PRES'22

Proceedings of the 25th Conference on Process Integration, Modelling and Optimisation for Energy Saving and Pollution Reduction, 5 September – 9 September 2022, Bol, Croatia

Editors: Jiří J. Klemeš, Sandro Nižetić, Petar S. Varbanov
Copyediting and Composition: Xuexiu Jia, Xuechao Wang, Hon Huin Chin

Open-Source Tool for Heat Exchanger Network Synthesis

Keegan Keysers Hall[a], Timothy Gordon Walmsley[a,*], Andreja Nemet[b], Michael Walmsley[a]

[a]Ahuora – Centre for Smart Energy Systems, School of Engineering, University of Waikato, Hamilton 3240, New Zealand
[b]Faculty of Chemistry and Chemical Engineering, University of Maribor, Smetanova ulica 17, 2000 Maribor, Slovenia
tim.walmsley@waikato.ac.nz

The synthesis of heat exchanger networks (HEN) is not a trivial task due to the mixed integer and non-linear nature of the optimisation problem, yet it can provide significant cost and energy use savings to an industrial processing site. Many HEN synthesis models are developed using propriety solvers, which make it difficult for industrial companies to realise the potential of optimisation. This paper introduces a new open-source tool for HEN synthesis built in Python using the GEKKO library, which interfaces with the solver APOPT. The tool features three different synthesis methods, two of which utilise a two-step method where step one aims to initialise or narrow the search space prior to global optimisation in the second synthesis step. A unique feature of this work compared to previous HEN synthesis studies is the application of a true driving force constraint, *dQ/dA*, (instead of the conventional, but at times crude, minimum approach temperature, ΔT_{min}) to provide feasible initial variable values to the second step. To demonstrate the current progress in developing the open-source tool, network solutions obtained by the tool were purposefully compared with the best reported solutions for three common case studies in HEN synthesis literature. For a small problem, the tool achieved a solution within 0.02 % of the best literature solution and attained a solution for the two larger problems within 4 % and 21% of the best literature solution. Before the tool can be made publicly available, further work is needed to implement a non-isothermal mixing model across stream splits, which is expected to yield a more optimal HEN.

1. Introduction

With an ever-increasing global population, in addition to a growing awareness of the effects of greenhouse gas (GHG) emissions, the concept of heat integration has become an important field within chemical and process engineering. Appropriate management of energy resources and assets can help an industrial process reduce operating costs and improve its environmental sustainability, particularly for industries which heavily rely upon fossil fuels. One of the most fundamental topics of heat integration is the design/synthesis of heat exchange networks (HENs), as poor allocation of capital investment in the form of recovery (process to process) heat exchangers results in high utility operating and capital costs. Subsequently, HEN synthesis continues to be an evolving field within the literature.

There are two major methodologies for HEN synthesis. Both Pinch Analysis (PA) and Mathematical Programming (MP) have been applied by numerous chemicals, processing and manufacturing companies to HEN synthesis with great success by minimising energy consumption and GHG emissions (Fu et al., 2017). In the case of PA, beginning with the pioneering work of Linnhoff and Flower (1978), many research groups have developed their own customised software platforms that have gradually progressed alongside their sequence of publications. Likewise, MP focused groups have developed mathematical programming models and code to solve a HEN superstructure to (near) optimality with better accuracy of the capital cost and energy consumption trade-off. However, many of these well-developed platforms, including optimisation solvers, are proprietary, and commercial licensing for many companies is difficult to justify without first knowing the benefits.

This study seeks to demonstrate the potential of a newly developed open-source tool built in Python to synthesise a HEN with the minimal total cost for a given set of process and utility streams and cost coefficients. The goal in demonstrating and publishing the use of open-source modelling languages and solvers is to encourage the process engineering community to realise the benefit, apply and develop more open-source tools for the benefit of all in facing up to the challenge of emission reduction and climate change.

2. Python GEKKO

GEKKO (Beal et al., 2018) is an object-orientated Python package incorporating an Algebraic Modelling Language (AML) for equation-based model optimisation. AML is intuitive and harnesses the ease of Python, which is arguably the most popular programming language for science and engineering applications. The package interfaces with APMonitor optimisation software containing powerful open-source solvers for linear, quadratic, nonlinear, and mixed integer programming. APMonitor utilises, besides others, APOPT, a recently developed solver which applies an active-set method for Non-Linear Programming (NLP) optimisation and a Branch and Bound technique (B&B) to handle the Mixed-Integer Programming (MIP). It is one of only a few open-source Mixed-Integer Non-Linear Programming (MINLP) solvers and comes packaged with GEKKO.
Within GEKKO there are four standard variable types: constants, parameters, solver variables and intermediates. Constants are not functionally different from Python variables but provide clarity when reading GEKKO model files. Parameters are similar to constants, except they may update with time (for control applications). Solver variables are manipulated by the solver to satisfy the model constraints and must be provided with an initial value, while bounds are optional. Intermediates are unique to GEKKO and assist with model reduction as they can be used to store temporary variable calculations and are evaluated explicitly by the solver after each iteration. A contextual example is the use of intermediate variables to store the calculation of the heat exchanger area based on constants, parameters and solver variables. Equations are constructed from these four variable types using Python syntax. All GEKKO variables and equations can utilise Python lists to minimise the lines of code and generalise to any size of problem input. The final element of the model is the objective function that resolves to a singular value to be minimised or maximised.

3. Method

3.1 Open-Source HEN Synthesis Tool

The development version of the new open-source tool for HEN synthesis built in Python features three different methods for HEN synthesis consisting of two two-step methods (Methods A and B) and a single step (Method C) method (Figure 1). Method A, like many others, follows a two-step process by first solving the transshipment based TransHEN model proposed by Nemet et al. (2019) followed by an optimisation of the reduced stage-wise superstructure model SynHEAT developed by Yee and Grossmann (1990) based on an isothermal mixing assumption. Method B first applies the key dQ/dA constraint from the Cost Derivative Method (CDM) developed by Walmsley et al. (2014) for optimal area allocation in a defined HEN using a stage-wise superstructure model (SynHEAT). Method C utilises only SynHEAT and serves as a benchmark.
Like many other HEN synthesis problems, the HEN synthesis tool first requires stream data detailing the process and utility stream properties. These properties with their associated nomenclature are shown in Table 1. The minimum approach temperature ΔT_{min} must also be defined.

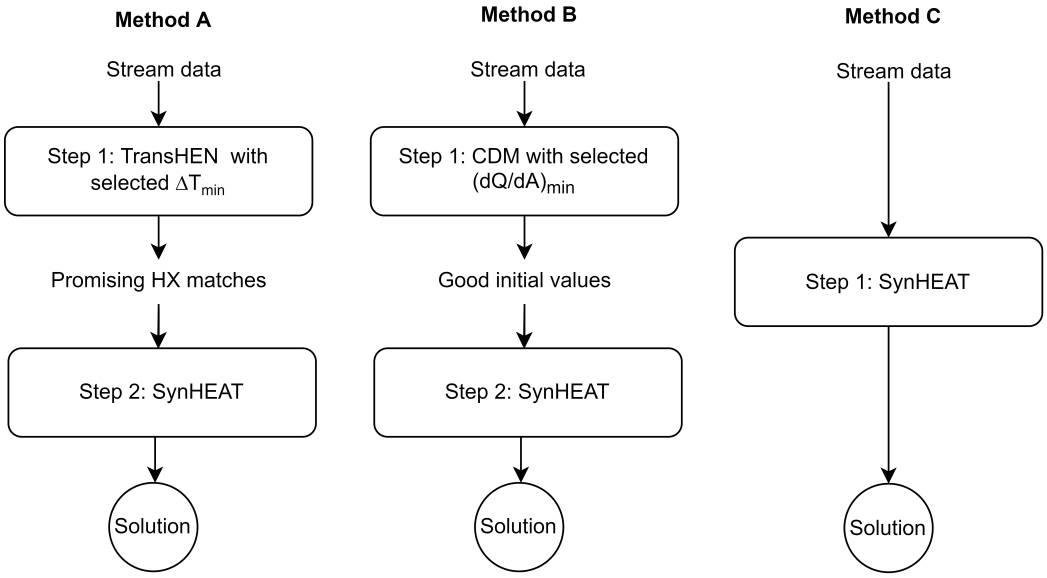

Figure 1: Visual representation of the three HEN synthesis methods used in this study

Table 1: Stream data list and symbol required for HEN synthesis problems

Stream Property	Unit	Stream type			
		Hot	Cold	Hot Utility	Cold Utility
Supply temperature	K	$T_{h,in}$	$T_{c,in}$	$T_{hu,in}$	$T_{cu,in}$
Target temperature	K	$T_{h,out}$	$T_{c,out}$	$T_{hu,out}$	$T_{cu,out}$
Heat flow capacity	kW/K	F_h	F_c	F_{hu}	F_{cu}
Film heat transfer coefficient (HTC)	kW/m^2-K	htc_h	htc_c	htc_{hu}	htc_{cu}
Stream cost	\$/kW-y	N/A	N/A	$Cost_{hu}$	$Cost_{cu}$

The capital investment cost of a heat exchanger $Cost_{HX}$ as a function of area A is also required and can be expressed through Eq.(1). The equation consists of a fixed unit cost C_{fixed}, variable cost $C_{variable}$ and area exponent n_a. These parameters are typically expressed in terms of annualised units which are specific to a particular site/region.

$$Cost_{HX} = C_{fixed} + C_{variable} A^{n_a} \tag{1}$$

When discussing the HEN synthesis models presented in this study a range of index sets must first be defined. These sets are implemented using GEKKO parameters and are as follows:

- Index *K* and *S* for the number of stage boundaries and intervals, $k \in S$. In TransHEN these stages represent hot and cold temperature intervals where $k \in S$ for hot and $kk \in S$ for cold intervals.
- Index *I* and *J* for the number of hot and cold streams, $i \in I$, $j \in J$.
- Index *A* for linear area intervals, $t \in AI$.

3.2 Temperature Interval Based Superstructure (TransHEN)

Nemet et al. (2019) developed a MILP transhipment model termed TransHEN which identifies from which hot stream the heat is released and also the temperature interval of release as well as it is known to which cold stream the heat is directed at which temperature interval to select thermodynamically feasible and economically favourable heat exchange matches for global optimisation in the second HEN synthesis step. The temperature intervals allow for the log-mean-temperature-difference (LMTD) to be calculated before optimisation for each match between a hot stream in a temperature interval equal to or greater than a cold stream, retaining a linear area calculation. Temperature boundaries of the intervals are generated analogously to the problem table in PA by considering the inlet and outlet stream temperatures. Nemet et al. (2019) state that additional intervals can be added to increase the accuracy of the area calculations by reducing the size of large intervals. However, this study considers only the inlet and outlet steam temperatures. Several simplifying conditions were made to reduce model complexity, namely an assumption of isothermal mixing and only a single hot and cold utility stream were considered. Compared to the initial transhipment models developed by Papoulias and Grossmann (1983), TransHEN provides a more accurate calculation of the trade-off between utility cost and capital investment which can be used to initialise the second HEN synthesis step and provide better solutions for large scale problems.

The primary difference between the TransHEN model used in this study and the model presented by Nemet et al. (2019) is the linearisation of the heat exchange capital cost function. In this study, the distribution of intervals is calculated such that the difference between the actual cost function and linearised cost function is minimised which results in a greater proportion of intervals at lower areas where the function is more linear as shown in Figure 2. The piecewise approximation can be represented by a disjunction in Eq(2) where A_t^{LB} and A_t^{UB} denote the lower and upper bounds of each interval. The cost function parameters are replaced by mt and bt which represent the slope and y-intercept of the function in each linear interval. The disjunction introduces an additional binary variable zt for each heat exchange match which determines what cost function is used.

$$\bigvee_{t \in AI} \begin{bmatrix} z_t \\ Cost_{HX} = A m_t + b_t \\ A_t^{LB} \le A \le A_t^{UB} \end{bmatrix} \tag{2}$$

Several additional equations were added to TransHEN to handle the linearised cost function. The disjunction in Eq(2) can be expressed with binary variables using the Big-M reformulation, allowing it to be solved with the MILP solver. The value of the 'Big-M' parameter *U* must be carefully selected so that it is large enough to render the inequality redundant when $z_t = 0$ but maintain a strong LP relaxation (Grossmann, 2021). It is unlikely that the area of a single heat exchanger will exceed 20,000 m$^{2;}$ hence a U value of 50,000 was deemed sufficient. The reformulated linear capital cost function for recovery heat exchange is shown in Eq(3). The capital cost

must always be greater or equal to the capital cost of the piecewise function from any area interval, hence the need for Big-M reformulation.

$$Cost_{HX} \geq A_{i,j} m_t + b_t - U\left(1 - z_{t,i,j}\right) \qquad \forall i \in I, j \in J, t \in AI \tag{3}$$

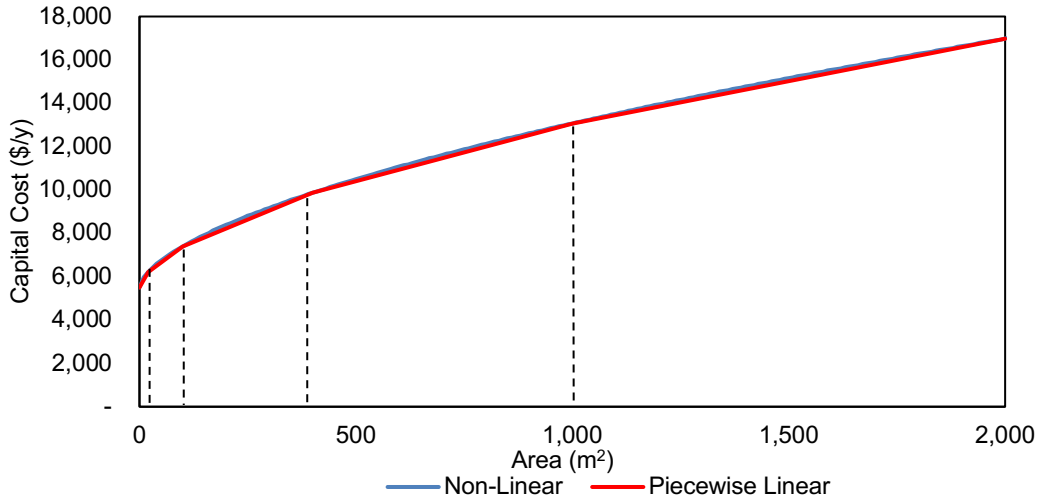

Figure 2: Comparison of example non-linear capital cost function with piecewise linear approximation

Eq(4) specifies that the area for a heat exchange match must belong to one of the area intervals and is selected via a binary variable.

$$\sum_{t \in AI} z_{t,i,j} A_t^{UB} \geq A_{i,j} \geq \sum_{t \in AI} z_{t,i,j} A_{t-1}^{UB} \qquad \forall i \in I, j \in J \tag{4}$$

A particular heat exchange match can only have one area and, consequently, active piecewise cost function. Hence, the amount of active binary area interval variables must be equivalent to the number of binary matches for each *i,j* match i.e. either 1 or 0.

$$\sum_{t \in AI} z_{t,i,j} = z_{i,j} \quad \forall i \in I, j \in J \tag{5}$$

Eq(2) to Eq(5) are repeated for matches been hot stream *i* with cold utility, and cold stream *j* with hot utility.

3.3 Stage-Wise Superstructure with Isothermal Mixing (SynHEAT)

Methods A, B and C all utilise an adaption of the stage-wise SynHEAT model (Yee and Grossmann, 1990). The SynHEAT model offers a reduced superstructure and increased robustness which allows an MINLP problem to be solved to near optimality. Therefore, it is used as the final stage of the HEN synthesis method. A desirable trait of the SynHEAT model is that when an isothermal mixing assumption is made, non-linearities are confined to only the objective function. The number of variables and equations in this study's adaption of SynHEAT model has been reduced due to the functionality of Python GEKKO. The adapted model will be presented in full.

Variables

In SynHEAT, a heat exchange match between hot stream *i* and cold stream *j* can occur once in any stage *s*. Utility matches can only occur at the end of each process stream in a separate stage. Consequently, any heat exchange match been two process streams (recovery) in stage *s*, hot stream and cold utility, or a cold stream and hot utility, can be represented with binary and heat transfer variables as shown in Table 2. Equations constraining a continuous variable to a binary variable are not required in GEKKO, as during initialisation, the variable can be designated as 'binary' and assigned its minimum and maximum values.

Table 2: Variables defining a heat exchange match between process and/or utility streams in SynHEAT

Variable	Recovery	Cold Utility	Hot Utility
Binary Match	$z_{i,j,k}$	$z_{i,cu}$	$z_{j,hu}$
Heat Transfer	$Q_{i,j,k}$	$Q_{i,cu}$	$Q_{j,hu}$

The temperature of each process stream at each stage boundary k is a variable used in area calculations. However, the stage boundary temperature nearest the stream inlet (k = 0 for streams and k = n for cold streams) are set as parameters since utility heat exchangers can only be placed near the stream outlet. This results in a reduced number of solver variables compared to the original SynHEAT model.

- $T_{i,k}$ for temperature of hot stream i at boundary k, $\forall\, k \in K, k \neq 0$
- $T_{j,k}$ for temperature of cold stream j at boundary k, $\forall\, k \in K, k \neq K_n$

A modification to the SynHEAT implementation in this study compared to the original model is the use of two (positive) approach temperature variables, $\theta_{1,i,j,k}$ and $\theta_{2,i,j,k}$, for the hot and cold sides of a recovery heat exchanger between streams i and j at stage k (with lower bounds of ΔT_{min}) instead of a single approach temperature variable in an array (or list). This modification was a critical change to enable the model to solve using APOPT.

Parameters

The following parameters are defined before optimisation using the GEKKO "parameter" variable type. The maximum amount of heat exchanged between hot stream i and cold stream j can be calculated as follows from the stream with the minimum heat duty:

$$Q_{max,i,j} = \min\{F_{h,i}(T_{h,i,in} - T_{h,i,out}), F_{c,j}(T_{c,j,out} - T_{c,j,in})\} \tag{6}$$

As the HTCs are known, the overall HTC for each recovery and utility match can be estimated as follows.

$$U_{i,j} = \frac{1}{(\frac{1}{htc_{h,i}} + \frac{1}{htc_{c,j}})} \quad \forall i \in I, j \in J \tag{7}$$

$$U_{i,cu} = \frac{1}{(\frac{1}{htc_{h,i}} + \frac{1}{htc_{cu}})} \quad \forall i \in I \tag{8}$$

$$U_{j,hu} = \frac{1}{(\frac{1}{htc_{c,j}} + \frac{1}{htc_{hu}})} \quad \forall j \in J \tag{9}$$

Equations

For aesthetics, all equations are written with the right-hand side being zero. The first set of equations define the energy balance of each stream which must be equal to the sum of all heat exchange and utility matches.

$$\sum_{k \in S} \sum_{j \in J} Q_{i,j,k} - F_{h,i}(T_{h,i,in} - T_{h,i,out}) - Q_{i,cu} = 0 \quad \forall\, i \in I \tag{10}$$

$$\sum_{k \in S} \sum_{i \in I} Q_{i,j,k} - F_{c,j}(T_{c,j,out} - T_{c,j,in}) - Q_{j,hu} = 0 \quad \forall\, j \in J \tag{11}$$

An additional set of energy balances at each stage are also needed as the heat recovery in each stage cannot be greater than the available heat content from each stream in that stage.

$$\sum_{j \in J} Q_{i,j,k} - F_{h,i}(T_{i,k+1} - T_{i,k}) = 0 \quad \forall\, i \in I, k \in S \tag{12}$$

$$\sum_{i \in I} Q_{i,j,k} - F_{c,j}(T_{j,k+1} - T_{j,k}) = 0 \quad \forall\, j \in J, k \in S \tag{13}$$

Logic equations are defined so that when the solver selects a binary match ($z_{i,j,k}$ = 1) it allows for heat flow.

$$Q_{i,j,k}(1 - z_{i,j,k}) = 0 \quad \forall\, i \in I, j \in J, k \in S \tag{14}$$

$$Q_{i,cu}(1 - z_{i,cu}) = 0 \quad \forall\, i \in I \tag{15}$$

$$Q_{j,hu}(1 - z_{j,hu}) = 0 \quad \forall\, j \in J \tag{16}$$

The approach temperature equations are expressed such that when a match is rejected by the solver in a particular stage, the value defaults to $\Delta T_{\min}$ (even though the heat flow must be zero). This is to ensure the area equations always use positive values for θ and avoid dividing by zero under any situation. These equations also replace the need for the monotonicity equations in the original SynHEAT model that normally ensure that stage boundary temperatures decrease with increasing stage numbers. Upper bounds are imposed through GEKKO on the approach temperature variables so that they do not exceed the maximum temperature difference of the streams involved in the match, which removes the need for an inequality equation.

$$\theta_{1,i,j,k} - z_{i,j,k}\left(T_{i,k} - T_{j,k}\right) - \Delta T_{min}\left(1 - z_{i,j,k}\right) = 0 \qquad \forall i \in I, j \in J, k \in S \tag{17}$$

$$\theta_{2,i,j,k+1} - z_{i,j,k}\left(T_{i,k+1} - T_{j,k+1}\right) - \Delta T_{min}\left(1 - z_{i,j,k}\right) = 0 \qquad \forall i \in I, j \in J, k \in S \tag{18}$$

Intermediates

With the GEKKO, intermediate variable type equations are constructed for the total hot and cold utility use, while the Chen equation (Chen, 1987) is used to approximate the LMTD and as part of calculating the area of each heat exchanger match. After a solution is obtained, the areas are re-calculated using the actual LMTD to obtain a more accurate total HEN cost.

$$Q_{cu,total} = \sum_{i \in I} Q_{i,cu} \tag{19}$$

$$Q_{hu,total} = \sum_{j \in J} Q_{j,hu} \tag{20}$$

$$A_{i,j,k} = \frac{Q_{i,j,k}}{U_{i,j}\left(0.5\left(\theta_{1,i,j,k}\theta_{2,i,j,k+1}\left(\theta_{1,i,j,k} + \theta_{2,i,j,k+1}\right)\right)\right)^{\frac{1}{3}}} \qquad \forall i \in I, j \in J, k \in S \tag{21}$$

$$A_{i,cu} = \frac{Q_{i,cu}}{U_{i,cu}\left(\frac{1}{2}\left(\left(T_{i,k=S} - T_{cu,out}\right)\left(T_{h,i,out} - T_{cu,in}\right)\left(\left(T_{i,k=S} - T_{cu,out}\right) + \left(T_{h,i,out} - T_{cu,in}\right)\right)\right)\right)^{\frac{1}{3}}} \quad \forall i \in I \tag{22}$$

$$A_{j,hu} = \frac{Q_{j,hu}}{U_{j,hu}\left(\frac{1}{2}\left(\left(T_{hu,in} - T_{c,j,out}\right)\left(T_{hu,out} - T_{c,k=0}\right)\left(\left(T_{hu,in} - T_{c,j,out}\right) + \left(T_{hu,out} - T_{c,k=0}\right)\right)\right)\right)^{\frac{1}{3}}} \quad \forall j \in J \tag{23}$$

The total capital cost for each match type is the sum of cost function, Eq(1), of each heat exchanger.

$$Cost_{HX,recovery,total} = \sum_{i \in I}\sum_{j \in J}\sum_{k \in S}\left(C_{fixed} + C_{variable}A_{i,j,k}^{n_a}\right) \tag{24}$$

$$Cost_{HX,cu,total} = \sum_{i \in I}\left(C_{fixed} + C_{variable}A_{i,cu}^{n_a}\right) \tag{25}$$

$$Cost_{HX,hu,total} = \sum_{j \in J}\left(C_{fixed} + C_{variable}A_{j,hu}^{n_a}\right) \tag{26}$$

The objective function is to minimise the network TAC resulting in an optimised HEN.

$$\min\left\{Cost_{hu}Q_{hu,total} + Cost_{cu}Q_{cu,total} + Cost_{HX,recovery,total} + Cost_{HX,cu,total} + Cost_{HX,hu,total}\right\} \tag{27}$$

3.4 Simplified Cost Derivative Method for HEN Synthesis

Walmsley et al. (2014) first developed the CDM for optimal area allocation of a pre-defined HEN, such as a network synthesised using PA or an MP model. The CDM attempts to add, remove or shift the heat exchanger area between recovery exchangers to where the greatest incremental cost benefit is returned. Analytical derivatives for incremental changes, as opposed to numerical derivatives (which are prone to floating-point errors), were found as the core contribution of the model. A key part of the CDM approach is to determine -dQ_{ut}/dA for each heat exchanger match to detect changes in area allocation that would result in less utility use. Two significant limitations of the CDM is that required the calculation of numerous “flow-on factors” to relate *dQ* of heat recovery for a match to *-dQ* of utility and that relied on other methods for HEN synthesis. This study applies the underlying *dQ/dA* concept of the CDM without the flow-on factor as the key constraint in a stage-wise superstructure that seeks to minimise utility (not TAC). The present study refers to this change as the

simplified CDM. The implementation of the *dQ/dA* constraint is remarkably simple and involves only an additional equation, Eq(28), and a change to the objective function in Eq(29).

$$\left(\left(\frac{dQ}{dA}\right)_{min}\left(T_{i,k+1}-T_{i,k}\right)-U_{i,j}\left(\theta_{1,i,j,k}\theta_{2,i,j,k+1}\right)\right)z_{i,j,k}\leq 0,\ \ \forall\, i\in I, j\in J, k\in S \tag{28}$$

$$min\left\{Q_{hu,total}\right\} \tag{29}$$

As part of the changes, the lower bounds on $\theta_{1,i,j,k}$ and $\theta_{2,i,j,k}$ are replaced with a small non-zero number (e.g., 0.1) to prevent an active heat exchange match having an approach temperature of 0 °C. The objective function of the simplified CDM model is set to minimise the amount of hot utility required. This is because the dQ/dA constraint already embeds the duty vs area trade-off that is inherent to the variable cost component of TAC.

3.5 Initialisation of SynHEAT

One of the limitations of SynHEAT, and optimisation problems in general, is the requirement of initial values for each solver variable and parameter. Multi-step HEN synthesis methods define methods for initialising the final synthesis step with good initial conditions that result in the synthesis of the 'optimal' HEN. Methods A and B represent two different approaches to initialisation.

Method A

In this method, TransHEN is used to populate the superstructure of SynHEAT with feasible and promising heat exchange matches. Unlike TransHEN, the stage/interval boundary temperatures in SynHEAT are variables resulting in a reoptimised HEN with greater accuracy and optimisation of heat exchange arrangement. Nemet et al. (2019) discusses various methods to populate the SynHEAT superstructure after solving TransHEN. This study employs the method where selected matches in TransHEN are allowed in all stages of SynHEAT by setting rejected matches to parameters in SynHEAT resulting in a reduced number of solver variables. A potential disadvantage of Method A is that the initial values for the boundary temperatures, approach temperatures and heat exchange variables remain guesses and, through many tests, are known to have a large influence on the solution. Changing the initial values can lead to the solver finding only infeasible or marginally better solutions. Though these values could be initialised from TransHEN, the method of treating matches between the same two streams in any temperature interval as one heat exchanger regardless of intermediate stream exchange and different superstructure result in poor initial values.

Method B

In this method, the simplified CDM model provides initial values (including temperatures and duties) to SynHEAT. The simplified CDM is highly effective at identifying all matches that have economic potential while turning off all matches that are highly unlikely to be economic. The results of the simplified CDM can directly initialise the SynHEAT model because it uses the same stage-wise superstructure. A potential limitation of this approach is that it does not currently reduce the size of the SynHEAT model, which may prove challenging for large problems that contain more than 100 streams.

3.6 Python Implementation Considerations

Creating the model with Python syntax simplifies several components associated with the model. The four standard variable types can be placed into nested arrays that can be iterated across with list comprehension. A benefit of list comprehension is that it is translated from set theory terminologically very intuitively as, for example, $i+1\ \forall\ i \in I$, can be written with the Python 'for', 'in' and 'range' function as "[i+1 for i in range(I)]". Evidently, reproducing optimisation models defined mathematically into GEKKO is an easier task.

Applying constraints to the variable types with Boolean logic is simplified with an intrinsic property of GEKKO. A variable created in GEKKO is actually created in the APOPT solver model file, and GEKKO creates a path for Python to locate the variable in the file. Consequently, a Python array consisting of GEKKO variables does not contain the variables themselves but instead the paths to the variable in the solver file. Boolean logic can then be easily applied to variables, particularly equations, outside of the equation, which is written as the 'expression' component of the array, without introducing slack variables into the model. A practical example is if two streams, i_2 and j_3 had incompatible fluids hence heat exchange matches between the streams should be forbidden. Using Python syntax, the recovery binary match and heat exchange variable array would then be modified so that all combinations of streams except the forbidden matches were created as variables, i.e., "[variable if i≠2 and j≠3 else parameter for i in range(I) for j in range(J)]". A further use of this GEKKO property was for model reduction as equations and variables for infeasible matches could be disallowed before solving the set of equations by creating an index set containing all feasible matches. If a combination of elements defining a match were not present in the index set, then an equation was not created for that match. This was particularly useful for

reducing the number of variables and equations in SynHEAT and TransHEN as the logic equations for a match were redundant if a stream was not present in an interval and hence could not exchange heat with another stream or utility.

4. Case Studies

Three case studies were used to demonstrate the different HEN synthesis methods presented in this study and compare with reported solutions in the literature. In particular, Case Study 1 and Case Study 2 are very common HEN synthesis problems as they represent a small and medium scale problem and are often used as benchmark problems in the literature. The stream data is presented in Tables 3, 4 and 5.

Table 3: Stream data for Case Study 1 proposed by Yee and Grossmann (1990)

Stream	T_{supply} (K)	T_{target} (K)	F (kW/K)	HTC (kW/m^2-K)	Cost ($/kW-y)
Hot 1	650	370	10	1	
Hot 2	590	370	20	1	
Cold 1	410	650	15	1	
Cold 2	350	500	13	1	
Hot Utility	680	680		5	80
Cold Utility	300	320		1	15
			$Cost_{HX}$ = 5,500 + 150A[1]		

Table 4: Stream data for Case Study 2 proposed by Linnhoff and Ahmad (1990)

Stream	T_{supply} (K)	T_{target} (K)	F (kW/K)	HTC (kW/m^2-K)	Cost ($/kW-y)
Hot 1	600.15	313.15	100	0.5	
Hot 2	493.15	433.15	160	0.4	
Hot 3	493.15	333.15	60	0.14	
Hot 4	433.15	318.15	400	0.3	
Cold 1	373.15	573.15	100	0.35	
Cold 2	308.15	437.15	70	0.7	
Cold 3	358.15	411.15	350	0.5	
Cold 4	333.15	443.15	60	0.14	
Cold 5	413.15	573.15	200	0.6	
Hot Utility	603.15	523.15		0.5	60
Cold Utility	288.15	303.15		0.5	6
			$Cost_{HX}$ = 2,000 + 70A[1]		

5. Results

The three case studies were solved on a Windows PC with an Intel Core i5-10500 CPU (3.1 GHz) and 16 GB of RAM. For all the three case studies the two step HEN synthesis Method B yielded a lower TAC as shown in Table 6, which is indicative of a more optimal HEN. For the larger problems, Case Studies 2 and 3, Method C synthesised and increasingly suboptimal HEN compared to the two-step Methods A and B which suggests that initialisation is critical for large problems. TransHEN in Method A was simplified compared to the other methods as discussed by Nemet et al. (2019) and a comprehensive adaptation might achieve better results.

Tables 7, 8 and 9 compare the best network solution obtained from this study with several results from the literature. Alongside the original solution from the source of the case study, recent literature entries were considered for comparison as these represent the state-of-the-art and contain considerable enhancements to the early works of Yee and Grossmann (1990) and Papoulias and Grossmann (1983). This ensures that a fair comparison is made by avoiding the issue of selecting only poor literature solutions. For Case Study 1, the open-source tool obtained one of the best solutions in the literature. The difference in TAC between this study and Caballero et al. (2021) may only be due to how GAMS and Python treat floating point numbers as the utility use and heat recovery are the same but the areas differ by 0.06 % causing a slightly different TAC. Note that for comparing literature solutions N/S signifies when the value was not specified in the paper.

Table 5: Stream data for Case Study 3 proposed by Björk and Pettersson (2003)

Stream	T_{supply} (K)	T_{target} (K)	F (kW/K)	HTC (kW/m^2-K)	Cost ($/kW-y)
Hot 1	453.15	348.15	30	2	
Hot 2	553.15	393.15	60	1	
Hot 3	453.15	348.15	30	2	
Hot 4	413.15	313.15	30	1	
Hot 5	493.15	393.15	50	1	
Hot 6	453.15	328.15	35	2	
Hot 7	473.15	333.15	30	0.4	
Hot 8	393.15	313.15	100	0.5	
Cold 1	313.15	503.15	20	1	
Cold 2	373.15	493.15	60	1	
Cold 3	313.15	463.15	35	2	
Cold 4	323.15	463.15	30	2	
Cold 5	323.15	523.15	60	2	
Cold 6	363.15	463.15	50	1	
Cold 7	433.15	523.15	60	3	
Hot Utility	598.15	598.15		1	80
Cold Utility	298.15	313.15		2	10
		$Cost_{HX} = 8{,}000 + 500A^{0.75}$			

Table 6: Comparison of TAC between different the different methods of the HEN synthesis tool

Problem	Method A ($/y)	Method B ($/y)	Method C ($/y)
Case Study 1	154,436	154,436	184,182
Case Study 2	3,311,186	3,002,982	3,788,431
Case Study 3	2,167,513	1,879,925	3,968,939

Table 7: Comparison with literature solutions for Case Study 1

Solution	Area (m^2)	Hot Utility (kW)	Cold Utility (kW)	Solver	TAC ($/y)	Isothermal Mixing
Yee and Grossmann (1990)	N/S	N/S	N/S	DICOPT (IBM308)	154,893	Yes
Chang et al. (2020)	337	490.6	2,140.5	CPLEX (GAMS)	154,911	No
Faria et al. (2015)	337.2	491.1	2,114.1	DICOPT (GAMS)	154,995	Yes
Caballero et al. (2021)	347.8	526.2	2,176.2	BARON (GAMS)	154.406	No
This study (Method B)	348.0	526.2	2,176.2	APOPT (GEKKO)	154,436	Yes

Table 8: Comparison with literature solutions for Case Study 2

Solution	Area (m^2)	Hot Utility (MW)	Cold Utility (MW)	Solver	TAC ($/y)	Isothermal Mixing
Linnhoff and Ahmad (1990)	17,400	25.31	33.02	N/S	2,930,000	N/S
Caballero et al. (2021)	17,709	26.97	31.59	BARON (GAMS)	2,891,064	No
Pavão et al. (2017)	18,054	24.13	31.85	Particleswarm (Matlab)	2,928,629	No
This study (Method B)	16,658	26.74	34.45	APOPT (GEKKO)	3,002,982	Yes

For Case Studies 2 and 3, shown in Tables 8 and 9, the best solution obtained by this study is within 4 % and 21 %, of the best solution reported in the literature. These results highlight areas for future development of the open-source tool as there are some possible reasons for the large deviation from the best literature solution. Firstly, this study made several simplifying assumptions, namely isothermal mixing, which may exclude promising matches at the end of stream splits or better allocation of the area of matches made in splits. Yee and Grossmann (1990) also state that the assumption of isothermal may lead to the overestimation of areas as the temperature difference across the exchanger may be higher when stream splits are present. The isothermal mixing assumption holds true for Case Study 1 as the optimum solution contains no stream splits which allowed this study to obtain a very good result. For Case Study 2 and 3, the optimal solution contains numerous stream splits so models in the literature with non-isothermal mixing were able to achieve a better solution, aside from

Chang et al. (2020) who made significant modifications to the SynHEAT model and employed an enumeration algorithm to find solutions with different energy targets. Compared to BARON and DICOPT, APOPT is relatively new solver and still in development so it may not be as refined for MINLP optimisation problems.

Table 9: Comparison with literature solutions for Case Study 3

Solution	Area (m^2)	Hot Utility (MW)	Cold Utility (MW)	Solver	TAC ($/y)	Isothermal Mixing
Björk and Pettersson (2003)	N/S	N/S	N/S	N/S	1,513,854	N/S
Chang et al. (2020)	4,359	10.24	7.68	CPLEX (GAMS)	1,501,004	Yes
Caballero et al. (2021)	4,128	10.01	7.68	BARON (GAMS)	1,492,869	No
Pavão et al. (2017)	4,028	10.21	7.86	Particleswarm (Matlab)	1,525,735	No
This study (Method B)	3,394	12.98	10.61	APOPT (GEKKO)	1,879,925	Yes

6. Conclusion

This study successfully demonstrates the use of an open-source tool for HEN synthesis with comparison against common literature HEN problems. The tool is created in Python and interfaced with the solver, APOPT, through the GEKKO library and features three different synthesis methods. Using benchmark case studies, the two-step methods provided a more optimal HEN compared to SynHEAT alone and is comparable to other solutions in the literature. One of the methods, Method B features a novel application of a CDM developed by Walmsley et al. (2014) to HEN synthesis which provides good initial values for global optimisation in the second synthesis step involving the Yee and Grossmann (1990) SynHEAT model which resulted in the lowest TAC of the three methods. The assumption of isothermal mixing simplifies the synthesis models but may be preventing the solver from attaining better solutions for larger problems which require stream splits. Future versions of the open-source tool will contain non-isothermal mixing models to allow for the synthesis of more optimal HEN's.

References

Beal L.D.R., Hill D.C., Martin R.A., Hedengren J.D., 2018, GEKKO Optimization Suite, Processes, 6(8), 106.

Björk K.-M., Pettersson F., 2003, Optimization of large-scale heat exchanger network synthesis problems.

Caballero J.A., Pavão L.V., Costa C.B.B., Ravagnani M.A.S.S., 2021, A Novel Sequential Approach for the Design of Heat Exchanger Networks, Frontiers in Chemical Engineering, 3, DOI: 10.3389/fceng.2021.733186.

Chang C., Liao Z., Costa A.L.H., Bagajewicz M.J., 2020, Globally optimal synthesis of heat exchanger networks. Part II: Non-minimal networks, AIChE Journal, 66(7), e16264.

Chen J.J.J., 1987, Comments on improvements on a replacement for the logarithmic mean, Chemical Engineering Science, 42(10), 2488–2489.

Faria D.C., Kim S.Y., Bagajewicz M.J., 2015, Global Optimization of the Stage-wise Superstructure Model for Heat Exchanger Networks, Industrial & Engineering Chemistry Research, 54(5), 1595–1604.

Fu C., Vikse M., Gundersen T., 2017, Challenges in Work and Heat Integration, Chemical Engineering Transactions, 61, 601–606.

Grossmann I.E., 2021, Advanced Optimization for Process Systems Engineering, Cambridge University Press.

Linnhoff B., Ahmad S., 1990, Cost optimum heat exchanger networks—1. Minimum energy and capital using simple models for capital cost, Computers & Chemical Engineering, 14(7), 729–750.

Linnhoff B., Flower J.R., 1978, Synthesis of heat exchanger networks: I. Systematic generation of energy optimal networks, AIChE Journal, 24(4), 633–642.

Nemet A., Isafiade A.J., Klemeš J.J., Kravanja Z., 2019, Two-step MILP/MINLP approach for the synthesis of large-scale HENs, Chemical Engineering Science, 197, 432–448.

Papoulias S.A., Grossmann I.E., 1983, A structural optimization approach in process synthesis—II: Heat recovery networks, Computers & Chemical Engineering, 7(6), 707–721.

Pavão L.V., Costa C.B.B., Ravagnani M.A.S.S., 2017, Heat Exchanger Network Synthesis without stream splits using parallelized and simplified simulated Annealing and Particle Swarm Optimization, Chemical Engineering Science, 158(September 2016), 96–107.

Walmsley T.G., Walmsley M.R.W., Morrison A.S., Atkins M.J., Neale J.R., 2014, A derivative based method for cost optimal area allocation in heat exchanger networks, Applied Thermal Engineering, 70(2), 1084–1096.

Yee T.F., Grossmann I.E., 1990, Simultaneous optimization models for heat integration—II. Heat exchanger network synthesis, Computers & Chemical Engineering, 14(10), 1165–1184.