 ± 131	272
0.80	193,031	25	44,502 ± 507	123
0.80	193,031	50	44,338 ± 530	290
0.80	96,515	25	44,757 ± 541	128
0.80	96,515	50	44,438 ± 334	235
0.80	48,257	25	44,248 ± 367	122
0.80	48,257	50	44,097 ± 124	265
0.80	24,128	25	44,346 ± 388	134
0.80	24,128	50	44,554 ± 530	198
0.70	193,031	25	47,247 ± 6163	83
0.70	193,031	50	44,294 ± 322	161
0.70	96,515	25	46,027 ± 3679	150
0.70	96,515	50	44,494 ± 471	165
0.70	48,257	25	44,858 ± 800	82
0.70	48,257	50	44,428 ± 572	200
0.70	24,128	25	45,689 ± 3744	84
0.70	24,128	50	46,453 ± 4237	214
0.60	193,031	25	44,388 ± 370	91
0.60	193,031	50	44,933 ± 710	135
0.60	96,515	25	46,044 ± 856	62
0.60	96,515	50	44,750 ± 856	111
0.60	48,257	25	46,102 ± 67	66
0.60	48,257	50	44,724 ± 561	156
0.60	24,128	25	44,479 ± 462	46
0.60	24,128	50	44,496 ± 342	100
0.50	193,031	50	45,241 ± 1758	76
0.50	193,031	25	45,459 ± 1191	40
0.50	96,515	50	45,522 ± 1581	93
0.50	96,515	25	46,970 ± 4340	43
0.50	48,257	50	44,923 ± 1046	85
0.50	48,257	25	46,804 ± 3354	51
0.50	24,128	50	46,208 ± 4447	98
0.50	24,128	25	47,188 ± 4400	48

Table 3. Results for the Annealing Schedule of Aarts and van Laarhoven

δ	Initial *Tsa* ($/yr)	*Nsa*	Avg. Cost ($/yr)	Avg. CPU Time (s)
1.0	193,031	25	43,952 ± 12	1520
1.0	193,031	50	43,951 ± 13	3273
1.0	96,515	25	43,976 ± 68	1583
1.0	96,515	50	43,938 ± 29	2996
1.0	48,257	25	43,947 ± 10	1443
1.0	48,257	50	43,950 ± 11	3313
1.0	24,128	25	43,961 ± 7	1604
1.0	24,128	50	43,941 ± 32	3338
0.5	193,031	25	43,948 ± 18	2105
0.5	193,031	50	43,944 ± 10	2981
0.5	96,515	25	43,956 ± 21	2134
0.5	96,515	50	43,949 ± 12	3313
0.5	48,257	25	43,956 ± 10	2040
0.5	48,257	50	43,949 ± 11	4475
0.5	24,128	25	43,954 ± 11	2030
0.5	24,128	50	43,956 ± 11	4375

of the search space, and a smaller value like $C^0/16$ freezes the procedure, and the convergence toward a good solution fails.

A large number of local solutions is obtained with both annealing schedules. With regard to average costs and standard deviations that characterize the search efficiency for Kirkpatrick's schedule, the reducing factor α must be chosen above 0.9. In a general way, we can note, on the one hand, an improvement in the results when the length of pseudotemperature is equal to 50, and on the other hand, an increase of CPU time for high values of Tsa^0.

For the annealing schedule of Aarts and van Laarhoven, all values of the speed parameter δ lead to good results, but with CPU time much higher than in the previous schedule. Results are similar for the two values of the length of tem-

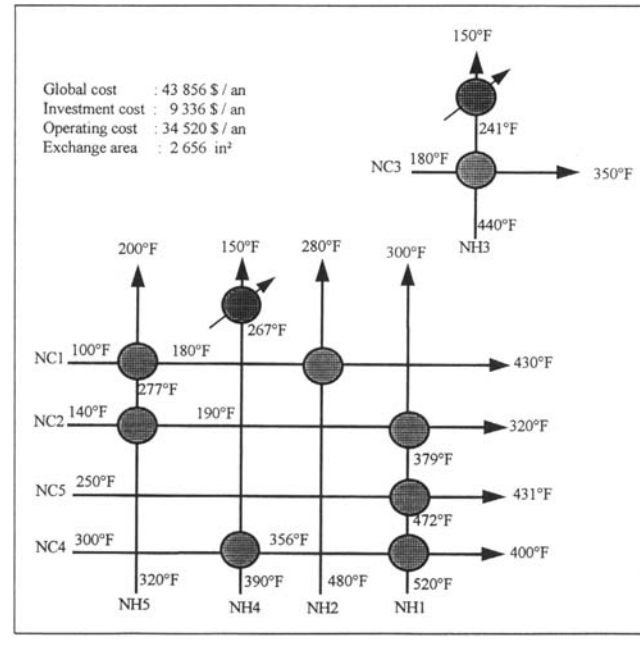

Figure 9. Optimal solution for the 10SP1 problem.

perature *Nsa*, and no clear conclusion can be drawn from the influence of the initial value of the pseudotemperature.

In conclusion the annealing schedule of Kirkpatrick seems better suited to the problem under consideration, insofar as a good fit of SA parameters can be made in terms of CPU times and the quality of results. A good arrangement consists of taking α equal to 0.95, *Nsa* at 50 (that is to say, $2 \cdot NH \cdot NC$), and $Tsa^0 = C^0/8$. For these values, the number of moves required to reach the solution is about 10,500. These values, obtained from a statistical analysis, are also used in the two following examples. Let us note that for the 10SP1 problem, these statistical estimates don't lead to the best solution (obtained for $\alpha = 0.9$, $Nsa = 2 \cdot NH \cdot NC$, and $Tsa^0 = C^0/8$).

Concerning the CPU time, about 98% of it is consumed by the nonlinear programming phase. For example, for the preceding values ($\alpha = 0.95$, $Nsa = 50$, and $Tsa^0 = C^0/8$), the CPU time is 683 s (see Table 2) for 10,500 moves, that is to say, the mean CPU time for optimizing a given network by the NOONF code is about 0.06 s. The remaining 2% is related to the generation of feasible configurations by the SA procedure by using efficient pointers and linked lists. Obviously, the CPU times strongly depend on the problem size, but the proportion 98%–2% remains quite constant.

About the termination test for the SA procedure, we have implemented Dolan et al.'s (1989) test in the three examples presented; that is to say, the search ends when the temperature becomes less than unity.

Example 2. This example, first presented by Masso and Rudd (1969) and called 5SP1, involves three cold streams, two hot streams, and one hot-and-cold available utility. It is an interesting case not for its size, but because it includes a forbidden match between streams *NC*1 and *NH*2, which implies possibilities of cold-to-cold or hot-to-hot matches.

As usual, the initial network comprises only utilities, and data are taken from Nishida et al. (1977). For managing the increase in the search space dimension due to the possibilities of cold-to-cold or hot-to-hot matches, the recommended value of the temperature length ($2 \cdot NH \cdot NC$) has been doubled, so its value is fixed at 24. The other paramters ($\alpha = 0.95$, $Tsa^0 = \$32{,}000$/year) and the termination test are the same as in the previous case.

In order to compare our results with the published ones, we have taken a minimum temperature approach of 20°F(−6.7°C). The optimal solution, represented in Figure 10, has a cost of \$39,649/year, and is found in 25 s of CPU time. This example shows the capability of SA to converge to a good solution, insofar as a solution of cost \$50,340/year was found by Linnhoff and Flower (1978), and another at a cost of \$39,706/year, by Grimes et al. (1982).

Example 3. This example, called 20SP1, involves 10 hot streams, 10 cold streams, and a hot and a cold utility one. The data are taken from Grossmann and Sargent (1978), and the problem is initialized as in the previous cases. The parameters of SA are the following: $\alpha = 0.95$, $Tsa^0 =$ \$53,620/year, $Nsa = 200$.

The problem was solved within 3.30 h CPU time, the optimal cost is \$23,230/year (\$23,432/year from Grossmann and Sargent (1978) and \$23,887/year from Belkebir (1989)). This optimal network is represented on Figure 11, where only input and output temperatures are given.

The solution is reached within 42,201 moves, only 587 in-

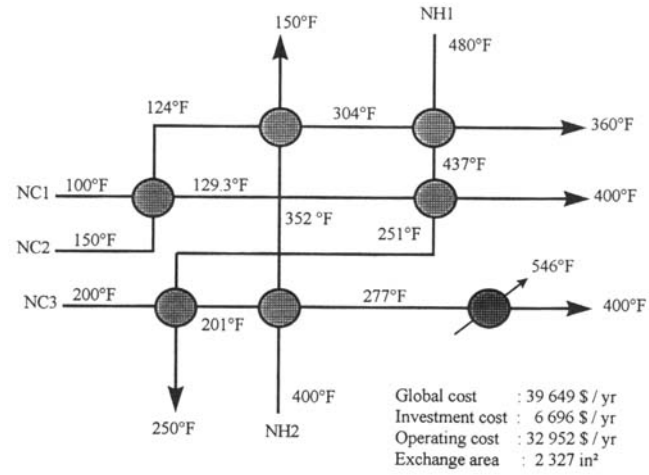

Figure 10. Optimal solution for the 5SP1 problem.
SI conversion: °C = (°F − 32)/1.8; mm = in. × 25.4.

feasible structures are generated, and 10,659 moves are accepted. Among these accepted moves, 5010 correspond to a heat exchanger addition, 2561 to a heat exchanger removal, and 3088 to a separator addition. In Figure 12, the global cost vs. the accepted moves is reported. It can be noted that the number of accepted moves is important at the beginning of the search because the annealing temperature is high. Then the objective function slowly decreases.

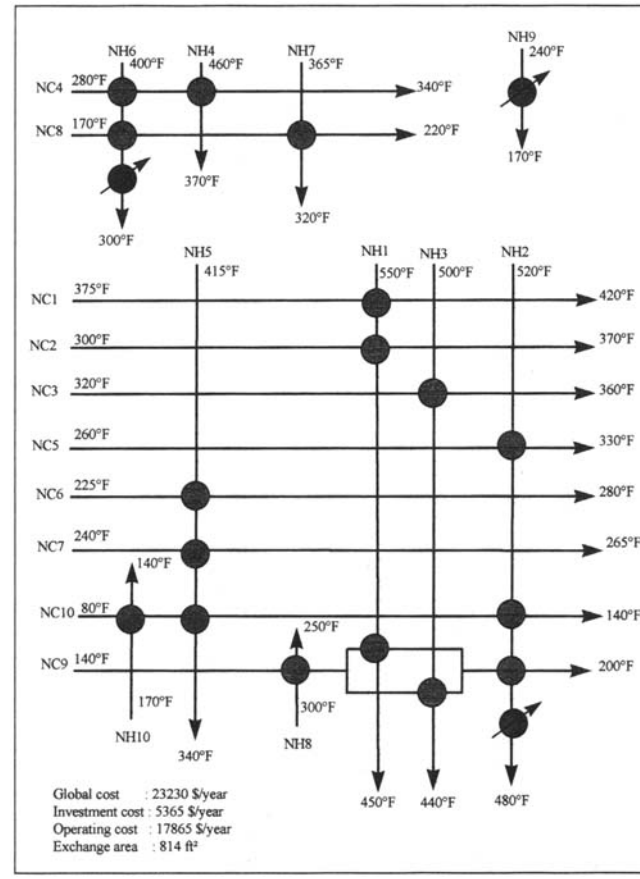

Figure 11. Optimal solution for the 20SP1 problem.
SI conversion: °C = (°F − 32)/1.8.

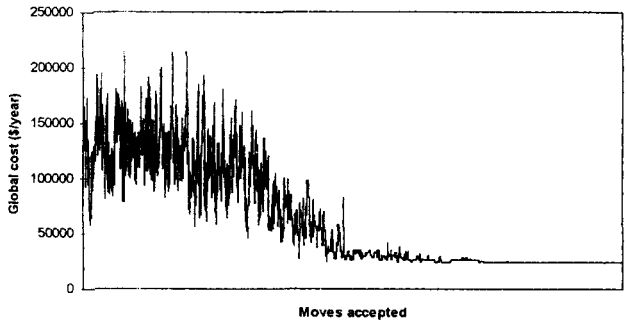

Figure 12. Evolution of the SA algorithm.

Conclusion

A new approach for the HENs problems was presented. On an upper level, the simulated annealing is implemented for generating HEN structures. This information is then used in a lower level within a nonlinear programming SQP frame to optimize operating variables. The most relevant points of this study are the use of pointers, linked lists, and data structures for model implementation. Different SA parameters have been statistically studied in order to fix their values. The annealing schedule of Kirkpatrick appears to be well suited to the problem under consideration, in terms of CPU time and consistency of results. Three examples were presented to illustrate the proposed approach and to compare the obtained solutions with some published works.

Notation

a, b, c = cost parameters
H = overall heat transfer coefficient
$LMTD$ = log mean temperature
Q = heat load exchanged
Q_{min} = minimum heat load exchanged

Subscripts and superscripts

ee' = relative to a heat exchanger
e = relative to the cold side of an exchanger
e' = relative to the hot side of an exchanger
k = relative to a process stream
m = relative to a mixer
n = relative to a node
s = relative to a splitter
I' = relative to a second input
O' = relative to a second output
0 = relative to the first temperature of a given stream
$*$ = relative to the last temperature of a given stream

Literature Cited

Aarts, E. H. L., and P. J. M. van Laarhoven, "Statistical Cooling: A General Approach to Combinatorial Optimization Problems," *Philips J. Res.*, **40**, 193 (1985).

Belkebir, M. K., "Conception de Réseaux d'Échangeurs de Chaleur," PhD Thesis, Inst. Nat. Polytech. de Toulouse, Toulouse (1989).

Ciric, A. R., and C. A. Floudas, "Heat Exchanger Network Synthesis without Decomposition," *Comput. Chem. Eng.*, **15**(6), 385 (1991).

Das, H., P. T. Cummings, and M. D. Le Van, "Scheduling of Serial Multiproduct Batch Processes via Simulated Annealing," *Comput. Chem. Eng.*, **14**(12). 1351 (1990).

Dolan, W. B., P. T. Cummings, and M. D. Le Van, "Heat Exchanger Network Design by Simulated Annealing," *Proc. Conf. on Foundations of Computer-Aided Process Operations*, Park City, Utah (1987).

Dolan, W. B., P. T. Cummings, and M. D. Le Van, "Process Optimization via Simulated Annealing: Application to Network Design," *AIChE J.*, **35**, 5 (1989).

Dolan, W. B., P. T. Cummings, and M. D. Le Van, "Algorithmic Efficiency of Simulated Annealing for Heat Exchanger Network Design," *Comput. Chem. Eng.*, **14**(20), 1039 (1990).

Floudas, C. A., A. R. Ciric, and I. E. Grossmann, "Automatic Synthesis of Optimum Heat Exchanger Network Configurations," *AIChE J.*, **32**, 2 (1986).

Floudas, C. A., and A. R. Ciric, "Strategies for Overcoming Uncertainties in Heat Exchanger Network Synthesis," *Comput. Chem. Eng.*, **13**, 1133 (1989).

Glauber, R. J., "Time-Dependent Statistics of the Ising Model," *J. Math. Phys.*, **4**, 294 (1963).

Grimes, E., M. D. Rychener, and W. Westerberg, "The Synthesis and Evolution of Networks of Heat Exchange that Feature the Minimum Number of Units," *Chem. Eng. Commun.*, **14**, 339 (1982).

Grossmann, I. E., and R. W. H. Sargent, "Optimum Design of Heat Exchanger Networks," *Comput. Chem. Eng.*, **2**, 1 (1978).

Guiglion, C., L. Pibouleau, and S. Domenech, "Thermodynamique des Réseaux d'Échangeurs de Chaleur, et Possibilité de Diminuer le Nombre d'Échangeurs dans ces Réseaux.—I. Thermodynamique des Réseaux d'Échangeurs," *Int. J. Heat Mass Transfer*, **35**(6), 1349 (1992).

Gundersen, T., and L. Naess, "The Synthesis of Cost Optimal Heat Exchanger Networks. An Industrial Review of the State of the Art," *Comput. Chem. Eng.*, **12**(6), 503 (1988).

Hohmann, E. C., "Optimum Networks for Heat Exchange," PhD Thesis, Univ. of Southern California (1971).

Jezowski, J., "Heat Exchanger Network Grassroot and Retrofit Design. The Review of the State of the Art: Part II. Heat Exchanger Network Synthesis by Mathematical Methods and Approaches for Retrofit Design," *Hung. J. Ind. Chem.*, Veszprém, **22**, 295 (1994).

Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, **220**, 671 (1983).

Linnhoff, B., and J. R. Flower, "Synthesis of Heat Exchanger Networks: I. Systematic Generation of Energy Optimals Networks," *AIChE J.*, **24**, 4 (1978).

Linnhoff, B., and E. Hindmarsh, "The Pinch Design Method for Heat Exchanger Networks," *Chem. Eng. Sci.*, **38**, 745 (1983).

Masso, A. H., and D. F. Rudd, "The Synthesis of System Designs: II. Heuristic Structuring," *AIChE J.*, **23**, 1 (1969).

Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *J. Chem. Phys.*, **27** (1953).

Nishida, N., S. Kobayashi, and A. Ichikawa, "Optimal Synthesis of Heat Exchange Systems. Necessary Conditions for Minimum Heat Transfer Area and Their Application to System Synthesis," *Chem. Eng. Sci.*, **26**, 1841 (1971).

Nishida, N., Y. A. Liu, and L. Lapidus, "Studies in Chemical Process Design and Synthesis: III. A Simple and Practical Approach to the Optimal Synthesis of Heat Exchanger Networks," *AIChE J.*, **23**, 1 (1977).

Painton, L. A., and U. M. Diwekar, "Synthesizing Optimal Design Configurations for a Brayton Cycle Power Plant," *Comput. Chem. Eng.*, **18**(5), 369 (1994).

Papoulias, A., and I. E. Grossmann, "Structural Optimization Approach in Process Synthesis: II. Heat Recovery Networks," *Comput. Chem. Eng.*, **7**(6), 707 (1983).

Patel, A. N., R. S. H. Mah, and I. A. Karimi, "Preliminary Design of Multiproduct Non Continuous Plants Using Simulated Annealing," *Comput. Chem. Eng.*, **15**(7), 451 (1991).

Pho, T. K., and L. Lapidus, "Topics in Computer-Aided Design: II. Synthesis of Optimal Heat Exchanger Networks by Tree Searching Algorithms," *AIChE J.*, **19**, 1182 (1973).

Ponton, J. W., and R. A. B. Donaldson, "A Fast Method for the Synthesis of Optimal Heat Exchanger Network," *Chem. Eng. Sci.*, **29**, 2375 (1974).

Schittkowski, K., "NLPQ: A Fortran Subroutine Solving Constrained Nonlinear Programming Problems," *Ann. Oper. Res.*, **5**, 485 (1986).

Trivedi, K. K., B. K. Neill, J. R. Roach, and R. M. Wood, "A New Dual-Temperature Design Method for the Synthesis of Heat Exchanger Networks," *Comput. Chem. Eng.*, **13**, 667 (1989).

Townsend, D. W., and B. Linnhoff, "Surface Area Targets for Heat Exchanger Networks," *Ind. Chem. Eng. Annu. Research Meeting*, Bath, England (1984).

Wells, D., *The Penguin Dictionary of Curious and Interesting Numbers*, Penguin, Baltimore (1987).
Wiswanathan, M., and L. B. Evans, "Studies in the Heat Integration of Chemical Process Plants," *AIChE J.*, **33**(11), 1781 (1987).
Yee, T. F., I. E. Grossmann, and Z. Kravanja, "Simultaneous Optimization Models for Heat Integration: I. Area and Energy Targeting and Modeling of Multi-stream Exchangers," *Comput. Chem. Eng.*, **14**(10), 1151 (1990).
Yee, T. F., and I. E. Grossmann, "Simultaneous Optimization Models for Heat Integration: II. Heat Exchanger Network Synthesis," *Comput. Chem. Eng.*, **14**(10), 1165 (1990).

Appendix: Internal State of the Different Objects

We have represented the internal states of the four different objects. The global part class is one of the components of the four application classes.

Global Part Class

- Class of the object (UTI, EXC, SPL, or MIX)
- Name of the stream
- Is it a hot or a cold stream?
- Three pointers on these objects

UTI Class

- Global part class
- Name and type of the utility (heat exchanger, furnace, etc.)
- Index of constraints for minimum temperature approach (Eq. 16b or Eq. 16c)
- Pointers on the optimization variables: y_u, t_u^I, and t_u^O
- Cost variables: a_u, b_u, and c_u
- Different economical parameters
- Latent heat, heat capacity, or calorific power: $Hlat_u$, Cp_u, or Pv_u
- Heat load exchanged: Q_u^{UC} or Q_u^{UH}
- Optimal operating parameters y_u, t_u^I, t_u^O, and w_u
- Global cost: C_u^{UC} or C_u^{UH}
- In a case of a heat exchanger:
  Minimum temperature approach $\Delta T_{\min}^{UC}$ or $\Delta T_{\min}^{UH}$
  Overall heat transfer coefficient, H_u
  Input and output temperature: T_u^V or T_u^{UCE} *and* T_u^{UCS}

EXC Class

- Global part class
- Name of the heat exchanger
- Cost variables: $a_{ee'}$, $b_{ee'}$, and $c_{ee'}$
- Pointers on the optimization variables: y_e, t_e^I and t_e^O
- Index of constraint for mass and heat balances (Eq. 13).
- Index of constraint for minimum-temperature approach (Eq. 16a)
- Index of constraints for heat load exchanges (Eqs. 17a and 17b)
- Index of constraints for heat exchanger size (Eqs. 18a and 18b)
- Minimum-temperature approach $\Delta T_{\min}$ and minimum heat load exchanged $Q_{\min}$
- Minimum and maximum size of exchange area $A_{\min ee'}$ and $A_{\max ee'}$
- Overall heat-transfer coefficient $H_{ee'}$
- Optimal operating parameters y_e, t_e^I, and t_e^O
- Heat load exchanged: $Q_{ee'}$
- Exchange area: $A_{ee'}$
- Investment cost: $C_{ee'}^{EXC}$

SPL Class

- Global part class
- Pointers on the optimization variables: t_s, y_s^I, y_s^O and $y_s^{O'}$
- Index of constraint for mass and heat balances (Eq. 11)
- Optimal operating parameters: t_s, y_s^I, y_s^O, and $y_s^{O'}$

MIX Class

- Global part class
- Pointers on the optimization variables: t_m^I, $t_m^{I'}$, t_m^O, y_m^I, $y_m^{I'}$, and y_m^O
- Index of constraints for mass and heat balances (Eqs. 12a and 12b)
- Optimal operating parameters: t_m^I, $t_m^{I'}$, t_m^O, y_m^I, $y_m^{I'}$, and y_m^O

Manuscript received Feb. 18, 1997, and revision received June 27, 1997.