Article

Optimal Design of Heat Exchangers: A Genetic Algorithm Framework

Manish C. Tayal, Yan Fu, and Urmila M. Diwekar

Ind. Eng. Chem. Res., **1999**, 38 (2), 456-467• DOI: 10.1021/ie980308n • Publication Date (Web): 30 December 1998

Optimal Design of Heat Exchangers: A Genetic Algorithm Framework

Manish C. Tayal[†] and Yan Fu[‡]

Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Urmila M. Diwekar*

Department of Engineering and Public Policy, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

Computer software marketed by companies such as the Heat Transfer Research Institute (HTRI), HTFS, and B-JAC International are used extensively in the thermal design and rating of HEs. A primary objective in HE design is the estimation of the minimum heat transfer area required for a given duty, as it governs the overall cost of the HE. However, because the possible design configurations of heat transfer equipment are numerous, an exhaustive search procedure for the optimal design is computationally intensive. This paper presents a genetic algorithm (GA) framework for solving the combinatorial problem involved in the optimal design of HEs. The problem is posed as a large-scale, combinatorial, discrete optimization problem involving a black-box model. The problem is derived from earlier work on HE design using simulated annealing (SA). SA and GAs are particularly suitable in this black-box model because they lack the crucial gradient information required for other mathematical programming approaches. A methodology based on a command procedure has been modified to run the HTRI design program iteratively coupled to both SA and GAs. In our earlier studies, SA was found to be a robust and computationally efficient technique for the optimal design of HEs subject to infeasibilities and vibration problems. This paper compares the performance of SA and GAs in solving this problem and presents strategies to improve the performance of the optimization framework.

1. Introduction

Heat exchangers (HEs) are used extensively and regularly in process industries and thus are very important during plant design and operation. They are directly related to heat transfer process requirements and form a significant portion of the associated process fuel, equipment and heating costs. In general, HE design is a continuous and ongoing process to satisfy the varying process requirements on a day-to-day basis. With increased simulation capabilities, several companies have software packages dealing with the design and rating of HEs for a given heat duty requirement. These packages incorporate numerous design options for the HEs including variations in tube length, number of shells and baffles, and tube and shell orientation, to name a few. Hence, a designer has several design options to choose from and can minimize the cost involved without compromising the performance. However, the number of discrete combinations possible grows significantly with the number of parameters considered and the number of alternatives available for each parameter. The large computational cost involved in evaluating each configuration coupled with the growing number of combinations makes it impractical for designers to do exhaustive searches. Besides, these design softwares have grown in sophistication and involve large numbers of heat and enthalpy balance equations. These equations are embedded in a black box where explicit relations are not transparent. There is an obvious need for developing a computationally efficient optimization framework for such black-box models. Thus, the optimal HE design problem can be posed as a large-scale, discrete, black-box, combinatorial optimization problem.

Most of the traditional integer-programming approaches such as branch and bound and cutting plane are not applicable in solving a black-box, discrete optimization problem because several of their requirements may remain unfulfilled. These include explicit expressions of the objective function, explicit relationships for the constraints, good initial estimates or bounds (or both), the convexity of the objective function, availability of objective function gradient information, and the ability to extrapolate the objective function value from one point to its neighborhood based on the gradient. However, there are probabilistic algorithms, such as simulated annealing (SA), genetic algorithms (GAs), and tabu searches, which are suitable for such problems. These algorithms do not guarantee global optimality but are successful and widely known to come very close to the global optimal solution (if not to the global optimal). They offer significant savings in computational costs by selectively searching a much smaller fraction of the solution space for problems involving large numbers of discrete variables. Because many real-life optimization problems do not fit into forms suitable for traditional approaches, these algorithms have recently gained much momentum and have been used in several diverse areas, such as HE network synthesis (Athier et al., 1997), heat transfer optimization in finned tubes (Fabbri, 1998), piping systems support design

* Corresponding author. Tel: 412-268-3003. Fax: 412-268-3757. E-mail: urmila@cmu.edu.
[†] E-mail: mtayal@andrew.cmu.edu.
[‡] E-mail: yfu@andrew.cmu.edu.

10.1021/ie980308n CCC: $18.00
Published on Web 12/30/1998

(Chiba et al., 1996), aircraft structure design (Dunn, 1997), truss design (Vazquez-Espi and Vazquez, 1997), molecular scale catalyst design (McLeod et al., 1997), on-line optimization of ethanol fermentation (Moriyama and Shimizu, 1996), virology and AIDS studies (Shapiro and Wu, 1997), DNA structure studies (Guarnieri and Mezei, 1996), Internet studies (Joseph and Kinsner, 1997), multidatabase systems (Subramanian, and Subramanian, 1998), mutual fund (Lettau, 1997) and market simulations (Price, 1997). For some specific problems, nested approaches coupling both SA and GAs have also been successfully tried (Wong and Wong, 1997). Although these algorithms have been applied in other large-scale combinatorial optimization problems, their real potential of also being applicable in black-box models has not yet been exploited fully.

In a recent study (Chaudhuri et al., 1997), SA was used to solve a HE design problem. SA was found to be a robust technique for this problem and showed computational savings of 98% (compared to an exhaustive search) and cost savings of up to 74% (compared to the base-case design). In addition in this study, SA was modified to obtain 10 close-to-optimal solutions instead of a single optimal solution; this was done to help the designer overcome unforeseen problems during implementation. Another equally efficient and popular technique is a GA, which derives its basis from fundamental laws of nature. In this method, a "genetic code", hereafter referred to as a solution string, is developed and represents a particular choice of design variables, just as DNA represents the structure of a person or a rabbit. A GA works collectively on a population of possible solutions and gradually improves the *fitness* (correlated to the HE cost) of the whole population. It should be noted here that most optimization approaches stop at one global optimal solution, but a GA has the capability of collectively searching for multiple optimal solutions for the same best cost. Alternately, SA takes one solution and efficiently moves it around in the search space, avoiding local optima. Thus, a GA is more effective in getting the 10 best solutions which do not differ significantly in cost. Such information could be very useful to a designer, because one configuration could be much easier to fabricate then another. The objective of this study is to observe the performance of a GA in this black-box problem, compare SA and GA approaches, and evaluate and explore various strategies of GAs to give more liberty to the HE designer.

The Heat Transfer Research Institute (HTRI) design program ST-5 has been used in this study as a black-box model for computing the heat transfer area required for a given design configuration and heat duty requirement. Because a GA works simultaneously on a population of solution strings, a coupled command structure which moves back and forth between the ST-5 and GA modules was developed. Initially, a purely random population is generated, and ST-5 is used to find the heat transfer area required for each choice of design configuration. This information then is transformed into a fitness solution string. Then, on the basis of the *survival of the fittest* and *crossover and mutation* techniques, a new, fitter population is developed, with different solution strings spanning the whole search space. ST-5 is used again to get the fitness of the whole population. This coupled process is repeated for several generations until the optimality criterion is satisfied. To make this study solve the more practical and real-life problems which occur during HE design, two new modules for *vibration* and *process infeasibility* constraints were incorporated into the command structure. These modules, which are based on the HTRI guidelines, indicate any process infeasibility or the possibility of acoustic or tube vibrations with a particular design. A very high cost of HE or, equivalently, a very low fitness (commonly termed *penalty*) is then associated with all such unsuitable solutions in the genetic pool. The heuristic, based on a GA, automatically discards all such low-fitness solutions in the next generation in its search for the optimal solution.

Three real-world test problems from the Amoco Chem. Co. have been studied with heat transfer area and HE cost as the objective function for each case. In the case in which the heat transfer area was taken as the objective function, two new GA strategies were developed. In the case in which the HE purchase cost was the objective function, two more (for a total of four) GA strategies were developed and tested for convergence and computational efficiency.

This study begins with a brief overview of GAs explaining the basic theory and parameters involved. A brief comparison of SA and GAs is also given, which looks at the marked similarities and differences between these algorithms, both in theory and in practice. This section is followed by a detailed description of the exact nature of the optimal HE design problem. This section also includes details on the origin and development of feasibility and vibration constraints incorporated in the respective modules. Subsequently, a detailed description with flowcharts is given explaining all of the crucial programming steps involved in various modules. Two case studies incorporating heat transfer area and HE purchase cost as the objective functions are considered. Four novel GA strategies are developed and applied to three test cases with different heat duty requirements. Finally, this study concludes by showing for the first time that GAs are equally effective as compared to SA in solving black-box optimization model problems. Both SA and GAs are reliable search methods for such large-scale, combinatorial, black-box optimal HE design problems. They show up to 76% HE cost savings and up to 84% heat transfer area savings, using at the most 2% *selective search* on the vast search space. ST-5 runs were found to be the major computational procedural bottleneck in the coupled program structure, consuming most of the computational time involved. The number of ST-5 calls ranged from 370 to 3350 (vs the 165 888 calls required for an exhaustive search), depending on the GA parameters and the type of strategy used. This indicates that customizing a GA for a particular problem can result in significant computational savings.

2. Genetic Algorithms: An Overview

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. On the basis of the idea of survival of the fittest, they combine fittest string structures with a structured yet randomized information exchange, to form a search algorithm with some of the innovative flair of human search (Goldberg, 1989). GAs were first developed by John Holland and his colleagues at The University of Michigan in the 1960s and 1970s, and the first full, systematic treatment appeared in Holland's book *Adaptation in Natural and Artificial Systems*, published in 1975. The consistent growth in interest since then

has increased markedly during the last 15 years. Applications include diverse areas such as biological and medical science, finance, computer science, engineering and operations research, machine learning, and social science. For specific examples, see section 1.

A GA is a search procedure modeled on the mechanics of natural selection rather than a simulated reasoning process. Domain knowledge is embedded in the abstract representation of a candidate solution, termed an organism. Organisms are grouped into sets called populations. Successive populations are called generations. A general GA creates an initial generation (a population or a discrete set of decision variables), $G(0)$, and for each generation, $G(t)$, generates a new one, $G(t+1)$. The general genetic algorithm is described below:

$t = 0$
  Generate initial population, $G(t)$
  Evaluate $G(t)$
  While (termination criterions not satisfied) do
$t = t + 1$
  Select $G(t)$
  Recombine $G(t)$
  Evaluate $G(t)$
until solution is found.

In most applications, an organism consists of a single chromosome. A chromosome, also called a solution string of length n, is a vector of the form $\{x_1, x_2, ..., x_i\}$ where each x_i is an allele or a gene representing a set of decision variable values. The domain of values from which the x_i value is chosen is called the alphabets of the problem. The initial population $G(0)$ can be chosen heuristically or randomly. The populations of the generation $G(t+1)$ are chosen from $G(t)$ by a randomized selection procedure which is composed of four operators: (1) reproduction, (2) crossover, (3) mutation, and (4) immigration.

Reproduction is a process in which individual strings are copied according to their objective function or fitness (f). Objective function f can be some measure of profit or goodness that we want to maximize. Alternatively, an objective function can represent a process cost or an effluent pollutant level that we want to minimize. In the process of reproduction, only solution strings with high fitness values are reproduced in the next generation. This means that the solution strings which are fitter and which have shown better performance will have higher chances of contributing to the next generation.

The crossover operator randomly exchanges parts of genes of two parent solution strings of generation $G(t)$ to generate two child solution strings of generation $G(t+1)$. Crossover serves two complementary search functions. First, crossover can provide new information about the hyperplanes already represented earlier in the population, and through the evaluation of new solution strings, GA gathers further knowledge about these hyperplanes. Second, crossover introduces representatives of new hyperplanes into the population. If this new hyperplane is a high-performance area of the search space, the evaluation of a new population will lead to further explorations in this subspace. Figure 1 shows three variants of crossover: one-point, two-point, and one-gene crossover. In a simple one-point crossover, a random cut is made and genes are switched across this point. A two-point crossover operator randomly selects two crossover points and then exchanges genes

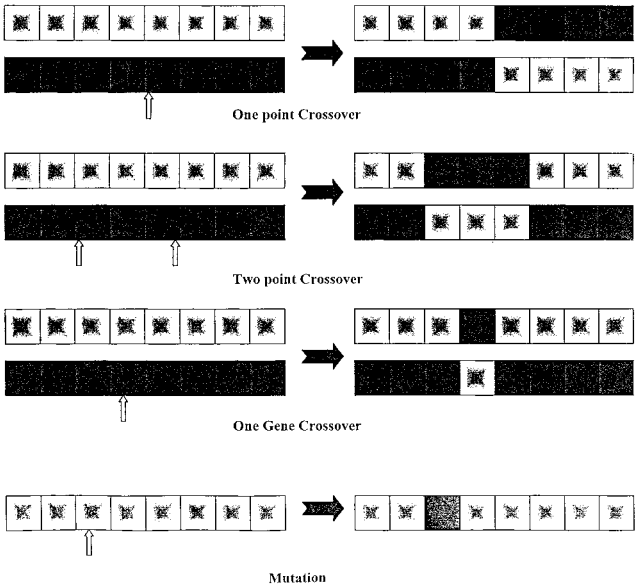

Figure 1. Crossover and mutation techniques in genetic algorithms.

in between. However, in a one-gene crossover, a single gene is exchanged between chromosomes of two parents at a random position.

Mutation is a secondary search operator which increases the variability of the population. As shown in Figure 1, a GA randomly selects a gene of the chromosome or solution string and then changes the value of this gene within its permissible range. A low level of mutation serves to prevent any given bit position from remaining fixed indefinitely (forever converged) to a single value in the entire population. A high level of mutation yields an essentially random search.

Immigration is a relatively new concept in GAs and is based on immigrations occurring between different nearby (not necessarily) societies in nature. In such scenarios, the fitness of immigrants from one society and their impact on the overall fitness of a new society to which they migrated becomes crucially important. It is analogous to the migration of smart individuals from rural to metropolitan areas in search of better prospects, and their integration and proliferation in the new society (the immigrants being fitter) enriches (increases the fitness) of this new society. Thus, immigration is the process of adding new, fitter individuals which replace some existing members in the current genetic pool. Two criteria for selecting immigrants are that they should be fit and should be quite different from the native population. Usually, immigration occurs between different populations, but it can be incorporated into a single population as well (Ahuja and Orlin, 1997), as was done in this study. In this case, when no further improvement was reported for several generations, new random solutions were added to increase the genetic diversity of the current population. (An immigration pool which contained random solutions was developed for this purpose, as is shown in Figure 4.) Immigration offers an alternative to mutation and is usually employed when there is danger of premature or local convergence.

Termination criterions of the GA may be triggered by finding an acceptable approximated solution, by fixing the total number of generations to be evaluated, or by setting some other special criterions which depend on the different approaches employed.

Table 1. SA and GA Comparison: In Theory and Practice

	simulated annealing	genetic algorithms
	In Theory	
analogous physical phenomena	statistical mechanics	biological evolution and natural selection
nature of algorithm	probabilistic	probabilistic
objective function	minimize the energy	maximize the fitness of a generation
mode of operation	works on a *single* solution string at any time	works on a *population* of solution strings at any time
initialization	random or heuristic set of decision variables	random population generated initially
change in decision variables for subsequent iteration	random perturbation	crossover, mutations and immigration
stopping criteria	low temperature no improvement for consecutive iterations	desired average fitness no improvement for consecutive generations
key algorithm parameters	temperature, decrement factor, no. of moves at each temperature	no. of solution strings in a population, percentage of reproduction crossover, and mutation
	In Practice	
type of optimization problems that can be solved	large-scale discrete combinatorial black-box nonconvex	large-scale discrete combinatorial black-box nonconvex
global optimization	asymptotically converges to global optima if move sequences are Markov chains	no proof for optimal convergence
optimization of nonconvex objective function	yes (does not require objective function gradient information)	yes (does not require objective function gradient information)
avoidance of local optima	yes (by accepting moves by Metropolis criteria)	yes (by crossover and mutation techniques)
recent applications	heat-exchanger networks (Atheir et al., 1997), multidatabase systems (Subramanian and Subramanian, 1998), DNA structure studies (Guarnieri and Mezei, 1996)	molecular design (Venkatasubramanian et al., 1994), aircraft design (Dunn, 1997), Internet (Joseph and Kinsner, 1997), virology and AIDS (Shapiro and Wu, 1997), truss design (Vazquez-Espi and Vazquez, 1997), market simulation (Price, 1997)

GAs have some unique characteristics which make them a more robust global search method than many traditional search techniques (Goldberg, 1989): a GA makes *no* assumption about the function to be optimized (Levine, 1997) and thus can also be used for nonconvex objective functions; a GA optimizes the tradeoff between exploring new points in the search space and exploiting the information discovered thus far; a GA is implicitly parallel; a GA is a randomized algorithm whose results are governed by probabilistic transition rules rather than deterministic rules; a GA operates on several solutions simultaneously, gathering information from current search points and using it to direct subsequent searches which makes a GA less susceptible to the problems of local optima and noise; a GA only uses objective function or fitness information, instead of using derivatives or other auxiliary knowledge, as are needed by traditional optimization methods.

However, a GA is not a panacea, and it often suffers from the problem of premature convergence. This means that after some generations most of the population members become identical, although the GA has still not reached the global optimal. Such a situation renders crossover operators useless. Currently, much effort has gone into refining the original simple GA for faster convergence and improved robustness. Two general approaches toward improving its performance have been explored: (1) the development of adaptive mechanisms within the genetic algorithm, such as involving more chromosomes, using hybrid algorithms, and performing parallel progress and (2) the optimization of static parameters such as mutation rate or population size. The efficiency of a GA depends on the customization of the parameters to the specific problems. Among the ideas that have been introduced are fitness scaling, generation gap, rank selection, and replacement criterions.

The key GA parameters, which are common to all strategies explained in later sections, are the population size in each generation (NPOP), the percentage of the population undergoing reproduction (R), crossover (C), and mutation (M), and the number of generations (NGEN). These can be crucial for customizing the GAs and can affect computational time significantly. These parameters govern the implementation of the algorithm to real-life optimization problems and must be determined a priori before the procedure is applied to any given problem. Four novel strategies are reported in this study, each incorporating a different set of GA parameters and each differing in the way the crossover and mutation operators are adopted. Some of these strategies have introduced additional parameters of their own. A detailed description of these strategies can be found in section 5.5.

3. Comparison of SA and GA

SA and GAs are both stochastic search techniques. These have been applied particularly in difficult large-scale optimization problems. Both techniques derive their motivation from processes found in nature (statistical mechanics and biological evolution). Table 1 illustrates the key features of these algorithms and highlights marked differences and similarities between the two approaches. For a detailed description of SA, the reader is referred to an earlier study (Chaudhuri et al., 1997).

4. The Optimal HE Design Problem

4.1. Problem Representation. The problem of representation in terms of a genetic code is one of the key aspects of GAs that leads to their success or failure (Ahuja and Orlin, 1997). Binary string representations for decision variables have been suggested for GA problems in earlier studies. Since then, researchers have considered many other representations for discrete, combinatorial problems. It is believed that Gray coding is in fact better for most practical problems than standard binary coding (Reeves, 1997). Here, the HE problem was represented as a large scale, discrete, combinatorial optimization problem, with the solution vector **X** containing elements x_i, with i varying from 1 to 8. Each x_i corresponds to a particular configuration choice of baffle types, the number of shells in series and tube passes, the tube length and tube layout pattern, the tube outer diameter (o.d.) and tube pitch, the number of shells in parallel, the shell orientation, the shell type, and the type of vibration record, respectively. The decision variable set (positions of the genes in this study) was derived from the earlier study (Chaudhuri et al., 1997). Thus, any choice of a vector **X** refers to a particular HE design configuration. The following alternatives are considered:

$$\mathbf{X} = \{x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5 \quad x_6 \quad x_7 \quad x_8\}$$

x_1 = three types of baffle choices involving "type" and "no-tubes-in-the-window" (NTIW) (corresponding to the integer variables 1−3). These include combinations of 1- and 2-segmental-type baffles with tubes-in-window (TIW) and NTIW preferences. x_2 = 9 types of shell choices involving the number of shells in series and the number of tube passes (corresponding to the integer variables 1−9). These include combinations of 1−4 shells in series with 1−4 tube passes. x_3 = 12 choices of tube data from set no. 1 involving the tube length and the tube layout pattern (corresponding to the integer variables 1−12). These include combinations of tube lengths ranging from 8 to 24 ft and tube layout patterns ranging from 30 to 90°. x_4 = 16 choices of tube data from set no. 2 involving the tube o.d. and the tube pitch (corresponding to the integer variables 1−16). These represent combinations of tube o.d. ranging from 0.5 to 2.0 in. and tube pitch ranging from 0.625 to 2.5 in. x_5 = 4 choices of shell configurations involving the number of shells in parallel (corresponding to the integer variables 1−4). These represent 1−4 shells in parallel. x_6 = 2 choices of shell layout involving the Shell orientation of 0 and 90° (corresponding to the integer variables 1 and 2). x_7 = 2 choices of shell types involving BEM and BJM shell types (corresponding to the integer variables 1 and 2). x_8 = 4 choices for the vibrations record for the single-segmental, NTIW baffle types which correspond to x_1 = 2 (corresponding to the integer variables 1−4). It must be noted here that these choices are available for x_1 = 2 only and so are indifferent for x_1 = 1 and x_1 = 3. On the basis of these choices for design variables, the solution space was calculated to have 165 888 combinations. So, an exhaustive search would require that many ST-5 calls, making it an impractical choice.

4.2. Objective Function. As stated earlier, this problem is derived from an earlier work (Chaudhuri et al., 1997) and considers heat transfer area as the objective function for case study A and HE purchase cost as that for case study B. The Amoco Chem. Co. chose these objective functions based on their requirements. Both of these case studies were undertaken subject to the process infeasibility and vibration constraints, which are explained in sections 4.3. and 4.4 below. The objective function is referred to as $f(\mathbf{X})$, where $\mathbf{X} = \{x_i, i = 1, ..., 8\}$, that gives the required area or HE purchase cost as a function of choice vector **X**.

Table 2. Optimal Heat Exchanger Design Problem

min $C(\mathbf{X})$ or $A(\mathbf{X})$
$\mathbf{X} \in \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8\}$
where
$x_1 = \{1, 2, 3\}$
$x_2 = \{1, 2, ..., 9\}$
$x_3 = \{1, 2, ..., 12\}$
$x_4 = \{1, 2, ..., 16\}$
$x_5 = \{1, 2, ..., 4\}$
$x_6 = \{1, 2\}$
$x_7 = \{1, 2\}$
$x_8 = \{1, 2, ..., 4\}$
subject to
feasibility constraints [FATAL_ERROR]
vibration constraints [VIBRATION]

The purchase cost is a more realistic index for the optimal design than the total heat transfer area, because the total heat transfer area and the number of shells affect the cost of the HE set for a given duty. The purchase cost of the HE was based on a linearized cost function typical of fixed-tube-sheet HEs (Peters and Timmerhaus, 1980):

$$C(\mathbf{X}) = 4368.0(n_{\text{series}} n_{\text{parallel}}) + (5.189 \text{ \$/sq ft}) A(\mathbf{X})$$

where $C(\mathbf{X})$ = purchase cost in dollars; $A(\mathbf{X})$ = total surface area in square feet, n_{series} = number of shells in series, n_{parallel} = number of shells in parallel, and $\mathbf{X}$ = a solution string (representing a particular design configuration).

Thus the HE design problem can be symbolically expressed as in Table 2.

4.3. Black-Box Model. The HTRI program ST-5 is used to compute the heat transfer area for a given duty requirement. For a given HE configuration, it can compute the heat transfer area required. However, the exact relationship of the objective function with the decision variables is not transparent. No explicit expression for the objective function can be deduced, and so, it lacks crucial objective function gradient information. It acts as a black-box model. The only way to know the functional topology (which varies from case to case depending on the heat duty requirement and case setup) is to do an exhaustive search, which demands enormous CPU time. This makes it practically infeasible. It should also be noted here that even a slightly different setup could cause dramatic changes in the search space topology and thus on the optimal design cost. This emphasizes the fact that even a slightly different setup requires a fresh search, be it exhaustive or selective.

4.4. Feasibility Constraints. The program structure was developed to incorporate checks for process infeasibilities in any design configuration. Process infeasibilities and programming errors caused during some iterative or convergence subroutine can result in premature abortion of the HTRI ST-5 program. This causes the ST-5 design program to write a "fatal error" message in the report file. The command procedure accounted for these cases by identifying and monitoring them and accordingly setting the objective function to a much

higher value, to signify an infeasible configuration. Identification of such an infeasibility was used to prompt the GA algorithm and to predict a new configuration replacing the infeasible configuration.

4.5. Vibration Constraints. Large HEs with increased shell-side flow velocities improve heat transfer and reduce fouling. However, increased flow velocities induce large shell-side pressure drops, which can be minimized to a certain extent by reducing the number of baffles. This, however, results in acoustic and tube vibrations. Such physical vibrations pose serious operating problems. So, incorporation of vibration constraints in the program structure was desirable, to help make possible practically applicable optimal HE designs.

HTRI provides guidelines that act as tradeoffs between increased flow velocities, reduced number of baffles, and the vibrations occurring therein. These guidelines allow designers to check for acoustic and tube vibrations (Amoco Report, 1994). These follow a step-by-step procedure and use the output data listed in the section "Flow-Induced Vibration Analysis" of the HTRI ST-5 report file to determine whether acoustic or tube vibrations are possible with a particular design. These guidelines check for vibrations in five main locations within a HE, namely, the inlet, central, outlet, shell entrance, and shell exit regions. The outcome of the analysis for a HE design usually results in one of the following three scenarios: (1) possible vibration problem, (2) probable vibration problem, (3) no vibration problem.

The objective function is penalized for possible and probable acoustic or tube vibrations as follows:

$$\text{Penalized } f(\mathbf{X}) = \text{Real } f(\mathbf{X}) + \text{wt}_{\text{pos}} \times \text{Real } f(\mathbf{X}) \quad \text{[possible vibration problem]}$$

$$\text{Penalized } f(\mathbf{X}) = \text{Real } f(\mathbf{X}) + \text{wt}_{\text{prob}} \times \text{Real } f(\mathbf{X}) \quad \text{[probable vibration problem]}$$

where wt_{pos} and wt_{prob} are the weights for the possible and probable vibration problem scenarios, respectively, and are user-defined. These values are read by the vibration routine. Thus, the amount of penalty imposed depends on the type of vibration and the various numbers of safe limits of critical parameters that are violated. It should be noted that this could lead to multiple penalties of varying amounts, thus representing a more realistic approach in incorporating vibration problems (during practical operation of HE) in our optimal configuration analysis.

The penalized objective function, as compared to the real objective function in the unconstrained case, is analyzed by the GA to predict new design variables. Such a large (penalized) objective function value implies an expensive design configuration; the survival of the fittest approach of the GA selectively eliminates such low-fitness configurations, thereby avoiding design configurations with possible vibration problems.

5. Generalized Program Structure

5.1. Basic Program Outline. A GA demands an efficient command structure involving a modular approach because it collectively works on a population of solution strings. Because the ST-5 program needs to be run repeatedly during a GA, an efficient file structure was also developed to manipulate the intermediate files that are created during execution of the program. Figure 2 shows the command structure and interactions between various key modules involved. These include the FATAL_ERROR and VIBRATION modules, which check for fatal errors, process infeasibilities, and physical vibrations. These, combined with ST-5, interact iteratively with the GA module. The GA module is strategy specific and is the most crucial step. It drives each current solution toward optimality in a highly efficient manner. Figure 3 gives a detailed outline of the whole procedure and it clearly explains how different modules interact. It also shows the elegant creation of new generations from the previous generations. ST-5 was found to be the computational procedural bottleneck during each iteration. Different GA strategies were aimed at minimizing the number of ST-5 calls while simultaneously improving the objective function.

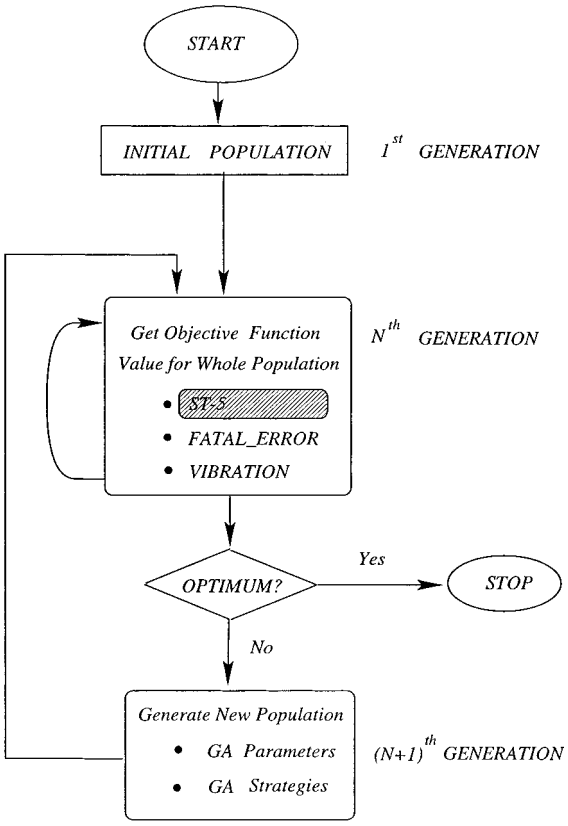

Figure 2. GA solution procedure with various key modules.

As shown in Figure 3, a random population is generated initially, and the objective function value or, equivalently, the fitness of the whole population is obtained by repeated calls to the ST-5, FATAL_ERROR, and VIBRATION modules. When using these, appropriate penalties are added to avoid physical vibrations and process infeasibilities. This fitness information is then communicated to the GA module, which develops a new and fitter population and directs the search. The intricate methodologies characteristic of different strategies which are involved in the GA module are explained in section 5.5.

5.2. Comparison of SA and GA Program Structure. The generalized program structure using a GA differs significantly from the SA structure developed in the earlier study (Chaudhuri et al., 1997). The first basic difference comes from the fact that a GA optimizes collectively over a large population of solutions whereas in SA individual solutions are developed and perturbed based on the simulated annealing theory. Second, they

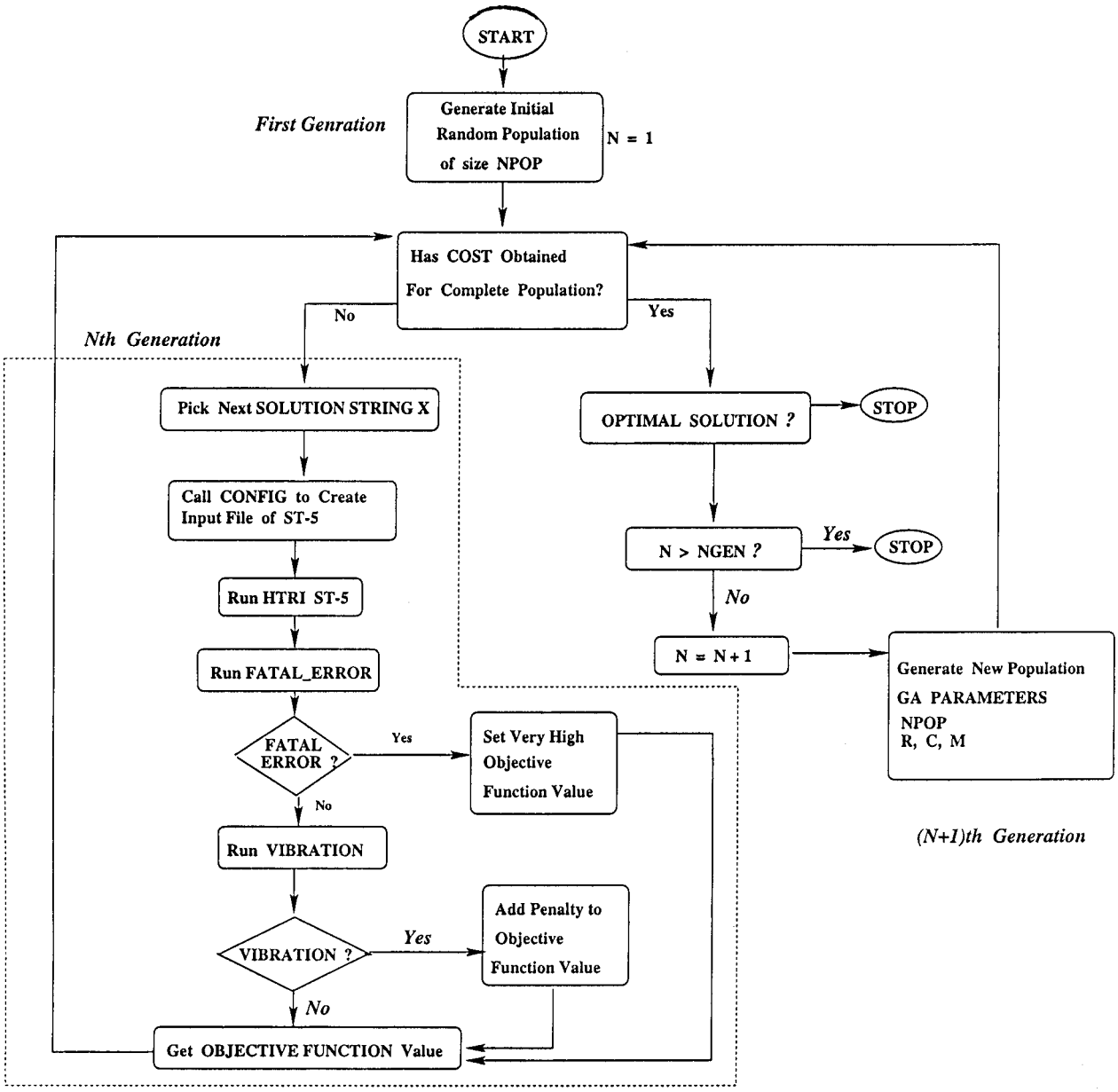

Figure 3. Generalized GA program structure for optimal HE design. (See section 5.5 for a description of GA parameters.)

differ largely in the way of perturbation in generating new solutions. In GAs, the crossover and mutation tactics provide a simple way of getting fitter solutions and span widely different regions of the whole search space. However in SA, several local perturbations are performed in the neighborhood of the current solution. Because of the above differences, we introduced an additional calculation loop in the GA. A more complex command structure (compared with SA) was developed, which took care of this aspect and maintained interactions and partial transparency between all intermediate data storage files as desired. The best solution string, cost, and area of each generation were stored in separate files for comparison. The feasibility and vibration constraints have been incorporated in both of the GA and SA program structures in a similar manner.

5.3. Initial Population. The initialization of a population in a nonrandom manner has not been successful in practice. GAs almost always do better sampling the full search space and recombining these solutions rather than starting in a user-specified area of the search space (Levine, 1997). So, an initial parent population was generated randomly with a particular population size, which varied with different strategies.

5.4. Stopping Criteria. GAs do not guarantee convergence to global optimum solution and so require suitable stopping criteria. In this study, the GA was terminated when there was no improvement in the objective function value for 25 consecutive generations and when these remained within a prespecified tolerance range. The algorithm was also terminated when the GA has already worked on a prespecified maximal number of generations. After such consecutive futile repetitions, it is reasonable and highly probable to expect no further improvement in the given framework. This also highlights the tradeoff involved in GAs between the level of exploration, the available CPU time, and the satisfaction (percentage improvement) with the best solution obtained in this fashion.

5.5. Four Strategies of GAs. Four strategies of GAs were explored in this study. The general key parameters for each strategy are shown in Table 3. The schematic diagram in Figure 4 explains the process of generating

Table 3. Comparison of GA Parameters for Four Novel Strategies[a]

	NPOP	NGEN	*R*	*C*	*M*	specific features
strategy 1	20	100	50	40[b]	10[b]	one-gene crossover repetitions removed by brute force
strategy 2	100	100	50	40[b]	10[b]	one-gene crossover repetitions removed by brute force
strategy 3	100	100	50	36	14	two-point crossover repetitions not removed
strategy 4	100	100	50	36	14	two-point crossover repetitions not removed controlled immigration of random population

[a] NPOP = no. of solution strings in each generation, NGEN = no. of generations enumerated, *R* = percentage of reproduction, *C* = percentage of crossover, and *M* = percentage of mutation. [b] These are initial values and the change during the iterations as dictated by the respective strategies.

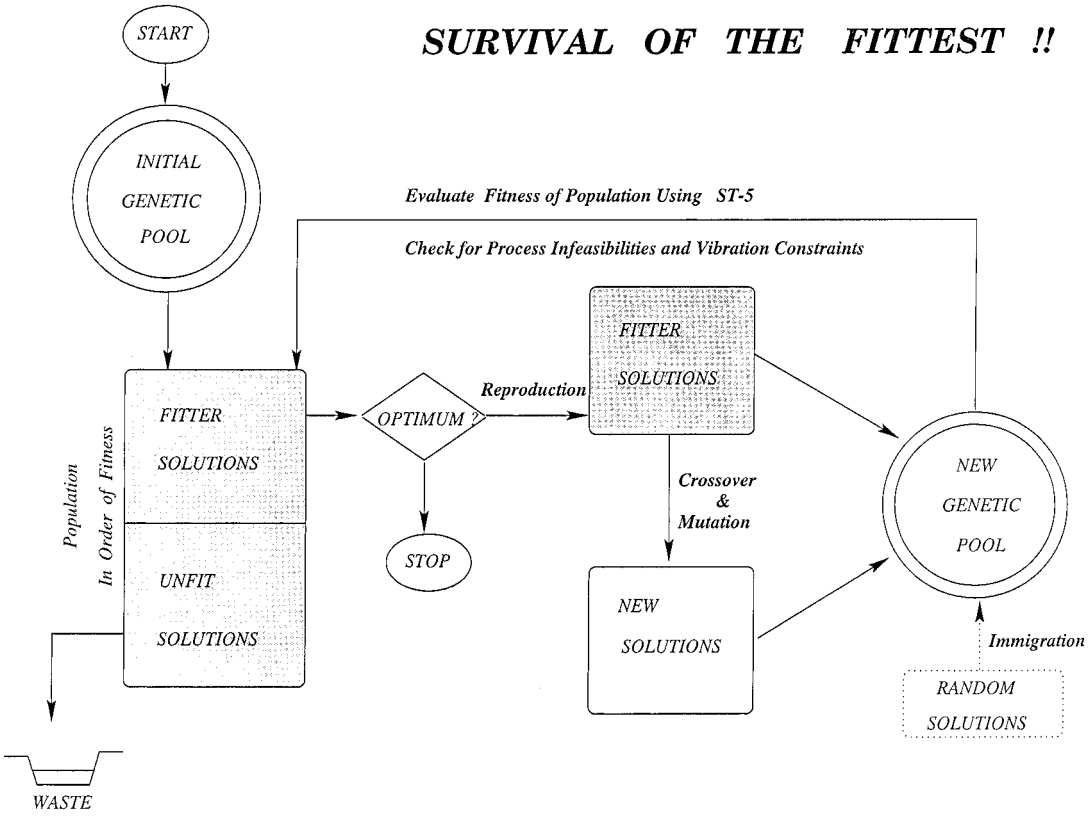

Figure 4. Schematic diagram of a GA with different strategies for developing the next generation using crossover, mutation, and immigration techniques.

new populations. Also highlighted are the individual roles played by different operators, key to GAs.

5.5.1. GA Parameters. In the strategies developed in this study, the first (initial) generation is sorted based on the fitness values. Top-half solutions, which are fitter, are given a "free pass" to the next generation, and unfit solutions perish. Next, a gene enrichment mechanism involving crossover, mutation, and immigration operators is employed. The solution strings generated thus take the place of the solutions rejected earlier. Now, fitness of this new collection of solution strings is obtained again. The fitter of these solutions again move on to the next generation, and the whole procedure is repeated. The *P*, *C*, and *M* parameters determine the number of solution strings which receive a free pass, the number which crossover, and the number which mutate, respectively. The parameter NPOP signifies the size of the population, and NGEN represents the number of generations explored.

The choice of these parameters can play a significant role in the efficacy of the algorithm. They govern the level of gene exploration and gene preservation and thus can affect the speed of convergence of the GAs. Several heuristics and rules of thumb (Holland, 1975; Goldberg, 1989) provide good initial guesses for these parameters. However, parameter optimization, which is often highly computationally intensive (especially if the objective function evaluation involves a black-box model), can be performed for high-impact problems. The inherent tradeoff lies between the additional computational time devoted to parameter optimization and tuning versus the additional savings achieved in the objective function value or the computational time of the algorithm.

5.5.2. GA Strategies. Various strategy specific approaches are explained below.

Strategy 1: In this strategy, the GA parameters chosen are a population size of 20, 50% reproduction 40% crossover, and 10% mutation. The chromosome length in this study is 8, and no information is available about the relationships between different decision variables. So, to avoid any disruption of better schemata obtained during any generation, a one-gene crossover

was employed in this strategy. In addition, if the genes at the cross point are the same, then the mutation operator was called upon to generate new values for these genes, thus producing new solution strings. There is a tradeoff involved between exploring new sample space and exploiting the information discovered so far. After a crossover was performed, mutation was performed on the remaining 10% of the solution strings.

The characteristics of this strategy are (1) when a crossover does not produce a new population (meaning the same gene values were present at the crossover point), the code automatically changes to mutation operator (so in fact, the percentages of crossover and mutation are not fixed, which increases the chance of diversity) and (2) after the population of one generation had been sorted, duplicate solution strings were checked, and when duplication was found, only one copy was kept, which increases diversity further. Because of these features, strategy 1 can successfully avoid premature problems and get good optimal solutions for both area and cost studies.

Strategy 2: This strategy is similar to strategy 1. However, in this strategy, the NPOP was increased from 20 to 100 because it was believed that increasing the number of generations would improve the performance of the GA. All remaining GA parameters were the same as those in strategy 1.

Strategy 3: This strategy differs from strategies 1 and 2 in GA parameters and in the way crossover and mutation are performed. In this strategy, the GA parameters chosen are a population size of 100, 50% reproduction, 36% crossover, and 14% mutation. A two-point crossover was performed by cutting genes of both the parents at two random positions and then crossing the genes in the middle between the two parents. Thus it differs from the one-point crossover in strategies 1 and 2, in which one cut was made and the parts were switched. Mutation was done by choosing a random position on the string (here a length of 8) and then replacing it by another value that is possible for that design variable. While creating new strings by crossover and mutation, no brute force was applied, unlike earlier strategies. This leads to a large number of multiple copies of the best solution after several generations. However, it was believed that multiple copies of fitter solutions would help in spreading the better genes to future generations.

Strategy 4: This strategy is similar to strategy-3 in some aspects and has the same GA parameters, namely, population size of 100, 50% reproduction, 36% crossover, and 14% mutation. However, a new concept of immigration was introduced here to tackle the problem of accumulating multiple copies of best-solution strings after several generations, as such an accumulation may lead to premature convergence.

Most GA success stories use hybridization of some kind which involve a customization to each particular kind of problem (Ross, 1997). GAs are good at finding promising areas of the search space but are not good at converging to the optimal solution from within those areas (Levine, 1997). It is also said that GAs lack "killer instinct"; they can get to the right neighborhood quickly but then cannot reliably find the optimum (Ross, 1997). Thus, many operation researchers believe that GAs often are more effective when employed in hybrid algorithms (Ahuja and Orlin, 1997). As a continuation of these ideas, a strategy was developed which incorporated the concept of immigration during the development of a new generation. Immigrants replace all multiple copies but one of the existing best solution in the current genetic pool. Note that multiple copies of the fittest solution are generated as a direct consequence of the inability of their algorithm to bring further improvement while simultaneously trying to preserve the fitter genes. Immigration was performed when the number of copies of a best solution crossed a prespecified limit. This limit of the number of multiple copies is an additional GA parameter for this strategy; here, this limit was chosen as six. Thus, immigrants replaced five or more multiple copies. This whole process of adding new random solutions acted as an intermittent feed of new genes into the genetic pool when no significant improvement was observed for several generations.

Table 4. Amoco Base-Case Design for Three Test Cases

design variables	test 1	test 2	test 3
baffle type	1-seg, TIW	1-seg, TIW	1-seg, NTIW
shell type	BEM	BEM	BEM
shell orientation (deg)	0	0	0
no. of shells in series	2	1	4
no. of shells in parallel	1	1	1
no. of tube passes	2	2	1
tube length (ft)	25.0	28.0	20.0
tube layout pattern (deg)	30	30	30
tube o.d. (in.)	0.625	0.625	0.750
tube pitch (in.)	0.781	0.875	1.000
area (sq ft)	426	6644	519
vibration problems	no	yes	no
purchase cost ($)	10 947	38 844	20 165

5.6. Performance Evaluators. The following performance evaluators were used for comparison. (1) Savings in Area: Percentage savings in the heat transfer area (P_a) can be defined by comparing the base-case (Amoco) design with the optimal design as follows.

$$P_a \quad \% \text{ savings in area} = 100(\text{base-case design area} - \text{optimal area})/\text{base-case design area}$$

(2) Savings in Cost: Percentage savings in the HE purchase cost (P_c) can be defined by comparing the base-case design with the optimal design as follows.

$$P_c \quad \% \text{ savings in cost} = 100(\text{base-case design cost} - \text{optimal cost})/\text{base-case design cost}$$

(3) Computational Savings: The computational saving (P_{CPU}) compared to an exhaustive search is based on the number of enumerations by the GA. It can be expressed as a percentage of the total number of possible combinations (165 888) as follows.

$$P_{CPU} \quad \% \text{ computational savings} = 100(165\,888 - \text{no. of enumerations by GA})/165\,888$$

6. Case Studies

Three test cases supplied by the Amoco Chem. Co. were studied here, each corresponding to a different set of liquid streams with various properties and temperature requirements. In sum, these correspond to various heat duty requirements for different setups of the HE. There are three distinct input files to the ST-5 program for each of these three cases.

Table 5. GA Optimal Design Using Strategy 1 (on the Basis of Heat Exchanger Area)

design variables	test 1	test 2	test 3
baffle type	1-seg, TIW	1-seg, TIW	1-seg, NTIW
shell type	BEM	BEM	BEM
shell orientation (deg)	90	0	0
no. of shells in series	4	4	2
no. of shells in parallel	2	2	1
no. of tube passes	3	3	4
tube length (ft)	12	16	10
tube layout pattern (deg)	30	90	30
tube o.d. (in.)	0.75	1	0.75
tube pitch (in.)	1.125	1.312	1.062
area (sq ft)	415	5027	87
vibration problems	no	no	no
purchase cost ($)	37 097	61 029	9187
no. of GA enumerations	350	360	280
area saving	2.60%	24.3%	83.2%
CPU saving	99.78%	99.78%	99.83%

Table 6. GA Optimal Design Using Strategy 2 (on the Basis of Heat Exchanger Area)

design variables	test 1	test 2	test 3
baffle type	1-seg, NTIW	1-seg, TIW	1-seg, NTIW
shell type	BEM	BEM	BEM
shell orientation (deg)	0	0	0
no. of shells in series	4	4	2
no. of shells in parallel	1	2	2
no. of tube passes	3	3	3
tube length (ft)	12	16	8
tube layout pattern (deg)	90	90	30
tube o.d. (in.)	0.750	1.000	0.500
tube pitch (in.)	1.000	1.312	0.625
area (sq ft)	415	5027	84
vibration problems	no	no	no
purchase cost ($)	10 889	61 029	17 908
no. of GA enumerations	1600	1600	1450
area savings	2.6%	24.3%	83.8%
CPU savings	99.03%	99.03%	99.12%

Table 7. GA Optimal Design Using Strategy 1 (on the Basis of Heat Exchanger Purchase Cost)

design variables	test 1	test 2	test 3
baffle type	1-seg, NTIW	2-seg, TIW	1-seg, NTIW
shell type	BJM	BEM	BEM
shell orientation (deg)	90	0	0
no. of shells in series	2	2	1
no. of shells in parallel	1	1	1
no. of tube passes	3	3	1
tube length (ft)	20	24	24
tube layout pattern (deg)	90	30	30
tube o.d. (in.)	0.625	0.750	0.500
tube pitch (in.)	0.812	1.000	0.625
area (sq ft)	446	5278	88
vibration problems	no	no	no
purchase cost ($)	11 050	36 124	4825
no. of GA enumerations	510	370	430
area savings		7.0%	76.1%
CPU savings	99.69%	99.77%	99.74%

Table 8. GA Optimal Design Using Strategy 2 (On the Basis of Heat Exchanger Purchase Cost)

design variables	test 1	test 2	test 3
baffle type	1-seg, TIW	2-seg, TIW	1-seg, NTIW
shell type	BEM	BEM	BEM
shell orientation (deg)	0	90	90
no. of shells in series	2	2	1
no. of shells in parallel	1	1	1
no. of tube passes	3	4	1
tube length (ft)	24	24	24
tube layout pattern (deg)	90	90	30
tube o.d. (in.)	0.750	1.000	0.500
tube pitch (in.)	1.062	1.250	0.625
area (sq ft)	415	5203	88
vibration problems	no	no	no
purchase cost ($)	10 889	35 734	4825
no. of GA enumerations	1900	3350	1950
area savings	0.01%	8.0%	76.1%
CPU savings	98.85%	97.98%	98.82%

Table 9. GA Optimal Design Using Strategy 3 (On the Basis of Heat Exchanger Purchase Cost)

design variables	test 1	test 2	test 3
baffle type	1-seg, NTIW	1-seg, NTIW	1-seg, NTIW
shell type	BEM	BEM	BEM
shell orientation (deg)	0	0	90
no. of shells in series	2	2	4
no. of shells in parallel	1	1	1
no. of tube passes	4	4	4
tube length (ft)	16	20	16
tube layout pattern (deg)	90	90	30
tube o.d. (in.)	0.625	0.750	0.500
tube pitch (in.)	0.812	1.125	0.688
area (sq ft)	440	5492	103
vibration problems	no	no	no
purchase cost ($)	11 019	37 234	4902
no. of GA enumerations	1500	1250	1250
area savings		4.14%	75.69%
CPU savings	99.09%	99.24%	99.24%

Table 10. GA Optimal Design Using Strategy 4 (On the Basis of Heat Exchanger Purchase Cost)

design variables	test 1	test 2	test 3
baffle type	1-seg, NTIW	1-seg, TIW	1-seg, TIW
shell type	BJM	BEM	BEM
shell orientation (deg)	90	0	0
no. of shells in series	4	4	2
no. of shells in parallel	1	1	1
No. of tube passes	3	4	3
tube length (ft)	24	20	24
tube layout pattern (deg)	30	90	30
tube o.d. (in.)	0.625	1.000	0.500
tube pitch (in.)	0.812	1.250	0.625
area (sq ft)	425	5412	95
vibration problems	no	no	no
purchase cost ($)	10 941	36 818	4860
no. of GA enumerations	1500	1500	1900
area savings	0.01%	5.21%	75.89%
CPU savings	99.09%	99.09%	98.85%

6.1. Amoco Base-Case Designs. Table 4 shows the Amoco base-case design configuration, corresponding heat transfer area, and HE purchase cost for each test case. These designs will be compared with the results obtained using different GA strategies.

6.2. Case Study A: Area as the Objective Function. A primary goal in HE design is the estimation of the minimum heat transfer area required for a given heat duty, as it governs the overall cost of the HE. In this case study, strategies 1 and 2 were used to optimize the design of HEs for a given duty based on the total heat transfer area required, subject to the vibration constraints and process infeasibilities. Each test case has 165 888 combination design configurations. Results are shown in Tables 5 and 6.

6.3. Case Study B: Cost as the Objective Function. In this case, the determination of the optimal design of HEs was undertaken keeping in view a more realistic index, i.e., the purchase cost as the objective function, subject to the vibration constraints and process infeasibilities. The search space considered here consists of 165 888 combinations, the same as case study A. In

Table 11. Comparison of Optimal Heat Exchanger Area

	test 1[a]			test 2			test 3		
	$A(\mathbf{X})$	P_{CPU}	N_s	$A(\mathbf{X})$	P_{CPU}	N_s	$A(\mathbf{X})$	P_{CPU}	N_s
base case (Amoco)	426			6644			519		
SA[b]	425	98.0	3047	5203	98.0	3052	80	98.0	3014
GA (strategy 1)	415	99.8	350	5027	99.8	360	87	99.8	280
GA (strategy 2)	415	99.0	1600	5027	99.0	1600	84	99.1	1450

[a] $A(\mathbf{X})$ = area objective function (sq ft), P_{CPU} = percentage of computational savings compared to an exhaustive search, and N_s = no. of ST-5 calls. [b] Based on earlier studies (Chaudhuri et al., 1997).

Table 12. Comparison of Optimal Heat Exchanger Purchase Cost

	test 1[a]			test 2			test 3		
	$C(\mathbf{X})$	P_{CPU}	N_s	$C(\mathbf{X})$	P_{CPU}	N_s	$C(\mathbf{X})$	P_{CPU}	N_s
base case (Amoco)	10 947			38,844			20 165		
SA[b]	11 019	98.0	3048	36 124	98.0	3049	4 960	98.0	3048
GA (strategy 1)	11 050	99.7	510	36 124	99.7	370	4 825	99.7	430
GA (strategy 2)	10 889	98.8	1900	35 734	99.8	3350	4 825	98.8	1950
GA (strategy 3)	11 019	99.0	1500	37 234	99.2	1250	4 902	99.2	1250
GA (strategy 4)	10 941	99.0	1500	36 818	99.0	1500	4 860	98.8	1900

[a] $C(\mathbf{X})$ = cost objective function in dollars, P_{CPU} = percentage computational savings compared to an exhaustive search, N_s = no. of ST-5 calls. [b] Based on earlier studies (Chaudhuri et al., 1997).

this case study, strategies 1−4 were used to optimize the design of the HEs. Results are shown in Tables 7−10, respectively.

6.4. Overall Results and Discussions. Results using SA (Chaudhuri et al., 1997) and GA strategies are summarized in Tables 11 and 12. These and earlier tables show that the optimal designs for the three cases were significantly improved compared to those of the base-case designs obtained by the Amoco Chem. Co. Heat transfer area savings of up to 84% and HE cost savings of up to 76% were obtained. Also, significant computational savings of more than 98% were observed in all cases as compared to those of exhaustive searches. Comparisons of different strategies show that strategy 1, which uses a smaller population size, gave significantly improved solutions with only a small number of ST-5 calls. On the contrary, strategies 2−4, which use larger populations and show computational efforts and results similar to those of SA, required significantly more ST-5 calls but did improve the solution obtained by strategy 1. However, the extra computational cost was enormous when compared to the improvement in the solution. For example, for test 1 in Table 12, the use of strategy 2 resulted in a savings of 1.5% in the HE cost objective function value but cost 4 times the computational effort of strategy 1.

7. Conclusions

This paper demonstrates the first successful application of genetic algorithms to optimal HE design with a black-box model. It also incorporates methodologies to avoid process infeasibilities and design vibrations. In addition, the paper compares the performance of SA and four GA strategies. From this study, we conclude that (1) The optimal design obtained using combinatorial algorithms such as GAs and SA significantly improves base-case designs; (2) these algorithms also result in considerable computational savings compared to an exhaustive search; (3) SA and GAs show comparable performance, however, a customized strategy (e.g., strategy 1) used in conjunction with these algorithms can provide larger savings in computational efforts without compromising the quality of the solution; and (4) GAs have an advantage over other methods in obtaining multiple solutions of the same quality, thus providing more flexibility to the designer.

Acknowledgment

This work was partially supported by the NSF. We greatly acknowledge useful discussions with Dr. Jefferey S. Logsdon from the Amoco Chem. Co. We also thank the Amoco Chem. Co. for providing the HTRI executable and other base-case examples.

Literature Cited

Ahuja, R. K.; Orlin, J. B. Developing Fitter Genetic Algorithms. *INFORMS J. Comput.* **1997**, *9* (3), 251−253.

Amoco Training Course on the Design and Rating of Heat Exchangers—Sensible Heat Transfer and Condensation; *Amoco Report*; Amoco Chem. Co.: Naperville, IL, 1994.

Athier, G.; Floquet, P.; Piblouleau, L. Synthesis of heat-exchanger network by simulated annealing and NLP procedures. *AIChE J.* **1997**, *43*, 3007−20.

Chaudhuri, P. D.; Diwekar, U. M.; Logsdon, J. S. An Automated Approach for the Optimal Design of Heat Exchangers. *Ind. Eng. Chem. Res.* **1997**, *36*, 3685−3693.

Chiba, T.; Okado, S.; Fujii, I. Optimum support arrangement of piping systems using genetic algorithm. *J. Pressure Vessel Technol.* **1996**, *118*, 507−12.

Dunn, S. A. Modified genetic algorithm for the identification of aircraft structures. *J. Aircr.* **1997**, *34*, 251−3.

Fabbri, G. Heat transfer optimization in internally finned tubes under laminar flow conditions. *Int. J. Heat Mass Transfer* **1998**, *41* (10), 1243−53.

Goldberg, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: Reading, MA, 1989.

Guarnieri, F.; Mezei, M. Simulated annealing of chemical potential: A general procedure for locating bound waters. Application to the study of the differential hydration propensities of the major and minor grooves of DNA. *J. Am. Chem. Soc.* **1996**, *118*, 8493−4.

Holland, J. H. *Adaptation in Natural and Artificial Systems*; The University of Michigan Press: Ann Arbor, MI, 1975.

Joseph, D.; Kinsner, W. Design of a parallel genetic algorithm for the Internet. *IEEE WESCANEX 97 Commun., Power Comput. Conf. Proc.* **1997**, 333−43.

Lettau, M. Explaining the Facts with Adaptive Agents: The case of Mutual Fund Flows. *J. Econ. Dynam. Control* **1997**, *21* (7), 1117−47.

Levine, D. Genetic Algorithms: A Practitioner's View. *INFORMS J. Comput.* **1997,** *9* (3), 256−259.

McLeod, A. S.; Johnston, M. E.; Gladden, L. F. Development of a genetic algorithm for molecular scale catalyst design. *J. Catal.* **1997**, *167*, 279−85.

Moriyama, H.; Shimizu, K. On-line optimization of culture temperature for ethanol fermentation using a genetic algorithm. *J. Chem. Technol. Biotechnol.* **1996**, *66*, 217−22.

Peters, M. S.; Timmerhaus, K. D. *Plant Design and Economics for Chemical Engineers*, 3rd ed.; McGraw-Hill: New York, 1980.

Price, T. C. Using Co-evolutionary Programming to Simulate Strategic Behavior in Markets. *J. Evolutionary Econ.* **1997,** *7* (3), 219−54.

Reeves, C. R. Genetic Algorithms: No Panacea, but a Valuable Tool for the Operations Researcher. *INFORMS J. Comput.* **1997**, *9* (3), 263−65.

Ross, P. What are Genetic Algorithms Good At? *INFORMS J. Comput.* **1997**, *9* (3), 260−262.

Shapiro, B. A.; Wu, J. C. Predicting RNA H-type pseudoknots with the massively parallel genetic algorithm. *Comput. Appl. Biosci.* **1997**, *13* (4), 459−472.

Subramanian, D. K.; Subramanian, K. Query optimization in multidatabase systems. *Distrib. Parallel Databases* **1998**, *6* (2), 183−210.

Vazquez-Espi, C.; Vazquez, M. Sizing, shape and topology design optimization of trusses using genetic algorithm. *J. Struct. Eng.* **1997**, *123*, 375−7.

Venkatasubramanian, V.; Chan, K.; Caruthers, J. M. Computer-Aided Molecular Design Using Genetic Algorithms. *Comput. Chem. Eng.* **1994**, *18* (9), 833−844.

Wong, K. P.; Wong, Y. W. Combined genetic algorithm/simulated annealing/fuzzy set approach to short-term generation scheduling with take-or-pay fuel contract. *IEEE Trans. Power Syst.* **1997**, *11*, 128−36.

Received for review May 19, 1998
Revised manuscript received November 4, 1998
Accepted November 5, 1998

IE980308N