PRACTICAL ASPECTS OF APPLYING EVOLUTIONARY ALGORITHMS FOR OPTIMIZING REFRIGERANT CIRCUITRY IN HEAT EXCHANGERS

DOMANSKI P.A.[(*)],YASHAR, D.A. [(*)], LEE, S. [(*)], WOJTUSIAK, J.[(**)]

[(*)] National Institute of Standards and Technology, Gaithersburg, MD, USA
piotr.domanski@nist.gov, david.yashar@nist.gov, sunil.lee@nist.gov

[(**)] Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA, USA
jwojtusi@mli.gmu.edu

ABSTRACT

A public domain finned-tube heat exchanger simulation program, EVAP-COND ver. 3.0, has been equipped with an evolutionary algorithm-based module, called ISHED – Intelligent System for Heat Exchanger Design, which facilitates optimization of refrigerant circuitry. Previous studies demonstrated that the best circuitry designs developed by the evolutionary methods in ISHED yielded capacity improvements over the original manufacturers' designs. Since evolutionary methods inherently involve the element of randomness, they do not guarantee achieving the optimum outcome within a given optimization run. This paper presents optimization experiments with ISHED and provides recommendations for selecting optional run parameters to expedite convergence to the optimal circuitry designs. This paper also presents the implementation of a design constraint which forces all tube connections on one side of the heat exchanger to be short return bends between two neighboring tubes; i.e., each connected pair of tubes resembles a large "hairpin", and evaluates this design constraint's influence on the optimization process.

1. INTRODUCTION

Specifying a refrigerant circuitry is, in most cases, the last decision a design engineer makes in the design process of a finned-tube heat exchanger. At this point the other design parameters have already been determined: the heat exchanger shape and general dimensions; the tube material, diameter and pattern; and the fin material, type and spacing. Nevertheless, the refrigerant circuitry design can have a significant influence on the heat exchanger's capacity because it determines the path of refrigerant through the heat exchanger, which in turn influences the refrigerant mass flux in each individual tube, the local refrigerant heat transfer coefficient, the refrigerant pressure and saturation temperature drop, and how temperatures of individual tubes are distributed in relation to the temperature of air flowing through the heat exchanger.

The importance of proper circuitry design on heat exchanger performance is well documented (e.g., Liang et al., 2001; Casson et al., 2002; Granryd and Palm, 2003). Typically, different refrigerants perform better with different circuitries due to differences in their thermophysical properties. Also, different inlet airflow distributions require different circuitry designs for good performance; several experimental papers reported substantial capacity degradation resulting from maldistributed inlet airflow (e.g., Fagan, 1980; Chwalowski et al., 1989) while an analytical study demonstrated that a significant part of this capacity degradation can be recovered by a proper circuitry design (Domanski and Yashar, 2007).

While it is possible to specify a workable refrigerant circuitry based on prior experience, specifying the circuitry that provides the maximum capacity can be exceedingly difficult, in particular for cases with a non-uniform inlet air distribution. The main source of the design challenge is the complex interdependency of various physical phenomena involved in the heat transfer between the refrigerant and air. An additional factor contributing to this difficulty is the large pool of possible solutions, e.g., a three-depth row heat exchanger with 12 tubes per row has approximately $2 \cdot 10^{45}$ possible circuitry designs. Although the overwhelming majority of these designs are easily recognized by visual inspection as impractical, the number of reasonable solutions is still far too large to perform an exhaustive evaluation. For this reason, an automated search method based on evolutionary computation, as implemented in ISHED (Domanski et al., 2004), is an attractive manner for optimizing refrigerant circuitry

architectures. A similar approach for determining the best refrigerant circuitry design was also taken by Wu et al. (2008) and Castillo Martinez et al. (2010).

2. ISHED

Detailed information on ISHED is given by Domanski et al., 2004 and Yashar et al., 2011. This section provides only general information about ISHED that is useful to a user of the program.

ISHED specifies connections between tubes in the heat exchanger assembly with a goal of maximizing the heat exchanger capacity; i.e., it conducts a single-objective optimization process for a given heat exchanger assembly and operating condition. Consistent with the concept of evolutionary computation, ISHED operates on one population (or generation) of solutions to the given optimization problem at a time. Each population is a set of possible circuitry designs. Each member of the population is evaluated by a heat exchanger model to provide each member's capacity as a single numerical fitness value. The circuitry designs and their fitness values are returned as an input for deriving the next generation of circuitry designs. Hence, the optimization process is iterative as ISHED attempts to generate new members for the next generation with improved fitness values each iteration.

Figure 1 shows the functional structure of ISHED. It uses two independent modules which alternatively generate circuit designs for the subsequent population: a knowledge-based computational module and symbolic learning computational module. The control module decides which of the two computational modules is active at a given time. The decision to switch from one module to another is based on the progress that the current computational module achieved within the recent populations, both in terms of the best individual design and the population overall. Both modules are capable of optimizing refrigerant circuitry without involving the other module in the optimization process; however, the dual mode approach, depicted in Figure 1, has proven to result in a more effective optimization outcome (Yashar et al., 2011).

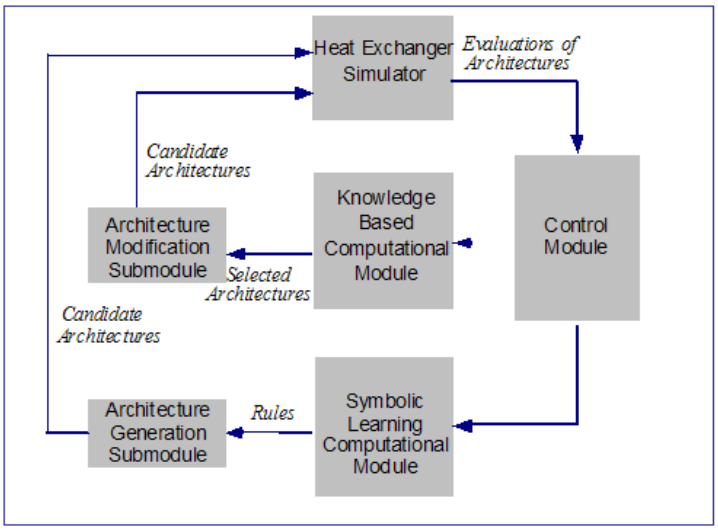


Figure 1. Functional structure of ISHED

3. IMPLEMENTATION OF ISHED WITHIN EVAP-COND

The ISHED optimization module is embedded in the EVAP-COND ver. 3.0 package and is accessible from the main EVAP-COND window through the 'Circuitry Optimization' pulldown menu.

3.1. Pre-Processing

Heat exchanger input data for an optimization run consists of the same data needed for execution of a simulation run by EVAP-COND (inputting refrigerant circuitry is optional) and some additional data defining how the optimization process is to be carried out. The set of data describing the geometry of the heat exchanger, heat transfer surfaces, and refrigerant are those solicited within the EVAP-COND *Coil Design Data* window. Once these items are specified, the user can enter the *ISHED Operating Conditions* and the *ISHED Control Parameters* window. Here the user can specify certain design rules and constraints for the allowable circuitry designs, the number of circuitry architectures in each generation, the number of generations to be examined in the

optimization run, and some other advanced parameters controlling the optimization process. The user also has the option to specify "seed" architectures; i.e., to insert specific designs as members to the first generation.

3.2. Optimization Run

Once the required input data have been entered, the user can initiate the execution of the optimization run. The optimization run may take a considerable amount of time depending on the computer's speed, the size of the heat exchanger, the specified refrigerant, and the entries for the ISHED control parameters. A typical optimization run may take several hours; ISHED will use 100 % of the available computing power from a single processor core but will not parallel process across multiple cores. Consequently, a computer with a multiple core processor will complete an optimization run faster than one with a single core processor of comparable clock speed because a multiple-core unit allows ISHED to run on a different core than the operating system.

During an optimization run ISHED implements the iterative process depicted in Figure 1 examining one generation of circuitry architectures after another. Throughout the execution, the program creates and updates several files containing intermediate results. Most importantly, the program maintains files in a results folder which contains the top ten performing circuitry architectures and updates them each iteration cycle throughout the optimization run. Also, ISHED maintains a log file documenting all optimization steps from the beginning of the execution onwards, called ishedtrace.log. A user can check on the progress during program execution and, in case of an unforeseen interruption to the run, can recover useful data from these files and prevent loss of information.

3.3. Post-Processing

At the end of a successful optimization run, the program displays a message indicating the completion along with the highest heat exchanger capacity obtained within the run. The user can access the ten best performing circuitry architectures within EVAP-COND by navigating to the ISHED Results folder. Most often, the user will find it necessary to modify the ISHED-generated circuitry architectures to accommodate manufacturing constraints; although the user has the option to limit ISHED's exploration by imposing a few design rules and constraints in the *ISHED Control Parameters* window, real world constraints are often much more involved. For this reason, it is very likely that the best performing designs, as returned by ISHED, will not appear to be realistic upon first look, and therefore will require a certain level of manual post-processing. During the post-processing effort, the user will have to "clean" the circuitry (i.e., reroute tube connections to remove crossovers, long return bends, etc.), while preserving the general design architecture. Figure 2 shows an optimized evaporator circuitry architecture produced by ISHED for a highly non-uniform air flow distribution, and a similar architecture that is the result of manually post-processing the ISHED design. The capacity of the heat exchanger changed insignificantly during this post-processing effort.

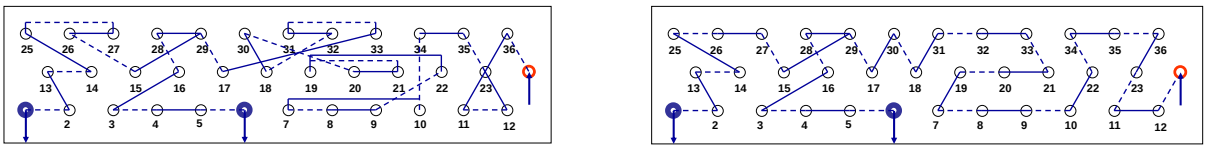


Figure 2. ISHED generated circuitry results, before (left) and after (right) post-processing. The solid lines denote the tube connections on the near side of the heat exchanger, and the dashed lines denote the tube connections on the far side.

4. PRE-PROCESSING GUIDANCE

The goal of optimization is to find the best solution for a given problem. The ISHED options selected during the pre-processing stage can influence the effectiveness of the optimization process. We examined the impact of the population size (number of members), the number of generations to be considered in the evolutionary process, and the goodness of "seed" circuitry architectures on ISHED optimization results.

4.1 Population Size and Number of Generations

While examining the influence that the population size and the number of generations has on finding the optimal solution during the optimization run, we should bear in mind that the search space explored by ISHED is multi-modal, meaning that multiple local optima and several global optima may exist (Singh and Deb, 2006). Because of this, ISHED is influenced by the randomly selected starting population – each time it is executed the

evolutionary process progresses in a different way. This behavior is well studied and known among evolutionary computation methods. For this reason ISHED is extremely unlikely to produce the same circuitry architecture during different optimization runs performed with identical settings of control parameters. The results of one optimization run will fare slightly better or worse than the results of another. This is in a contrast to calculus-based optimization methods, which produce the same results each time.

For the above reasons we performed multiple optimization runs to evaluate the influence of the population size and the number of generations with the aim of formulating recommendations that would increase the chance of identifying the optimal circuitry. We performed optimization runs for four basic design cases involving a small heat exchanger and a large heat exchanger operating with both a low-pressure and a high-pressure refrigerant, R1234yf and R32, respectively. Both heat exchangers had the same number of tubes per depth row (25) but differed by the number of depth rows. The small heat exchanger had two depth rows, and the large heat exchanger had four depth rows. Other than the depth, all other geometric and material specifications were the same. The optimization runs were based on uniform air flow velocity with 26.7 °C inlet air dry-bulb temperature and 50 % relative humidity. The imposed refrigerant parameters were 0.20 inlet quality, 7.2 °C outlet saturation temperature, and 10.8 °C outlet superheat. The optimization runs used the following five combinations of population size and number of generations: 5x2000, 10x1000, 20x500, 40x250, and 80x125; therefore each combination resulted in evaluating 10 000 circuitry designs and consumed a comparable amount of CPU time. Furthermore, we executed seven optimization runs for each of these data sets to examine the variability in performance of the obtained designs. In total, we executed 5x7=35 optimization runs for each of the four basic cases identified by the size of the heat exchanger and refrigerant.

Figure 3 presents the best evaporator capacities for the four basic cases, where the five capacities identified for each case were obtained using the five different sets of population size and number of generations. These capacities were the best out of seven runs executed with the same control settings for a given case. Figure 4 presents standard deviation of capacities obtained for each control setting for all four basic cases.

We can review Figure 3 to learn which control settings provided the best outcome (highest capacity) for the cases studied. While differences between the obtained results are small (except the R1234yf/large hx), the setting with 40 members in the population and 250 generations (40x250) provided the best capacity for three cases: R32/small hx, R1234yf/small hx, and R1234yf/large hx, and came in on the second position for the R32/ large hx case. This observation indicates that using a population size of 40 members should be the preferable option from those studied. From Figure 4 we may note that the capacities obtained using the 40x250 setting exhibited the second smallest variation for the four design cases. This means that, with this population size, the user may not have to execute an excessive number of optimization runs to find the optimal or close-to-optimal solution.

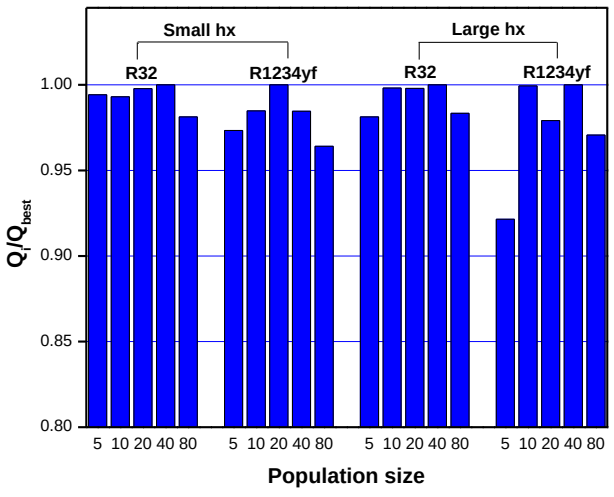


Figure 3. Capacities obtained for four refrigerant/hx cases for five settings of population size and number of generations (5x2000, 10x1000, 20x500, 40x250, and 80x125). Each capacity is the best of seven obtained for a given setting. All capacities are normalized by the best capacity obtained for a given refrigerant hx case.

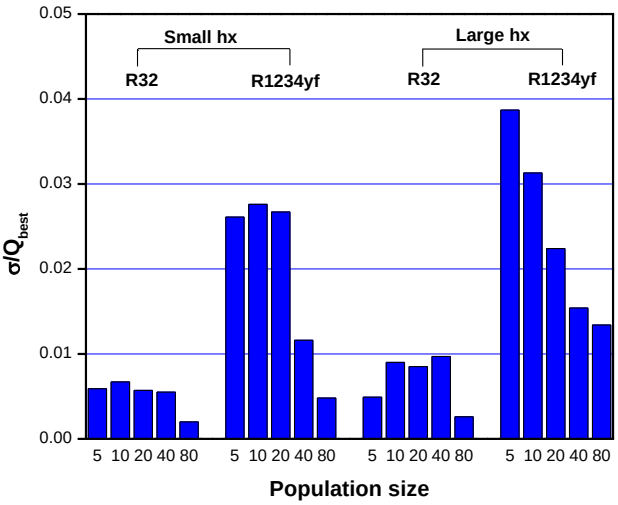


Figure 4. Standard deviation of capacities from seven optimization runs performed for each setting of population size and number of generations (5x2000, 10x1000, 20x500, 40x250, and 80x125) referenced to the best capacity obtained within a given group of seven optimizations.

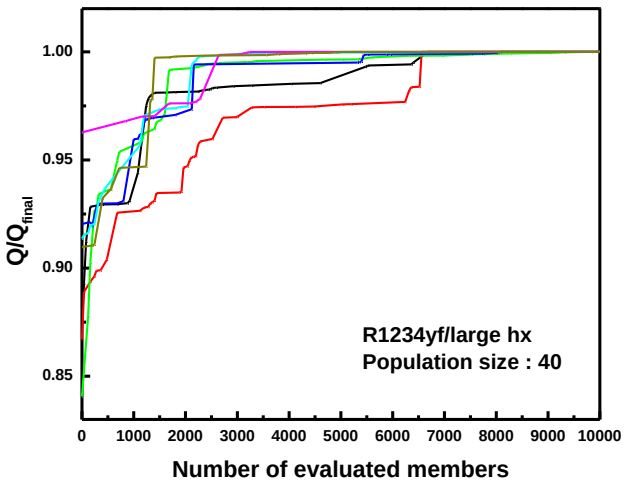


Figure 5. The best capacity progression of seven optimization runs for R1234yf/large hx case using 40 member population, normalized by the final capacity of each run.

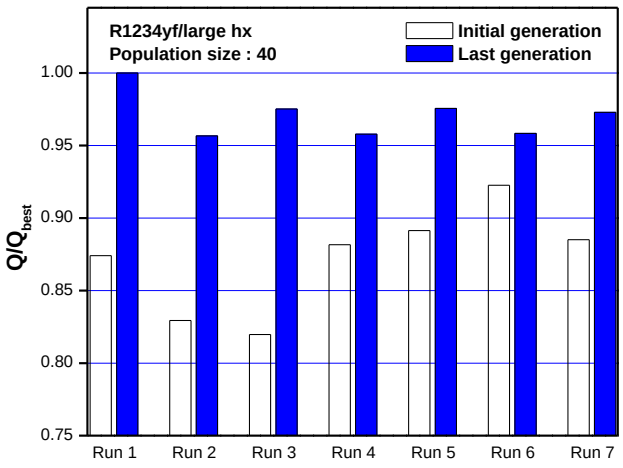


Figure 6. The best seed capacity and final capacity of seven optimization runs for R1234yf/large hx using 40x250 setting, normalized by the final capacity from Run 1 (best final capacity out of seven runs).

Besides choosing a population size, it is also of practical interest to know the number of generations required for an effective optimization run. With this goal in mind, we examined the improvement of capacity as a function of the number of designs evaluated for the four base cases and five population sizes. The review of the log files showed insignificant capacity improvement beyond 8 000 simulations for all population sizes. For example, Figure 5 presents a progression of capacity for seven optimization runs for a 40 member population. The presented R1234yf/large hx example had the largest disparity of the improvement paths from all four refrigerant/heat exchanger cases. Still the 8 000 limit of evaluated members seems to be the most practical.

We may also make an additional observation that the optimizations involving the low-pressure R1234yf showed larger deviations than the high-pressure R32, and that the deviations were larger for the large evaporator than for the small one for both refrigerants. This result can be attributed the fact that a low-pressure refrigerant requires a lower refrigerant mass flux than a high-pressure refrigerant. This means that 1234yf will generally perform better with a larger number of parallel refrigerant circuits which, in turn, may provide a larger variability in capacities. Also, a larger evaporator has a larger search space of possible refrigerant circuitry architectures than a small heat exchanger, which can also result in larger differences in obtained capacities.

4.2 Fitness of First Generation

The first generation of possible solutions must be populated before the optimization process depicted in Figure 2 can start. ISHED is equipped with an embedded seeding algorithm for generating circuitry architectures, but it also gives the user the option to specify members of the first generation. The user may populate the whole first generation, or may partially populate it, or may not provide any seed architecture at all. In any case, ISHED generates as many architectures as are needed to complete the seeding process of the first generation.

The input information to the ISHED seeding algorithm for an evaporator (or condenser) is gathered from the pre-processing data and consists of the number of members in a population, the number of seed design architectures provided by the user, the minimum number of heat exchanger inlet (outlet) tubes, the maximum number of inlet (outlet) tubes, and the maximum number of tubes allowed to be fed from (fed into) a single tube. The seeding algorithm starts the seeding process by generating simple designs that use the same number of tube inlets and outlets (no splits or multi feeds) and then provides architectures with more complex layouts, as allowed by the input information provided by the user.

All optimization runs in this study were performed without user-specified seed designs; i.e., the first generations of designs were populated entirely by ISHED. Since the ISHED seeding algorithm is stochastic by nature, the members of the first generation and their fitness values (capacities) were all different. It is then of interest to examine the influence that the fitness of the first generation had on the level of the final capacity achieved in a given optimization run. We examined results for the selected 40 member population for the R1234yf/large hx

case because the final capacities from its seven runs had the highest deviation, as shown in Figure 4. For each of the seven optimization runs, Figure 6 presents the capacity of the best design from the first generation and the best final capacity. These results indicate no correlation. Run 6 had the fittest member in the first population of all seven runs but it achieved only fifth highest capacity of the final designs and was outscored by Run 3, which had the lowest score in the first population. The winner in this R1234yf/large hx case is Run 1, which achieved the highest final capacity having only the fifth highest capacity in the initial population. Since the fitness of the best seed architecture does not correlate with the best optimization outcome, a user-specified high capacity seed architecture is not likely to guarantee achieving the highest final capacity either.

5. HAIRPIN DESIGN CONSTRAINT

5.1 Description of Hairpin Design Constraint

ISHED allows the user to impose a few design constraints which were derived from either good practice or manufacturing considerations. In the original version of ISHED used in previous publications the following three constraints were selectable: (1) Predecessor to exit tube must be adjacent to the exit tube; (2) Inlet and exit tubes must not be adjacent; and (3) Connections to all inlet tubes and from all exit tubes must be on the same side of the heat exchanger. The current version of ISHED also incorporates the hairpin constraint and the option to define the location of the inlet and outlet tubes in the tube coil assembly.

The hair-pin constraint allows the designer to develop circuitry designs that will comply with the common hairpin manufacturing preference. To provide design flexibility, three modes for creating hairpin designs are available. The choice of the mode depends on manufacturing requirements and specific user's preferences. The different modes are illustrated in Figure 7.

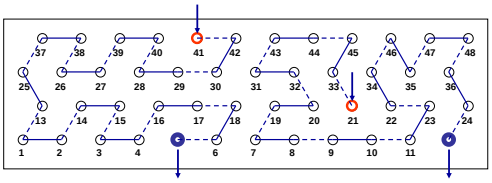
- Mode 1 - no predetermined hairpin pattern: ISHED creates hairpin designs by enforcing connections between adjacent tubes without any guidance regarding the hairpin pattern. This method is computationally intensive because ISHED searches through all possible designs and selects only those that fulfil the above constraint. Figure 7a presents a design ISHED developed using Mode 1. It includes hairpins with both hairpin tubes located in the same depth row (hairpin return bents are horizontal) as well the hairpins with the tubes placed in different depth rows.

- Mode 2 - use a standard predetermined hairpin pattern. ISHED creates hairpin designs using one of three predefined hairpin patterns, based on the number of tube depth rows and the number of tubes in a row. When more than one design pattern is possible, the program randomly chooses one design, which is applied to all designs in a given ISHED run. The three predefined patters are:
  - Return bends of all hairpins are parallel to the air inlet plane of the heat exchanger; i.e., both tubes of each hairpin are located in the same depth row. This pattern is feasible only for heat exchangers with even number of tubes in depth rows. Figure 7b shows a design ISHED developed using this pattern.
  - Return bends of all hairpins are not parallel to the air inlet plane of the heat exchanger; i.e., the tubes of each hairpin are placed in two neighbouring tube depth rows. The availability of this pattern is limited only to heat exchangers with four tube depth rows because the simplified hairpin representation (explained later) used in ISHED can't function for two depth row coil with the non-parallel hairpin pattern.
  - Return bends of most hairpins are parallel to the air inlet plane. The only hairpins with return bends not parallel to the inlet plane are those located at the right side of the coil assembly; their tubes are placed in two neighbouring odd and even depth rows. This hairpin pattern was predefined for heat exchangers of an odd number of tubes in depth rows and even number of depth rows.

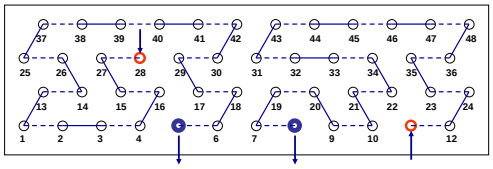
  After selecting a specific pattern, the heat exchanger representation is simplified; i.e., it is transformed to a reduced domain such that each hairpinned pair of tubes is represented by a single tube. Then, ISHED applies design operators in both knowledge-based learning and symbolic learning modes to designs in this simplified search domain. The designs are then expanded back into the original domain by transferring all non-hairpin connections to one side of the exchanger and connecting them to the ends of the hairpinned pairs of tubes. Connections to each tube of each hairpinned pair of tubes are determined so that the intersection of different connections is avoided and the overall connection length is minimized. In this method, the optimization search space is significantly reduced by the transformation, allowing for faster program operation and convergence to an optimal design.

- Mode 3 - use a hairpin pattern specified by the user: The EVAP-COND graphical interface allows the user to specify a hairpin pattern, which ISHED checks for technical consistency and uses in the optimization run.

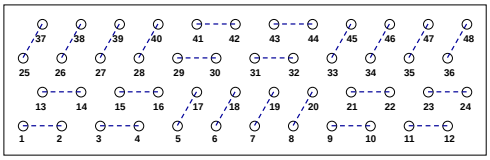
Similar to Mode 2, the designs are "shrunk" during the optimization process to improve its computational efficiency. Figures 7c and 7d show an example of a user-defined hairpin pattern and circuitry architecture generated by ISHED, respectively.



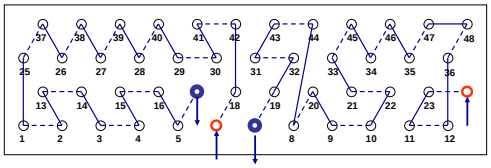
a) Hairpin design generated using Mode 1



b) Circuitry design generated using Mode 2



c) Hairpin pattern specified by ISHED user



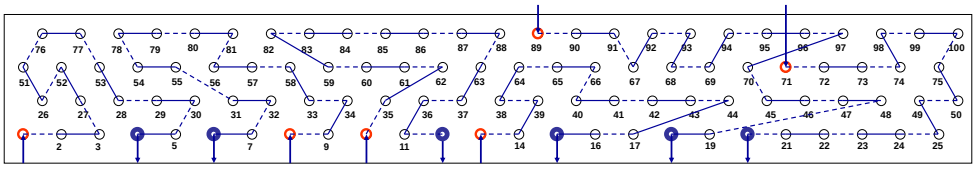
d) Circuitry design generated based on hairpin pattern shown in Figure 7c (Mode 3)

Figure 7. Different hairpin placements and circuitry designs. The return bends that are the integral part of hairpins are located on the far side of the heat exchanger and are denoted by broken lines connecting the tubes.

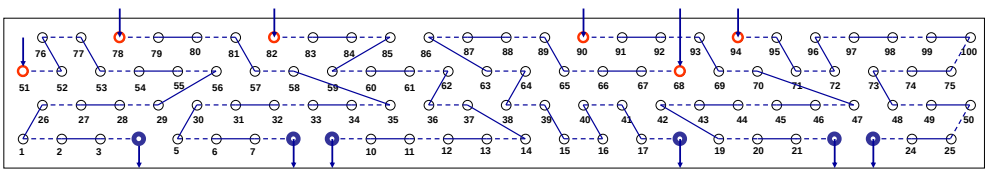
5.2 Impact of Hairpin Design Constraint on Heat Exchanger Performance

We performed circuitry optimizations to evaluate the impact of the hairpin constraint on the capacity of the heat exchanger. We examined this by running ISHED with and without this constraint for the four evaporators described in section 4.1. The optimization results showed that the hairpin constraint does not impact the performance of the optimized heat exchanger. On average, the capacities of the designs developed with the hairpin constraint were within 0.4 % of the capacities of the unconstrained designs.

For illustration, Figure 8 presents optimized refrigerant circuitries for R32 obtained from unconstrained and hairpin-constrained optimizations. The near side of the both heat exchangers have a similar level of complexity; however, the far side of the hairpin-constrained heat exchanger has a much more orderly layout consisting of horizontal return bends in all cases except the two return bends connecting tubes 25-50 and 75-100. These two non-horizontal connections were necessitated by the odd number of tubes in the depth rows. The capacities of the two heat exchangers were within 0.3 %. Note that the circuitry architectures shown in Figure 8 were manually "cleaned" based on ISHED-generated architectures but maintained the basic outline of the original ISHED designs.



a) Unconstrained optimization



b) Hairpin-constrained optimization

Figure 8. R32 circuitry designs for large evaporator from unconstrained and hairpin-constrained optimization

6. CONCLUDING REMARKS

The process of optimizing refrigerant circuitry has inherent elements of randomness and is rather unlikely to produce the same optimization outcome from different optimization runs. For this reason, performing multiple optimization runs is recommended to achieve optimal designs.

The outcome of the optimization process is affected by the population size and number of generations. The recommended values for these process control parameters are 40 and 200, respectively.

The experimentations with ISHED demonstrated no correlation between the fitness of the first generation and the optimization outcome. It follows that user-specified seed design architectures do not guarantee the optimal outcome of the optimization process. Using the seed algorithm embedded in ISHED to populate the first generation seems to be a practical and effective choice.

The hairpin constraint does not impact the performance of heat exchangers optimized by ISHED.

Optimized designs developed by ISHED require manual post-processing. During the post-processing effort, the user has to "clean" the circuitry (i.e., reroute tube connections, eliminate long return bends) to improve coils manufacturability. Further upgrade of ISHED's capabilities to reduce the designer's post-processing effort is sought.

7. NOMENCLATURE

hx – heat exchanger
Q – capacity
б – standard deviation

8. REFERENCES

1. Casson, V., Cavallini, A., Cecchinato, L., Del Col, D., Doretti, L., Fornasieri, E., Rossetto, L., Zilio, C. 2002. Performance of finned coil condensers optimized for new HFC refrigerants, *ASHRAE Trans.* 108(2): 517-527.
2. Castillo Martinez, L.C., Parise, J.A.R., Yana Motta, S.F., Vera, Becerra, E. 2010. Plate-fin and Tube Heat exchangers Refrigerant Circuiting Optimization Optimization in Vapor Compression Refrigeration System, *Proc. Int. Refrigeration and Air Conditioning Conference at Purdue*, July 12-15, 2010.
3. Chwalowski, M., Didion, D. A., Domanski, P. A. 1989. Verification of Evaporator Computer Models and Analysis of Performance of an Evaporator Coil, *ASHRAE Trans.* 95(1).
4. Domanski, P.A., Yashar, D. 2007. Optimization of Finned-Tube Condensers Using an Intelligent System, *Int. J. Refrig*. 30(4): 482-488.
5. Domanski, P.A., Yashar, D., Kaufman, K.A., and Michalski, R.S. 2004. Optimized Design of Finned-Tube Evaporators Using Learnable Evolution Methods, *Int. J. HVAC&R Research* 10(2), 201-212.
6. Fagan T.J. 1980. The effects of air flow maldistribution on air-to-refrigerant heat exchanger performance, *ASHRAE Trans*. 86(2): 699-713.
7. Granryd, E. and Palm, B. 2003. Optimum number of parallel sections in evaporators. *Proc. 21st IIR Int. Congress Refrig.*, Paper ICR0077, Washington, DC, USA, August 17-22, 2003.
8. Liang, S.Y., Wong, T.N., Nathan, G.K. 2001. Numerical and experimental studies of refrigerant circuitry of evaporator coils, *Int. J. Refrig*. 24(8): 823-833.
9. Singh, G., Deb, K. 2006. Comparison of multi-modal optimization algorithms based on evolutionary algorithms, *Proc. 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, WA, USA, July 08-12, 2006. GECCO '06. ACM Press, New York, NY, 1305-1312.
10. Wu, Z., Ding, G., Wang, K., Fukaya, M. 2008. Application of a genetic algorithm to optimize the refrigerant circuit of fin-and-tube heat exchangers for maximum heat transfer of shortest tube, *Int. J. Thermal Sciences* 47: 985-997.
11. Yashar, D., Wojtusiak, J., Kaufman, K., Domanski, P.A. 2011. A Dual Mode Evolutionary Algorithm for Designing Optimized Finned-tube Heat Exchangers, accepted for publication in *Int. J. HVAC&R Research.*