2014

Computational Strategies for Large-Scale MILP Transshipment Models for Heat Exchanger Network Synthesis

Yang Chen
Carnegie Mellon University

Ignacio E. Grossmann
Carnegie Mellon University, grossmann@cmu.edu

David C. Miller
U.S. Department of Energy

Computational Strategies for Large-Scale MILP Transshipment Models for Heat Exchanger Network Synthesis

Yang Chen [a,*], Ignacio E. Grossmann [a], David C. Miller [b]

[a] *Department of Chemical Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, United States*

[b] *U.S. Department of Energy, National Energy Technology Laboratory, 626 Cochrans Mill Road, Pittsburgh, PA 15236, United States*

Abstract

Determining the minimum number of units is an important step in heat exchanger network synthesis (HENS). The MILP transshipment model (Papoulias and Grossmann, 1983) and transportation model (Cerda and Westerberg, 1983b) were developed for this purpose. However, they are computationally expensive when solving for large-scale problems. Several approaches are studied in this paper to enable the fast solution of large-scale MILP transshipment models. Model reformulation techniques are developed for tighter formulations with reduced LP relaxation gaps. Solution strategies are also proposed for improving the efficiency of the branch and bound method. Both approaches aim at finding the exact global optimal solution with reduced solution times. Several approximation approaches are also developed for finding good approximate solutions in relatively short times. Case study results show that the MILP transshipment model can be solved for relatively large-scale problems in reasonable times by applying the approaches proposed in this paper.

Key Words

heat exchanger network synthesis (HENS), transshipment model, mixed-integer linear programming, computational strategies, model reformulation

* Corresponding author. Tel.: +1 412-386-4798.
Email address: yangchen@andrew.cmu.edu.

Highlights

- Model reformulations and several solution strategies are proposed for finding the optimal solution of the MILP transshipment model.
- Several approximation approaches are developed for finding near optimal solutions.
- Some relatively large-scale MILP transshipment models are solved in reasonable times by using the approaches proposed in this study.

1. Introduction

Heat exchanger network synthesis (HENS) has been an important topic in power, refining and chemical industries for several decades due to its crucial role in energy savings and cost reduction. Recently it is also of increased interest in broader areas, including carbon capture and storage (CCS), water treatment and energy polygeneration. HENS has been extensively studied in process systems engineering research, and a number of methodologies have been developed. Linnhoff and Hindmarsh (1983) proposed the pinch design method, which is based on physical insight for the maximum heat recovery in heat exchanger networks. Mathematical programming based approaches were first developed by Papoulias and Grossmann (1983) and Cerda and Westerberg (1983a, 1983b). Both methods are now widely used in grassroots design and retrofit of heat exchanger networks. Detailed reviews on developments of HENS methods can be found in Gundersen and Naess (1988), Furman and Sahinidis (2002), Morar and Agachi (2010) and Klemeš and Kravanja (2013).

Two different types of HENS approaches have been studied: sequential and simultaneous. In the sequential approach the HENS problem is solved in three steps (Biegler et al., 1997): first, the utility cost (or consumption) is minimized with a linear programming (LP) model (Papoulias and Grossmann, 1983; Cerda and Westerberg, 1983a); second, the number of heat exchangers is minimized with a mixed-integer linear programming (MILP) model to determine the optimal stream matches (Papoulias and Grossmann,

1983; Cerda and Westerberg, 1983b); finally, the total investment cost is minimized with a nonlinear programming (NLP) model, and the optimal heat exchanger network structure is derived (Floudas et al., 1986). In contrast to the target-based sequential approach, the simultaneous synthesis approach optimizes energy, number of units and total heat exchanger area simultaneously in a mixed-integer nonlinear programming (MINLP) model (Yee and Grossmann, 1990; Ciric and Floudas, 1991). Due to its computational complexity, the simultaneous approach usually can only solve problems with small to medium sizes. The sequential approach, on the other hand, decomposes the HENS problem into several smaller subproblems that are much easier to solve, and hence is still considered the most practical way to solve industrial-scale HENS problems.

In the sequential approach, minimization of the number of heat exchangers is a key step to determine the optimal structure of heat exchanger networks. The MILP transhipment and MILP transportation models are basic tools to calculate the minimum number of units. The major difference between the two is that the former uses a heat cascade while the latter uses direct matches, which makes the size of the former model significantly smaller. These models have been further developed during the last twenty years. A vertical MILP transshipment model was proposed by Gundersen and Grossmann (1990) and Gundersen et al. (1996), in which non-vertical heat transfer (i.e., criss-crossing) was minimized and the optimal solution with smallest heat exchange area could be identified. Floudas and Grossmann (1986) extended the MILP transshipment model for multiperiod operations. The MILP transshipment/transportation models were also incorporated into the simultaneous approach for heat exchanger network design (Shethna et al., 2000; Barbaro and Bagajewicz, 2005) and retrofit (Nguyen et al., 2010). In process synthesis, the MILP transshipment model has been applied to the optimal design of heat exchanger networks in a wide range of systems and processes, such as refrigeration systems (Shelton and Grossmann, 1986), batch/semi-continuous processes (Zhao et al., 1998), water utilization systems (Bagajewicz et al., 2002) and hybrid transportation fuel production processes (Elia et al., 2010).

Despite significant advances in MILP solvers (e.g., CPLEX, GUROBI, XPRESS) and application of the MILP transhipment/transportation models, solving the MILP itself is still quite challenging. Furman and Sahinidis (2001) proved that the minimum number of matches problem is $\mathcal{NP}$-hard in the strong sense due to its combinatorial nature. So far the MILP transshipment/transportation models are quite difficult to solve for large-scale problems, as will be shown later, rendering the minimum number of units problem as the major bottleneck in the HENS procedure. Only a few papers have investigated efficient solution approaches for large-scale MILP transshipment/transportation models. Gundersen et al. (1997) developed a MILP transshipment formulation with tighter heat transfer upper bounds and indicated that the gap between the MILP solution and its LP relaxation could be reduced by using the tighter formulation. Anantharaman et al. (2010) systematically studied approaches for improving the solution performance for the MILP transshipment model. The authors proposed three major approaches: pre-processing to reduce model size using insight and heuristics, model modification/reformulation, and improving efficiency of the branch and bound method. Several ideas for model modification and reformulation were investigated in this article, including decreasing the upper bound, adding integer cuts and reformulating the original model to set-partitioning formulations. The authors tested these ideas for several cases with the size of up to 22 process streams, and showed that the LP relaxation was significantly tightened. However, the above two papers did not present the effect on solution times. Hence, the actual computational performance of these model reformulations is unknown. Pettersson (2005) developed an approximate approach, which includes match set reduction and grouping, and solved the minimum number of matches problem with up to 39 process streams in reasonable times. The minimum number of units problem was also solved by evolutionary methods (Mocsny and Govind, 1984; Shethna and Jezowski, 2006). These approximate approaches, however, cannot guarantee the global optimal solution and also cannot indicate how far the obtained solution is with respect to the global optimum.

This paper investigates several rigorous and approximate approaches for reducing the computational time required to solve large-scale MILP transshipment models. Both LP relaxation and solution time results

are presented. The remaining part of this paper is organized as follows. Section 2 presents solution times of MILP transshipment models for a variety of cases with small to large sizes, discusses reasons for the slow computation, and then shows results for weighted matches. Section 3 proposes several reformulations, including model disaggregation and addition of integer cuts, and then compares their results with the original model. Several solution strategies, e.g., branching priority, strong branching, parallel computing, are discussed in Section 4. Finally, Section 5 describes several approximation approaches, including using a relative optimality gap, a combined model, a reduced MILP model and a NLP reformulation. The paper is concluded in Section 6.

2. Preliminary Results

2.1 MILP Transshipment Model – Case Study and Results

The MILP transshipment model is usually difficult to solve for the full heat exchanger network due to its computational complexity. Instead, the full network is partitioned into several subnetworks defined by the pinch points, and then the MILP transshipment model is solved for each subnetwork. This is a reasonable procedure because stream matches across the pinch are usually avoided (for exceptions see Wood et al. (1985)).

Before solving the MILP transshipment model, the following LP transshipment model (Papoulias and Grossmann, 1983) is solved, which provides the minimum utility consumption and the location of pinch points that partition the full network into subnetworks:

$$\min Z = \sum_{m \in S} c_m Q_m^S + \sum_{n \in W} c_n Q_n^W \qquad \text{(M0)}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j \in C_k} Q_{ijk} + \sum_{n \in W_k} Q_{ink} = Q_{ik}^H \qquad \forall i \in H'_k \qquad \forall k \in K$$

$$R_{mk} - R_{m,k-1} + \sum_{j \in C_k} Q_{mjk} - Q_m^S = 0 \qquad \forall m \in S'_k \qquad \forall k \in K$$

$$\sum_{i\in H_k} Q_{ijk} + \sum_{m\in S_k} Q_{mjk} = Q_{jk}^{C} \qquad \forall j \in C_k \qquad \forall k \in K$$

$$\sum_{i\in H_k} Q_{ink} - Q_n^{W} = 0 \qquad \forall n \in W_k \qquad \forall k \in K$$

$$R_{ik},\ R_{mk},\ Q_{ijk},\ Q_{mjk},\ Q_{ink},\ Q_m^{S},\ Q_n^{W} \geq 0$$

$$R_{i0} = R_{iK} = 0$$

where K is the index set for all temperature intervals; c_m and c_n are the unit cost of hot utility m and cold utility n, which are known parameters; Q_{ik}^{H} and Q_{jk}^{C} are the heat content of hot process stream i and cold process stream j at temperature interval k, which are known parameters; Q_m^{S} and Q_n^{W} are the heat load of hot utility m and cold utility n; Q_{ijk}, Q_{mjk}, and Q_{ink} are the amount of heat exchanged between hot stream i and cold stream j, hot utility m and cold stream j, and hot stream i and cold utility n at interval k; R_{ik} and R_{mk} are the heat residual of hot stream i and hot utility m exiting interval k. Pinch points are identified by these temperature intervals for which all the heat residuals are zero. The index sets are defined below:

$$H_k = \{ i \mid \text{hot stream } i \text{ supplies heat to interval } k \}$$

$$H'_k = \{ i \mid \text{hot stream } i \text{ is present at interval } k \text{ or at a higher interval} \}$$

$$C_k = \{ j \mid \text{cold stream } j \text{ demands heat from interval } k \}$$

$$S_k = \{ m \mid \text{hot utility } m \text{ supplies heat to interval } k \}$$

$$S'_k = \{ m \mid \text{hot utility } m \text{ is present at interval } k \text{ or at a higher interval} \}$$

$$W_k = \{ n \mid \text{cold utility } n \text{ extracts heat from interval } k \}$$

Next, the MILP transshipment model is solved for each subnetwork, obtaining the minimum number of units and the optimal matches between hot and cold streams. In this model, the heat loads of hot and cold

utilities are fixed at values obtained in Model (M0). Both hot process streams and hot utilities (cold process streams and cold utilities) are considered as additional hot streams (cold streams), and the common index *i* (*j*) is used for them. The MILP transshipment model for each subnetwork *q* is formulated as follows (Papoulias and Grossmann, 1983):

$$\min \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q} \tag{M1}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^{H} \qquad \forall i \in H_k' \qquad \forall k \in K_q \tag{M1-1}$$

$$\sum_{i\in H_k} Q_{ijk} = Q_{jk}^{C} \qquad \forall j \in C_k \qquad \forall k \in K_q \tag{M1-2}$$

$$\sum_{k\in K_q} Q_{ijk} - Q_{ij}^{U,q} y_{ij}^{q} \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \tag{M1-3}$$

$$R_{ik},\ Q_{ijk} \ge 0 \qquad y_{ij}^{q} \in \{0,1\}$$

where K_q is the index set for all temperature intervals in subnetwork *q*; y_{ij}^{q} is the binary variable that indicates the existence of match between hot stream *i* and cold stream *j* in subnetwork *q*; $Q_{ij}^{U,q}$ is the upper bound of heat exchanged between hot stream *i* and cold stream *j* in subnetwork *q*; H_q and C_q are index sets for all hot streams and cold streams present in subnetwork *q*. All other variables and index sets have the same meaning as in Model (M0). The upper bound $Q_{ij}^{U,q}$ is traditionally given by the smaller of the total heat content of hot stream *i* and cold stream *j* in the subnetwork: $Q_{ij}^{U,q} = \min\left\{\sum_{k\in K_q} Q_{ik}^{H},\ \sum_{k\in K_q} Q_{jk}^{C}\right\}$. Recently a tighter upper bound has been proposed by Gundersen et al. (1997), which is used in this study:

$$Q_{ij}^{U,q} = \min\left\{\sum_{k\in K_q} Q_{ik}^{H},\ \sum_{k\in K_q} Q_{jk}^{C}, \max\left[\min\left(FCp_i^{H}, FCp_j^{C}\right)\cdot\left(T_{in,i}^{H} - T_{in,j}^{C} - \mathrm{EMAT}\right), 0\right]\right\} \tag{1}$$

where FCp_i^H and FCp_j^C are heat capacity flow rates of hot stream *i* and cold stream *j*; $T_{in,i}^H$ and $T_{in,j}^C$ are inlet temperatures of hot stream *i* and cold stream *j*; the exchanger minimum approach temperature (EMAT) is set to be zero.

Model (M1) is tested for multiple cases. The stream information for all cases, including heat capacity flow rate (FCp), inlet temperature (T_{in}) and outlet temperature (T_{out}), is listed in Tables A1 and A2 in the Appendix. These cases are categorized into two types: balanced streams and unbalanced streams. Cases with balanced streams have similar FCps within the same order of magnitude (0.8 - 2.8); while cases with unbalanced streams have dissimilar FCps, whose values can span several orders of magnitude (0.2 - 14). Streams of all cases are selected as the subset of streams listed in Tables A1 and A2: The case with m hot streams and n cold streams (mH, nC) uses the first m hot streams and the first n cold streams in Table A1 (or A2). Two utilities, high-pressure steam (500°C) and medium-pressure steam (350°C), and one cold utility, cooling water (20-30°C), are utilized. In all cases, the heat recovery approach temperature (HRAT) is set to 10K.

The test problems are solved on a computer with single core 3.33 GHz CPU (except for the parallel computing case studies as will be shown later), 24.0 GB memory and running Windows 7. GAMS 24.1.3 (McCarl et al., 2013) is used to formulate the model. Two MILP solvers, CPLEX 12.5 (CPLEX, 2013) and GUROBI 5.5 (GUROBI, 2013), are employed to solve the model to global optimality. Only the solver time reported by GAMS is reported here. Since the objective function of Problem (M1) can only be integer values, an absolute gap of 0.99 is used for the termination criterion. The optimal objective values and solution times for all cases are listed in Table 1 in which the reported CPU times are the sum of times for the solution of the MILP for each subnetwork.

Table 1. Optimization results and solution times for different cases by CPLEX and GUROBI

Case [b]	Optimal Objective Value [a]	CPU Time (s) [a]	
		CPLEX	GUROBI
Balanced Streams			
5H, 5C	24	0.5	0.5
8H, 8C	35	35.9	58.4
10H, 10C	42	1,017.9	949.9
12H, 12C	48	68,688.6	62,478.2
15H, 15C [c]	57	> 100,000	> 100,000
Unbalanced Streams			
5H, 5C	26	0.3	0.1
10H, 10C	39	25.7	16.2
15H, 15C	55	660.1	3,135.1
17H, 17C [d]	67	> 100,000	> 100,000
20H, 20C [c]	77	> 100,000	> 100,000

[a] Absolute gap 0.99 is applied.
[b] mH, nC means m hot process streams and n cold process streams. HRAT = 10K.
[c] Global optimal solutions are not confirmed due to very long computational times. For these cases, best solutions obtained so far are present.
[d] Global optimal solution is obtained by using advanced computational strategies, which will be discussed later.

For both solvers, the solution time increases exponentially with problem size. Despite the very significant progress of MILP solvers in recent years, the transshipment model (M1) can only be solved for problems with small to medium sizes (up to 15H, 15C). It is found that problems with unbalanced streams are easier to solve than those with balanced streams. This observation is expected because the matches are more restricted in cases with unbalanced streams, especially for those streams with large FCps. Note that most industrial cases actually have unbalanced streams, while cases with balanced streams are mostly used in academic papers. Comparing the performance of the two MILP solvers, CPLEX achieves similar solution times as GUROBI for balanced cases but shorter times for unbalanced cases. Hence, the overall performance of CPLEX is better than GUROBI for solving the MILP transshipment model. CPLEX is thus chosen as the MILP solver for all the following case studies.

2.2 Discussion

There are several reasons for slow computation of Model (M1):

a) Same coefficients in the objective function.

b) Symmetry in the problem structure (Nemhauser and Wolsey, 1988).

c) Large LP relaxation gap.

Reason (a) is obvious since all binary variables in the objective function are multiplied by the coefficient, one. This tends to introduce degeneracy, that is, multiple optimal solutions with the same objective value. It is not an appropriate approach to change those coefficients to other values because the model then loses its physical meaning, i.e, the minimum number of matches. However, we can try to modify the coefficients to reflect potential heat exchange areas so that the computational speed is accelerated. We will discuss this approach in Section 2.3: Weighted Model.

Reason (b) also follows from having the same coefficients in the objective function. The existence of symmetry in the model, which implies many alternative solutions with the same objective value, decreases branch and bound efficiency since many nodes with equivalent solutions are explored, which significantly increases the computational time (Margot, 2003). This also explains why unbalanced cases are easier to solve, since they may have less symmetry. It is difficult to develop symmetry breaking constraints in the MILP transshipment model. One possible way to reduce the effect of the symmetry is to introduce branching priority for binary variables. Some CPLEX options, e.g., strong branching, may also be helpful. These approaches will be discussed in Section 4: Solution Strategies.

Reason (c) is verified by Table 2, which shows fairly large gaps between the optimal objective values and their LP relaxations for multiple cases (between 23.2 % and 33.0 %). The LP relaxation can be reduced

by introducing tighter formulations and adding integer cuts, which will be covered in Section 3: Model Reformulations.

Table 2. LP relaxation for different cases

Case	Optimal Objective Value	LP Relaxation Value
Balanced Streams		
5H, 5C	24	16.30
8H, 8C	35	24.36
10H, 10C	42	28.85
12H, 12C	48	32.14
15H, 15C	57	40.39
Unbalanced Streams		
5H, 5C	26	17.93
10H, 10C	39	29.97
15H, 15C	55	41.42
17H, 17C	67	48.84
20H, 20C	77	56.59

Remark

The significant performance difference between the balanced and unbalanced cases, in which only the values of FCps are different, indicates that parameter values may have a strong influence on both optimal results and solution times. To investigate the sensitivity of parameters on the solution and performance, the nominal values of FCps in both balanced and unbalanced cases are perturbed between ±10% from their base values (as shown in Table A1 and A2). Each case is run ten times with random values for FCps in the above interval. To keep the problems simple, all random values of FCps are rounded to the nearest tenth. The range of optimal objective values and solution times obtained from these ten runs for all cases are listed in Table 3. The results show that the selection of parameters has a significant impact on the solution time, but a much smaller impact on the optimal objective value. However, for most of cases, the average solution times with random values of FCps are still within the same order of magnitude as solution times with base values of FCps.

Table 3. Case study results with random values of FCps

Case	With Nominal Values of FCps [a]		With Random Values of FCps [b]		
	Optimal Value	CPU Time (s)	Range of Optimal Values	Average CPU Time (s)	Range of CPU Times (s)
Balanced Streams					
5H, 5C	24	0.5	23 - 25	0.4	0.3 – 0.5
8H, 8C	35	35.9	32 - 36	24.4	3.3 – 73.2
10H, 10C	42	1,017.9	40 - 43	2,523.0	49.1 – 16,772.4
12H, 12C	48	68,688.6	45 - 48	71,610.0	4,796.9 – > 100,000
15H, 15C	57	> 100,000	56 - 60	> 100,000	> 100,000 – > 100,000
Unbalanced Streams					
5H, 5C	26	0.3	24 - 26	0.2	0.1 – 0.3
10H, 10C	39	25.7	38 - 40	15.6	3.9 – 29.3
15H, 15C	55	660.1	55 - 59	8,610.7	369.5 – 40,293.0
17H, 17C	67	> 100,000	67 - 70	> 100,000	78,280.7 – > 100,000
20H, 20C	77	> 100,000	77 - 83	> 100,000	> 100,000 – > 100,000

[a] Base values of FCps are taken from Table A1 and A2.
[b] Random values of FCps are randomly generated within the interval between 90% and 110% of their base values. To keep simplicity, all random values contain at most one decimal.

2.3 Weighted Model

We first try to develop the weighted model with non-uniform coefficients in the objective function, which is the easiest approach. The idea of introducing weight factors to the objective function of the MILP transshipment model has been studied previously. Papoulias and Grossmann (1983) mentioned that weight factors could be generated by some pre-defined priorities for matches. Elia et al. (2010) introduced weight factors that were calculated by the order of the distance between two units and the order of stream flowrates (or equipment heat transfer rates). However, these weighted models were aimed at reducing the number of optimal solutions. Accelerating the computational speed was not the purpose of these articles, and no solution times were reported.

In this study, weight factors are added to the objective function in Model (M1) to reflect the potential heat exchange areas in terms of heat transfer coefficients and temperature driving forces. The following weighted model is formulated:

$$\min \sum_{i\in H_q}\sum_{j\in C_q} w_{ij}^q y_{ij}^q \tag{M2}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^H \qquad \forall i \in H_k' \qquad \forall k \in K_q$$

$$\sum_{i\in H_k} Q_{ijk} = Q_{jk}^C \qquad \forall j \in C_k \qquad \forall k \in K_q$$

$$\sum_{k\in K_q} Q_{ijk} - Q_{ij}^{U,q} y_{ij}^q \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q$$

$$R_{ik},\ Q_{ijk} \ge 0 \qquad y_{ij}^q \in \{0,1\}$$

All equations in Model (M2) are the same as (M1) except for the objective function. w_{ij}^q is the weight factor for the match (*i*,*j*) in subnetwork *q* (or y_{ij}^q), and is defined as:

$$w_{ij}^q = \frac{Q_{ij}^{U,q}}{\Delta T_{ij}^q} \tag{2}$$

where $Q_{ij}^{U,q}$ is the upper bound of heat transfer for the match (*i*,*j*) in subnetwork *q*; ΔT_{ij}^q is the logarithmic mean temperature difference (LMTD) between hot stream *i* and cold stream *j* that can exchange heat in subnetwork *q*, which is defined as:

$$\Delta T_{ij}^q = \frac{\Delta T_{\text{out},ij}^q - \Delta T_{\text{in},ij}^q}{\ln \dfrac{\Delta T_{\text{out},ij}^q}{\Delta T_{\text{in},ij}^q}}, \qquad \text{if} \quad \Delta T_{\text{in},ij}^q \ne \Delta T_{\text{out},ij}^q,$$

$$\Delta T_{ij}^q = \Delta T_{\text{in},ij}^q, \qquad \text{if} \quad \Delta T_{\text{in},ij}^q = \Delta T_{\text{out},ij}^q, \tag{3}$$

where $\Delta T_{\text{in},ij}^q$ and $\Delta T_{\text{out},ij}^q$ are inlet and outlet temperature differences between hot stream *i* and cold stream *j* in subnetwork *q*. $\Delta T_{\text{in},ij}^q$ and $\Delta T_{\text{out},ij}^q$ are defined as:

$$\Delta T_{\text{in},ij}^q = T_{\text{in},i}^{H,q} - \min\left(T_{\text{out},j}^{C,q},\ T_{\text{in},i}^{H,q} - \text{HRAT}\right),$$

$$\Delta T_{\text{out},ij}^q = \max\left(T_{\text{out},i}^{H,q},\ T_{\text{in},j}^{C,q} + \text{HRAT}\right) - T_{\text{in},j}^{C,q}, \tag{4}$$

where $T_{\text{in},i}^{H,q}$ and $T_{\text{out},i}^{H,q}$ are inlet and outlet temperatures of hot stream *i* in subnetwork *q*; $T_{\text{in},j}^{C,q}$ and $T_{\text{out},j}^{C,q}$ are inlet and outlet temperatures of cold stream *j* in subnetwork *q*.

For simplicity we assume all matches have the same overall heat transfer coefficient in Eq (2), but if the heat transfer coefficients are available, the weight factors in Model (M2) can be trivially modified as:

$$w_{ij}^{q} = \frac{Q_{ij}^{U,q}}{U_{ij}\Delta T_{ij}^{q}} \tag{5}$$

where U_{ij} is the heat transfer coefficient for the match (*i*,*j*). Note that the weight factor of a match is proportional to its heat transfer area. This means that a stream match with a smaller heat transfer area is associated with a smaller weight factor, and hence it is favored in the optimal solution. Therefore, this weighted model (M2) may not only reduce the solution time, but may also obtain solutions with potentially smaller total heat transfer areas compared to the original transshipment model (M1).

The optimal results and solution times for the weighted model are compared with those of the original model in Table 4. Solution times for all cases are significantly reduced. However, the weighted model is still very difficult to solve for large-scale problems (e.g., 15H, 15C with balanced streams and 20H, 20C with unbalanced streams). The optimal objective values of the weighted model are not listed in the table. Instead, the values of the sum of all binary variables are presented in order to provide the original physical meaning of the model. The results show that more units are introduced by the weighted model since the fixed cost may be different for different matches. The weighted model may lead to networks with more matches but with smaller total heat transfer area and lower total capital costs (as discussed before). This topic, however, is outside of the scope of this paper.

Table 4. Optimization results and solution times for the weighted model

Case	Original Model (M1) [a]		Weighted Model (M2) [b]	
	$\sum y_{ij}$ [c]	CPU Time (s)	$\sum y_{ij}$ [c]	CPU Time (s)
Balanced Streams				
5H, 5C	24	0.5	25	0.3
8H, 8C	35	35.9	38	20.3
10H, 10C	42	1,017.9	47	706.5
12H, 12C	48	68,688.6	53	5,938.1
15H, 15C	57	> 100,000	65	> 100,000
Unbalanced Streams				
5H, 5C	26	0.3	26	0.2
10H, 10C	39	25.7	44	6.1
15H, 15C	55	660.1	60	281.2
17H, 17C	67	> 100,000	76	1,399.3
20H, 20C	77	> 100,000	84	6,8839.9

[a] Absolute gap 0.99 is applied.
[b] Relative gap 1% is applied.
[c] Sum of matches in all subnetworks.

3. Model Reformulations

In order to reduce the LP relaxation gap, some tighter formulations for Model (M1) are studied here. Two reformulations are discussed in this section: using disaggregated models and adding additional integer cuts. Part of these approaches have been investigated in Anantharaman et al. (2010); however, only LP relaxations were reported in that article, and its largest case study included only 22 process streams. In this study, we present both LP relaxations and solution times with case studies of up to 40 process streams.

3.1 Disaggregated Models

The large upper bound in constraint (M1-3) is a major reason for the loose LP relaxation in Model (M1). Decreasing the upper bound is an effective way to tighten the LP relaxation and improve the solution time. Constraint (M1-3) in Model (M1) is first disaggregated at each temperature interval *k* so that the heat exchanged at the interval should be equal to or less than a new smaller upper bound multiplying the binary variable. The new disaggregated MILP transshipment model is shown below:

$$\min \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q} \tag{M3}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^{H} \qquad \forall i \in H_k' \qquad \forall k \in K_q \tag{M3-1}$$

$$\sum_{i\in H_k} Q_{ijk} = Q_{jk}^{C} \qquad \forall j \in C_k \qquad \forall k \in K_q \tag{M3-2}$$

$$Q_{ijk} - Q_{ijk}^{U,q} y_{ij}^{q} \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q \tag{M3-3}$$

$$R_{ik},\ Q_{ijk} \ge 0 \qquad y_{ij}^{q} \in \{0,1\}$$

The upper bound $Q_{ijk}^{U,q}$ is defined as: $Q_{ijk}^{U,q} = \min\left\{\sum_{\substack{l\in K_q \\ l\le k}} Q_{il}^{H},\ Q_{jk}^{C}\right\}$, which is much smaller than $Q_{ij}^{U,q}$ in Model (M1).

Using a similar reasoning, the MILP transportation model (Cerda and Westerberg, 1983b) can be disaggregated, so that a new upper bound is defined for the heat transfer for each hot stream *i* in temperature interval *k* to each cold stream *j* in temperature interval *l*.

$$\min \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q} \tag{M4}$$

$$\text{s.t. } \sum_{\substack{l\in K_q \\ l\ge k}}\sum_{j\in C_k} q_{ik,jl} = Q_{ik}^{H} \qquad \forall i \in H_k \qquad \forall k \in K_q \tag{M4-1}$$

$$\sum_{\substack{l\in K_q \\ l\le k}}\sum_{i\in H_l} q_{il,jk} = Q_{jk}^{C} \qquad \forall j \in C_k \qquad \forall k \in K_q \tag{M4-2}$$

$$q_{ik,jl} - Q_{ik,jl}^{U} y_{ij}^{q} \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q \qquad \forall l \in K_q \tag{M4-3}$$

$$q_{ik,jl} \ge 0 \qquad y_{ij}^{q} \in \{0,1\}$$

where $q_{ik,jl}$ is the heat exchanged between hot stream *i* at interval *k* and cold stream *j* at interval *l*. The upper bound $Q^U_{ik,jl}$ is thus defined as: $Q^U_{ik,jl} = \min\left\{Q^H_{ik},\ Q^C_{jl}\right\}$, which is expected to be the smallest upper bound among Model (M1), (M3) and (M4).

The LP relaxations of Models (M1), (M3) and (M4) are compared in Table 5. The tightness of the LP relaxations of these models is ranked as follows: transportation model > disaggregated transshipment model > original transshipment model. Disaggregated models generally have much tighter LP relaxations than the original model; however, the improvement of the LP relaxation is not very significant considering the large gap between the true solution and the relaxation.

Table 5. LP relaxation for disaggregated models

Case	**Optimal Objective Value**	**LP Relaxation Value**		
		Original Transshipment Model (M1)	Disaggregated Transshipment Model (M3)	Transportation Model (M4)
Balanced Streams				
5H, 5C	24	16.30	16.72	16.80
8H, 8C	35	24.36	24.76	24.79
10H, 10C	42	28.85	30.08	30.10
12H, 12C	48	32.14	33.40	33.63
15H, 15C	57	40.39	42.39	42.58
Unbalanced Streams				
5H, 5C	26	17.93	20.07	20.65
10H, 10C	39	29.97	31.90	32.61
15H, 15C	55	41.42	43.48	44.64
17H, 17C	67	48.84	52.64	53.55
20H, 20C	77	56.59	61.85	63.23

Tighter reformulations do not always lead to faster computation if they are larger in size. The number of constraints in (M3) and (M4) is greatly increased. Assuming the total number of hot streams and cold streams in subnetwork *q* are N^q_H and N^q_C, respectively, and the total number of temperature intervals in subnetwork *q* is N^q_K, then the maximum number of constraints, including binary variables in Model (M1),

(M3) and (M4), are equal to $N_H^q N_C^q$, $N_K^q N_H^q N_C^q$ and $\left(N_K^q\right)^2 N_H^q N_C^q$, respectively. Thus, there is a trade-off between the tightness of the LP relaxation and model size. Tighter models would be helpful to reduce the number of nodes in the branch and bound tree, but the larger problem size would also increase the solution time at each node, possibly deteriorating the overall performance. The solution times for the various models are listed in Table 6. The transportation model shows the worst performance among the three models because of its large size, even though it has a slightly tighter LP relaxation. The disaggregated transshipment model, on the other hand, realizes a more optimal trade-off between the tightness and problem size in all cases except one, and it achieves the best overall performance. Hence, the disaggregated transshipment model (M3) is used as the base model in all the following computational tests.

Table 6. Solution times for disaggregated models

Case	**CPU Time (s)** [a]		
	Original Transshipment Model (M1)	Disaggregated Transshipment Model (M3)	Transportation Model (M4)
Balanced Streams			
5H, 5C	0.5	0.5	0.4
8H, 8C	35.9	34.9	91.1
10H, 10C	1,017.9	1,011.4	3,075.1
12H, 12C	68,688.6	36,356.6	> 100,000
15H, 15C	> 100,000	> 100,000	> 100,000
Unbalanced Streams			
5H, 5C	0.3	0.2	0.4
10H, 10C	25.7	21.1	150.1
15H, 15C	660.1	1,043.1	> 100,000
17H, 17C	> 100,000	76,676.3	> 100,000
20H, 20C	> 100,000	> 100,000	> 100,000

[a] Absolute gap 0.99 is applied.

3.2 Additional Integer Cuts

Model (M3) can be further tightened by introducing additional integer cuts. One type of enhanced integer cut is enforcing the total number of matches for each hot or cold stream to be at least one.

$$\sum_{j \in C_q} y_{ij}^q \geq 1 \qquad \forall i \in H_q,$$

$$\sum_{i \in H_q} y_{ij}^q \geq 1 \qquad \forall j \in C_q. \tag{6}$$

When the total heat content of a hot (or cold) stream is larger than that of all cold (or hot) streams, multiple matches can be enforced for that hot (or cold) stream. The tighter integer cuts are added as:

$$\sum_{j \in C_q} y_{ij}^q \geq \left\lceil \frac{\sum_{k \in K_q} Q_{ik}^H}{\max_{j \in C_q} \sum_{k \in K_q} Q_{jk}^C} \right\rceil \qquad \forall i \in H_q,$$

$$\sum_{i \in H_q} y_{ij}^q \geq \left\lceil \frac{\sum_{k \in K_q} Q_{jk}^C}{\max_{i \in H_q} \sum_{k \in K_q} Q_{ik}^H} \right\rceil \qquad \forall j \in C_q. \tag{7}$$

In Eq (7), the minimum number of matches for each hot (or cold) stream is enforced to be the smallest integer value that is larger than or equal to the ratio of its heat content to the maximum heat content of all cold (or hot) streams. The other type of integer cut is limiting the total number of stream matches to be smaller than or equal to the total number of hot and cold streams minus one, for each subnetwork *q*, as indicated by Hohmann (1971):

$$\sum_{i \in H_q} \sum_{j \in C_q} y_{ij}^q \leq N_H^q + N_C^q - 1 \tag{8}$$

We should note, however, that this constraint is not completely rigorous as there are exceptions to the rule for minimum number of units (Wood et al., 1985).

By adding Eqs (7) and (8) to Model (M3), we obtain a disaggregated transshipment model with additional integer cuts as follows:

$$\min \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q} \qquad \text{(M5)}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^{H} \qquad \forall i \in H_k' \qquad \forall k \in K_q$$

$$\sum_{i\in H_k} Q_{ijk} = Q_{jk}^{C} \qquad \forall j \in C_k \qquad \forall k \in K_q$$

$$Q_{ijk} - Q_{ijk}^{U,q} y_{ij}^{q} \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q$$

$$\sum_{j\in C_q} y_{ij}^{q} \ge \left\lceil \sum_{k\in K_q} Q_{ik}^{H} \Big/ \max_{j\in C_q} \sum_{k\in K_q} Q_{jk}^{C} \right\rceil \qquad \forall i \in H_q$$

$$\sum_{i\in H_q} y_{ij}^{q} \ge \left\lceil \sum_{k\in K_q} Q_{jk}^{C} \Big/ \max_{i\in H_q} \sum_{k\in K_q} Q_{ik}^{H} \right\rceil \qquad \forall j \in C_q$$

$$\sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q} \le N_H^{q} + N_C^{q} - 1$$

$$R_{ik},\ Q_{ijk} \ge 0 \qquad y_{ij}^{q} \in \{0,1\}$$

Table 7. LP relaxation for models with and without additional integer cuts

Case	Optimal Objective Value	LP Relaxation Value	
		Without Integer Cuts (M3)	With Integer Cuts (M5)
Balanced Streams			
5H, 5C	24	16.72	17.63
8H, 8C	35	24.76	24.81
10H, 10C	42	30.08	30.80
12H, 12C	48	33.40	34.08
15H, 15C	57	42.39	42.47
Unbalanced Streams			
5H, 5C	26	20.07	20.10
10H, 10C	39	31.90	31.90
15H, 15C	55	43.48	43.53
17H, 17C	67	52.64	52.79
20H, 20C	77	61.85	61.89

Table 8. Solution times for models with and without additional integer cuts

Case	CPU Time (s) [a]	
	Without Integer Cuts (M3)	With Integer Cuts (M5)
Balanced Streams		
5H, 5C	0.5	0.5
8H, 8C	34.9	33.2
10H, 10C	1,011.4	878.3
12H, 12C	36,356.6	33,869.2
15H, 15C	> 100,000	> 100,000
Unbalanced Streams		
5H, 5C	0.2	0.3
10H, 10C	21.1	30.7
15H, 15C	1,043.1	749.8
17H, 17C	76,676.3	28,682.4
20H, 20C	> 100,000	> 100,000

[a] Absolute gap 0.99 is applied.

LP relaxations and solution times for Models (M3) and (M5) are compared in Table 7 and Table 8, respectively. After introducing additional integer cuts, the LP relaxations become somewhat tighter, especially for the balanced problems. The solution times for most cases are also reduced by these additional integer cuts. The results show that the effect of integer cuts on solution time is more significant for unbalanced cases, although the LP relaxation was only slightly improved. Since the additional integer cuts are helpful to improve the model performance, Model (M5) is used as the base model in all the following studies.

4. Solution Strategies

Results of the previous sections indicate that we cannot significantly reduce the solution time by only model reformulations or by adding integer cuts. To further improve the computational performance, several additional strategies are tested to improve the MILP solution process, including specifying branching priorities for binary variables, selecting the node branching rule, using a feasibility pump heuristic, employing relaxation induced neighborhood search (RINS), and optimizing on a parallel

processor. These solution strategies are covered in this section, and all of them can be implemented through CPLEX options (CPLEX, 2013).

4.1 Branching Priority for Binary Variables

Stream matches with larger upper bounds on heat exchange usually have more a significant impact on the heat exchanger network. If the values of binary variables with larger upper bounds are determined early in the branch and bound procedure, the solution efficiency may be improved. Hence, we provide higher branching priority to the binary variables with larger upper bounds. The following specification is added to the GAMS model: $y_{ij}^{q}.\text{prior} = 1/Q_{ij}^{U,q}$.

Table 9. Solution times for models with and without branching priority for binary variables

Case	**CPU Time (s)** [a]	
	Without Branching Priority	With Branching Priority $y_{ij}^{q}.\text{prior} = 1/Q_{ij}^{U,q}$
Balanced Streams		
5H, 5C	0.5	0.5
8H, 8C	33.2	29.6
10H, 10C	878.3	607.0
12H, 12C	33,869.2	24,400.8
15H, 15C	> 100,000	> 100,000
Unbalanced Streams		
5H, 5C	0.3	0.3
10H, 10C	30.7	31.3
15H, 15C	749.8	1,527.2
17H, 17C	28,682.4	> 100,000
20H, 20C	> 100,000	> 100,000

[a] Disaggregated transshipment model (M5) is used. Absolute gap 0.99 is applied.

The solution times for the different problems with and without branching priorities are compared in Table 9. The proposed branching priority for binary variables seems only effective for balanced cases; however, it will also be effective for unbalanced cases when combined with other strategies, as will be discussed in following subsections.

4.2 Node Branching Rule

A node branching rule is a pre-defined priority for selecting the branching variable at the node that has been branched. It can be specified by the CPLEX option varsel (CPLEX, 2013). Four different node branching rules are studied here: branch on variable with maximum infeasibility (varsel 1), branch based on pseudo costs (varsel 2), strong branching (varsel 3), and branch based on pseudo reduced costs (varsel 4). Strong branching is particularly interesting because under this rule a number of subproblems with tentative branches are partially solved and the most promising branch is then selected. This rule is potentially effective on large, difficult problems, such as the problems in this study, but the rule itself can be computationally intensive. The base case uses the default CPLEX setting, in which the branch variable is automatically selected (varsel 0).

Table 10. Solution times with different node branching rules (without branching priority)

Case	**CPU Time (s)** [a]				
	Branch Variable Automatically Selected (Base Case) (varsel 0) [b]	Branch on Variable with Maximum Infeasibility (varsel 1) [b]	Branch Based on Pseudo Costs (varsel 2) [b]	Strong Branching (varsel 3) [b]	Branch Based on Pseudo Reduced Costs (varsel 4) [b]
Balanced Streams					
5H, 5C	0.5	0.5	0.3	0.5	0.4
8H, 8C	33.2	62.1	35.6	120.3	33.9
10H, 10C	878.3	1,848.0	955.5	2,106.1	754.7
12H, 12C	33,869.2	> 100,000	34,799.6	92,240.2	40,949.7
15H, 15C	> 100,000	> 100,000	> 100,000	> 100,000	> 100,000
Unbalanced Streams					
5H, 5C	0.3	0.2	0.2	0.3	0.2
10H, 10C	30.7	24.9	20.8	74.8	41.2
15H, 15C	749.8	14,114.6	846.9	4,932.8	1,274.5
17H, 17C	28,682.4	> 100,000	> 100,000	> 100,000	23,102.8
20H, 20C	> 100,000	> 100,000	> 100,000	> 100,000	> 100,000

[a] Disaggregated transshipment model (M5) is used. Absolute gap 0.99 is applied.
[b] CPLEX option is listed in the parentheses.

Table 11. Solution times with different node branching rules (with branching priority: y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$)

Case	CPU Time (s) [a]				
	Branch Variable Automatically Selected (Base Case) (varsel 0) [b]	Branch on Variable with Maximum Infeasibility (varsel 1) [b]	Branch Based on Pseudo Costs (varsel 2) [b]	Strong Branching (varsel 3) [b]	Branch Based on Pseudo Reduced Costs (varsel 4) [b]
Balanced Streams					
5H, 5C	0.5	0.5	0.4	0.4	0.5
8H, 8C	29.6	26.0	27.0	37.3	26.8
10H, 10C	607.0	481.0	515.3	628.1	563.1
12H, 12C	24,400.8	22,606.6	24,509.3	31,545.2	24,643.4
15H, 15C	> 100,000	> 100,000	> 100,000	> 100,000	> 100,000
Unbalanced Streams					
5H, 5C	0.3	0.2	0.2	0.2	0.2
10H, 10C	31.3	14.8	6.8	10.2	8.5
15H, 15C	1,527.2	693.7	692.2	471.2	742.7
17H, 17C	> 100,000	36,240.9	> 100,000	19,178.8	21,759.8
20H, 20C	> 100,000	> 100,000	> 100,000	> 100,000	> 100,000

[a] Disaggregated transshipment model (M5) is used. Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. Absolute gap 0.99 is applied.
[b] CPLEX option is listed in the parentheses.

Solution times for cases with different node branching rules are listed in Table 10 and Table 11. Table 10 shows results without any branching priority for binary variables, while Table 11 shows results with the binary branching priority proposed in Section 4.1. It is definitely not useful to specify any of the node branching rules if no binary branching priority is used. However, when binary branching priority y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$ is applied, most of the problems with a specific node branching rule perform better than the base case. Strong branching (varsel 3) and branch on pseudo reduced costs (varsel 4) plus branching priority y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$ achieve shorter solution times than the base case (without branching priority) for both balanced and unbalanced cases. Hence, these are two promising approaches for further studies.

4.3 Feasibility Pump

The feasibility pump heuristic (Fischetti et al., 2005) can be used to find a feasible integer solution more quickly, which may be helpful to solve the MILP transshipment model faster. Two types of feasibility pump heuristics are implemented in CPLEX: a feasibility pump with an emphasis on finding a feasible solution (fpheur 1), and one with an emphasis on finding a feasible solution with a good objective value (fpheur 2) (CPLEX, 2013). The latter will likely obtain a better solution, but may also fail to find a feasible solution. These two feasibility pump heuristics are studied for all problems. The base case is set to be automatically choosing whether or not to use feasibility pump (fpheur 0). The results indicated that the feasibility pump heuristics are not helpful for improving the solution efficiency compared to the base case and are, therefore, not reported in this paper.

4.4 Relaxation Induced Neighborhood Search (RINS)

In CPLEX, RINS is a heuristic that explores a neighborhood around the current incumbent to try to find a new, improved solution (CPLEX, 2013). It formulates the neighborhood exploration as an MILP subproblem, called a sub-MIP. In this sub-MIP, binary variables with the same values in the incumbent and its LP relaxation are fixed, and the remaining variables are then solved. The sub-MIP is not solved to global optimality; instead its solution is truncated by limiting the number of nodes explored in the search tree. RINS may greatly improve the solution quality and, hence, increase the computational speed for MILP problems. RINS is only invoked at every k^{th} node in the tree, where k is specified by the CPLEX option rinsheur (CPLEX, 2013).

Table 12. Solution times with different RINS rules with branching priority: y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$

Case	CPU Time (s) [a]			
	Automatic (Base Case) (rinsheur 0) [b]	RINS Invoked Every 2,000th Node (rinsheur 2000) [b]	RINS Invoked Every 3,000th Node (rinsheur 3000) [b]	RINS Invoked Every 4,000th Node (rinsheur 4000) [b]
Balanced Streams				
5H, 5C	0.5	0.4	0.4	0.4
8H, 8C	29.6	37.1	37.2	36.1
10H, 10C	607.0	466.9	423.5	458.0
12H, 12C	24,400.8	30,582.3	26,817.0	27,894.0
15H, 15C	> 100,000	> 100,000	> 100,000	> 100,000
Unbalanced Streams				
5H, 5C	0.3	0.3	0.3	0.2
10H, 10C	31.3	6.7	6.6	6.8
15H, 15C	1,527.2	553.9	511.1	657.9
17H, 17C	> 100,000	17,827.4	18,605.7	25,758.9
20H, 20C	> 100,000	> 100,000	> 100,000	> 100,000

[a] Disaggregated transshipment model (M5) is used. Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. Absolute gap 0.99 is applied.
[b] CPLEX option is listed in the parentheses.

The solution times with different RINS frequencies are compared in Table 12. In the base case, RINS is implemented in the automatic mode (rinsheur 0). RINS achieves faster computation for most of cases, especially for unbalanced cases. The RINS invoked at every 3000th node together with binary branching priority y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$ shows the best overall performance for all cases.

Remark

Multiple solution strategies can be implemented together. Table 13 lists solution times with node branching rules combined with RINS, which were previously shown to be effective strategies. In most cases, the performance of the combined strategies is worse than the best individual strategy. This means that solution efficiency cannot be easily improved by just combining several strategies together.

Table 13. Solution times with combined strategies (with branching priority: y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$)

Case	CPU Time (s) [a]			
	Automatic (Base Case) (varsel 0 + rinsheur 0) [b]	Node Branching Rule + RINS		
		(varsel 1 + rinsheur 3000) [b]	(varsel 3 + rinsheur 3000) [b]	(varsel 4 + rinsheur 3000) [b]
Balanced Streams				
5H, 5C	0.5	0.4	0.3	0.4
8H, 8C	29.6	25.7	37.4	40.1
10H, 10C	607.0	500.1	617.6	494.9
12H, 12C	24,400.8	25,701.9	36,408.1	29,197.2
15H, 15C	> 100,000	> 100,000	> 100,000	> 100,000
Unbalanced Streams				
5H, 5C	0.3	0.3	0.2	0.3
10H, 10C	31.3	16.6	12.0	6.6
15H, 15C	1,527.2	1,923.1	475.5	371.1
17H, 17C	> 100,000	30,593.7	22,371.8	20,109.7
20H, 20C	> 100,000	> 100,000	> 100,000	> 100,000

[a] Disaggregated transshipment model (M5) is used. Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. Absolute gap 0.99 is applied.
[b] CPLEX option is listed in the parentheses.

Summary

The branching priority for binary variables (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) plus the RINS invoked at every 3000th achieves the best overall performance for most of cases; therefore, it is implemented with Model (M5) as the basis in the following studies.

4.5 Parallel Computing

Parallel computing technology can be applied to reduce time to solution if a multi-core CPU is available. CPLEX is capable of implementing parallel computing by using the threads option, where threads m means m cores are used for solution. Two different parallel modes are employed in CPLEX: deterministic and opportunistic. These can be realized by the CPLEX option: parallelmode (CPLEX, 2013). The deterministic mode (parallelmode 1) uses the same solution path for all runs and repeats the same results, while the opportunistic mode (parallelmode -1) may produce different solution paths and, consequently,

different optimal solutions and solution times. The opportunistic mode usually outperforms the deterministic mode because less synchronization is required between threads. In this study, six cores are used for parallel computing, with a speed of 3.33 GHz for each core of an Intel Xeon X5680 processor. All other hardware and software settings are the same as previous case studies. Parallel solution results from both the deterministic and opportunistic modes are listed in Table 14. The solution times with six cores are much smaller, but still larger than 1/6 of those with the single core due to overhead associated with synchronization between cores. The fast solution of large-scale cases still seems impossible even with parallel computing. Nevertheless, the reductions in CPU times are quite significant in most cases. The results also demonstrate that the opportunistic mode provides better performance than the deterministic mode for most cases. Parallel computing is clearly a good option, but will only be used in Sections 5.1 and 5.3. Note that for small-scale cases (e.g., 5H, 5C and unbalanced 10H, 10C), more time is required because additional time is spent in synchronization between the cores.

Table 14. Solution times with parallel computing

Case	**Single Core** [a]	**Six Cores - Deterministic** [a]		**Six Cores - Opportunistic** [a]	
	CPU Time (s)	CPU Time (s)	Speedup	CPU Time (s)	Speedup
Balanced Streams					
5H, 5C	0.4	0.5	0.79	0.4	0.82
8H, 8C	37.2	10.9	3.40	8.1	4.60
10H, 10C	423.5	160.0	2.65	115.1	3.68
12H, 12C	26,817.0	5,180.9	5.18	10,844.8	2.47
15H, 15C	> 100,000	> 100,000		> 100,000	
Unbalanced Streams					
5H, 5C	0.3	0.2	1.50	0.2	1.38
10H, 10C	6.6	8.6	0.77	2.7	2.42
15H, 15C	511.1	184.4	2.77	124.8	4.10
17H, 17C	18,605.7	6,951.6	2.68	6,184.5	3.01
20H, 20C	> 100,000	> 100,000		> 100,000	

[a] Disaggregated transshipment model (M5) is used. Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node. Absolute gap 0.99 is applied.

5. Approximation Approaches

Although some model reformulation techniques and solution strategies developed in the previous sections can improve the solution performance for the MILP transshipment model, solving large-scale problems in relatively short times is still a very difficult task. Instead of obtaining the exact global optimal solution, several approximation approaches are proposed in this section in order to quickly find good approximate solutions of Model (M5).

5.1 Relative Optimality Gap

A simple way to obtain an approximation is to terminate the MILP search by selecting an appropriate relative optimality gap. Case study results for Model (M5) with a 10% relative gap are listed in Table 15. By applying a 10% relative gap, the optimal objective values are identical or very close to their global optimal values, which are obtained with an absolute gap of 0.99. Note that the solution time can be reduced by one order of magnitude for some large-scale cases. Therefore, choosing a 10% relative gap for large-scale MILP transshipment models is an effective option.

Table 15. Optimization results and solution times with relative optimality gap

Case	**Absolute Gap 0.99** [a,b]		**Relative Gap 10%** [a]	
	Optimal Objective	CPU Time (s)	Optimal Objective	CPU Time (s)
Balanced Streams				
5H, 5C	24	0.4	24	0.4
8H, 8C	35	37.2	35	25.3
10H, 10C	42	423.5	42	143.3
12H, 12C	48	26,817.0	48	4,654.1
15H, 15C	57	> 100,000	58	> 100,000
Unbalanced Streams				
5H, 5C	26	0.3	26	0.2
10H, 10C	39	6.6	40	5.1
15H, 15C	55	511.1	56	134.1
17H, 17C	67	18,605.7	68	3,406.7
20H, 20C	77	> 100,000	79	20,422.2

[a] Disaggregated transshipment model (M5) is used. Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node.
[b] Global optimal solutions or best solutions obtained so far.

The solution time can be further reduced by using the 10% relative gap together with a parallel computing approach, as shown in Table 16; however, it is still impossible to solve some large-scale cases (e.g., 15H, 15C with balanced streams) within a reasonable time.

Table 16. Optimization results and solution times with 10% relative optimality gap and parallel computing

Case	Optimal Objective [a]	CPU Time (s) [a]	
		Single Core	Six Cores [b]
Balanced Streams			
5H, 5C	24	0.4	0.5
8H, 8C	35	25.3	7.2
10H, 10C	42	143.3	52.7
12H, 12C	48	4,654.1	1,035.9
15H, 15C	58	> 100,000	> 100,000
Unbalanced Streams			
5H, 5C	26	0.2	0.2
10H, 10C	40	5.1	5.4
15H, 15C	56	134.1	95.5
17H, 17C	68	3,406.7	2,119.9
20H, 20C	79	20,422.2	8,813.1

[a] Disaggregated transshipment model (M5) is used. Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node. Relative gap 10% is applied.
[b] Six CPU cores are used. Deterministic parallel mode is applied.

5.2 Combined Model

Another approximation scheme is to add the utility cost terms to the objective function of Model (M5) in order to optimize both the utility cost and weighted contribution of number of units. This scheme tends to reduce the degeneracy caused by unity coefficients of all the binary variables. Assuming that the identity of the subnetworks remains unchanged, the combined model is as follows:

$$\min \sum_{m\in S_q} c_m Q_m^{S,q} + \sum_{n\in W_q} c_n Q_n^{W,q} + \alpha_w \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q} \qquad \text{(M6)}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^{H} \qquad \forall i \in H_k' \qquad \forall k \in K_q$$

$$\sum_{i \in H_k} Q_{ijk} = Q_{jk}^{C} \qquad \forall j \in C_k \qquad \forall k \in K_q$$

$$Q_{ijk} - Q_{ijk}^{U,q} y_{ij}^{q} \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q$$

$$\sum_{j \in C_q} y_{ij}^{q} \ge \left\lceil \sum_{k \in K_q} Q_{ik}^{H} \Big/ \max_{j \in C_q} \sum_{k \in K_q} Q_{jk}^{C} \right\rceil \qquad \forall i \in H_q$$

$$\sum_{i \in H_q} y_{ij}^{q} \ge \left\lceil \sum_{k \in K_q} Q_{jk}^{C} \Big/ \max_{i \in H_q} \sum_{k \in K_q} Q_{ik}^{H} \right\rceil \qquad \forall j \in C_q$$

$$\sum_{i \in H_q} \sum_{j \in C_q} y_{ij}^{q} \le N_H^{q} + N_C^{q} - 1$$

$$R_{ik},\ Q_{ijk} \ge 0 \qquad y_{ij}^{q} \in \{0,1\}$$

where S_q and W_q are index sets for all hot utilities and cold utilities present in subnetwork *q*; $Q_m^{S,q}$ and $Q_n^{W,q}$ are the heat load of hot utility *m* and cold utility *n* in subnetwork *q*; α_w is the weight factor, for the minimum number of units term in the objective. The value of α_w can be tuned to achieve both a good approximate solution and a short solution time.

The results for the combined model (M6) under different α_w are compared in Table 17. For better comparison, only the sum of the binary variables is reported. The results demonstrate a trade-off between solution quality and solution time. By using a small value for α_w (α_w = 10), Model (M6) is the least similar to the base model (M5) and more similar to the LP transshipment model (M0). Thus, the CPU times are quite short but the solutions differ by up to 10 units in the largest instance. With a larger value for α_w (α_w = 50) Model (M6) becomes more similar to (M5), and the solution quality is improved (showing a discrepancy of up to 6 units in the largest case), but the CPU times greatly increase. It is difficult to determine a suitable value of α_w for all cases. Generally, the combined model achieves good solution quality for balanced cases and short solution times for unbalanced cases.

Table 17. Optimization results and solution times for combined models

Case	Base Model (M5) [a,b]		Combined Model (M6) [a,c]					
			$\alpha_w = 10$		$\alpha_w = 25$		$\alpha_w = 50$	
	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)
Balanced Streams								
5H, 5C	24	0.4	25	0.2	25	0.4	25	0.5
8H, 8C	35	37.2	36	3.0	35	41.0	35	79.8
10H, 10C	42	423.5	45	5.0	44	96.0	43	1,748.1
12H, 12C	48	26,817.0	52	3.3	50	174.9	48	18,836.6
15H, 15C	57	> 100,000	64	5.7	63	29,390.4	61	> 100,000
Unbalanced Streams								
5H, 5C	26	0.3	26	0.1	26	0.1	26	0.1
10H, 10C	39	6.6	45	0.6	45	0.8	43	0.9
15H, 15C	55	511.1	61	3.2	61	3.7	59	24.5
17H, 17C	67	18,605.7	73	39.8	75	6.8	71	330.2
20H, 20C	77	> 100,000	87	46.2	88	29.0	83	190.2

[a] Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node.
[b] Absolute gap 0.99 is applied.
[c] Relative gap 1% is applied.

5.3 Reduced MILP Model

Another solution approach is to fix part of the binary variables in Model (M5) and solve a reduced MILP model. The solution of the LP relaxation of Model (M5) can be used to fix the binary variables. The basic idea is that binary variables with the value of zero in the LP relaxation will tend to have the value of zero in the MILP solution. We define a set for these "zero" binary variables and fix them to zero in the full MILP model. However, this assumption may not hold true for all cases. Therefore, we develop a test to exclude some “zero” binary variables that could possibly be one in the final solution, fixing only those binary variables which have the highest probability to be zero in (M5) and then solving the reduced model.

The procedure to derive the reduced MILP model is as follows:

Initial: Define the set of binary variables with the value of zero as Y_0. *Set* $Y_0 = \emptyset$.

Step 1: Solve the LP relaxation problem, which can be the LP relaxation of the original transshipment model (M1), the disaggregated transshipment model (M3), or the transportation model (M4). Let $y_{ij}^q \in Y_0$ if $y_{ij}^q = 0$ in the solution of LP relaxation. Record the reduced cost of y_{ij}^q as rc_{ij}^q.

Step 2: Solve the following reduced MILP model:

$$\min \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^q \qquad \text{(M5-R)}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^H \qquad \forall i \in H_k' \qquad \forall k \in K_q$$

$$\sum_{i\in H_k} Q_{ijk} = Q_{jk}^C \qquad \forall j \in C_k \qquad \forall k \in K_q$$

$$Q_{ijk} - Q_{ijk}^{U,q} y_{ij}^q \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q$$

$$\sum_{j\in C_q} y_{ij}^q \ge \left\lceil \sum_{k\in K_q} Q_{ik}^H \Big/ \max_{j\in C_q} \sum_{k\in K_q} Q_{jk}^C \right\rceil \qquad \forall i \in H_q$$

$$\sum_{i\in H_q} y_{ij}^q \ge \left\lceil \sum_{k\in K_q} Q_{jk}^C \Big/ \max_{i\in H_q} \sum_{k\in K_q} Q_{ik}^H \right\rceil \qquad \forall j \in C_q$$

$$\sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^q \le N_H^q + N_C^q - 1$$

$$R_{ik},\ Q_{ijk} \ge 0$$

$$y_{ij}^q = 0 \qquad \forall y_{ij}^q \in Y_0$$

$$y_{ij}^q \in \{0,1\} \qquad \forall y_{ij}^q \notin Y_0$$

If Model (M5-R) is feasible, record the optimal values of the binary variables y_{ij}^q as $\hat{y}_{ij}^q$, go to Step 3. If Model (M5-R) is infeasible, for every (*i'*, *j'*) such that $y_{i'j'}^q \in Y_0$, check the value of reduced cost $rc_{i'j'}^q$; keep $y_{i'j'}^q$ in Y_0 if $rc_{i'j'}^q = 0$, and remove $y_{i'j'}^q$ from Y_0 if $rc_{i'j'}^q \ne 0$; go to Step 4.

Step 3: The goal is to test whether any $y_{ij}^q \in Y_0$ from Step 2 should be set to one. For every (*i', j'*) such that $y_{i'j'}^q \in Y_0$, solve the following LP test problem for $y_{i'j'}^q = 1$:

$$\min \sum_{i \in H_q} \sum_{j \in C_q} y_{ij}^q \qquad \text{(M5-T)}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j \in C_k} Q_{ijk} = Q_{ik}^H \qquad \forall i \in H_k' \qquad \forall k \in K_q$$

$$\sum_{i \in H_k} Q_{ijk} = Q_{jk}^C \qquad \forall j \in C_k \qquad \forall k \in K_q$$

$$Q_{ijk} - Q_{ijk}^{U,q} y_{ij}^q \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q$$

$$R_{ik},\ Q_{ijk} \ge 0$$

$$y_{i'j'}^q = 1$$

$$y_{ij}^q = 0 \qquad \forall y_{ij}^q \in Y_0 \setminus y_{i'j'}^q$$

$$y_{ij}^q = \hat{y}_{ij}^q \qquad \forall y_{ij}^q \notin Y_0$$

Check the value of heat exchange for (*i', j'*): $Q_{i'j'}^q = \sum_{k \in K_q} Q_{i'j'k}$. If $Q_{i'j'}^q = 0$, keep $y_{i'j'}^q$ in Y_0; if $Q_{i'j'}^q > 0$, remove $y_{i'j'}^q$ from Y_0.

Step 4: For Set Y_0 determined in Step 2 or 3, solve Model (M5-R), and obtain the final solution, which is the approximate solution of Model (M5).

Results for reduced MILP models derived from the LP relaxations of different formulations are listed in Table 18. The reduced model achieves good approximate solutions with significantly reduced solution times for most of cases. In some cases, the reduced model successfully solved the problem to the exact solution in less than one tenth of the original time. The LP relaxation formulation has significant influence on the performance of the reduced model. By using the LP relaxation of the original

transshipment model, which may be loose, Model (M5-R) could be infeasible in Step 2 of the above procedure; then set Y_0 is refined by the reduced cost of y_{ij}^q in the LP relaxation instead of solving a series of test problems and, hence, more binary variables tend to be fixed to zero in the final reduced MILP model, which leads to fast solution times but poor solution quality. The solution quality can be improved by using tighter LP relaxations, such as relaxation of disaggregated models. The reduced MILP, however, is still not able to solve the largest problems, but it provides good approximations for medium- to large-scale problems.

Table 18. Optimization results and solution times for reduced MILP models

Case	Full (Base) Model (M5) [a]		Reduced Model (M5-R) [a]					
			LP Relaxation of Original Transshipment Model [b]		LP Relaxation of Disaggregated Transshipment Model [b]		LP Relaxation of Transportation Model [b]	
	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)
Balanced Streams								
5H, 5C	24	0.4	25	2.0	24	3.6	24	2.3
8H, 8C	35	37.2	36	6.2	35	6.9	36	5.9
10H, 10C	42	423.5	44	7.6	43	26.1	43	62.0
12H, 12C	48	26,817.0	52	62.7	48	319.8	48	1,222.8
15H, 15C	57	> 100,000	58	> 100,000	59	> 100,000	59	> 100,000
Unbalanced Streams								
5H, 5C	26	0.3	26	1.7	26	2.0	26	1.8
10H, 10C	39	6.6	42	3.7	41	4.9	41	10.9
15H, 15C	55	511.1	61	12.5	56	45.0	55	310.8
17H, 17C	67	18,605.7	72	61.9	70	53,486.5	70	6,631.7
20H, 20C	77	> 100,000	86	159.0	79	> 100,000	82	> 100,000

[a] Branching priority (y_{ij}^q.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node. Absolute gap 0.99 is applied.
[b] LP relaxation problem that determines which integer variables to be fixed at zero.

Since both the reduced MILP model and the parallel computing option are the most promising, we can combine them for the solution of large-scale problems. Table 19 shows the solution times for reduced MILP models with a multi-core CPU. The reduced model derived from the LP relaxation of the original transshipment model is not studied here because it obtains relatively poor solutions as previously shown in Table 18. The reduced MILP model plus parallel computing achieves the best overall performance

among all options studied so far in this paper, and it is the only option that obtains a good approximate solution for the case of balanced streams, 15H, 15C, within reasonable time. The reduced model derived from the LP relaxation of disaggregated transshipment model outperforms that of transportation model in most cases.

Table 19. Optimization results and solution times for reduced MILP models with parallel computing

Case	Full (Base) Model (M5) [a,b]		Reduced Model (M5-R) [a,b]			
			LP Relaxation of Disaggregated Transshipment Model		LP Relaxation of Transportation Model	
	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)	$\sum y_{ij}$	CPU Time (s)
Balanced Streams						
5H, 5C	24	0.5	24	3.8	24	3.5
8H, 8C	35	10.9	35	7.4	36	7.3
10H, 10C	42	160.0	43	12.6	43	25.7
12H, 12C	48	5,180.9	48	110.2	48	243.1
15H, 15C	57	> 100,000	59	13,657.5	59	> 100,000
Unbalanced Streams						
5H, 5C	26	0.2	26	2.3	26	2.2
10H, 10C	39	8.6	41	5.3	41	8.5
15H, 15C	55	184.4	56	38.2	55	133.7
17H, 17C	67	6,951.6	70	4,353.7	70	984.4
20H, 20C	77	> 100,000	79	> 100,000	82	> 100,000

[a] Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node. Absolute gap 0.99 is applied.
[b] Six CPU cores are used. Deterministic parallel mode is applied.

5.4 NLP Reformulation

The last approximation scheme is to reformulate the MILP model into a continuous NLP model to avoid combinatorial search and to take advantage of the fast speed of NLP solvers. The binary variables are first relaxed as continuous variables. To enforce the integrality of these binary variables in the spirit of complementarity problems (Biegler and Grossmann, 2004), we can either add the penalty term $\sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^{q}\left(1-y_{ij}^{q}\right)$ to the objective function, or add the inequalities $y_{ij}^{q}\left(1-y_{ij}^{q}\right)\le \varepsilon$ ($\forall i \in H_q,\ \forall j \in C_q$, where ε is a small positive number). The latter option usually causes numerical difficulties for finding

feasible solutions. Hence, it is not selected in this study. The NLP reformulation of Model (M5) or (M3) is presented below:

$$\min Z_{\text{nlp}} = \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^q + \beta_{\text{nlp}} \sum_{i\in H_q}\sum_{j\in C_q} y_{ij}^q\left(1-y_{ij}^q\right) \tag{M7}$$

$$\text{s.t. } R_{ik} - R_{i,k-1} + \sum_{j\in C_k} Q_{ijk} = Q_{ik}^H \qquad \forall i \in H_k' \qquad \forall k \in K_q$$

$$\sum_{i\in H_k} Q_{ijk} = Q_{jk}^C \qquad \forall j \in C_k \qquad \forall k \in K_q$$

$$Q_{ijk} - Q_{ijk}^{U,q} y_{ij}^q \le 0 \qquad \forall i \in H_q \qquad \forall j \in C_q \qquad \forall k \in K_q$$

$$R_{ik},\ Q_{ijk} \ge 0 \qquad y_{ij}^q \in [0,1]$$

where β_{nlp} is the penalty factor to enforce the integrality of all y_{ij}^q.

In this study, β_{nlp} is set to be 1000, which is large enough to ensure integrality of y_{ij}^q. Since NLP solvers are often trapped in local optimal solutions, a multi-start NLP solver is used to try to obtain a high-quality solution that is close to the global optimum. The following procedure is implemented to improve the solution quality:

Initial: Define the upper bound of the objective of Model (M7) as $Z_{\text{nlp}}^{\text{up}}$. Set $Z_{\text{nlp}}^{\text{up}} = +\infty$. Add the equation $Z_{\text{nlp}} \le Z_{\text{nlp}}^{\text{up}}$ to the constraints of Model (M7). The obtained new model is denoted (M7-R).

Step 1: Solve Model (M7-R) by using a multi-start NLP solver (e.g., OQNLP in this study). Record the optimal objective value as $Z_{\text{nlp}}^{\text{l}}$.

Step 2: Update $Z_{\text{nlp}}^{\text{up}} = Z_{\text{nlp}}^{\text{l}} - 1$.

Repeat Step 1 and 2 *Until* Model (M7-R) is infeasible.

The above procedure tries to force the NLP solver to find a better solution by gradually reducing the upper bound of the objective. The results for the NLP reformulation are shown in Table 20. Despite the relatively short solution times, the NLP reformulation fails to find good approximate solutions, especially for large-scale cases, overestimating the number of units by up to 18.

Table 20. Optimization results and solution times for NLP reformulation

Case	MILP (Base) Model (M5) [a]		NLP Model (M7) [b]	
	Optimal Value	CPU Time (s)	Optimal Value	CPU Time (s)
Balanced Streams				
5H, 5C	24	0.4	26	82.0
8H, 8C	35	37.2	39	233.6
10H, 10C	42	423.5	48	202.5
12H, 12C	48	26,817.0	53	458.5
15H, 15C	57	> 100,000	70	1,291.9
Unbalanced Streams				
5H, 5C	26	0.3	26	81.9
10H, 10C	39	6.6	46	864.8
15H, 15C	55	511.1	66	1,242.7
17H, 17C	67	18,605.7	79	1,674.0
20H, 20C	77	> 100,000	95	14,804.0

[a] Branching priority (y_{ij}^{q}.prior = $1/Q_{ij}^{U,q}$) is selected. RINS is invoked every 3,000th node. Absolute gap 0.99 is applied.
[b] β_{nlp} = 1000. OQNLP is used as the NLP solver.

6. Conclusions

In this paper, it is shown that the solution time of the MILP transshipment model increases exponentially with the problem size due to the combinatorial explosion in the selection of potential matches. Problems with unbalanced streams, which may be less symmetric, are easier to solve than those with balanced streams. By using weight factors in the objective function, the solution time is reduced but more units are usually introduced in the optimal solution.

Several different approaches have been developed for faster solution of the MILP transshipment model. Model reformulations, including model disaggregation and adding integer cuts, can both strengthen the

LP relaxation and reduce the solution time. The disaggregated transshipment model with additional integer cuts was found to be the best formulation in this study.

Several additional solution strategies have been investigated. The branching priority, which first branches the binary variable with the largest upper bound, together with RINS is a promising approach for faster solutions. Branching priority with strong branching is another good option. When a multi-core CPU is available, parallel computing can significantly reduce time to solution. In fact among all options to obtain a rigorous solution this was found to be the most effective.

Instead of obtaining the exact global optimal solution, several approximation approaches have been studied for finding a good approximate solution in short time. A 10% relative optimality gap is a good approach, which solves some large-scale cases in reasonable times. The combined model with utility costs greatly reduces the solution time, although it is difficult to determine a proper weight factor for obtaining high-quality solutions. The reduced MILP model is another suggested approach, which is very effective, reducing the solution time by one to two orders of magnitudes, while still finding good approximate or even exact solutions. The reduced MILP model combined with parallel computing achieves the best overall performance among all the options presented in this paper with relatively modest overestimation of the minimum number of units (see Table 19). The NLP reformulation was fast but produced poor solutions for large-scale problems.

In summary, by applying the proposed approaches in this paper, the MILP transshipment model can be solved for relatively large-scale problems, that is, 12H, 12C with balanced streams and 17H, 17C with unbalanced streams, in reasonable times. However, it is still quite difficult to solve problems above 15H, 15C with balanced streams, and 20H, 20C with unbalanced streams, even with approximation schemes.

Acknowledgements

The authors acknowledge financial support through U.S. Department of Energy, Office of Fossil Energy, National Energy Technology Laboratory (NETL) (Grant Number: 4000.2.673.062.001.641.000.004). This project was conducted as a part of the Carbon Capture Simulation Initiative (CCSI) program.

Appendix

A.1 Stream Information

Table A1. Stream information for balanced streams [a]

Hot Streams [b]				**Cold Streams** [b]			
Stream No.	FCp (MW/°C)	T_{in} (°C)	T_{out} (°C)	Stream No.	FCp (MW/°C)	T_{in} (°C)	T_{out} (°C)
1	1	400	120	1	1.5	160	400
2	2	340	120	2	1.3	100	250
3	1.5	380	150	3	2.5	50	300
4	2.5	300	100	4	2.8	200	380
5	1.7	420	160	5	1.9	150	450
6	0.8	390	110	6	0.8	100	180
7	1.2	360	200	7	1.7	200	350
8	1.8	280	130	8	1.6	120	330
9	1.1	250	80	9	0.9	110	220
10	1.3	330	170	10	2.1	190	360
11	2.1	430	300	11	1.8	260	420
12	2.2	200	100	12	1.2	80	180
13	1.2	150	70	13	1.6	130	390
14	1.6	330	180	14	1.4	180	260
15	1.9	370	115	15	2	155	365
16	1.4	355	105	16	1	95	480
17	0.9	310	130	17	1.1	175	385
18	1.3	260	90	18	1.5	130	290
19	1.1	300	115	19	2.2	210	430
20	2.3	265	190	20	1.7	230	370

[a] Hot utility: high-pressure steam (500°C), medium-pressure steam (350°C). Cold utility: cooling water (20-30°C).
[b] Each case study selects a subset of streams in this table. A case with mH and nC means that the first m hot streams and first n cold streams in this table are selected.

Table A2. Stream information for unbalanced streams [a]

Hot Streams [b]				Cold Streams [b]			
Stream No.	FCp (MW/°C)	T_{in} (°C)	T_{out} (°C)	Stream No.	FCp (MW/°C)	T_{in} (°C)	T_{out} (°C)
1	6	400	120	1	14	160	400
2	2	340	120	2	3	100	250
3	0.5	380	150	3	0.4	50	300
4	8	300	100	4	2.5	200	380
5	3	420	160	5	2	150	450
6	4	390	110	6	6	100	180
7	0.2	360	200	7	1.5	200	350
8	0.6	280	130	8	0.2	120	330
9	1.5	250	80	9	5.5	110	220
10	4	330	170	10	3	190	360
11	12	430	300	11	8	260	420
12	8	200	100	12	12	80	180
13	5	150	70	13	0.3	130	390
14	0.6	330	180	14	4.5	180	260
15	0.3	370	115	15	1	155	365
16	6	355	105	16	0.1	95	480
17	0.9	310	130	17	7	175	385
18	3	260	90	18	2	130	290
19	1	300	115	19	0.5	210	430
20	0.3	265	190	20	1.7	230	370

[a] Hot utility: high-pressure steam (500°C), medium-pressure steam (350°C). Cold utility: cooling water (20-30°C).
[b] Each case study selects a subset of streams in this table. A case with mH and nC means that the first m hot streams and first n cold streams in this table are selected.

A.2 Problem Sizes

Table A3. Problem sizes of MILP transshipment models for cases with balanced streams

Case	Number of Streams	Number of Binary Variables	Number of Continuous Variables	Number of Constraints
5H, 5C				
Subnetwork 1	3H, 3C	12	38	32
Subnetwork 2	5H, 5C	30	94	61
Subnetwork 3	5H, 4C	25	105	65
8H, 8C				
Subnetwork 1	5H, 4C	24	85	61
Subnetwork 2	8H, 7C	63	236	121
Subnetwork 3	8H, 6C	56	250	127
10H, 10C				
Subnetwork 1	5H, 5C	30	96	69
Subnetwork 2	10H, 9C	99	471	199
Subnetwork 3	10H, 8C	90	492	209
12H, 12C				
Subnetwork 1	6H, 6C	42	148	95
Subnetwork 2	11H, 10C	119	665	249
Subnetwork 3	11H, 9C	109	678	262
15H, 15C				
Subnetwork 1	7H, 8C	64	242	137
Subnetwork 2	13H, 13C	181	975	339
Subnetwork 3	14H, 12C	176	1477	453

Table A4. Problem sizes of MILP transshipment models for cases with unbalanced streams

Case	Number of Streams	Number of Binary Variables	Number of Continuous Variables	Number of Constraints
5H, 5C				
Subnetwork 1	3H, 3C	12	38	32
Subnetwork 2	5H, 5C	30	94	61
Subnetwork 3	5H, 4C	25	105	65
10H, 10C				
Subnetwork 1	5H, 5C	30	96	69
Subnetwork 2	7H, 7C	56	172	99
Subnetwork 3	10H, 10C	110	791	286
15H, 15C				
Subnetwork 1	7H, 8C	64	242	137
Subnetwork 2	14H, 15C	220	1476	454
Subnetwork 3	11H, 9C	108	976	309
17H, 17C				
Subnetwork 1	8H, 10C	90	407	195
Subnetwork 2	15H, 15C	239	1685	476
Subnetwork 3	15H, 13C	205	1966	541
20H, 20C				
Subnetwork 1	8H, 12C	108	482	226
Subnetwork 2	18H, 18C	339	3369	743
Subnetwork 3	18H, 14C	265	2435	647

Reference

Anantharaman R, Nastad I, Nygreen B, Gundersen T. The sequential framework for heat exchanger network synthesis—The minimum number of units sub-problem. Comput. Chem. Eng., 2010; 34(11):1822-1830.

Bagajewicz M, Rodera H, Savelski M. Energy efficient water utilization systems in process plants. Comput. Chem. Eng., 2002; 26(1):59-79.

Barbaro A, Bagajewicz MJ. New rigorous one-step MILP formulation for heat exchanger network synthesis. Comput. Chem. Eng., 2005; 29(9):1945-1976.

Biegler LT, Grossmann IE. Retrospective on optimization. Comput. Chem. Eng., 2004; 28(8):1169-1192.

Biegler LT, Grossmann IE, Westerberg AW. Systematic methods of chemical process design. New Jersey: Prentice Hall PTR; 1997.

Cerda J, Westerberg AW. Minimum utility usage in heat exchanger network synthesis: A transportation problem. Chem. Eng. Sci., 1983; 38(3):373-387.

Cerda J, Westerberg AW. Synthesizing heat exchanger networks having restricted stream/stream matches using transportation problem formulation. Chem. Eng. Sci., 1983; 38(10):1723-1740.

CPLEX. Cplex Solver Manual. 2013. http://www.gams.com/dd/docs/solvers/cplex.pdf.

Ciric AR, Floudas CA. Heat exchanger network synthesis without decomposition. Comput. Chem. Eng., 1991; 15(6):385-396.

Elia JA, Baliban RC, Floudas CA. Toward novel hybrid biomass, coal, and natural gas processes for satisfying current transportation fuel demands, 2: Simultaneous heat and power integration. Ind. Eng. Chem. Res., 2010; 49(16):7371-7388.

Fischetti M, Glover F, Lodi A. The feasibility pump. Math. Program., 2005; 104(1):91-104.

Floudas CA, Ciric AR, Grossmann IE. Automatic synthesis of optimum heat exchanger network configurations. AIChE J., 1986; 32(2):276-290.

Floudas CA, Grossmann IE. Synthesis of flexible heat exchanger networks for multiperiod operation. Comput. Chem. Eng., 1986; 10(2):153-168.
Furman KC, Sahinidis NV. Computational complexity of heat exchanger network synthesis. Comput. Chem. Eng., 2001; 25(9-10):1371-1390.
Furman KC, Sahinidis NV. A critical review and annotated bibliography for heat exchanger network synthesis in the 20th century. Ind. Eng. Chem. Res., 2002; 41(10):2335-2370.
Gundersen T, Duvold S, Hashemi-Ahmady A. An extended vertical MILP model for heat exchanger network synthesis. Comput. Chem. Eng., 1996; 20(S1):S97-S102.
Gundersen T, Grossmann IE. Improved optimization strategies for automated heat exchanger network synthesis through physical insights. Comput. Chem. Eng., 1990; 14(9):925-944.
Gundersen T, Naess L. The synthesis of cost optimal heat exchanger networks: An industrial review of the state of the art. Comput. Chem. Eng., 1988; 12(6):503-530.
Gundersen T, Traedal P, Hashemi-Ahmady A. Improved sequential strategy for the synthesis of near-optimal heat exchanger networks. Comput. Chem. Eng., 1997; 21(S):S59-S64.
GUROBI. Gurobi Solver Manual. 2013. http://www.gams.com/dd/docs/solvers/gurobi.pdf.
Hohmann EC. Optimum networks for heat exchange. Ph.D. Thesis, University of Southern California. 1971.
Klemeš JJ, Kravanja Z. Forty years of heat integration: pinch analysis (PA) and mathematical programming (MP). Current Opinion in Chemical Engineering, 2013: 2(4):461-474.
Linnhoff B, Hindmarsh E. The pinch design method of heat exchanger networks. Chem. Eng. Sci., 1983; 38(5):745-763.
Margot F. Exploiting orbits in symmetric ILP. Math. Program., 2003; 98(1-3):3-21.
McCarl BA, Meeraus A, Eijk P, Bussieck M, Dirkse S, Steacy P, Nelissen F. McCarl GAMS User Guide (Version 24.0); 2013. http://www.gams.com/dd/docs/bigdocs/gams2002/mccarlgamsuserguide.pdf.
Mocsny D, Govind R. Decomposition strategy for the synthesis of minimum-unit heat exchanger networks. AIChE J., 1984; 30(5):853-856.
Morar M, Agachi PS. Review: Important contributions in development and improvement of the heat integration techniques. Comput. Chem. Eng., 2010; 34(8):1171-1179.
Nemhauser GL, Wolsey LA. Integer and combinatorial optimization. New York: John Wiley & Sons, Inc.; 1988.
Nguyen DQ, Barbaro A, Vipanurat N, Bagajewicz MJ. All-at-once and step-wise detailed retrofit of heat exchanger networks using an MILP model. Ind. Eng. Chem. Res., 2010; 49(13):6080-6103.
Papoulias SA, Grossmann IE. A structure optimization approach in process synthesis – II. Heat recovery networks. Comput. Chem. Eng., 1983; 7(6):707-721.
Pettersson F. Synthesis of large-scale heat exchanger networks using a sequential match reduction approach. Comput. Chem. Eng., 2005; 29(5):993-1007.
Shelton MR, Grossmann IE. Optimal synthesis of integrated refrigeration systems—i: Mixed-integer programming model. Comput. Chem. Eng., 1986; 10(5):445-459.
Shethna HK, Jezowski JM. Near independent subnetworks in heat exchanger network design. Ind. Eng. Chem. Res., 2006; 45(13):4629-4636.
Shethna HK, Jezowski JM, Castillo FJL. A new methodology for simultaneous optimization of capital and operating cost targets in heat exchanger network design. Appl. Therm. Eng., 2000; 20(15-16):1577-1587.
Wood RM, Wilcox RJ, Grossmann IE. A note on the minimum number of units for heat exchanger network synthesis. Chem. Eng. Commun., 1985; 39(1-6):371-380.
Yee TF, Grossmann IE. Simultaneous optimization models for heat integration - II. Synthesis of heat exchanger networks. Comput. Chem. Eng., 1990; 14(10):1165-1184.
Zhao XG, O'neill BK, Roach JR, Wood RM. Heat Integration for Batch Processes: Part 2: Heat Exchanger Network Design. Chem. Eng. Res. Des., 1998; 76(6):700-710.