HENSSA: A PROTOTYPE CAD PACKAGE FOR HEAT EXCHANGER NETWORK SYNTHESIS BY SIMULATED ANNEALING

W. E. Dolan, P. T. Cummings and M. D. LeVan
Department of Chemical Engineering
University of Virginia
Charlottesville, Virginia

ABSTRACT

Simulated annealing, a new multivariable optimization technique developed for designing very large scale integrated circuits, has recently been extended and applied with great success to the design of heat exchanger networks (Dolan et al., 1987, 1988a). In this paper, the simulated annealing algorithm for heat exchanger network design and its successes are briefly reviewed. A prototype for an interactive package for implementing the simulated annealing algorithm, called HENSSA (Heat Exchanger Network Synthesis by Simulated Annealing), is described. HENSSA is written in C and runs within the Suntools windowing environment on a Sun engineering workstation. Its features include the ability to enter or modify a heat exchanger network using a mouse-driven user interface, to choose options for the simulated annealing process from pull-down menus and to interrupt the annealing process at any time to permit user modifications of the network.

NOMENCLATURE

Symbol | Meaning
---|---
N | number of hot and cold streams in a heat exchanger design problem
q | heat load (in units of energy per unit time) on a heat exchanger
T_{sa} | simulated annealing temperature
ΔC | difference in cost between two heat exchanger network designs

INTRODUCTION

The heat exchanger network (HEN) synthesis problem in process engineering can be stated succinctly as follows: Find the combination of heat exchangers and utilities (usually steam and cooling water) that minimizes the overall (annualized capital plus operating) cost of meeting the heating and cooling requirements in a process. As a result of the sharp rise in energy costs in the early 1970s, HEN synthesis has become one of the most studied design synthesis problems in process engineering. The four main approaches to HEN synthesis developed over the past one and a half decades are:

1. the evolutionary/heuristic approach of Liu and coworkers (Nishida et al., 1977)
2. the energy pinch concept developed by Linnhoff and
co–workers (Linnhoff and Flower, 1978; Linnhoff and Hindmarsh, 1983) based on thermodynamic arguments about heat transfer across the so–called pinch point in an enthalpy–temperature diagram.

3. the mixed integer linear programming (MILP) and mixed integer non–linear programming (MINLP) approaches of Grossmann and Floudas (Papoulias and Grossman, 1983; Floudas et al., 1986)

4. the simulated annealing algorithm developed by Dolan, Cummings and LeVan (1987, 1988a).

Excellent reviews of first three of these approaches are given by Gundersen and Naess (1988) and Liu (1987). As it is the most recently developed HEN synthesis methodology, simulated annealing is not described in these reviews. The most complete description of simulated annealing for HEN synthesis is given by Dolan et al. (1988a) to which the reader is referred.

The first three methods all rely to some extent on the use of heuristics. Heuristics are clearly the foundation of the heuristic/evolutionary approach. The pinch technology approach begins with the assumption that a network with minimum energy consumption and minimum number of units is nearly cost–optimal, and that a cost–optimal network can be found by limited relaxation of these assumptions. The MILP and MINLP approaches generally assume that the pinch technology heuristics of no heat transfer across the pinch and minimum number of units characterize cost–optimal networks and use these heuristics to limit the search space in the minimization. In contrast, simulated annealing assumes no heuristics and searches the feasible solution space efficiently to find the cost–optimal (or very nearly cost–optimal) network.

Simulated annealing is a combinatorial optimization technique based on the theory of Markov processes. The method was developed by Kirkpatrick et al. (1983) for the optimal design of very large scale integrated circuits. One objective in VLSI design is to minimize the wire length and at the same time not allow the local wire density to become high enough to cause signal corruption. An analogy between physical annealing (the cooling of atoms into a solid state) and combinatorial optimization is drawn. This analogy introduces a fictitious temperature, the “annealing temperature” \( T_{\text{rs}} \), into the combinatorial optimization process and draws a parallel between cost of a design and the energy of a physical system. Simulated annealing amounts to starting an optimization problem at a high annealing temperature, allowing the variables which characterize the design to come to equilibrium at each annealing temperature by changing them according to the Metropolis algorithm (Metropolis et al., 1953) used in statistical mechanics to equilibrate the positions of molecules in a Monte Carlo simulation of a physical system, and slowing lowering the annealing temperature until the design variables “freeze” into their lowest cost state.

In developing a simulated annealing algorithm for HEN synthesis, Dolan et al. faced two difficulties. The first difficulty is that in HEN synthesis one does not know a priori how many process and utility heat exchangers will be required in the final network, nor are the areas of these known. Thus there is a structural optimization problem (the number of exchangers and their placement within the network) as well as a parametric optimization problem (the areas—and, therefore, heat loads—of the exchangers). This difficulty was solved by drawing an analogy between optimization of HENs and physical annealing of open systems which are free to exchange particles with their surroundings at rates determined by the chemical potentials of the particles. In statistical mechanics, such systems are said to be grand canonical. By modifying the Monte Carlo simulation algorithm for the grand canonical ensemble (Adams, 1975), Dolan et al. developed the grand canonical simulated annealing (GCSA) algorithm (Dolan et al., 1987, 1988a) which has proven to be very successful in producing cost–optimal heat exchanger networks.

The second difficulty is that the efficiency of simulated annealing as a design algorithm is predicated on developing a fast method for the evaluation of the difference between the cost of two designs. By using a linked list data structure (Dolan et al., 1988b, 1989; Dolan, 1989), Dolan et al. were able to find an \( O(N) \) algorithm (where \( N \) is the number of streams in the HEN synthesis problem) for computing the change in cost between two designs. This yielded a two orders of magnitude increase in speed between early versions of GCSA and the most recent version, and permitted the implementation of GCSA on an entry level engineering workstation (a Sun 3/50).

The GCSA algorithm is described in detail elsewhere (Dolan et al., 1987, 1988a; Dolan 1989). Briefly, the algorithm is as follows:

1. Initialize the locations and heat loads of exchangers. Usually, this means starting with a far–from–optimal network containing no process heat exchangers and with all heating and cooling duties provided by utilities.

2. Modify the network randomly by one of the following network modifications (which we refer to as moves):
   (a) add an exchanger at a randomly chosen point in the network
   (b) delete a randomly chosen exchanger
In some cases, the HEN found by GCSA is only slightly lower in cost than that obtained by other techniques; even in these cases, however, the GCSA algorithm has usually found a network that is inconsistent with one or more of the heuristics implicit in the other design methodologies. In the case of constrained network problems (HEN synthesis problems where pairs of streams are excluded from matching because of safety and/or physical layout considerations), GCSA has found designs as much as 30% lower in cost than previously published designs by non-intuitive matching of streams. These matches exchange heat between streams so that a stream which must ultimately be heated (cooled) is first cooled (heated). Dolan et al. refer to these as hot-to-hot and cold-to-cold matches (Dolan et al., 1987, 1988a). They were found independently within a MILP formulation by Viswanathan and Evans (1987).

An example is shown in Figure 1. This HEN synthesis problem is known as the constrained 4SP1 problem (Papoulis and Grossmann, 1983) and consists of two hot streams (streams 1 and 2 which start at high temperatures—320 °F and 480 °F respectively—and which must be cooled to low temperatures—200 °F and 280 °F) and two cold streams (streams 3 and 4 which start at low temperature—140 °F and 240 °F respectively—and which must be heated up to high temperatures—320 °F and 500 °F). The heat capacity flow rates (equal to the product of heat capacity and flow rate for each stream) are shown in the right hand side of the figure. The constraint in this HEN synthesis problem is that stream 1 and stream 3 cannot be matched with each other for heat transfer. [This may be due to potential hazards resulting from the chemical composition of the streams or may reflect a layout difficulty.] Heat exchangers are shown in the network as vertical connections between streams. Intermediate temperatures are written on each stream segment. Hot and cold utilities are denoted by circles containing the letters H and C respectively. The solution of Papoulis and Grossmann is shown in the upper half of the figure. The simulated annealing solution was found using the GCSA algorithm described above. The initial design used as the starting point for GCSA had no heat exchangers and all heating and cooling requirements were met by utilities. This is a far from optimal design. The GCSA algorithm was executed without intervention by the user and produced the design shown in the lower half of Figure 1 featuring a cold-to-cold match between streams 3 and 4. The annualized cost of the network obtained by GCSA is 35% lower than the design obtained by Papoulis and Grossmann (1983) who used a mixed integer programming method. Papoulis and Grossmann did not anticipate hot-to-hot and cold-to-cold matches in the

3An additional constraint is that the minimum approach temperature for all exchangers in this HEN synthesis problem s set at 18 F. This is an easily relaxed constraint in GCSA (Dolan et al., 1988a).

Figure 1: Solutions to constrained 4SP1 HEN synthesis problem: top, solution found by Papoulis and Grossman (1983); bottom, solution found by simulated annealing (Dolan et al., 1987)

(c) add a random amount of heat load \( q \) to an exchanger

(d) delete a random amount of \( q \) from an exchanger

(e) shift a random amount of \( q \) from one exchanger to another

(f) add a split with an exchanger at a randomly chosen point in the network.

3. Accept the new network design with probability

\[
\exp\left(-\frac{\Delta C}{T_m}\right)
\]

4. Go to 2, periodically reducing \( T_m \).

Note that in step 3 the new network configuration is accepted automatically if \( \Delta C \), the difference in the cost between the new and the old networks, is negative (i.e., if the new network is lower in cost than the old network). The cost function used in most of our work is that of Nishida et al. (1977) (in which cost is a non-linear function of heat transfer area), although one of the strengths of the GCSA algorithm is that the cost function can be arbitrarily complicated and need not be neither a continuous nor monotonic function of exchanger area (Dolan et al., 1988a). Similarly, no minimum approach temperature for the network need be specified.

Using this algorithm, Dolan et al. have found HENs that in every case studied to date are lower in cost than solutions published in the literature (Dolan et al., 1987, 1988a).
optimal design. Consequently, they did not consider a solution of the type obtained by simulated annealing to be feasible and did not search for such solutions.

In this paper, we describe a prototype interactive design package for Heat Exchanger Network Synthesis by Simulated Annealing (HENSSA). HENSSA was developed to permit the user to view the evolution of heat exchanger networks during the annealing process. The goal is therefore twofold: from a research point of view, HENSSA can be used to assist in the development of new ways of modifying the network to reach cost optimality more rapidly; from an applications point of view, it makes it possible (and simple) for users to apply the GCSA algorithm with no programming effort on their part.

At the present time, HENSSA is limited to the design of heat exchanger networks in which the model adopted for individual exchangers is limited to single shell exchangers with countercurrent heat exchange and with heat transfer coefficients that are independent of fluid velocity. However, the addition of a more detailed cost function for the heat exchangers is a trivial addition to HENSSA since the GCSA algorithm is transparent to the nature of the cost function. [In fact, Dolan et al. (1988a) describe the application of GCSA to the design of HENs with a discontinuous cost function arising from the use of off-the-shelf exchangers.]

In the next section of the paper, we describe the structure of HENSSA and its interactive features. In the conclusions section, we describe future directions for the refinement of HENSSA and expansion of its capabilities.

HENSSA STRUCTURE AND CAPABILITIES

The HENSSA software is designed to implement the GCSA algorithm of Dolan et al. (1988a) for the design of HENs on a Sun workstation using a graphical interface. It is composed of two distinct parts: the GCSA implementation and the user interface. The simulator software runs the simulated annealing algorithm. The user interface is a graphical representation of the HEN combined with point and click mouse features to modify the network.

GCSA Implementation

The HENSSA implementation of the GCSA algorithm, which we refer to subsequently as the optimizer, consists of the simple set of steps illustrated in Figure 2. The first step is to read in the stream data and then place that information in the structure used to represent the network. Then the optimizer enters the main loop of the program. The first step of the main loop is to execute a move and evaluate the effect of the move on the cost of the network. The change in cost caused by the move is then used to decide whether or not to accept the move based on the acceptance criterion (Dolan et al., 1987, 1988a). Statistics are gathered such as the average cost of the network, the variance of the cost of the network, and the acceptance rate of individual moves. Some of the statistical information is used, after a set of moves have been executed, in updating the annealing temperature (Dolan et al., 1988a; Aarts and van Laarhoven, 1985).

When a move is executed, several other procedures are invoked, many of which are common to all moves. The execution of a move first requires the modification of the network either structurally (such as the insertion and deletion of exchanger moves) or parametrically (such as the change $q$ and shift $q$ moves). Once the network has been modified, a material balance on the network is performed if necessary. Once the flow rates within the network are known, the energy balances can be solved yielding all of the temperatures in the network. Thus, the resulting change in size of any heat exchanger in the network can be calculated along with any change in utility demands. Finally, given the sizes of the exchangers and the utility demands, the resulting change in cost for the network, $\Delta C$, can be calculated. A flow chart of this set of steps is provided in Figure 3.

User Interface

The interface is intended to provide the user with the capability to enter a HEN (as the starting configuration for an annealing process) and of modifying a HEN at any point during annealing or at the conclusion of annealing. The
very nature of the GCSS algorithm makes incorporating a user interface quite straightforward. In essence, since simulated annealing is an evolutionary algorithm, many of the modification procedures developed for use in the optimizer can be utilized by the user interface to perform the user-requested modifications of the network. The only difference between the user and the optimizer specifying a network modification is the source of the modification.

The interaction of the optimizer and the user with the different moves is shown in Figure 4. As can be seen in the figure, all but two functions are shared by the user and optimizer. Specifically, the user can remove a split directly, whereas the optimizer can attempt this step only after deletion of an exchanger. Conversely, the optimizer can shift \( q \) between two exchangers, whereas the user must do this indirectly by the successive application of the change \( q \) move. (That is, the user deletes/adds heat load to one exchanger in the network, then adds/deletes the same amount to another exchanger in the network, effectively shifting heat load from one exchanger to another).

The graphical interface has been developed to run in the Sun windowing system, SunTools, and is written using the SunCGI graphics facilities. [A more technical discussion of some of the implementation details is given in the Appendix below.] The user interacts with the optimizer by selecting an object (using the mouse in point-and-click fashion) within the graphical representation of the network. Depending on the type of object selected (exchanger, stream, splitter) different subwindows are presented to the user. If the user picks an exchanger then he or she has the following options: delete the exchanger, change the heat load \( q \) of the exchanger, add a split around the exchanger or add an exchanger. A reproduction of the user interface illustrating the exchanger subwindow and the HEN representation is shown in Figure 5. Alternatively, if the user selects a splitter in the network, a subwindow appears offering the following options: change the split ratio or delete the splitter. An example of this is given in Figure 6.

Across the top of the main graphics window, the user can select several items which control various aspects of the GCSS algorithm. The options on the panel consist of the following:

- **Anneal**: Choosing this option instructs the optimizer to carry out the simulated annealing algorithm for a set number of moves.
- **Options**: When the user chooses this option, another menu appears which permits the user to change how often the network is redrawn, the initial annealing temperature, and the moves per annealing temperature.
- **MER**: Choosing this option results in the GCSS optimizer minimizing the total utilities consumption of
the network rather than minimizing overall cost. This yields maximum energy recovery (MER) networks.

- Dump Canvas: This enables the user to capture the graphic representation of the current network configuration into a file for printing.

- Structural Crossovers: Unless this last item is chosen by the user, the optimizer will not permit structural crossovers in the network.

Structural crossovers are non-optimal network configurations which (unless specifically excluded) can arise during the course of GCSA. They would be represented as crossing, non-vertical lines in the grid representation of the network. One can avoid them in the course of a GCSA algorithm by giving them infinite cost: thus any move which yields a structural crossover will be automatically rejected. We are in the process of evaluating whether excluding structural crossovers in this way improves the algorithm substantially.

CONCLUSIONS

A prototype version of HENSSA, an interactive, highly flexible design tool for heat exchanger network synthesis using the GCSA algorithm, has been presented and described in this paper. Current research is aimed at improving the functionality of HENSSA (to permit easy user specification of cost functions, for example) as well as developing a more sophisticated version of HENSSA which can be applied to retrofit design problems.

REFERENCES


APPENDIX: Some Technical Notes on the Graphical Interface

The interface has been developed in C using the SunCGI Pixwins 2-D graphics tools. It consists of a main window which is partitioned into a canvas section (on which the graphical representation of the HEN is drawn) and a panel section (which contains the options for controlling the GCSA algorithm). All the drawing functions are done using the SunCGI utilities. Input, output (i.e., mouse locator functions and mouse button functions) within the canvas are performed using facilities within the Sun workstation. Thus the code is independent of any particular Sun hardware configuration and can be compiled and executed on any Sun workstation capable of running the Suntools windowing environment. In addition, the SunCGI implementation is a very nearly complete implementation of the draft ANSI CGI (Computer Graphics Interface) standard, so that migration to other hardware and operating systems should be straightforward.

When the user chooses an item in the network by locating and clicking the mouse on the canvas, this triggers the display of a subwindow containing panel items which offer the user options for modifying the network appropriate to the item selected. In fact, these subwindows are always present, but are shown only when the corresponding item is chosen within the canvas. This mechanism forces the user to first choose an item within the canvas and then choose the action to be performed. This procedure was adopted to work around the fact that within SunCGI choosing items from the panel and from the canvas are events at the same level of notification, with the result that enforcing an ordering on the choice of panel and canvas items displayed simultaneously is somewhat difficult.

All software is being developed within the Software Code Control System (SCCS) environment to allow for proper documentation and a consistent manner of updating the code. Currently the optimizer exists within one file and the user interface is spread over many files. The compilation and linking of the files are done using the UNIX make facilities.

Sun Microsystems, Inc., February 1986