Imperial College London
Department of Computing

Symmetry and Degeneracy in Nonconvex Optimisation Problems: Application to Heat Recovery Networks

Georgia Kouyialis

Advisor: Dr. Ruth Misener
Submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of the
Diploma of Imperial College London, January 18, 2019

Declaration of Originality

I herewith certify that all material in this dissertation which is not my own work has been properly acknowledged.

Copyright Declaration

Georgia Kouyialis

Abstract

Many optimisation problems are formulated with nonconvexities in the objective function and the set of constraints. Nonconvex optimisation has applications in a wide range of disciplines and this thesis examines scheduling and process network design problems. Two main solution approaches are used to deal with such problems: exact and approximation algorithms. Exact algorithms guarantee to solve a problem to global optimality but may require exponential time. On the other hand, approximation algorithms can generate near-optimal solutions in reasonable time. Both sets of algorithms could benefit from insights on the special structure of optimisation problems, e.g. symmetry and degeneracy.

This thesis proposes novel structures i.e. matrices and graphs, for detecting symmetry in Quadratically Constrained Quadratic Programs. In several critically important engineering applications, such as Heat Exchanger Network Synthesis (HENS), symmetry and degeneracy have not been characterised yet. This work investigates the minimum number of matches, e.g. heat exchanger units, which is the current bottleneck in designing HENS. We classify special cases with many equivalent optimal solutions and define symmetry and degeneracy. Due to the aforementioned complexities, we report via computational results that state-of-the-art approaches cannot solve the minimum number of matches problem to global optimality for moderately-sized instances. Hence this thesis develops three classes of heuristics with performance guarantees to the minimum number of matches problem. Each of these heuristics is either novel or provably the best in its class. Our work has interesting implications for solving the problem exactly, e.g. the analysis into reducing big-M parameters or the possibility of quickly generating good primal feasible solutions.

Detecting special structures in optimisation problems and dealing with instances of HENS is neither trivial nor easy. This thesis provides an in-depth analysis of these problems and develops fundamental tools to efficiently solve challenging optimisation problems via both exact and approximation approaches.

Acknowledgements

My family told me that nothing comes easy and luck is when opportunity meets commitment to my goals. My advisor has shown me that career success takes hard work, dedication and the willingness to take personal responsibility for your actions. My PhD experience has taught me that luck is the contribution of all the amazing people around me, without whom the successful completion of this journey would not have been possible.

Firstly, I would like to express my gratitude to my advisor Dr. Ruth Misener for her continuous support of my PhD studies. Dr. Ruth Misener is an inspiring example of a scientist with a vision and has the ability to convey that to the others. I have learned that what she loves most about her job is working with her students. Her willingness to invest time and adjust her training style to one's needs and to guide with her valuable knowledge distinguishes her as an exceptional advisor. She has the ability to not only see the potentials of each student that she works with but to maximise their accomplishments. I would never have imagined that I would be able to stand out with my work in the Department of Computing and Chemical Engineering, coming from the Department of Mathematics. Dr. Ruth Misener, however, believed in my abilities and pushed me to reach my full potential and exceed myself. I am happy to call these four years a successful journey of research and self- development highly correlated to her support and guidance. Whether I work in academia or not, I hope that I will be as enthusiastic and lively as Dr. Ruth Misener and be able to command an audience as well as she does. I am hugely respectful and appreciative to her.

Similar profound gratitude goes to Dr. Dimitrios Letsios. He has been a tremendous mentor to me. I am indebted for several discussions that helped me to focus on my work and evolve as a researcher. Dr. Dimitrios Letsios with his patience and knowledge, introduced me to the world of approximation algorithms and taught me exciting tricks and properties of several combinatorial optimisation problems. Working with him was truly a pleasant experience and a productive period that led to the publication of my favourite piece of work. His passion for academia inspired me to overcome the limitations that I had set for myself and complete my PhD. I keep all his advice and move forward with his words: to set a big goal (that probably scares me) and to constantly work towards that.

I would also like to thank Professor Leo Liberti and Professor Nilay Shah for serving as my thesis committee members. I want to thank them for letting my viva be an enjoyable moment of constructive discussion and for their insightful comments and suggestions. Moreover, in 2015 Professor Leo Liberti had generously invited me to École Polytechnique in France from where

I keep fond memories and appreciation to the member of his group.

Furthermore, I would like to thank Dr. Michael Kapsos and Professor Berc Rustem. Back in 2013, Dr. Michael Kapsos was a former member of the COG (Computational Optimisation Group) group at Imperial College and introduced me to his advisor Professor Berc Rustem. Dr. Michael Kapsos is the person who encouraged me to embark into the world of computational optimisation and Professor Berc Rustem kindly supported my PhD application. I would also like to thank the rest of the members of the COG group and the Department of Computing. I would like to thank Dr. Panos Parpas who has served as my second advisor and with Professor Michael Huth as examiners in the early stage review. Dr. Wolfram Wiesemann gave me the opportunity to work as a tutorial assistant at the Imperial College Business School and see the applications of optimisation from a different viewpoint. The technical and emotional support from all my seniors is greatly appreciated. From their personal experience, I learned that a PhD is a bumpy road and they all encouraged me to believe that I would conquer that. I want to thank Dr. Chin Pang Ho (Clint) for helping me realise that a PhD is a long journey of self-development. Every discussion with him challenged me to understand and improve my academic skills. Dr. Grani A. Hanasusant and Dr. Vladimir Roitch helped me to learn useful technical tools at the very beginning of my PhD. I will never forget the guidance of Dr. Juan Campos Salazar on the flow of my first formal presentation. Advice that I continue to use for my presentations since then. Tutoring with him was also a fun experience. Dr. Vahan Hovhannisyan has encouraged me to stay calm and positive with his chill mood and vibes. Dr. Sei Howe also explained that it is expected to go through all the ups and downs and I am happy that she is still around. Besides my seniors, I am glad to confess that Dr. Ruth Misener manages to attract the nicest people in her group. There is a strong spirit of support and collaboration (among related and unrelated topics!) within all the group members which made a great environment to spend most of my everyday time during the PhD. I would like to thank Radu Baltean for the interesting discussions on our work and PhD experience. Miten Mistry is my tech wizard. He forced me to start using tikz and his attention to details has provided me with great feedback on the outline and the presentation of my work. Simon Olofsson, Johannes Wiebe and Francesco Ceccon are all great examples of hard-working and talented fellow PhD students. I would like to extend my thanks to Dr. Jeremy Cohen, Dr. Layal Hakim, Haoyang Wang and Alexander Thebelt from 302 office, Dr. Vu Ngoc Duy Luong and Dr. Quang Kha Tran alumni members of the COG group.

Thank you all for the stimulating discussions, for the long days (and few late nights) at the office and for all the fun we have had in the last four years. Even during the darkest days, I would always find a reason to smile because of you! I am lucky to call you colleagues and I am sure

that we will continue the team activities outside Imperial College as friends.

I am also appreciative to all the members of the Centre for Process System Engineering from the Department of Chemical Engineering at Imperial College. My participation in events, meetings and workshops as a tutorial assistant was a great experience. Their work has broadened my interest in the petroleum industry, the field of my first and current job. Sincere thanks to Professor Stratos Pistikopoulos, Professor Costas Pantelides and Dr. Dimitri Papageorgiou for their scientific advice on my work and career prospects. Thanks are also due to the EPSRC for their financial support and for which without I would not have been able to develop my scientific skills. Attending major events and conferences, presenting my work and meeting with great researchers and industrial people in my field was one of the best parts of this journey.

Besides the people directly linked with my PhD group, I could not be more thankful to my 'PhDaes gang' at Imperial College, Nicholas Miscourides, Kyriakos Nikiforou, Renos Karamanis, Christina Koutsoumba, Constantinos Papayiannis and Hilda Xue (Drs. to be). I would like to thank each and every one of them! I am lucky and glad to have met them and I treasure every lunch, break and moments of laughter, discussions and complains about the PhD life (okay I was the drama queen!). You made my years there a wonderful experience and turned all the highs higher and lows less frequent.

I am particularly indebted to all the people from UCL during my undergraduate studies that believed in me from the very beginning and encouraged me to extend my studies. My deepest appreciation is also towards my final year project advisor Dr. Isidoros Strouthos. His kindness and patience were crucial for introducing me to the exciting world of scientific research and for successfully completing a thesis and a presentation day to remember. His love for teaching is contagious. Dr. Roberts Bowl and Dr. Mark Roberts provided great recommendations and advice as both tutors and lecturers. Special thanks to my sister Elena Kouyialis and cordial friends Erini Cleanthous and Dr. Isabelle Kamenou. They truly learned what it takes to go through a PhD and their continuous support and encouragement were precious. I would like to extend my thanks to my friends Alexandra Markidou, Andria Petrou, Aris Agathangelou, Cristian Louca, Georgoulla Petrou, Maria Scotti and Niki Constantinidou for all the special memories that I will always cherish and for still being around.

Finally, special mention goes to Nicos, Marina, Elena and Mikaela Kouyialis. I consider myself privileged to have them as my family. Nicos and Marina are both great role models of parents and career professionals. Their thirst for self-cultivation through life experiences and education constantly inspires me to evolve and progress as an individual and accomplish my goals. Elena and Mikaela make me realise the strength of the bond called sisterhood. Their support and affection are my strength. I would like to thank them for their unconditional love and for always believing in me and encouraging me to pursue my passions and follow my dreams.

This thesis is dedicated to them.

List of Publications and Presentations

0.0.1 Publications

- Letsios D., **Kouyialis G.**, Misener R. Heuristics with Performance Guarantees for the Minimum Number of Matches Problem in Heat Recovery Network Design. *Computers & Chemical Engineering*; 2018.

- **Kouyialis G.**, Misener R. Detecting Symmetry in Designing Heat Exchanger Networks. In C. Maravelias, J. Wassick, E. Ydstie, and L. Megan, (eds), Proceedings of *FOCAPO/CPC*. AZ; 2017.

- Ceccon F., **Kouyialis G.**, Misener R. Using Functional Programming to Recognize Named Structure in an Optimization Problem: Application to Pooling. *AIChE Journal*; 2015.

0.0.2 Presentations [* Presenter]

- Letsios D., **Kouyialis G.***, Misener R. Heuristics with Performance Guarantees for the Minimum Number of Matches in Heat Recovery Networks. INFORMS, Arizona, USA; 11/2018.

- Letsios D., **Kouyialis G.**, Misener R. Heuristics with Performance Guarantees for the Minimum Number of Matches in Heat Recovery Networks. AIChE Annual Meeting, Minneapolis, USA; 11/2017.

- **Kouyialis G.***, Misener R, Detecting Symmetry in Designing Heat Exchanger Networks. 1^{st} PSE@ResearchDayUK, Imperial College London, UK; 07/2016.

0.0.3 Posters [* Presenter]

- Letsios D., **Kouyialis G.***, Misener R. Heuristics with Performance Guarantees for the Minimum Number of Matches in Heat Recovery Networks. SIAM UKIE Annual Meeting, Southampton, UK; 01/2018.

- Letsios D., **Kouyialis G.***, Misener R. Heuristics with Performance Guarantees for the Minimum Number of Matches in Heat Recovery Networks. CPSE, Annual Industrial Consortium Meeting, Imperial, UK; 12/2017.
- Letsios D., **Kouyialis G.***, Misener R. Heuristics with Performance Guarantees for the Minimum Number of Matches in Heat Recovery Networks. 2^{nd} PSE@ResearchDayUK, Imperial, UK; 06/2017.
- **Kouyialis G.*** G., Misener R, Detecting Symmetry in Designing Heat Exchanger Networks. Foundations of Computer Aided Process Operations/Chemical Process Control. Tucson, AZ; 01/2017.
- **Kouyialis G.***, Misener R. Data Structures for Representing Symmetry in Quadratic Programs. CPSE, Annual Industrial Consortium Meeting. Imperial, UK; 12/2015.
- **Kouyialis G.***, Misener R. Exploiting Symmetry in Mixed-Integer Nonlinear Optimisation. Dep of Computing, PhD Google Poster Competition. Imperial, UK; 03/2015.

0.0.4 Honours and Awards

- **Funded** by EPSRC Doctoral Training Account Studentship (EP/P008739/1), Imperial College London, UK 2014 - 2018.
- **Awarded** 1^{st} Poster Prize (out of 37) at the UK/Ireland Annual Meeting of the Society for Industrial & Applied Mathematics, Southampton, UK 2018.
- **Awarded** UK/Ireland Annual Meeting of the Society for Industrial & Applied Mathematics, Travel Grant, Southampton, UK 2018.
- **Awarded** Best Poster (out of 19) at PSE@ResearchDayUK , London, UK 2017.
- **Awarded** 2^{nd} Poster Prize (out of 24) at the CPSE Annual Industrial Consortium Meeting, London, UK 2017.
- **Awarded** FOCAPO/CPC Travel Grant, Tucson, AZ 2017.
- **Awarded** 3^{rd} prize (out of 21) for 1st year PhD students in the Department of Computing Google Poster Competition London, UK 2015.

Contents

Chapter 1

Mathematical Toolkit

1.1 List of Abbreviations

P^L - LP	Linear Program
P^{MIL} - MILP	Mixed-Integer Linear Program
P^{MINL} - MINLP	Mixed-Integer Nonlinear Program
P^Q - QCQP	Quadratically Constrained Quadratic Program
P^{MIQCQ} - MIQCQP	Mixed-Integer Quadratically-Constrained Quadratic Program
P^{LQ} - LQP	Linearised Program
B&B	Brunch-and-Bound Algorithm
RLT	Reformulation Linearisation Technique
BLG	Binary Layered Graph
DAG	Directed Acyclic Graph
HENS	Heat Exchanger Network Synthesis Problem

Table 1.1: Table of notation for symmetry in mathematical programming

Symbol	Description	Symbol	Description
x_i, y_i, z	Variables	I	Identity element
$\mathbf{x}, \mathbf{y}$	Vectors of variables	π, σ	Permutations
α	Coefficient	Π	Set of permutations
$\mathbf{a}, \mathbf{c}, \mathbf{b}, \mathbf{p}$	Vectors of parameters	S^n	Symmetric group of order n
$\mathbf{A}, \mathbf{Q}, \mathbf{B}, \mathbf{X}$	Matrices of parameters/variables	Ω, Y, X	Sets
θ	Variables	f, ξ, ϕ	Functions
$\mathbf{M}, \mathbf{I}, \mathbf{J}, \mathbf{K}$	Sparse representations of matrices	$\mathbf{G}, \mathbf{H}$	Graphs
$\mathscr{F}$	Set of feasible solutions	E, V	Set of edges, vertices
$\mathscr{G}, \tilde{\mathscr{G}}$	Symmetry groups	e	Edges in the graph
$W, Z, \mathbf{K}$	Groups	u, v	Nodes in the graph
C_n	Cyclic group order n	r_k, λ	Constants
X_{ij}, Y_{ij}, Z_{ij}	Auxiliary variables	d, z, g, w, k	Group elements

1.2 Lists of Notation

1.3 Mathematical Preliminaries

This section presents basic definitions and notation of group theory (Clark 1984, Robinson 1996) and graph theory (Chartrand 1977, Gibbons 1985).

1.3.1 Definitions of Convexity

The definition of convexity is used in a subsequent proof (Bertsekas et al. 2003).

Definition 1.1 *Convex set*

A set $X \subset \mathbb{R}^n$ is called convex if

$$\forall \mathbf{x}, \mathbf{y} \in X, \lambda \in [0,1] \qquad \lambda \mathbf{x} + (1-\lambda)\mathbf{y} \in X$$

For any two points in the set, a line segment between them includes only points that are inside the set.

Definition 1.2 *Convex function*

Table 1.2: Notation for the minimum utility cost problem (an LP optimisation model)

Name	Description
Cardinalities, Indices, Sets	
ns, ms	Number of hot, cold streams
nu, mu	Number of hot, cold utilities
k	Number of temperature intervals
$i \in HS \cup HU$	Hot stream, utility
$j \in CS \cup CU$	Cold stream, utility
$t \in TI$	Temperature interval
HS, CS	Set of hot, cold streams
HU, CU	Set of hot, cold utilities
TI	Set of temperature intervals
Parameters	
FCp_i, FCp_j	Flowrate heat capacity of hot stream i, cold stream j
$T^{HS}_{\text{in},i}, T^{HS}_{\text{out},i}$	Inlet, outlet temperature of hot stream i
$T^{CS}_{\text{in},j}, T^{CS}_{\text{out},j}$	Inlet, outlet temperature of cold stream j
$T^{HU}_{\text{in},i}, T^{HU}_{\text{out},i}$	Inlet, outlet temperature of hot utility i
$T^{CU}_{\text{in},j}, T^{CU}_{\text{out},j}$	Inlet, outlet temperature of cold utility j
$\Delta T_{\min}$	Minimum heat recovery approach temperature
$\kappa^{HU}_i, \kappa^{CU}_j$	Unitary cost of hot utility i, cold utility j
$\sigma^{HS}_{i,t}$	Heat supply of hot stream i in interval t
$\delta^{CS}_{j,t}$	Heat demand of cold stream j in interval t
Variables	
$\sigma^{HU}_{i,t}$	Heat supply of hot utility i in interval t
$\delta^{CU}_{j,t}$	Heat demand of cold utility j in interval t
R_t	Residual heat exiting temperature interval t

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called convex if

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1] \qquad f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y})$$

Definition 1.3 *Concave function*

A function $f : \mathbb{R}^n \to \mathbb{R}$ is called concave if function $(-f)$ is convex on X.

$$\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \lambda \in [0,1] \qquad f(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \geq \lambda f(\mathbf{x}) + (1-\lambda) f(\mathbf{y})$$

1.3.2 Group Theory

Group theory studies the algebraic structure of a wide range of objects with or without special properties. An important class of groups is the one of transformations (symmetries). We use the class of permutation groups, e.g. the symmetric and the cyclic group, to describe the

transformations in geometric objects (Armstrong 1988, Cameron 1999).

Definition 1.4 *Group*

A group $(W, \cdot)$ is a nonempty set W with a binary operation $\cdot$ on W satisfying the following properties:

1. *If $g, z \in W$, then $g \cdot z$ is also in W;*
2. *$g \cdot (z \cdot d) = (g \cdot z) \cdot d$ for all $g, z, d \in W$;*
3. *Every group W has an identity element I, such that $g \cdot I = I \cdot g = g$, $for each g \in W$;*
4. *If $g \in W$, $\exists\, g^{-1} \in W$ such that $g \cdot g^{-1} = g^{-1} \cdot g = I$.*

Definition 1.5 *Subgroup*

A subgroup Z of a group W is a nonempty subset of W that forms a group itself under the operation induced by W.

Definition 1.6 *Isomorphic groups*

Two groups W, Z are isomorphic if $\exists$ a bijective function $\phi : W \to Z$ that satisfies:

1. *$\phi(I) = I$ for an identity element I;*
2. *$\phi(g^{-1}) = \phi(g)^{-1}$, $\forall g \in W$;*
3. *$\phi(gz) = \phi(g)\phi(z)$, $\forall g, z \in W$.*

A group automorphism is an isomorphism from a group to itself.

To define the composition of groups, Clark (1984) uses the notion of external and internal direct product.

Definition 1.7 *External direct product*

Given a finite sequence of groups $K_1, \ldots, K_n$, their external direct product is $\Pi_{i=1}^{n} K_i = K_1 \times K_2 \times \cdots \times K_n$, with elements the tuples $(k_1, k_2, \ldots, k_n)$ for each $k_i \in K_i, \forall i$ and the operation of their product $(k_1, \ldots, k_n)(k_{1'}, \ldots, k_{n'}) = (k_1 k_{1'}, \ldots, k_n k_{n'})$.

Given a finite sequence of subgroups $K_1, \ldots, K_n \leq W$, W is their internal direct product if the following properties hold:

Definition 1.8 *Internal direct product*

1. $W = K_1 \cdots K_n$ *with tuples* $(k_1 \cdots k_n)$ *for* $k_i \in K_i$;
2. *For each* $K_i \cap K_j$ *a trivial subgroup is generated;*
3. *Each* K_i *is a normal subgroup of* W *i.e.* $\forall k \in K_i$, $\forall g \in W$, $gkg^{-1} \subseteq K_i$ *and is denoted as* $K_i \lhd W$.

Definition 1.9 *Permutation*

A permutation of a set $Y = \{1, \ldots, n\}$ *is a bijective function* $\pi : Y \longrightarrow Y$.

Definition 1.10 *Permutation group*

A permutation group Π^n *is a finite group whose elements are permutations of a given set Y and whose group operation is composition of permutations in the group.*

For permutations $\pi \in \Pi^n$, $\sigma \in \Pi^m$, $\mathbf{A}(\pi, \sigma)$ is a matrix obtained by permuting the columns of $\mathbf{A}$ by π and the rows of $\mathbf{A}$ by σ.

Definition 1.11 *Symmetric group*

The symmetric group S^n *is the group of all permutations of a given set Y.*

In mathematics, representation theory studies ways to represent the elements of groups as linear transformations of vector spaces.

Definition 1.12 *Symmetry group*

The symmetry group of an object is the group of all transformations under which the object is invariant with group operation the composition of such transformations.

Definition 1.13 *Cyclic group*

A cyclic group is a group that can be generated by a single element e.g. $C_n = \langle r | r^n = 1, n \in Z \rangle$ *is the cyclic group of order n.*

An object has cyclic symmetries if it is invariant under the transformations in a cyclic group.

1.3.3 Graph Theory

Definition 1.14 *Graph*

A graph is a tuple $\mathbf{G} = (V, E)$ *where* V *is a (finite non-empty) set of vertices and* $v \in V$ *is called a vertex and* $E \subset V \times V$ *is a finite collection of edges and* $e = \{u, v\} \in E$ *is called an edge.*

Definition 1.15 *Loop*

An edge from a vertex to itself $e = \{u\}$ *is said to be a loop.*

Definition 1.16 *Weighted graph*

A weighted graph $\mathbf{K}$ *is a triplet* $\mathbf{K} = (V, E, \mathbf{w})$ *where* $w : E(\mathbf{K}) \to \mathbb{R}$.

Definition 1.17 *Vertex coloured graph*

A ℓ*-colouring of a labelled graph* $\mathbf{G} = (V, E, c)$ *is a function* $c : V(\mathbf{G}) \to \{0, 1, \ldots, \ell - 1\}$ *where* k *is the number of colours. The vertices of one colour form a colour class.*

Definition 1.18 *Graph isomorphism*

Two simple graphs $\mathbf{G} = \{V(\mathbf{G}), E(\mathbf{G})\}$ *and* $\mathbf{H} = \{V(\mathbf{H}), E(\mathbf{H})\}$*, are isomorphic, denoted* $\mathbf{G} \cong \mathbf{H}$*, if* $\exists$ *a bijective function* $f : V(\mathbf{G}) \to V(\mathbf{H})$*, such that for each edge* $\{u, v\} \in E(\mathbf{G})$ *there is an edge* $\{f(u), f(v)\} \in E(\mathbf{H})$*. Under this relation, any set of adjacent vertices* $E(V) = \{\{u, v\} | u, v \in V, u \neq v\}$ *remains adjacent.*

An automorphism is an isomorphism of a graph to itself.

Definition 1.19 *Graph automorphism*

Given a graph $\mathbf{G}$*, a permutation* π *of* $V(\mathbf{G})$ *is an automorphism of* $\mathbf{G}$*, if* $\forall u, v \in V(\mathbf{G})$ *there exist an edge* $\{u, v\} \in E(\mathbf{G})$ *then under any permutation* π *remains in the set of edges as* $\{\pi(u), \pi(v)\} \in E(\mathbf{G})$.

Definition 1.20 *Colour - preserving isomorphism*

Consider pairs $(\mathbf{G}, c)$ *where* $\mathbf{G}$ *is a graph and* $c : V(G) \to \{0, \ldots, \ell - 1\}$ *is a* ℓ*-colouring of* $\mathbf{G}$*. A colour - preserving isomorphism from* $(\mathbf{G}, c)$ *to* (H, c') *is a bijection* $\pi : V(\mathbf{G}) \to V(\mathbf{H})$ *such that* π *is an isomorphism from* $\mathbf{G}$ *to* $\mathbf{H}$ *and* $c(v) = c'(\pi(v))$ $\forall v \in V(\mathbf{G})$.

A colouring of the vertices is also referred to as a partition, and the colour classes as the cells of the partition.

Definition 1.21 *Graph partitioning*

A graph partitioning of $\mathbf{G}$ *into* ℓ *parts is a collection of nonempty disjoint subsets* $V_0, \ldots, V_{\ell-1}$ *for* $\ell \in \mathbb{Z}$ *whose union is V, i.e.* $V = V_0 \cup \ldots \cup V_{\ell-1}$ $\forall \ell$.

1.3.4 Basic Computational Complexity Classes

We briefly introduce basic computational complexity classes (Papadimitriou 1994, Arora and Barak 2009). In a computational problem, we are given an input and we want to return as output a solution satisfying some property: a computational problem is then described by the property that the output has to satisfy given the input. In a decision problem, given a problem instance, the solution consists of an answer of YES or NO. An algorithm is a step-by-step process for solving a problem. A *polynomial algorithm* produces a solution for a computational problem with a running time polynomial to the size of the problem instance. A decision problem belongs to class $\mathscr{P}$ if there is a deterministic algorithm that solves it in polynomial time. The class of $\mathscr{NP}$ is the set of decision problems for which a "YES" instance can be *verified* deterministically in polynomial time. A decision problem P is $\mathscr{NP}$-hard when for every problem Q in $\mathscr{NP}$, there is a polynomial-time reduction from Q to P. There is also the class of $\mathscr{NP}$*-complete problems* for which we do not know whether they admit a polynomial algorithm or not. The question of whether $\mathscr{NP}$-complete problems admit a polynomial algorithm is known as the $\mathscr{P} = \mathscr{NP}$ question. In general, it is conjectured that $\mathscr{P} \neq \mathscr{NP}$, i.e. $\mathscr{NP}$-complete problems are not solvable in polynomial time. An optimisation problem is $\mathscr{NP}$*-hard* if its decision version is $\mathscr{NP}$-complete. A computational problem is *strongly* $\mathscr{NP}$*-hard* if it remains $\mathscr{NP}$-hard when all parameters are bounded by a polynomial to the size of the instance.

Table 1.3: Notation for the minimum number of matches problem, presenting the standard transportation and transshipment MILP models

Name	Description
Cardinalities	
n	Number of hot streams
m	Number of cold streams
k	Number of temperature intervals
v	Number of matches (objective value)
Indices	
$i \in H$	Hot stream
$j \in C$	Cold stream
$s,t,u \in T$	Temperature interval
$b \in B$	Bin (single temperature interval problem)
Sets	
H, C	Hot, cold streams
T	Temperature intervals
M	Set of matches (subset of $H \times C$)
$C_i(M), H_j(M)$	Cold, hot streams matched with $i \in H$, $j \in C$ in M
B	Bins (single temperature interval problem)
$A(M)$	Set of valid quadruples (i,s,j,t) with respect to a set M of matches
$A_u(M)$	Set of quadruples $(i,s,j,t) \in A(M)$ with $s \leq u < t$
$V^H(M)$	Set of pairs $(i,s) \in H \times T$ appearing in $A(M)$ (transportation vertices)
$V^C(M)$	Set of pairs $(j,t) \in C \times T$ appearing in $A(M)$ (transportation vertices)
$V^C_{i,s}(M)$	Set of pairs $(j,t) \in V^C(M)$ such that (i,s,j,t) belongs to $A(M)$
$V^H_{j,t}(M)$	Set of pairs $(i,s) \in V^H(M)$ such that (i,s,j,t) belongs to $A(M)$
Parameters	
h_i	Total heat supplied by hot stream i ($h_i = \sum_{s \in T} \sigma_{i,s}$)
$h_{\max}$	Maximum heat among all hot streams ($h_{\max} = \max_{i \in H}\{h_i\}$)
c_j	Total heat demanded by cold stream j ($c_j = \sum_{t \in T} \delta_{j,t}$)
$\sigma_{i,s}$	Heat supply of hot stream i in interval s
$\delta_{j,t}$	Heat demand of cold stream j in interval t
$\vec{\sigma}, \vec{\delta}$	Vectors of all heat supplies, demands
$\vec{\sigma}_t, \vec{\delta}_t$	Vectors of all heat supplies, demands in temperature interval t
R_t	Residual heat exiting temperature interval t
$U_{i,j}$	Upper bound (big-M parameter) on the heat exchanged via match (i,j)
$\lambda_{i,j}$	Fractional cost approximation of match (i,j) (Lagrangian relaxation)
$\vec{\lambda}$	Vector of all fractional cost approximations $\lambda_{i,j}$
Variables	
$y_{i,j}$	Binary variable indicating whether i and j are matched
$q_{i,j,t}$	Heat of hot stream i received by cold stream j in interval t
$q_{i,s,j,t}$	Heat exported by hot stream i in s and received by cold stream j in t
$\vec{y}, \vec{q}$	Vectors of binary, continuous variables
$r_{i,s}$	Heat residual of heat of hot stream i exiting s
x_b	Binary variable indicating whether bin b is used
$w_{i,b}$	Binary variable indicating whether hot stream i is placed in bin b
$z_{j,b}$	Binary variable indicating whether cold stream j is placed in bin b
Other	
N	Minimum cost flow network
G	Solution graph (single temperature interval problem)
$\phi(M)$	Filling ratio of a set M of matches
$\vec{y}^f, \vec{q}^f$	Optimal fractional solution
α_i, β_j	Number of matches of hot stream i, cold stream j
$L_{i,j}$	Heat exchanged from hot stream i to cold stream j
I	Instance of the problem
r	Remaining heat of an algorithm

Chapter 2

Symmetry and Degeneracy in Mathematical Programming

2.1 Introduction

Mixed-Integer Nonlinear Programming (MINLP) is one of the most challenging computational optimisation problems as it combines combinatorial aspects with non-linearities that arise in a broad range of applications in fields such as manufacturing, chemical and biological sciences and engineering design (Floudas et al. 2005, Boukouvala et al. 2016, Liberti 2018). MINLP are $\mathscr{NP}$-hard in the most general case (Bussieck and Pruessner 2003). Subclasses of MINLP include; Nonlinear Programming (NLP), Mixed-Integer Linear Programming (MILP), Linear Programming (LP), Mixed-Integer Quadratically-Constrained Quadratic Programming (MIQCQP) and Quadratically-Constrained Quadratic Programming (QCQP).

Many state-of-the-art algorithms have been proposed to solve MINLP to global optimality. Floudas and his coworkers establish the literature of this field as presented in (Floudas and Gounaris 2008). While solving an MINLP problem a series of techniques may applied to simplify its solvability and attain a global solution i.e. cutting planes, reformulation and linearisation methods, and solving substreams of the original problem (Belotti et al. (2009a), Skjäl et al. (2012)). Moreover, it has been shown that, in several classes of problems, considering *symmetry*

and *degeneracy* could speed up the performance of the algorithms (Costa et al. 2013, Ostrowski et al. 2015).

2.2 Mathematical Optimisation

Mathematical programming formulates all the aforementioned classes of optimisation problems. It consists of the input parameters, the decision variables as output that optimise the objective function and a set of constraints which restrict the feasible region. The property of nonconvexity in mathematical programs imposes more difficulties when we try to solve them. As shown in Figure 2.1, nonconvexity causes the existence of multiple local optima when we may be seeking a global solution that gives the best optimal value.

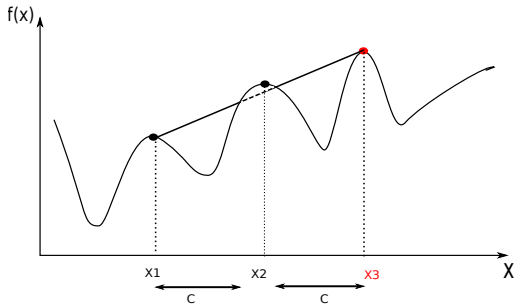


Figure 2.1: Graph: Maximisation function of multiple optima

2.2.1 Mixed-Integer Nonlinear Programming

We refer to a number of optimisation models that have both discrete and continuous variables, dealing with nonlinear functions in the objective function and/or the constraints. A general MINLP formulation is as follows:

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x},\mathbf{y}) \\ \text{s.t.} \quad & \phi(\mathbf{x},\mathbf{y}) = 0 \\ & \xi(\mathbf{x},\mathbf{y}) \leq 0 \\ & \mathbf{x} \in X \\ & \mathbf{y} \in Y \end{aligned} \tag{MINLP}$$

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $\phi, \xi : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ and polyhedra sets $X = \{\mathbf{x} | \mathbf{x} \in \mathbb{R}^n, \mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u, \mathbf{B}\mathbf{x} \leq \mathbf{b}\}$ and $Y = \{\mathbf{y} | \mathbf{y} \in \{0,1\}^m, \mathbf{A}\mathbf{y} \leq \mathbf{a}\}$ have values of decision continuous and discrete variables respectively. Generic solvers for MINLP include ANTIGONE (Misener and Floudas 2014), BARON (Sahinidis 1996), Couenne (Belotti et al. 2009b), LINDO (Youdong and Linus 2009), and SCIP (Achterberg 2009).

2.2.2 Mixed-Integer Quadratically Constrained Quadratic Programming

A subclass of MINLP restricted to quadratic and bilinear nonlinearities is called Mixed-Integer Quadratically Constrained Programming (MIQCP) problems. Several geometry problems are mathematically formulated as MIQCQP (Kucherenko et al. 2007, Kallrath 2009).

A general formulation is given:

$$\begin{aligned} \min_{x,y} \quad & f_0(\mathbf{x}, \mathbf{y}) \\ \text{s.t.} \quad & f_k(\mathbf{x}, \mathbf{y}) \leq 0 \quad \forall k = 1, \cdots, m \\ & \mathbf{x} \in \mathbb{R}^n \\ & \mathbf{y} \in \{0,1\}^m \end{aligned} \tag{MIQCQP}$$

where the functions $f_k : \mathbb{R}^n \to \mathbb{R}$, have the form: $f_k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{Q_k} \mathbf{x} + \mathbf{p_k} \mathbf{x} + \mathbf{r_k} \mathbf{y}$, $\forall k = 0, \cdots, m$ and $\mathbf{Q_0} \cdots \mathbf{Q_m} \in \mathbb{R}^{\mathrm{n} \times \mathrm{n}}$, are n by n matrices and $\mathbf{x} \in \mathbb{R}^{\mathrm{n}}$ with $\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u$, $\mathbf{p_k} \in \mathbb{R}^{\mathbf{n}}$, $r_k \in \mathbb{R}$.

2.2.3 Mixed-Integer Linear Programming

A large number of optimisation problems in Process Systems Engineering (PSE) can be described by Mixed-Integer Linear Programming (MILP) models since the integrality constraints capture the discrete nature of decisions. Examples include the optimisation of production operations e.g. planning and scheduling (Maravelias and Grossmann 2004), multiple period optimisation (Lotero et al. 2016), and process synthesis using simplified models without nonlinearities (Biegler et al. 1997). The formulation follows MINLP.

Consider the MINLP formulation. If the functions in the objective $f(\mathbf{x}, \mathbf{y})$ and the constraints are linear and $X \neq \emptyset$, $Y \neq \emptyset$, i.e. variables take both continuous and integer values, then the

MINLP formulation actually describes a MILP.

2.3 Resource Allocation Problems

Resource allocation problems are widely studied using mathematical programming models (Koopman 1953, Katoh et al. 2013). The objective is to achieve the best assignment of limiting resources with some consideration to costs and benefits of such allocations. Efficient algorithms have been developed, based on the type of the objective function, constraints and variables. The concept of allocation of resources is wide spread in industry and several industrial problems belong to this class. Applications include: load distribution, petroleum refining, supply-chain operations and computer resource allocation. In these thesis we study and categorise special cases of packing, scheduling and network design problems.

2.3.1 Scheduling Problems

Scheduling optimisation problems allocate resources to perform a set of tasks over time. The resources are modelled as machines and the requests for resources are modelled as jobs with a service requirement. Given this broad definition, scheduling problems frequently arise in the manufacturing industries such as production planning in factories, supply chain planning of pharmaceutical companies (Sousa et al. 2011) and scheduling of heat-integrated plants (Papageorgiou et al. 1994). A vast amount of literature on machine scheduling, including job shop scheduling, has been published (Brucker 2001, Conway et al. 2012).

Typically, scheduling problems aim to minimise the total amount of time (or cost) required to complete all the tasks. The most basic version is as follows: We are given n jobs of varying processing times, which need to be scheduled on m machines with varying processing power, while trying to minimise the makespan.

2.3.1.1 Job Shop Scheduling Problem

The following simple scheduling problem incorporates also time limitations. Given m identical machines for scheduling, indexed by the set $i \in \{1,\dots,m\}$. There are n given jobs, indexed by

the set $j \in \{1,\ldots,n\}$, with processing time $p_{i,j}$ units and resources at the rate $c_{i,j}$. Each task j has release time zero and deadline d_j. All the jobs, randomly chosen, can be processed on any of the m machines. The task is to assign jobs to machines so that the completion time, also called the makespan, is minimised. The formulation is given by Ierapetritou and Floudas (1998), Hooker (2005):

$$\min \quad z \tag{2.1}$$

$$\text{s.t.} \quad \sum_{it} x_{ijt} = 1 \qquad \forall j \tag{2.2}$$

$$\sum_{j} \sum_{t' \in T_{ijt}} c_{ij} x_{ijt} \leq C_i \qquad \forall i,t \tag{2.3}$$

$$z \geq \sum_{it} (t + p_{ij}) x_{ijt} \qquad \forall j \tag{2.4}$$

$$x_{ijt} = 0 \qquad \forall j,t \text{ with } d_j - p_{ij} < t < r_i \text{ or } t > n - p_{ij} \tag{2.5}$$

$$x_{ijt} \in \{0,1\} \qquad \forall i,j,t \tag{2.6}$$

There are N discrete times and any job j at machine i might start any time from $t = 0$ and $T_{ijt} = \{t' | t - p_{ij} < t' \leq t\}$. Constraints ensure that each job is allocated to one machine and starts once. Moreover, there are time windows and each machine has to be within a limit. The total rate of jobs consumption on machine i is never more than C_i at any given time.

2.3.2 Point Packing Problems

We study and categorise classes of computational geometry problems as packing problems (Kallrath 2009, Kucherenko et al. 2007). An example of such problems is to maximise the number of circles that can fit in a given shape.

Problem 2.1 *Maximise the minimum distance θ between every pair of points (x_i, y_i) in a unit*

square.

$$\begin{aligned} \max \quad & \theta \\ s.t. \quad & (x_i - x_j)^2 + (y_i - y_j)^2 \geq \theta \qquad 1 \leq i < j \leq n \\ & x \in [0,\,1]^n \\ & y \in [0,\,1]^n \end{aligned}$$

The nonconvexity in several packing problems appears due to the distance constraint $(x_i - x_j)^2 + (y_i - y_j)^2 \geq \theta$, $\theta \in \mathbb{R}$ which is a nonconvex function.

Statement 2.1

$$(x_i - x_j)^2 + (y_i - y_j)^2 \geq \theta \tag{2.7}$$

is a nonconvex function

Proof:

We prove this statement by using the method of contradiction.

Let the set:

$$\Omega = \left\{ (\mathbf{x}, \mathbf{y} \in \mathbb{R}^n \times \mathbb{R}^n : (x_i - x_j)^2 + (y_i - y_j)^2 \geq \theta \; \forall i, j = 1, \ldots, n, i \neq j \right\} \tag{2.8}$$

We will show that $\exists$ two points:

$$\hat{x}, \hat{y} \in \Omega, \quad \text{and a} \quad \hat{\lambda} \in [0,1], \text{ for which: } \hat{\lambda}\hat{x} + (1 - \hat{\lambda}\hat{y}) \notin \Omega$$

Consider points $(x_1, y_1), (x_2, y_2), (x_2, y_2 - \varepsilon), (x_1, y_1 - \varepsilon) \in \Omega$ for $\varepsilon > 0$ as shown in the graph, satisfying the conditions in Ω.

A point say $\hat{X}$ on the line segment between $(x_1, y_1), (x_2, y_2 - \varepsilon)$ is of the form:

$$\lambda(x_1, y_1) + (1 - \lambda)(x_2, y_2 - \varepsilon) = (\lambda x_1 + (1 - \lambda)x_2, \quad \lambda y_1 + (1 - \lambda)(y_2 - \varepsilon)) \tag{2.9}$$

A point say $\hat{Y}$ on the line segment between (x_2, y_2), $(x_1, y_1 - \varepsilon)$ is of the form:

$$\lambda(x_2, y_2) + (1-\lambda)(x_1, y_1 - \varepsilon) = (\lambda x_2 + (1-\lambda)x_1, \quad \lambda y_2 + (1-\lambda)(y_1 - \varepsilon)) \tag{2.10}$$

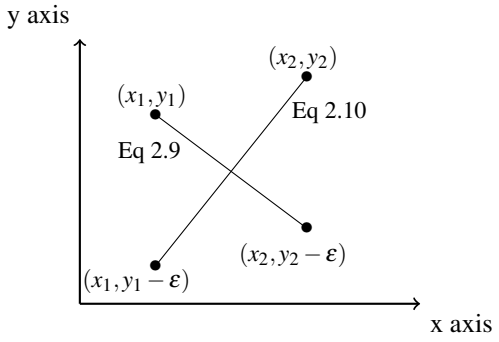


Figure 2.2: Graph illustrating the proof of nonconvexity of Point Packing Problems.

Subtract the two arrays to get the distance between the points:

$$\begin{pmatrix} \lambda x_1 - \lambda x_2 + x_2 - \lambda x_2 - x_1 + \lambda x_1 \\ \lambda y_1 - \lambda y_2 + y_2 - \lambda y_2 - y_1 + \lambda y_1 \end{pmatrix}$$

Now choose $\hat{\lambda} = \frac{1}{2}$, and the distance is equal to $|\hat{x} - \hat{y}|^2 = 0 < \theta$ which contradicts the conditions of Ω.

We conclude that the function is nonconvex. ∎

2.4 Relaxation Techniques

Real world problems tend to be large and exhibit an exponential complexity with the problem size. Relaxation techniques are used to transform such optimisation problems into simpler related problems.

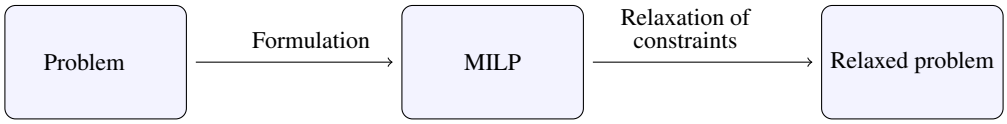


Figure 2.3: Steps of a Relaxation Technique

2.4.1 Relaxation for Mixed-Integer Linear Programming

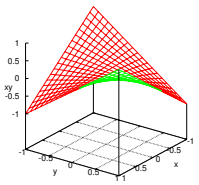
The basic idea of relaxation is illustrated in Figure 2.3. We relax some constraints of the original MILP and obtain another problem. We use optimisation algorithms to solve this problem. The solution to the relaxed program can be used to gain information about the solution to the original integer program. However, it is not necessarily feasible for the original problem but is rather considered as a valid bound bound in the integer model's solution.

Linear programming relaxation in the most general form of MILP relaxes the integrality constraints and generates an LP. Although the concept of relaxations is applied generally to most nonconvex problems, in this thesis the performance of the LP solver is of paramount importance in the solution of MILP problems and is examined in Chapter 6.

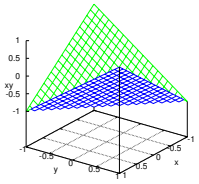
2.4.2 Convex Relaxation for Quadratically Constrained Quadratic Programming

There are several convex relaxation techniques for global optimisation problems (Liberti 2004). McCormick (1976) achieves a convex relaxation of quadratically constrained quadratic problems (formulation MIQCQP with no integral component) of such problems by adding inequality constraints generated on new auxiliary variables which combine the given ones. More precisely, a Reformulation Linearisation Technique (RLT) is the McCormick convex and concave relaxation for bilinear terms.

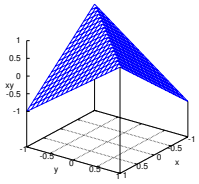
This part follows Anstreicher (2009) and Qualizza et al. (2012) to derive convex relaxation of the original $QCQP$. For each bilinear term set $X_{ij} = x_i x_j$, the McCormick hull forms under and



Bilinear Surface



Lower convex envelope



Upper concave envelope

Figure 2.4: McCormick convex and concave relaxation for bilinear terms

overestimator constraints, related to the variable bounds $x_i^L \leq x_i \leq x_i^U, \;\; x_j^L \leq x_j \leq x_j^U$

$$\begin{cases} X_{ij} \geq x_i^L x_j + x_j^L x_i - x_i^L x_j^L \\ X_{ij} \geq x_i^U x_j + x_j^U x_i - x_i^U x_j^U \\ X_{ij} \leq x_i^U x_j + x_j^L x_i - x_i^U x_j^L \\ X_{ij} \leq x_i^L x_j + x_j^U x_i - x_i^L x_j^U \end{cases}$$

Statement 2.2 *Any quadratic program (QCQP) can be linearised by using the reformulation linearisation technique.*

To derive the McCormick (1976) convex and concave relaxation for bilinear terms, consider any quadratic equation of the form $f_k(\mathbf{x}) = \mathbf{x}^T \mathbf{Q}_k \mathbf{x} + \mathbf{p}_k{}^T \mathbf{x} + r_k \leq 0 \forall k = \{0, \ldots, m\}$ and define:

$$\mathbf{X} = \mathbf{x}\mathbf{x}^T = \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \begin{pmatrix} x_1 & \cdot & \cdot & \cdot & x_n \end{pmatrix} = \begin{pmatrix} x_1x_1 & x_1x_2 & \cdot & \cdot & x_nx_n \\ x_2x_1 & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ x_nx_1 & \cdot & \cdot & \cdot & x_nx_n \end{pmatrix}$$

Rewrite each quadratic expression using the inner product:

$$\mathbf{x}^T \mathbf{Q}_k \mathbf{x} = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{Q}_{kij} x_i x_j = \mathbf{Q}_k \bullet \mathbf{X} = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{Q}_{kij} X_{ij}$$

Now use the variable bounds of x_i,x_j to obtain the constraints of the following linearised optimisation problem *LQP*. Note that for $i \neq j$, $X_{ij} = X_{ji}$ as the matrices are symmetric above their diagonal, and $\mathbf{X} = \mathbf{X}^T$. Hence we can define the linear form of *QCQP* as

Definition 2.1

$$
\begin{aligned}
\max \quad & \mathbf{Q}_0 \bullet \mathbf{X} + \mathbf{p}_0{}^T \mathbf{x} + r_0 \\
s.t. \quad & \mathbf{Q}_k \bullet \mathbf{X} + \mathbf{p}_k{}^T \mathbf{x} + r_k \leq 0 \qquad \forall k = \{1, \ldots, m\} \\
& \mathbf{X} - \mathbf{x}^L \mathbf{x}^T - \mathbf{x}(\mathbf{x}^L)^T \geq -\mathbf{x}^L(\mathbf{x}^L)^T \\
& \mathbf{X} - \mathbf{x}^U \mathbf{x}^T - \mathbf{x}(\mathbf{x}^U)^T \geq -\mathbf{x}^U(\mathbf{x}^U)^T \\
& \mathbf{X} - \mathbf{x}^L \mathbf{x}^T - \mathbf{x}(\mathbf{x}^U)^T \leq -\mathbf{x}^L(\mathbf{x}^U)^T \\
& \mathbf{X} - \mathbf{x}^U \mathbf{x}^T - \mathbf{x}(\mathbf{x}^L)^T \leq -\mathbf{x}^U(\mathbf{x}^L)^T \\
& \mathbf{X} = \mathbf{X}^T \\
& \mathbf{x} \in [\mathbf{x}^L, \mathbf{x}^U]
\end{aligned}
\tag{LQP}
$$

where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{Q}_0, \ldots, \mathbf{Q}_m \in \mathbb{R}^{n \times n}$, *are n by n matrices and* $\mathbf{p}_k \in \mathbb{R}^n$, $r_k \in \mathbb{R}$.

2.5 Solution Procedures

2.5.1 Branch-and-Bound

A widely used divide-and-conquer algorithm for solving mathematical programming problems to global optimality is the branch-and-bound (B&B). A tree search strategy initially proposed by Land and Doig (1960) for solving discrete programming. The B&B algorithm is carried out on a search tree rooted at a node representing the full set of candidate solutions. The search process moves forward through dividing the initial problem into subproblems; generating a search tree in which every node represents a subset of solutions. Basic elements of this algorithm are bounding functions for computing upper and lower bound for each subproblem. Fathoming takes place when (1) a global optimum for the subproblem in that node is found or (2) the node is shown to be infeasible or (3) the lower bound of a subproblem has higher value of the value of the current best optimum as shown in Figure 2.5. The main components of B&B are: separation, relaxation and fathoming.

For MILP problems a standard process for relaxation is to replace integer variables with bounded continuous variables to form LP. Branching strategy is applied to integer variables only and the

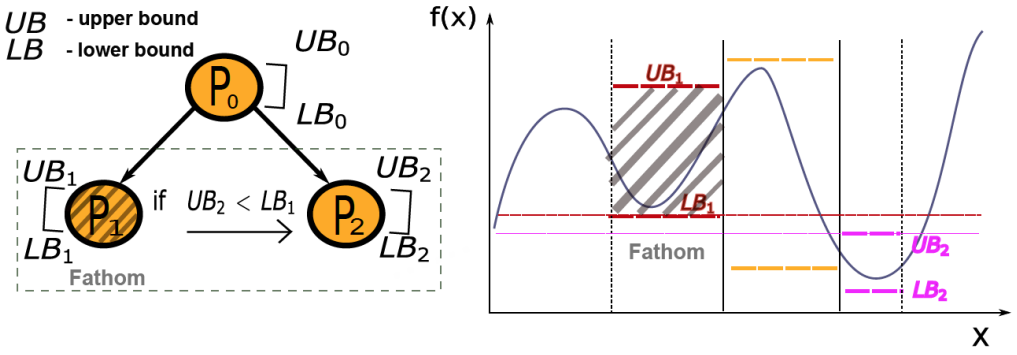


Figure 2.5: A branch-and-bound framework showing the fathoming stage of a minimisation problem.

complexity of this algorithm is known to be exponential even if the algorithm is guaranteed to terminate finitely.

Branch-and-cut (Elf et al. 2001) combines cutting plane method as first proposed by Gomory (1963) with a branch-and-bound algorithm for MILP. The goal of adding valid inequalities to a formulation is to try to get better bounds so that more nodes of the branch-and-bound tree can be fathomed as early as possible. The framework of this algorithm is to solve the relaxation at every subproblem with a small subset of all constraints and check if the optimum solution violates any other of the constraints. If so, append them to the relaxation. This generates a cutting plane method at each node of the enumeration tree that arises by branching on integer variables. Several combinatorial optimisation problems: matching, traveling salesman problem and set packing favour via this solving technique. Branch-and-cut methods are also common in modern MINLP solving (Tawarmalani and Sahinidis 2005, Misener et al. 2015).

Spatial B&B (Smith and Pantelides 1999) and several variations in literature extend this technique for solving nonlinear programs (MINLP) with nonconvex terms (Adjiman et al. 1998, Belotti et al. 2009a). The relaxation process generates convex envelopes as shown in Section 2.4. Branching takes place both on integer and continuous variables and the worst case complexity is also exponential. Such algorithms terminate finitely when and ε tolerance for optimality is specified in advance.

According to Smith and Pantelides (1999), the deterministic Spatial branch-and-bound algorithm divides the problem into subproblems. The algorithm generates a converging sequence of

upper and lower bounds on the optimal solution. The algorithm converges as the gap between the original nonconvex problem and the convex relaxation decreases as the size of variable domain is reduced. This solution is set as an optimal solution of the original problem.

More precisely, consider initially a nonconvex minimisation problem. Any factorable programming problem can be reformulated into a standard form based on binary and unary operations (McCormick 1976). Then a convex relaxation is constructed for each term which results to a convex relaxation for the whole problem.

1. Set a convergence tolerance $\varepsilon > 0$ on the difference between the upper and lower bounds. Initialise the search by setting the best objective value found, a domain that comprises the whole set of variable ranges and a list of regions (depending on the variables) that need to be examined and the domain for each one.

2. *Range tightening*. To accelerate the convergence of the algorithm, the range tightening tries to reduce the interval without changing the optimal value of the problem. Methods include *Optimality based bound tightening* (Liberti 2006), *Feasibility based bound tightening* (Belotti et al. 2009a). There has recently been significant work in the area of making optimality-based bounds tightening more robust (Caprara and Locatelli 2010, Gleixner et al. 2017).

3. *Choice of node*. Generally if it is a minimisation problem choose the one node with the lowest bound and if it is maximisation choose the one with the highest upper bound.

4. *Pruning*. Prune a branch when: (1) the local lower bound is greater than the global upper bound, which suggests that the current branch can not generate better solutions, (2) another branch representing the same solution set has been investigated, then there is no need to further explore, (3) there is no feasible point in the current region.

5. *Objective function upper bound.* Any feasible point can be taken of the original problem. Such points can be obtained by using local optimisation methods. Heuristics can generate a good starting point for this purpose. If a local minimum cannot be achieved or if the objective function upper bound exceeds the best objective value, then proceed to branching. Otherwise, set this as the best found and delete all subregions for which objective function lower bound is greater than the best found. If the upper and lower objective function

values are close enough, delete the subregion and choose a new one.

6. *Objective function lower bound.* Generate a convex relaxation of the original problem and solve the relaxed problem to determine the objective function lower bound.

7. *Branching:* Choose a variable to branch on. Then create two subropoblems and delete the current one. Several heuristics exist for selecting branching variable and value (Belotti et al. 2009a).

8. Delete the subregion from the list and go to step 3.

9. *Evaluation*. Quit when the global upper and lower bound are close enough which depends on ε and set the best objective value found.

B&B algorithms are extremely useful when solving optimisation programming problems to global optimality. However, in several cases such algorithms may not be able to effectively solve large problems due to the exponential number of subproblems and their complexity.

2.5.2 Approximation Algorithms

An alternative method for solving optimisation problems is using heuristics and approximation algorithms. Heuristics can generate good solutions with worst-case running time polynomial to the problem size. A heuristic with a performance guarantee is usually called an *approximation algorithm* (Vazirani 2001, Williamson and Shmoys 2011).

An approximation algorithm is a polynomial algorithm producing a near-optimal solution to an optimisation problem. Formally, consider an optimisation problem, without loss of generality minimisation, and a polynomial Algorithm A for solving it (not necessarily to global optimality). For each problem instance I, let $C_A(I)$ and $C_{OPT}(I)$ be the algorithm's objective value and the optimal objective value, respectively. Algorithm A is ρ-approximate if, for every problem instance I, it holds that:

$$C_A(I) \leq \rho \cdot C_{OPT}(I).$$

That is, a ρ-approximation algorithm computes in polynomial time a solution with an objective value at most ρ times the optimal objective value. The value ρ is the *approximation ratio* of

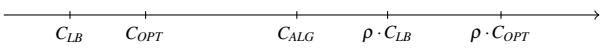


Figure 2.6: Analysis of an Approximation Algorithm

Algorithm A. To prove a ρ-approximation ratio, we proceed as depicted in Figure 2.6. For each problem instance, we compute analytically a lower bound $C_{LB}(I)$ of the optimal objective value, i.e. $C_{LB}(I) \leq C_{OPT}(I)$, and we show that the algorithm's objective value is at most ρ times the lower bound, i.e. $C_A(I) \leq \rho \cdot C_{LB}(I)$. The ratio of a ρ-approximation algorithm is *tight* if the algorithm is not $\rho - \varepsilon$ approximate for any $\varepsilon > 0$. An algorithm is $O(f(n))$-approximate and $\Omega(f(n))$-approximate, where $f(n)$ is a function of an input parameter n, if the algorithm does not have an approximation ratio asymptotically higher and lower, respectively, than $f(n)$.

As shown later in this thesis, heuristics and approximation algorithms are also known to be useful for generating good solutions early in a branch-and-bound tree.

In the following sections we describe symmetry and degeneracy in mathematical programming. Information about such special structures may lead to advances in generating better cuts and improve the efficiency of MILP solvers. Hence investigating the existence and effect of such numerical difficulties in MILP problems is the major goal of this thesis.

2.6 Symmetry in Mathematical Programming

Margot (2010) defines symmetry in Integer Linear Programming (ILP) of the form $P^L = \min_{\mathbf{x} \in \mathbb{Z}^n} \{\mathbf{c}^T \mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$ where $\mathbf{A} \in \mathbb{R}^{m \times n}$ and vectors $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$. Symmetry is the set of variable permutations under which any feasible solution remains feasible and the objective function value is invariant. Let $\mathscr{F}$ be the set of feasible solutions of any problem P^L. The symmetry group is:

$$\mathscr{G}(P^L) = \{\pi \in \Pi^n \mid \forall \quad \hat{\mathbf{x}} \in \mathscr{F}, \quad \pi(\hat{\mathbf{x}}) \in \mathscr{F} \quad \text{and} \quad \mathbf{c}^T \pi(\hat{\mathbf{x}}) = \mathbf{c}^T \hat{\mathbf{x}}\}$$

Liberti (2012a) studies and extends the definition of symmetry to mixed-integer nonlinear optimisation problems.

Symmetric structure in optimisation may be viewed through the lens of group theory for *QCQP* (Bödi et al. 2013, Herr et al. 2013). In many situations though, it is difficult to detect the symmetry of the original problem and its polyhedron representation (Bremner et al. 2014). The formulation group is a subgroup of the symmetry group and reflects the symmetric properties of the variables and the constraints of an optimisation problem. Symmetry handling approaches present methodologies to associate optimisation problems with graph representations from which the graph automorphism is generated by using software tools (Puget 2005, Berthold and Pfetsch 2009, Margot 2010, Liberti 2012b, Bödi et al. 2013, Knueven et al. 2017).

Many researchers exploit the above information and the insights of several problems; covering problems (Margot 2003a, 2007), scheduling and packing problems (Ostrowski et al. 2010, Costa et al. 2013) and engineering problems as the unit commitment problem and heat exchanger network synthesis (Ostrowski et al. 2015, Alemany et al. 2016, Kouyialis and Misener 2017). They identify the presence of symmetry and in some cases propose symmetry handling approaches for problems with known symmetric structure. The improved performance of the solvers validates the efficiency of these techniques (Pfetsch and Rehn 2015). However, they are problem specific and cannot be generalised to other problems.

There are several methods to exploit symmetry which are categorised as static and dynamic methods. Static methods adjoin new constraints to the formulation in order to make some symmetric optima infeasible. Sherali and co-workers add symmetry breaking constraints or perturb the objective function (Sherali and Smith. 2001, Ghoniem and Sherali 2011). Other researchers investigate the orbitopes of a problem (Berthold and Pfetsch 2009, Faenza and Kaibel 2009, Kaibel et al. 2011): convex hull of 0-1 matrices that represent possible solutions to packing and partitioning constraints. The new constraints yield to a reformulation which is guaranteed to keep at least one symmetric optimum feasible. Orbitopes have additionally been considered for cutting planes (Friedman 2007, Hojny and Pfetsch 2015). Liberti (2008) automatically generates symmetry handling inequalities, whereas other works study inequalities which exploit multiple variable orbits (Liberti and Ostrowski 2014, Dias and Liberti 2015); the groups of variables that can be sent to each other under some actions (permutations in the group) which are equivalent with respect to symmetry of the problem.

In the dynamic category are approaches which modify the solution method i.e. the search tree

algorithm to recognise and exploit symmetry dynamically as it goes along. For example, constraints can be derived for each node in the tree to forbid the isomorphic nodes (Gent and Smith 2000, Gent et al. 2005, Ramani and Markov 2005). Another way of exploiting symmetry in B&B is given by isomorphism pruning (Margot 2002, 2003b,a) and orbital/constrained orbital branching (Ostrowski et al. 2008, 2011). By introducing artificial variables, Fischetti et al. (2017) reformulate the problem to a reduced problem which considers only variables of symmetry orbits instead of all variables, so-called orbital shrinking.

While most of these works consider the symmetry representation as a step enclosed by the scope of handling symmetry, Liberti (2012a) is the first who stated the importance of a practical and general representation of symmetry. He uses expression trees to explicitly capture the structure of an optimisation problem and develops the ROSE (Liberti et al. 2010) reformulation software engine that produces a file representation of the problem as *Directed Acyclic Graphs* (*DAG*).

The work introduced in this thesis concerns with the improvements on symmetry detection, which is the first phase of symmetry handling techniques. Symmetry representation is an elementary process given to the software package `nauty`, on which all the following steps to break symmetry depend. Hence it is very essential to guarantee and increase its correctness and efficiency.

2.6.1 Motivation

To isolate this phenomenon we present a prototype circle packing problem. Visually consider a problem of locating two identical circles (c_1, c_2) with centre coordinates (x, y), (x', y') in a unit square. Figure 2.6.1 illustrates this problem. The optimisation problem is to make the circles as large as possible without overlapping. There are four ways to locate these circles and they are related by rotations and reflections. Mathematically speaking, there are four sets of feasible (approximated) solutions which give the same objective value; distance between their centre coordinates (Specht 2018).

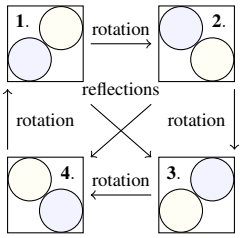


Figure 2.7: Example that shows four different ways of locating two circles in a unit square which lead to the same optimal minimum distance between their centre.

As shown in Figure 2.6.1:

$$1 = \{(0.293, 0.293), (0.707, 0.707)\}$$

$$2 = \{(0.293, 0.707), (0.707, 0.293)\}$$

$$3 = \{(0.707, 0.707), (0.293, 0.293)\}$$

$$4 = \{(0.707, 0.293), (0.293, 0.707)\}$$

However only one can be considered as unique as all the others can be obtained by permuting the variables of the problem. Consider solution 1 and permute variables (yy') to get solution $2 = \{(0.293, 0.707), (0.707, 0.293)\}$, (xx') to get $4 = \{(0.707, 0.293), (0.293, 0.707)\}$, apply both permutations $(yy')(xx')$ to get solution $3 = \{(0.707, 0.707), (0.293, 0.293)\}$. Permutations (xy) and/or $(x'y')$ take solution 1 to itself (Costa et al. 2013). The exchange of the variables of the problem which leaves the set of feasible solutions and the objective function value unaffected is the *symmetry* in an optimisation problems. In a branch-and-bound framework, symmetry is an optimal solution with different configuration leading to the same objective function value. In worst case, *B&B* exhaustively enumerates all feasible solutions. Hence, the presence of symmetry can cause unexpectedly large trees which immediately affects the time that is taken for the algorithm to terminate and the problem to be solved. Hence exploiting symmetry is a challenge. Identifying and classifying problem symmetries is an important step towards exploiting tree-based algorithms such as branch-and-cut. This subsequently allows state-of-the-art solver software to omit symmetric solutions.

2.7 Degeneracy and Multiplicity of Solutions

Degeneracy is a phenomenon that may cause efficiency and convergence problems when solving MILP problems that produce difficult linear subproblems (George and Osborne 1993, Gal 2003) using branch-and-cut algorithms. Roos et al. (1998) characterise an LP model as degenerate if either the primal problem or its dual has multiple optimal solutions. This definition relates the degeneracy of an LP model to the degeneracy of the optimal faces (Tijssen and Sierksma 1998). A theoretical aspect of degeneracy arises when the solving method may repeat a series of iterations and enumerate many equivalent optimal bases by randomly pivoting on variables with zero reduced cost. In linear programming (Charnes 1952), the practical implication of degeneracy is shown within the context of the following examples.

Example 2.1

$$\begin{aligned} \max \quad & 2x_1 + x_2 \\ s.t. \quad & 2x_1 + x_2 \leq 8 \\ & x_1 + x_2 \leq 4 \\ & x_1, x_2 \geq 0 \end{aligned}$$

The practical implication of this condition indicates that the first constraint is redundant i.e. it could be removed without affecting the feasible solution space.

Moreover, there exists multiplicity of solutions that differ in the values of variables, but produce identical values for the objective function leading to duplicate optimal solutions as shown in the next example.

Example 2.2

$$\begin{aligned} \max \quad & 2x_1 + x_2 \\ s.t. \quad & 2x_1 + x_2 \leq 8 \\ & x_1, x_2 \geq 0 \end{aligned}$$

There exists a linear segment such that every solution with different configuration is optimal.

LP degeneracy is classified into primal and dual (Oberdieck et al. 2016). A convex optimisation problem may illustrated as a polyhedron formed by the constrained feasible set and hyperplanes of the objective function. Primal degeneracy is considered as multiple bases defining one vertex of the polyhedron. In this case there is one or more redundant constraints on which the optimal solution lies, which intersects the feasible set without changing it. Hence the active set of constraints at the optimal point is not unique. Dual degeneracy as a complement of primal is explained as a facet of the polyhedron parallel to the objective function. Hence the objective function linearly depends on an active constraint; moving anywhere along that constraint all points are optimal solutions.

Branch-and-bound algorithms solve LP relaxations through general LP solvers. Valid inequalities are usually used to eliminate any infeasible integer points and tighten the LP problems; called cuts (Balas et al. 1996). Pure cutting-plane approaches though might cause numerical difficulties for LP and slow down the convergence of the original MILP. Hence advanced cut generation is integrated with branch-and-bound, leading to branch-and-cut schemes.

Within the context of MILP, Lodi and Tramontani (2013) discuss the performance variability of multiple runs of branch-and-cut solvers. They emphasise the effect of degeneracy associated with linear programming relaxations and identify it as a prominent cause of MILP performance variability (Koch et al. 2011).

Elhallaoui et al. (2011) propose decomposition based methods to address and reduce primal degeneracy in LP models. Stephen et al. (2017) investigate the effect of such techniques for dual degeneracy cases. A fundamental component of this method is to decompose the original problem into a reduced and complimentary one based on degenerate and non-degenerate variables classified by a pivoting rule of Omer et al. (2015). The reduced problem eliminates the degeneracy and further constraints are imposed to both. The algorithm iteratively updates the problems and solves them until a feasible solution is achieved. This method is implemented within the SCIP software solver (Stephen et al. 2017) which uses commercial solvers for dealing with the LP relaxed problem involved. An alternative approach to exploit dual degeneracy is lexicographic optimisation by finding an optimal basis and then generating stable cutting planes. Letsios and Misener (2018) further show that this lexicographic structure is useful for hedging

against uncertainty. Zanette et al. (2011) explain that generating cutting planes comes with the risk of introducing cuts almost parallel to the objective function. The purpose of their work is to use information about degeneracy and choose the best LP solution among the equivalent optima that eventually leads to a practical convergence of their method. Fischetti et al. (2016) following a similar approach aim to collect different LP optimal and generate better cuts via a k-sample method. LP solution polishing algorithms that target improvements on the quality of an existing optimum play a leading role on the final performance of SCIP software that incorporates such algorithms for solving MILP problems. On the same context CPLEX uses algorithms which fixes several variables and try to explore LP solutions to generate better cuts recognising the important and difficulties that arise from degeneracy when solving MILP problems and the need to exploit this phenomenon.

2.7.1 Degeneracy in Mixed-Integer Linear Programming

We abbreviate the interpretation of dual degeneracy and consider a new concept of degeneracy in MILP model when solved via tree search strategies. When the solution method fixes (decides) the integer (decision) variables of a MILP then a degenerate sub-instance of continuous variables remained to be solved.

Definition 2.2 *MILP Degeneracy*
We call MILP with relaxations that are dual degenerate to be MILP degenerate. This is similar to Lodi and Tramontani (2013).

2.8 Summary

To distinguish these special properties, consider a job shop scheduling problem following the formulation and description in Section 2.3. An optimal solution is illustrated in Figure 2.8. The number and type of tasks to be performed in a continuous time formulation of scheduling problems lead to complicate forms of symmetry and degeneracy with infinite equivalent solutions. Maravelias and Grossmann (2004) address such cases and generate integer cuts to exclude previously examined solutions.

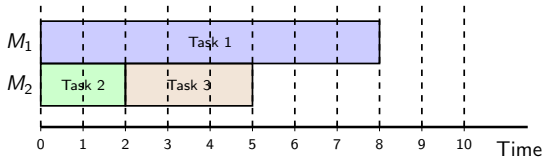


Figure 2.8: Optimal schedule for the problem described in Section 2.8.

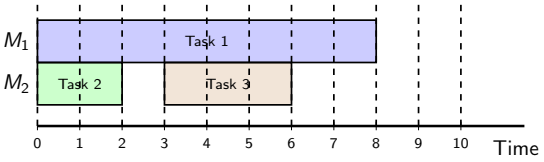


Figure 2.9: Optimal solution representing a degenerate case of this problem.

In the optimal solution, the decision variables allocating each job to each machine have been decided. Due to the time incorporated in this problem (Yee and Shah 1998, Stefansson et al. 2011, Maravelias 2005) more optimal solutions can be generated by shifting the tasks leading to degenerate cases as shown in Figures 2.9, 2.10.

A simple case of symmetry in this problem is caused by the presence of identical machines. For example, we can exchange the machines at which each job is allocated as shown in Figure 2.11.

Furthermore, due to symmetry and degeneracy, more cases of equivalent solutions might appear when solving this problem. i.e. as presented in Figure 2.12.

Summarising the literature of this specific field one could claim that approximation algorithms and heuristics which generate multiple optimal solutions may be favoured by degeneracy and produce important information by exploiting it. This statement nicely links the storyline of

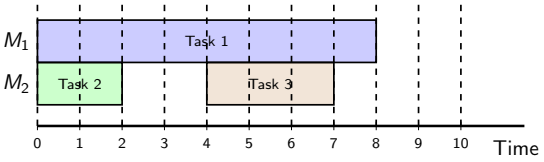


Figure 2.10: Optimal solution representing a degenerate case of this problem.

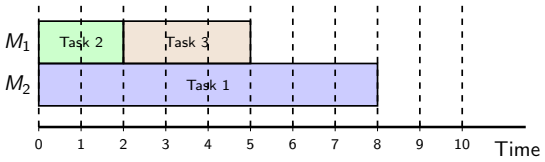


Figure 2.11: An equivalent solution of the problem in figure due to symmetry that arises in the problem from the identical machines.

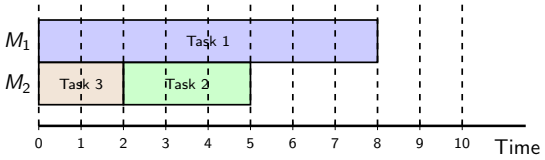


Figure 2.12: Another case that presents an equivalent optimal solution.

this thesis. We investigate the special properties of symmetry and degeneracy in optimisation problems and identify the complexities that arise when state-of-the-art commercial solvers are employed to deal with such cases. Furthermore we develop approximation algorithms and heuristics that not only provide good near-optimal solutions in efficient running times but as a complementary contribution can be practically used to improve the performance of branch-and-cut algorithms via exploiting the above properties.

Chapter 3

Heat Recovery Networks

3.1 Literature Review

Heat recovery is a major component of industrial processes: a quarter of the 2012 European Union energy consumption came from industry and industry uses 73% of this energy on heating and cooling (European Commission 2016). Heat exchanger network synthesis (HENS) minimises cost and improves energy recovery in chemical processes (Biegler et al. 1997, Smith 2000, Elia et al. 2010, Baliban et al. 2012). HENS exploits excess heat by integrating hot and cold process streams and improves energy efficiency by reducing utility usage (Floudas and Grossmann 1987, Gundersen and Naess 1988, Furman and Sahinidis 2002, Anantharaman et al. 2010, Escobar and Trierweiler 2013). Floudas et al. (2012) review the critical role of heat integration for energy systems producing liquid transportation fuels (Niziolek et al. 2015). Other important applications of HENS include: refrigeration systems (Shelton and Grossmann 1986), batch semi-continuous processes (Bancheva et al. 1996, Zhao et al. 1998, Castro et al. 2015, Kong and Shah 2017) and water utilisation systems (Bagajewicz et al. 2002). In their review articles, Furman and Sahinidis (2002) and Escobar and Trierweiler (2013) report two main synthesis approaches: pinch- and optimisation-based methodologies. Optimisation methods automatically generate the best design taking into consideration both the investment and the operation cost (Grossmann 1990). But there are difficulties when we try to model and solve these problems. Heat exchanger network design is a mixed-integer nonlinear optimisa-

tion (MINLP) problem (Yee and Grossmann 1990, Ciric and Floudas 1991, Papalexandri and Pistikopoulos 1994, Hasan et al. 2010) with nonlinear terms including bilinear stream mixing, concave cost functions, and the logarithmic mean temperature difference (LMTD). Approaches to overcome difficulties related to LMTD include using the Paterson (1984) or Chen (1987) approximations. Mistry and Misener (2016) apply strong relaxation methods to approach a global optimum. Further to the *simultaneous* way of solving this problem another approach is the *sequential* method which decomposes the objective function of minimising the overall cost into three simpler task: (1) minimum utility cost, (2) minimum number of matches and (3) minimum investment cost. But HENS remains a difficult problem with many nonconvex nonlinearities. Each possible match between two streams introduces a binary decision variable, so the number of binary variables may grow quadratically with the number of streams. Mathematical symmetry in the problem structure combinatorially increases the possible stream configurations and deteriorates the performance of exact, tree-based algorithms (Kouyialis and Misener 2017). Furman and Sahinidis (2001) proved that HENS is $\mathscr{NP}$ - hard in a strong sense as even the subproblem of the minimum number of matches problem is proved to be $\mathscr{NP}$ - hard due to its combinatorial nature. The standard MILP formulations for the minimum number of matches contain big-M constraints, i.e. the on/off switches associated with weak continuous relaxations of MILP. We choose to study the complexities that arise in the sequential method because each subproblem nicely isolates computational difficulties associated with solving the full simultaneous model; these studies will give us a new handle on approaching simultaneous synthesis.

Moreover, state-of-the-art approaches cannot deal with few dozen stream-problems to global optimality (Chen et al. 2015a) and engineers develop experience-motivated heuristics (Linnhoff and Hindmarsh 1983, Cerda et al. 1983, Furman and Sahinidis 2004, Letsios et al. 2018).

This thesis develops new heuristics and provably efficient approximation algorithms for the minimum number of matches problem. These methods have guaranteed solution quality and efficient run-time bounds.

The chapter explains the two main optimisation approaches in Section 3.2. Section 3.3 defines the parameters of the general heat exchanger network design required for minimising utility cost. Next, we present a way of obtaining temperature intervals and Section 3.4 presents the LP for minimising utility cost. Section 3.5 describes the minimum number of matches problem and

Section 3.6 defines the minimum investment cost problem.

3.2 Optimisation Approaches for Heat Exchanger Network Synthesis

Solving the HENS *simultaneously*, i.e. generating the optimal network without decomposition (Furman and Sahinidis 2002), requires a mixed-integer nonlinear programming (MINLP) formulation to account for stream mixing and the nonlinear nature of heat exchange (Yee and Grossmann 1990, Ciric and Floudas 1991, Papalexandri and Pistikopoulos 1994). Mistry and Misener (2016) recently showed that expressions incorporating logarithmic mean temperature difference, i.e. the nonlinear nature of heat exchange, may be reformulated to decrease the number of nonconvex nonlinear terms in the optimisation problem. There are several formulations proposed, with the simultaneous method for minimising the running cost being the one that can guarantee global solution to such problem and MINLP SYNHEAT model by (Yee and Grossmann 1990), and MINLP model proposed by (Ciric and Floudas 1991) being widely used. An alternative approach and a way to generate good HENS solutions is to use the so-called *sequential method* (Furman and Sahinidis 2002).The sequential method decomposes the original HENS MINLP into three tasks: (i) minimising utility cost, (ii) minimising the number of matches, and (iii) minimising the investment cost. This method optimises the three mathematical models sequentially with: (i) a linear program (LP) (Cerda et al. 1983, Papoulias and Grossmann 1983), (ii) a mixed-integer linear program (MILP) (Cerda and Westerberg 1983, Papoulias and Grossmann 1983), and (iii) a nonlinear program (NLP) (Floudas et al. 1986). The sequential method is less computationally difficult than the simultaneous method, but the sequential method cannot guarantee global optimality of the original problem.

3.3 General Heat Exchanger Network Design

An instance of the general heat exchanger network design consists of a set $HS = \{1,2,\ldots,ns\}$ of hot streams, a set $CS = \{1,2,\ldots,ms\}$ of cold streams, a set $HU = \{1,2,\ldots,nu\}$ of hot utilities and a set $CU = \{1,2,\ldots,mu\}$ of cold utilities. Each hot stream $i \in HS$ (cold stream $j \in CS$)

has initial inlet, outlet temperatures $T^{HS}_{\text{in},i}$, $T^{HS}_{\text{out},i}$ (resp. $T^{CS}_{\text{in},j}$, $T^{CS}_{\text{out},j}$) and flowrate heat capacity FCp_i (resp. FCp_j). Each hot utility $i \in HU$ (cold utility $j \in CU$) is associated with inlet, outlet temperatures $T^{HU}_{\text{in},i}$, $T^{HU}_{\text{out},i}$ (resp. $T^{CU}_{\text{in},j}$, $T^{CU}_{\text{out},j}$) and a cost κ^{HU}_i (resp. κ^{CU}_j).

Table 3.1: Minimum Utility Cost Notation

Name	Description
Cardinalities, Indices, Sets	
ns, ms	Number of hot, cold streams
nu, mu	Number of hot, cold utilities
k	Number of temperature intervals
$i \in HS \cup HU$	Hot stream, utility
$j \in CS \cup CU$	Cold stream, utility
$t \in TI$	Temperature interval
HS, CS	Set of hot, cold streams
HU, CU	Set of hot, cold utilities
TI	Set of temperature intervals
Parameters	
FCp_i, FCp_j	Flowrate heat capacity of hot stream i, cold stream j
$T^{HS}_{\text{in},i}, T^{HS}_{\text{out},i}$	Inlet, outlet temperature of hot stream i
$T^{CS}_{\text{in},j}, T^{CS}_{\text{out},j}$	Inlet, outlet temperature of cold stream j
$T^{HU}_{\text{in},i}, T^{HU}_{\text{out},i}$	Inlet, outlet temperature of hot utility i
$T^{CU}_{\text{in},j}, T^{CU}_{\text{out},j}$	Inlet, outlet temperature of cold utility j
$\Delta T_{\min}$	Minimum heat recovery approach temperature
$\kappa^{HU}_i, \kappa^{CU}_j$	Unitary cost of hot utility i, cold utility j
$\sigma^{HS}_{i,t}$	Heat supply of hot stream i in interval t
$\delta^{CS}_{j,t}$	Heat demand of cold stream j in interval t
Variables	
$\sigma^{HU}_{i,t}$	Heat supply of hot utility i in interval t
$\delta^{CU}_{j,t}$	Heat demand of cold utility j in interval t
R_t	Residual heat exiting temperature interval t

3.3.1 Temperature Intervals

The sequential method begins by computing a set $TI = \{1, 2, \ldots, k\}$ of k temperature intervals (Linnhoff and Flower 1978, Ciric and Floudas 1989). A minimum heat recovery approach temperature $\Delta T_{\min}$ specifies the minimum temperature difference between two streams exchanging heat. In order to incorporate $\Delta T_{\min}$ in the problem's setting, we enforce that each temperature interval corresponds to a temperature range on the hot stream side shifted up by $\Delta T_{\min}$ with respect to to its corresponding temperature range on the cold stream side. Let TI^H and TI^C be the temperature intervals on the hot and cold side, respectively. Consider, on the hot side, all $k+1$ discrete temperature values $T_1 > T_2 > \ldots > T_{k+1}$ belonging to the set $\{T^{HS}_{\text{in},i} : i \in HS\} \cup \{T^{HU}_{\text{in},i} : i \in HU\} \cup \{T^{CS}_{\text{in},j} + \Delta T_{\min} : j \in CS\} \cup \{T^{CU}_{\text{in},j} + \Delta T_{\min} : j \in CU\}$. Then, we define $TI^H = \bigcup_{t=1}^{k} \{[T_t, T_{t+1}]\}$ and $TI^C = \cup_{t=1}^{k} \{[T_t - \Delta T_{\min}, T_{t+1} - \Delta T_{\min}]\}$. We set $TI = TI^H$ and we observe that TI^C contains exactly the same temperature intervals with TI shifted by $\Delta T_{\min}$. Moreover, we set $\Delta T_t = T_t - T_{t+1}$, for $t \in TI$.

For each temperature interval $t \in TI$, we are now able to compute the quantity $\sigma^{HS}_{i,t}$ of heat load exported by hot stream $i \in HS$ as well as the amount $\delta^{CS}_{j,t}$ of heat load received by cold stream $j \in CS$ in $t \in TI$. Specifically, for each $i \in HS$ and $t \in TI$, we set

$$\sigma^{HS}_{i,t} = \begin{cases} FCp_i \cdot \Delta T_t, & \text{if } T^{HS}_{\text{in},i} \geq T_t \text{ and } T^{HS}_{\text{out},i} \leq T_{t+1} \\ FCp_i \cdot (T_t - T^{HS}_{\text{out},i}), & \text{if } T^{HS}_{\text{in},i} \geq T_t \text{ and } T^{HS}_{\text{out},i} > T_{t+1} \\ 0, & \text{if } T^{HS}_{\text{in},i} < T_t \end{cases}$$

Similarly, for each $j \in CS$ and $t \in TI$,

$$\delta^{CS}_{j,t} = \begin{cases} FCp_j \cdot \Delta T_t, & \text{if } T^{CS}_{\text{in},j} \leq T_{t+1} - \Delta T_{\min} \text{ and } T^{CS}_{\text{out},j} \geq T_t - \Delta T_{\min} \\ FCp_j \cdot (T^{CS}_{\text{out},j} - (T_{t+1} - \Delta T_{\min})), & \text{if } T^{CS}_{\text{in},j} \leq T_{t+1} - \Delta T_{\min} \text{ and } T^{CS}_{\text{out},j} < T_t - \Delta T_{\min} \\ 0, & \text{if } T^{CS}_{\text{in},j} > T_{t+1} - \Delta T_{\min} \end{cases}$$

3.4 Minimum Utility Cost

This problem is solved in order to compute the minimum amount of utility heat so that there is heat balance in the network. For each hot utility $i \in HU$ and cold utility $j \in CU$ the continuous variables $\sigma_{i,t}^{HU}$ and $\delta_{j,t}^{CU}$ correspond to the amount of heat of i and j, respectively, in temperature interval t. The LP uses a heat residual variable R_t, for each $t \in TI$. Let TI_i be the set of temperature intervals to which hot utility $i \in HU$ can transfer heat, feasibly. Similarly, let TI_j be the set of temperature intervals from which cold utility $j \in CU$ can receive heat. The minimum utility cost problem can be solved by using the following LP model (see Cerda et al. (1983), Papoulias and Grossmann (1983)).

$$\min \sum_{i \in HU} \sum_{t \in TI} \kappa_i^{HU} \cdot \sigma_{i,t}^{HU} + \sum_{j \in CU} \sum_{t \in TI} \kappa_j^{CU} \cdot \delta_{j,t}^{CU} \tag{3.1}$$

$$\sum_{i \in HS} \sigma_{i,t}^{HS} + \sum_{i \in HU} \sigma_{i,t}^{HU} + R_t = \sum_{j \in CS} \delta_{j,t}^{CS} + \sum_{j \in CU} \delta_{j,t}^{CU} + R_{t+1} \quad t \in TI \tag{3.2}$$

$$R_1, R_{k+1} = 0 \tag{3.3}$$

$$\sigma_{i,t}^{HU} = 0 \qquad i \in HU, t \in TI \setminus TI_i \tag{3.4}$$

$$\delta_{j,t}^{CU} = 0 \qquad j \in CU, t \in TI \setminus TI_j \tag{3.5}$$

$$\sigma_{i,t}^{HU}, \delta_{j,t}^{CU}, R_t \geq 0 \qquad i \in HU, j \in CU, t \in TI \tag{3.6}$$

Expression 3.1 minimises the utility cost. Constraints 3.2 and 3.3 ensure energy balance. Constraints 3.4 and 3.5 ensure that heat flows from a temperature interval to the same or a lower temperature interval.

3.5 Minimum Number of Matches

Given an optimal solution of the minimum utility cost problem, we obtain an instance of the minimum number of matches problem as follows. All utilities are considered as streams, i.e. $H = HS \cup HU$, $C = CS \cup CU$, $n = ns + nu$ and $m = ms + mu$. Furthermore, $T = TI$. Finally, for each $i \in H$ and $t \in T$ the parameter $\sigma_{i,t}$ is equal to $\sigma_{i,t}^{HS}$ or $\sigma_{i,t}^{HU}$ depending on whether i was originally a hot stream or utility. The parameters $\delta_{j,t}$ are obtained similarly, for each $j \in C$ and

$t \in T$.

This section defines the minimum number of matches problem and analytically represents the heat exchanger network synthesis problem using the standard transportation and transshipment MILP models. Table 4.1 contains the notation.

3.5.1 Problem Definition

The minimum number of matches problem posits a set of *hot process streams* to be cooled and a set of *cold process streams* to be heated. Each stream is associated with an initial and a target temperature. This set of temperatures defines a collection of *temperature intervals*. Each hot stream exports (or supplies) heat in each temperature interval between its initial and target temperatures. Similarly, each cold stream receives (or demands) heat in each temperature interval between its initial and target temperatures. Subsection 3.3.1 formally defines the temperature range partitioning. Heat may flow from a hot to a cold stream in the same or a lower temperature interval, but not in a higher one. In each temperature interval, the *residual heat* descends to lower temperature intervals. A zero heat residual is a *pinch point*. A pinch point restricts the maximum energy integration and divides the network into subnetworks.

A problem instance consists of a set $H = \{1,2,\ldots,n\}$ of hot streams, a set $C = \{1,2,\ldots,m\}$ of cold streams, and a set $T = \{1,2,\ldots,k\}$ of temperature intervals. Hot stream $i \in H$ has heat supply $\sigma_{i,s}$ in temperature interval $s \in T$ and cold stream $j \in C$ has heat demand $\delta_{j,t}$ in temperature interval $t \in T$. Heat conservation is satisfied, i.e. $\sum_{i\in H}\sum_{s\in T}\sigma_{i,s} = \sum_{j\in C}\sum_{t\in T}\delta_{j,t}$. We denote by $h_i = \sum_{s\in T}\sigma_{i,s}$ the total heat supply of hot stream $i \in H$ and by $c_j = \sum_{t\in T}\delta_{j,t}$ the total heat demand of cold stream $j \in C$.

A feasible solution specifies a way to transfer the hot streams' heat supply to the cold streams, i.e. an amount $q_{i,s,j,t}$ of heat exchanged between hot stream $i \in H$ in temperature interval $s \in T$ and cold stream $j \in C$ in temperature interval $t \in T$. Heat may only flow to the same or a lower temperature interval, i.e. $q_{i,s,j,t} = 0$, for each $i \in H$, $j \in C$ and $s,t \in T$ such that $s > t$. A hot stream $i \in H$ and a cold stream $j \in C$ are *matched*, if there is a positive amount of heat exchanged between them, i.e. $\sum_{s,t\in T} q_{i,s,j,t} > 0$. The objective is to find a feasible solution minimising the number of matches (i, j).

3.5.2 Mathematical Models

The minimum number of matches problem is a mixed-integer linear program (MILP) based on transportation and transshipment models.

3.5.2.1 Transportation Model (Cerda and Westerberg 1983)

As illustrated in Figure 3.1a, the transportation model represents heat as a commodity transported from supply nodes to destination nodes. For each hot stream $i \in H$, there is a set of supply nodes, one for each temperature interval $s \in T$ with $\sigma_{i,s} > 0$. For each cold stream $j \in C$, there is a set of demand nodes, one for each temperature interval $t \in T$ with $\delta_{j,t} > 0$. There is an arc between the supply node (i,s) and the destination node (j,t) if $s \leq t$, for each $i \in H$, $j \in C$ and $s,t \in T$.

In the MILP formulation, variable $q_{i,s,j,t}$ specifies the heat transferred from hot stream $i \in H$ in temperature interval $s \in T$ to cold stream $j \in C$ in temperature interval $t \in T$. Binary variable $y_{i,j}$ whether streams $i \in H$ and $j \in C$ are matched or not. Parameter $U_{i,j}$ is a big-M parameter bounding the amount of heat exchanged between every pair of hot stream $i \in H$ and cold stream $j \in C$, e.g. $U_{i,j} = \min\{h_i, c_j\}$. The problem is formulated:

$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j} \tag{3.7}$$

$$\sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{i,s} \qquad i \in H, s \in T \tag{3.8}$$

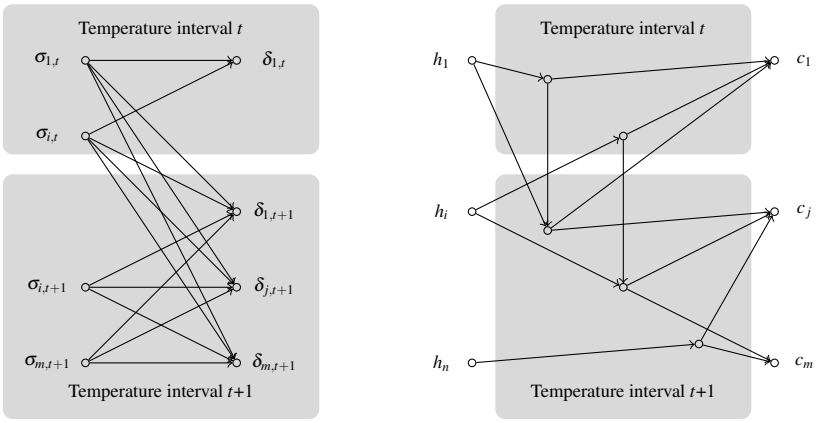
$$\sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{j,t} \qquad j \in C, t \in T \tag{3.9}$$

$$\sum_{s,t \in T} q_{i,s,j,t} \leq U_{i,j} \cdot y_{i,j} \qquad i \in H, j \in C \tag{3.10}$$

$$q_{i,s,j,t} = 0 \qquad i \in H, j \in C, s,t \in T : s \leq t \tag{3.11}$$

$$y_{i,j} \in \{0,1\}, q_{i,s,j,t} \geq 0 \qquad i \in H, j \in C,\ s,t \in T \tag{3.12}$$

Expression (3.7), the objective function, minimises the number of matches. Equations (3.8) and (3.9) ensure heat conservation. Equations (3.10) enforce a match between a hot and a cold stream if they exchange a positive amount of heat. Equations (3.10) are *big-M constraints*. Equations (3.11) ensure that no heat flows to a hotter temperature.



(a) Transportation Model (b) Transshipment Model

Figure 3.1: In this model (Cerda and Westerberg 1983), each hot stream i supplies $\sigma_{i,t}$ units of heat in temperature interval t which can be received, in the same or a lower temperature interval, by a cold stream j which demands $\delta_{j,t}$ units of heat in t. Following the transshipment model (Papoulias and Grossmann 1983), there are also intermediate nodes transferring residual heat to a lower temperature interval. This figure is adapted from Furman and Sahinidis (2004).

3.5.2.2 Transshipment Model (Papoulias and Grossmann 1983)

As illustrated in Figure 3.1b, the transshipment formulation transfers heat from hot streams to cold streams via intermediate transshipment nodes. In each temperature interval, the heat entering a transshipment node either transfers to a cold stream in the same temperature interval or it descends to the transshipment node of the subsequent temperature interval as residual heat.

Binary variable $y_{i,j}$ is 1 if hot stream $i \in H$ is matched with cold stream $j \in C$ and 0 otherwise. Variable $q_{i,j,t}$ represents the heat received by cold stream $j \in C$ in temperature interval $t \in T$ originally exported by hot stream $i \in H$. Variable $r_{i,s}$ represents the residual heat of hot stream $i \in H$ that descends from temperature interval s to temperature interval $s+1$. Parameter $U_{i,j}$ is a big-M parameter bounding the heat exchanged between hot stream $i \in H$ and cold stream $j \in C$, e.g. $U_{i,j} = \min\{h_i, c_j\}$. The problem is formulated:

$$\min \sum_{i \in H} \sum_{j \in C} y_{i,j} \tag{3.13}$$

$$\sum_{j \in C} q_{i,j,s} + r_{i,s} = \sigma_{i,s} + r_{i,s-1} \qquad i \in H, s \in T \tag{3.14}$$

$$r_{i,k} = 0 \qquad i \in H \tag{3.15}$$

$$\sum_{i \in H} q_{i,j,t} = \delta_{j,t} \qquad j \in C, t \in T \tag{3.16}$$

$$\sum_{t \in T} q_{i,j,t} \leq U_{i,j} \cdot y_{i,j} \qquad i \in H, j \in C \tag{3.17}$$

$$y_{i,j} \in \{0,1\},\ q_{i,j,t}, r_{i,s} \geq 0 \qquad i \in H, j \in C, s, t \in T \tag{3.18}$$

Expression (3.13) minimises the number of matches. Equations (3.14)-(3.16) enforce heat conservation. Equation (3.17) allows positive heat exchange between hot stream $i \in H$ and cold stream $j \in C$ only if (i, j) are matched.

3.6 Minimum Investment Cost Superstructure

The last problem solved sequentially is a nonlinear program that develops the heat exchanger network with a minimum investment cost. The networks may involve stream splitting, mixing, and bypassing. The LP transshipment model provides the corresponding heat duties of the utilities and the MILP transshipment model the matches that take place for the process streams and utilities indexed by the set $MA = \{(i, j)\}$ hot stream/ utility i exchanges heat with cold stream/ utility j with heat exchanged $Q_{i,j} = \sum_{t \in T} q_{i,j,t}$ for $i \in H$, $j \in C$. This derivation of the stream superstructure relies on the optimal area $(A = Q/U.LMTD)$ of the exchangers subject to stream interconnections, heat load distribution and temperature of each stream.

Table 3.2: Minimum investment cost superstructure

Name	Description
Cardinalities, Indices, Sets	
$p \in HC$	Hot and cold stream, utility
$l \in N_p$	Streams involved in the superstructure of each stream p
$s \in S_p$	Splitter stream
s_0	Initial splitting point in the streams superstructure p
$m \in M_p$	Mixer stream
m_o	Final mixing point in the streams superstructure p

HC	Set of hot and cold stream, utility
N_p	Set of streams involved with stream p
S_p	Set of splitter streams
M_p	Set of mixed streams
$S_p^{\text{in}}(s), S_p^{\text{out}}(s) = \{l \mid l \in N_p\}$	Set of relation of inlet/outlet stream to/from splitter $s \in S_p$
$M_p^{\text{in}}(m), M_p^{\text{out}}(m) = \{m \mid m \in N_p\}$	Set of relation of inlet/outlet stream to/from mixer $m \in M_p$
$E^H_{\text{in},(i,j)} = \{\eta \mid \eta \in N_i\}$	Inlet stream of hot stream i to unit $(i,j) \in MA$
$E^H_{\text{out},(i,j)} = \{\nu \mid \nu \in N_i\}$	Outlet stream of hot stream i to unit $(i,j) \in MA$
$E^C_{\text{in},(i,j)} = \{\mu \mid \mu \in N_j\}$	Inlet stream of cold stream j to unit $(i,j) \in MA$
$E^C_{\text{out},(i,j)} = \{\rho \mid \rho \in N_j\}$	Outlet stream of cold stream j to unit $(i,j) \in MA$
$F_p = \{F_i^H, F_j^C\}$	Set of flowrate heat capacities of streams i, j
$T_p^{\text{in}} = \{T^H_{\text{in},i}, T^C_{\text{in},j}\}$	Inlet temperature
$T_p^{\text{out}} = \{T^H_{\text{out},i}, T^C_{\text{out},j}\}$	Outlet temperature

Parameters

$Q_{i,j}$	Heat exchanged for each match $(i,j) \in MA$
$b_{i,j}$	Cost exponent
$c_{i,j}$	Cost coefficient
ΔH^H, ΔH^C	Specific enthalpy of hot, cold utility
$U_{i,j}$	Overall heat transfer coefficient

Variables

$A_{i,j}$	Area of each exchanger for match $(i,j) \in MA$
f_l^p	Variable of flowrate heat capacity
t_l^p	Variable temperature
$\text{LMTD}_{i,j}$	Log mean temperature difference for match $(i,j) \in MA$

Definition of LMTD

$dt^1_{i,j} = t^i_\eta - t^j_\mu$	Minimum temperature approach
$dt^2_{i,j} = t^i_\nu - t^j_\rho$	Minimum temperature approach
$\text{LMTD}_{i,j} = \frac{dt^1_{i,j} - dt^2_{i,j}}{\ln \frac{dt^1_{i,j}}{dt^2_{i,j}}}$	For $i \in HS$, $j \in CS$

Specifications

$f_l^p = F_p$	Flowrate heat capacity for $l \in S_p^{\text{in}}(s_0)$, $p \in HS \cup CS$
$t_l^p = T_p^{\text{in}}$	Inlet temperature for $l \in S_p^{\text{in}}(s_0)$, $p \in HC$
$t_l^p = T_p^{\text{out}}$	Outlet temperature for $l \in M_p^{\text{out}}(m_0)$, $p \in HC$

The problems is formulated (Floudas et al. 1986):

$$\min \sum_{(i,j)\in MA} c_{i,j} A_{i,j}^{b_{i,j}} \tag{3.19}$$

$$\sum_{l\in S_p^{\text{in}}(s)} f_l^p - \sum_{l\in S_p^{\text{out}}(s)} f_l^p = 0 \qquad s \in S_p, p \in HC \tag{3.20}$$

$$\sum_{l\in M_p^{\text{in}}(m)} f_l^p - \sum_{l\in M_p^{\text{out}}(m)} f_l^p = 0 \qquad m \in M_p, p \in HC \tag{3.21}$$

$$\sum_{l\in M_p^{\text{in}}(m)} f_l^p t_l^p - \sum_{l\in M_p^{\text{out}}(m)} f_l^p t_l^p = 0 \qquad m \in M_p, p \in HC \tag{3.22}$$

$$Q_{i,j} - f_l^i (t_\eta^i - t_\nu^i) = 0 \qquad \eta \in E^H_{\text{in},(i,j)}, \nu \in E^H_{\text{out},(i,j)}, i \notin HU', (i,j) \in MA \tag{3.23}$$

$$Q_{i,j} f_l^i \Delta H_i^H = 0 \qquad \eta \in E^H_{\text{in},(i,j)}, \nu \in E^H_{\text{out},(i,j)}, i \in HU', (i,j) \in MA \tag{3.24}$$

$$Q_{i,j} - f_l^j (t_\mu^j - t_\rho^j) = 0 \qquad \mu \in E^C_{\text{out},(i,j)}, \rho \in E^C_{\text{in},(i,j)}, j \notin CU', (i,j) \in MA \tag{3.25}$$

$$Q_{i,j} f_l^j \Delta H_j^C = 0 \qquad \mu \in E^C_{\text{out},(i,j)}, \rho \in E^C_{\text{in},(i,j)}, j \in CU', (i,j) \in MA \tag{3.26}$$

$$t_\eta^i - t_\mu^j \geq \Delta T_{min}, \qquad \eta \in E^H_{\text{in},(i,j)}, \mu \in E^C_{\text{out},(i,j)}, (i,j) \in MA \tag{3.27}$$

$$t_\nu^i - t_\rho^j \geq \Delta T_{min}, \qquad \nu \in E^H_{\text{out},(i,j)}, \rho \in E^C_{\text{in},(i,j)}, (i,j) \in MA \tag{3.28}$$

$$t_l^p = t_\nu^k \qquad l \in S_p^{\text{in}}(s), \nu \in S_p^{\text{out}}(s), s \in S_p, p \in HC \tag{3.29}$$

$$f_l^p \geq 0 \qquad l \in N_p, p \in HC \tag{3.30}$$

$$A_{i,j} = Q_{i,j} U_{i,j}^{-1} (LMTD)_{i,j}^{-1} \tag{3.31}$$

Expression (3.19) minimises the investment cost based on the coefficients and the areas. Equations (3.20) and (3.21) ensure mass conservation where equations (3.22-3.26) ensure heat conservation. Equations (3.27, 3.28) for minimum temperature approaches constraints. Equations (3.29) ensure equalities for inlet and outlet temperatures of splits and equations 3.30 ensure that flowrate heat capacity variables are nonnegative. Finally the area of each exchanger is expressed

via equation 3.31 in terms of the given heat loads and the log mean temperature difference for each match of streams (i, j).

Chapter 4

Data Structures for Representing Symmetry in Quadratically Constrained Quadratic Programs

4.1 Introduction

Symmetry in mathematical programming may lead to a multiplicity of solutions. In nonconvex optimisation, it can negatively affect the performance of the branch-and-bound algorithm. Symmetry may induce large search trees with multiple equivalent solutions, i.e. with the same optimal value. Dealing with symmetry requires detecting and classifying it first. Symmetries of a mathematical program are classified by Margot (2010) and Liberti (2012a) as the *symmetry* and the *formulation* group. This chapter develops methods for detecting groups of symmetry in the formulation of quadratically constrained quadratic optimisation problems via adjacency matrices. Using graph theory, we transform these matrices into *Binary Layered Graphs (BLG)* and enter them into the software package `nauty` (McKay and Piperno 2014). `Nauty` generates important symmetric properties of the original problem.

This chapter proceeds as follows: Section 4.2 formally defines symmetry in quadratically con-

Table 4.1: Table of Notation.

Symbol	Description	Symbol	Description
x_i	Variables	I	Identity element
$\mathbf{x}$	Vectors of variables	π, σ	Permutations
α	Coefficient	Π^n	Set of all permutations
$\mathbf{c}, \mathbf{b}, \mathbf{p}$	Vectors of parameters	S_n	Symmetric group order n
$\mathbf{A}, \mathbf{Q}$	Matrices of parameters	$\mathbf{Y}$	Sets
$\mathbf{X}$	Matrix of auxiliary variables	f, h, ϕ	Functions
M, IM, JM, KM	Sparse representations of matrices	$\mathbf{G}, \mathbf{H}$	Graphs
$\mathscr{F}$	Set of feasible solutions	E, V	Set of edges, vertices
$\mathscr{G}, \tilde{\mathscr{G}}$	Symmetry groups	e	Edges in the graph
W, Z	Groups	u, v	Nodes in the graph
s	Number of nonzero elements	ℓ	Number of unique coefficients
i, j, k, r	Indices	L	Number of layers

strained quadratic optimisation problems and identifies the role of integrality and nonconvexity in such cases. Section 4.3 evolves around the formulation symmetries in optimisation problems and the graph structures that currently exist in literature for detecting such symmetries. Section 4.4 suggests two different methods on forming a problem as an adjacency matrix and explains how to convert these matrices into graphs. Section 4.5 introduces *binary layered graphs* and describes how to use the proposed matrices and construct these graphs. Section 4.6 shows how to automatically detect symmetry using software package `nauty`. This work concludes in Section 4.7 with a discussion on the proposed structures and comparison to other methods.

4.2 Symmetry Group of Quadratically Constrained Quadratic Programs

A Quadratically Constrained Quadratic program *QCQP* is a subclass of *MIQCQP*.

Definition 4.1

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} \quad & f_0(\mathbf{x}) \\ s.t. \quad & f_k(\mathbf{x}) \leq 0 \qquad \forall k = 1,\ldots,m \\ & x_i \in [x_i^L, x_i^U] \quad \forall i = 1,\ldots,n \end{aligned} \tag{QCQP}$$

where

$$f_k(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i \alpha_{ij}^k x_j + \sum_{i=1}^{n} \alpha_{i0}^k x_i + \alpha_{00}^k \; \forall k = 0\ldots,m \tag{4.1}$$

with coefficients $\alpha_{ij}^k \in \mathbb{R}$ *for* $i \in \{0,\ldots,n\}$, $j = \{0,\ldots,n\}$ *and* $k \in \{0,1,\ldots,m\}$ *for* $x_i \in [x_i^L, x_i^U]$, $i \in \{1,\ldots,n\}$.

After surveying the available sources for detecting symmetry, we contemplate the explanation of symmetry given by Margot (2010) which is also presented by Liberti (2012a) on different problems. Under a set of permutations of the problem variables, each feasible solution can be mapped to another solution having the same value and the whole set of feasible solutions $\mathscr{F}$ can be mapped to itself.

Modifying this definition to the case of quadratic problems we define symmetry in QCQP:

Definition 4.2 *Symmetry group*

$$\tilde{\mathscr{G}}(P^Q) = \{\pi \in \Pi^n \mid \forall \quad \hat{\mathbf{x}} \in \mathscr{F}, \quad \pi(\hat{\mathbf{x}}) \in \mathscr{F} \quad \textit{and} \quad f_0(\pi(\hat{\mathbf{x}})) = f_0(\hat{\mathbf{x}})\}$$

The symmetry group is based on the feasible set of solutions of an optimisation problem. Deriving this set is impractical in our work. Hence, the scope of this chapter is to efficiently associate data structures with optimisation problems which can generate the formulation group: a set of permutations that fix the problem formulation. Liberti (2012a) proves that the formulation group is a subset of the symmetry group. Definitions and relevant structures of this group are provided in the next section.

This work discusses the symmetry in nonlinear QCQPs, as well as the symmetry in linearised cases of *LQP* (formulation in Chapter 2, Section 2.4) which arise after applying the RLT to the QCQP formulation.

Relaxing the problem using the RLT technique adds constraints which may alter the symmetric structure of the problem. We evaluate and state how the integrality and nonlinearities affect the symmetry group of the original problem. The following examples show that another challenging aspect of detecting symmetry is the fact that symmetry in the original problem does not imply the same symmetry in the relaxed problem and vice versa.

Example 4.1

$$\begin{aligned} \min \quad & x_1 + x_2 \\ s.t. \quad & 2x_1 + x_2 \leq 2 \\ & x_1, x_2 \in \{0,1\} \end{aligned}$$

The set of feasible solutions is $\{(0,0),(1,0),(0,1)\}$. Hence, the symmetry group is characterised by permutation $\pi = (x_1, x_2)$ under which any feasible solution remains within the set of feasible solutions and the objective function value is invariant. On the other hand, if we relax the integrality constraints over a continuous range $x, y \in [0,1]$ the feasible solution $(0.5,1)$ under permutation $\pi(0.5,1) = (1,0.5)$ violates the linear constraint. Hence, if the integrality restrictions in a MIQCQP are relaxed and the symmetry group is defined over the feasible set of solutions, then it is not necessarily a subgroup of the symmetry group in the relaxation. The next example shows that the symmetry group of the relaxation is also not a subgroup of the symmetry group in the original problem.

Example 4.2

$$\begin{aligned} \min \quad & x_1 + x_2 \\ s.t. \quad & x_1 + x_2 \leq 1 \\ & x_1 \in [0,1] \\ & x_2 \in \{0,1\} \end{aligned}$$

Similar to Example 4.1 permutation $\pi = (x_1, x_2)$ is the symmetry of the relaxed problem, but solution $(0.5,1)$ is not feasible under this permutation for the original problem.

Another example shows that the original optimisation problem might not inherit any symmetry. But under relaxation, e.g. McCormick, symmetries arise.

Example 4.3

$$\begin{aligned} \min \quad & -x_1 - x_2^2 \\ s.t. \quad & x_2 + x_3 \geq 1 \\ & x_1 \geq 2x_3 - 1 \\ & x_1 \leq x_3 \\ & x_1, x_2, x_3 \in \{0,1\} \end{aligned} \qquad (QP)$$

$$\begin{aligned} \min \quad & -x_1 - x_4 \\ s.t. \quad & x_2 + x_3 \geq 1 \\ & x_1 \geq 2x_3 - 1 \\ & x_1 \leq x_3 \\ & x_4 \geq 2x_2 - 1 \\ & x_4 \leq x_2 \\ & x_1, x_2, x_3, x_4 \in \{0,1\} \end{aligned} \qquad (LQP)$$

For the symmetry group the feasible set of solutions:
$\mathscr{F}(QP) = \{(1\,1\,1), (0\,1\,0), (1\,0\,1)\}$ with $\tilde{\mathscr{G}}(QP) = \{I\}$.
$\mathscr{F}(LQP) = \{(1\,1\,1\,1), (1\,0\,1\,0), (0\,1\,0\,1)\}$ with
$\tilde{\mathscr{G}}(LQP) = \{I, (x_1\,x_3), (x_2\,x_4), (x_1\,x_3)(x_2\,x_4), (x_1\,x_2)(x_3\,x_4), (x_1\,x_4)(x_2\,x_3)\}$ hence, $\mathscr{G}(LPQ) \not\leq \mathscr{G}(QP)$.

4.3 Formulation Symmetry Detection via Directed Acyclic Graphs

The formulation group of a mathematical optimisation problem is defined by Liberti (2012b) as the set of permutations of the variable indices for which the objective function and the constraints are the same. Hence, for *QCQP*:

Definition 4.3 *Formulation group of QCQP*

$$\mathscr{G}(P^Q) = \{\pi \in \Pi^n \mid \quad f_0(\pi(\mathbf{x})) = f_0(\mathbf{x}) \quad \textit{and} \quad \exists \quad \sigma \in \Pi^m (\sigma f_k(\pi(\mathbf{x})) = f_k(\mathbf{x}))\}$$

4.3.1 Expression Trees

To compare two functions, Liberti (2012b) suggests to compare their expression trees. An expression tree as first introduced by Crawford et al. (1996) and explained by Ramani and Markov (2005) for Constrained Programming (CP) is used to represent algebraic functions, since it can visually present the structural relation of its components. To guarantee that a tree correctly represents an algebraic expression, it should contains all of the component i.e. operations, constants and variables. Therefore, tree nodes are categorised into three types: operator nodes, constant nodes, and variable nodes. All the actions to modify an expression tree, like removing parentheses and merging similar terms, are in accordance with the laws of algebra. The rank of a node v is the maximum number of edges taken to reach a node and all the leaf nodes are of rank zero. The basic rules are:

1. Operators are distinguished in: binary (difference, power) and k-ary (sum and product) for positive integer k.
2. Leaf nodes: labelled with variable symbols and numerical constants.
3. Non-leaf nodes: labelled with operator symbols.

A simple example is provided in Figure 4.1:

Example 4.4

$$3x_1 + 2x_4^2 + 2x_2x_3$$

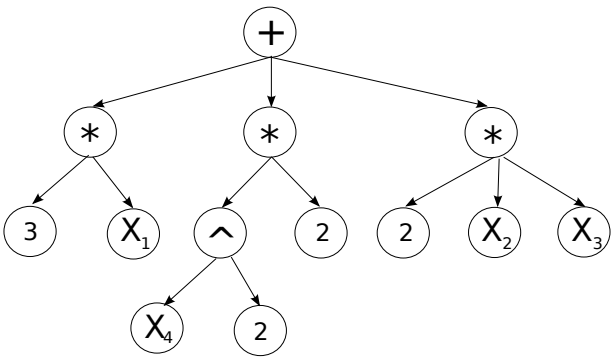


Figure 4.1: Example of expression tree representation.

An algorithm for the tree comparison starts at the root node by comparing their attributes and values. If the current nodes are equal, then it descends down to the child nodes and carries on the comparison in the same way. The comparison steps are recursively executed until the test function reports the existence of equivalence or detects that the two trees violate an equivalence criteria.

Designing equivalence test functions seems reasonable, however, it is not practical. Such tests might require a large number of numerical comparisons, and so they would be algorithmically intractable. To validate the correctness of this method, Liberti (2012b) claims that $f_1, f_2 : \mathbb{R}^n \rightarrow \mathbb{R}$, are equivalent if they have the same range of feasible domain, i.e. $dom(f_1) = dom(f_2)$ and $\forall x \in dom(f_1) f_1(x) = f_2(x)$. Based on which he proves that the formulation group of a mathematical program is a subset of its symmetry group, i.e. $\mathscr{G}(\mathscr{P}) \leq \tilde{\mathscr{G}}(P)$.

Moreover, in terms of the problem formulation, the role of convex relaxation is highly significant. The definition must be strictly applied in every nonlinear term otherwise the symmetric properties of the problem can be affected as in the following example.

Example 4.5 *Consider the original problem:*

$$\begin{aligned} \min \quad & -x_1^2 - x_2^2 \\ s.t. \quad & 0 \leq x_1 \leq 1 \\ & 0 \leq x_2 \leq 1 \end{aligned}$$

with $\tilde{\mathscr{G}}(P^Q) = \mathscr{G}(P^Q) = \{I, (x_1, x_2)\}$

If we generate RLT constraints only for x_1^2, let $x_4 = x_1 x_1$ then we get (P^{LQ}) which has no symmetric properties, $\tilde{\mathscr{G}}(P^{LQ}) = \{I\}$. Following the definition of convex relaxation,
let $x_4 = x_1 x_1, x_3 = x_2 x_2$ then we get $P^{LQ'}$ with $\mathscr{G} = \{I, (x_1, x_2)(x_3, x_4)\}$.

4.3.2 Directed Acyclic Graphs

Liberti (2012a) reduces the formulation symmetry problem to the graph isomorphism problem.

As an extension of the expression tree structures for single functions, he introduces a coloured *DAG* for multiple functions appearing in mathematical programs. Such functions have the same variable (argument) list, so the trees can share the same variable leaf nodes. Further simplifications for duplicated nodes and algebraic equivalence are applied by Liberti (2012b).

A major component of these structures is an equivalence relation on the graph vertices which determines the interchangeability of two vertices. Subsequently, a graph colouring partitions the *DAG* vertices and identifies the subsets of nodes which can be permuted.

The vertex set of an expression graph is partitioned according to the following rules:

1. Root nodes that represent the constraints can be permuted iff they have the same RHS.
2. Variable nodes can be permuted, iff they are of the same type and same range.
3. Constant nodes can be permuted, iff they have the same rank level and value.
4. Operator nodes can be permuted, iff they have the same rank level and value.
5. The order of a child node can not be exchanged iff the operator node is non-commutative.

Two important theoretical results support the correctness of *DAG* constructions. Ramani and Markov (2005) prove that:

Theorem 4.1 *The symmetries of the constraints of the given mathematical problem, correspond one-to-one to the symmetries of the graph.*

Liberti (2012b) proves how to map the automorphism group of a *DAG* graph to the formulation group of the original problem.

Theorem 4.2 *A subgroup of the automorphism group of a DAG that fixes the variable nodes of the graph is equivalent to the formulation group of the original problem.*

4.4 Symmetry Representation via Matrices

This section proposes and discusses structures to detect the formulation group which captures the symmetric nature of a given linear and nonlinear programming problem.

4.4.1 Matrix Structures

This part suggests two different methods of forming a problem as a matrix. The definition of the formulation group of a problem depends on these matrices. We transform each matrix into a graph for detecting and classifying the automorphism group which reveals the symmetry of the original problem. The presence of linear and bilinear terms in quadratic problems though indicates their difficulty. Consider the formulation of *QCQP* with functions

$$f_k(\mathbf{x}) = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i \alpha_{ij}^k x_j + \sum_{i=1}^{n} \alpha_{i0}^k x_i + \alpha_{00}^k \; \forall k = 0\ldots,m$$

with coefficients $\alpha_{ij}^k \in \mathbb{R}$ for $i \in \{0,\ldots,n\}$, $j = \{0,\ldots,n\}$ and $k \in \{0,1,\ldots,m\}$ for $x_i \in [x_i^L, x_i^U]$, $i \in \{1,\ldots,n\}$.

Method 4.1 *Create a tensor* $\mathbf{A}^Q \in \mathbb{R}^{(n+1)\times(n+1)\times(m+1)}$ *with entries* a_{ij}^k.

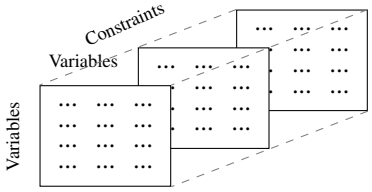


Figure 4.2: Matrix illustration of Method 4.1.

Each matrix corresponds to a QCQP equation and the rows and columns to a constant element and the variables of the QCQP, capturing the relations between the bilinear term. The following example shows this idea.

Example 4.6

$$\begin{aligned}
\max \quad & 3x_1 + 3x_4 + 2x_2x_3 && (c_0) \\
& x_2 + {x_1}^2 + 1 \leq 0 && (c_1) \\
& x_3 + x_4^2 + 1 \leq 0 && (c_2) \\
& x_2 + x_3 + 1 \leq 0 && (c_3) \\
& x_1, x_2, x_3, x_4 \in [0,1]
\end{aligned} \qquad (QP_1)$$

$\mathbf{A}^{Q_1} =$

$$A_{c_0} = \begin{pmatrix} 0 & 3 & 0 & 0 & 3 \\ 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 \end{pmatrix} A_{c_1} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} A_{c_2} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} A_{c_3} = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Consider the problem *LQP* which incorporates the constraints of the original problem and the RLT constraints formed by McCormick relaxation for each nonlinear term as presented in Chapter 2. Let $\hat{m} = (1 + m + (\#\text{ of non linear terms}) \times 4)$ and $\hat{n} = (1 + n + \#\text{ of non linear terms})$.

Method 4.2 *Create a 2 dimensional matrix* $\mathbf{A}^{LQ} \in \mathbb{R}^{\hat{m} \times \hat{n}}$*, with entries the coefficients of LQP* a_{kj}.

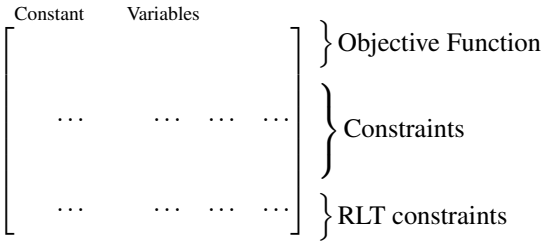


Figure 4.3: Matrix illustration of Method 4.2.

The number of columns set as $\hat{n}$ consists of a constant element, each variable and the auxiliary variables introduced for nonlinear terms. The number of rows say $\hat{m}$ consists of the objective function and all the constraints of the problem. Note that the maximum number of nonlinear terms is: $\frac{n(n+1)}{2}$. Consider the linearised form of QP_1 by introducing the auxiliary variables, $X_{23} = x_2x_3$, $X_{11} = x_1^2$, $X_{44} = x_4^2$ and add the McCormick relaxation constraints. Adding the RLT constraints lead to the following example:

Example 4.7

$$
\begin{aligned}
\max \quad & 3x_1 + 2X_{23} + 3x_4 && (c_0) \\
s.t. \quad & X_{11} + x_2 + 1 \leq 0 && (c_1) \\
& x_3 + X_{44} + 1 \leq 0 && (c_2) \\
& x_2 + x_3 + 1 \leq 0 && (c_3) \\
& x_2 + x_3 - X_{23} - 1 \leq 0 && (c_4) \\
& X_{23} - x_2 \leq 0 && (c_5) \\
& X_{23} - x_3 \leq 0 && (c_6) \\
& 2x_1 - X_{11} - 1 \leq 0 && (c_7) \\
& X_{11} - x_1 \leq 0 && (c_8) \\
& 2x_4 - X_{44} - 1 \leq 0 && (c_9) \\
& X_{44} - x_4 \leq 0 && (c_{10}) \\
& X_{23}, X_{11}, X_{44} \geq 0 \\
& x_1, x_2, x_3, x_4 \in [0,1]
\end{aligned}
\qquad (LQP_1)
$$

$\mathbf{A}^{LQ_1} =$

$$
\begin{array}{c}
\begin{array}{cccccccc} I & x_1 & x_2 & x_3 & x_4 & X_{11} & X_{23} & X_{44} \end{array} \\
\begin{pmatrix}
0 & 3 & 0 & 0 & 3 & 0 & 2 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
-1 & 2 & 0 & 0 & 0 & -1 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 0 & 2 & 0 & 0 & -1 \\
0 & 0 & 0 & 0 & -1 & 0 & 0 & 1
\end{pmatrix}
\begin{array}{c} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \\ C_5 \\ C_6 \\ C_7 \\ C_8 \\ C_9 \\ C_{10} \end{array}
\end{array}
$$

If we compare these two methods we observe that Method 4.2 has potentially considerably fewer entries. Consider the case where a problem has n variables. Method 4.2 requires 4 new constraint for each nonlinear term. Then in the worst case scenario there are $(1+n)(1+\frac{n}{2})(1+m+2n^2+2n)$ entries in contrast to the Method 4.1 which has $(1+n)(1+n)(1+m)$. In most cases, Method 4.1 has fewer entries than Method 4.2. There exist pathological cases, e.g. fully dense

formulations as $m > 3n^2 + 6n + 3$, where Method 4.2 has fewer entries. The graph transformation is based on the number of entries of these matrices. Hence, dealing with smaller graphs reduces their complexity and the procedure time taken to generate their symmetric properties and to compare them.

Next, we define the formulation group of a matrix; necessary for detecting symmetry.

Definition 4.4 *Formulation group of the matrix* $\mathbf{A}^{LQ}$

$$\mathscr{G}(\mathbf{A}^{LQ}) = \{\pi \in \Pi^{\hat{n}} \mid \exists \sigma \in \Pi^{\hat{m}} \quad \textit{such that} \quad \mathbf{A}(\sigma, \pi) = \mathbf{A}\}$$

The set of permutations of the columns of $\mathbf{A}^{LQ}$ such that there is a corresponding permutation of the rows that when applied yields the original matrix. For permutations $\pi \in \Pi^n$, $\sigma \in \Pi^m$, $\mathbf{A}(\pi, \sigma)$ is a matrix obtained by permuting the columns of $\mathbf{A}$ by π and the rows of $\mathbf{A}$ by σ.

Definition 4.5 *Formulation group of the matrix* $\mathbf{A}^{Q}$

$$\mathscr{G}(\mathbf{A}^{Q}) = \{\pi \in \Pi^{n} \mid \exists \sigma \in \Pi^{m} \quad \textit{such that} \quad \mathbf{A}(\pi, \pi, \sigma) = \mathbf{A}\}$$

The set of permutations of the columns and rows of each matrix in $\mathbf{A}^{Q}$ under which the matrix yields to its original form. The same permutation π acts both on rows and columns of the matrix $\mathbf{A}^{Q}$ which represent the same number and type of variables.

Matrix representation shows that exchanging the columns and rows of a matrix which subsequently means changing the position of the variables and constraints of the problem, leads to an equivalent problem. Margot (2002, 2003b) proves that the formulation group of similar matrices is a subset of the symmetry group of the original problem, i.e $\mathscr{G} \leq \tilde{\mathscr{G}}$. The corresponding set of permutations that fixes the problem formulation is called the problem symmetry. Liberti (2008) proposes an automatic way to compute permutations of the formulation group and proves that is a subset of the symmetries in the solution group of a MILP in the general form. As far as we know, this is the only approach for automatic symmetry detection that does not reduce the problem to a graph. Computational experiments report that finding elements of the symmetry automatically is too costly in terms of CPU time.

4.4.2 Converting Matrices to Edge-Labelled Vertex-Coloured Graphs

The matrix representations proposed in this chapter include all the elements of an optimisation problem by construction: variables, constraints and coefficients. The main idea of this work is to convert such matrices into edge-labelled vertex-coloured graphs associated with the basic elements of the problem and then map the graph automorphisms to the original problem symmetries. Margot (2010) states that mapping the instance of a problem to a coloured graph is a standard procedure (Ramani and Markov 2005, Salvagnin 2005, Ramani et al. 2006). Colour preserving automorphisms of such graphs correspond to problem symmetries. A similar idea on how to convert the matrices in Section 4.4.1 to edge-labelled vertex-coloured graphs is given here.

Since many of the matrix values **A** are 0, a sparse matrix representation of Method 4.1 is used to reduce space in memory and time accessing all the coefficient of the problem. Consider a tuple $\mathbf{A} = (\mathbf{M}, \mathbf{I}, \mathbf{J}, \mathbf{K})$ of vectors $\mathbf{M}, \mathbf{I}, \mathbf{J}, \mathbf{K} \in \mathbb{R}^s$ with maximum size $s = (n+1)(n+1)(m+1)$.

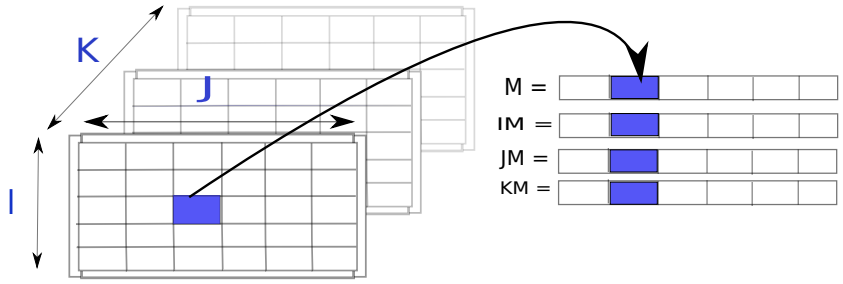


Figure 4.4: Sparse matrix representation.

- **M** = $(M_1, \ldots, M_s)$ is a vector with all non zero entries of a matrix stored from left to right and from top to bottom.
- **J** = $(J_1, \ldots, J_s)$ represent the column indices correspond to non zero entries.
- **I** = $(I_1, \ldots, I_s)$ represent the row indices correspond to non zero entries.
- **K** = $(K_1, \ldots, K_s)$ represent the matrix (constraint) indices correspond to non zero entries.

Recall Example QP_1. For Method 4.1, since each matrix is symmetric, without loss of generality we consider only the upper triangular matrices of each case.

- $\mathbf{M_{QP_1}} = (3\,3\,2\,1\,1\,1\,1\,1\,1\,1\,1\,1)$

- $\mathbf{I_{QP_1}} = (002001004000)$
- $\mathbf{J_{QP_1}} = (143021034023)$
- $\mathbf{K_{QP_1}} = (000111222333)$

Note that this is not the case for Method 4.2 since the matrix illustration of Example LQP_1 is not symmetric.

Using these vectors, we construct edge-labelled vertex-coloured graphs which are variants of constraint/variable incidence graphs. Consider a graph $\mathbf{G} = (V, E, c)$ corresponding to an instance $\mathbf{M}, \mathbf{I}, \mathbf{J}, \mathbf{K}$. The function $c : E \to r$, for $r \in \{0, \ldots, \ell - 1\}$ is an edge colouring and $\ell \in \mathbb{Z}^+$ is the unique number of different coefficients in $\mathbf{M}$. Each of these unique elements in vector $\mathbf{M}$ is stored in a vector $\mathbf{U} \in R^n$. The vertex set is partitioned (coloured) into four subsets as explained in Section 4.3.2, V_F a set containing a node for the objective function, V_C nodes for the constraints, V_S a constant node and V_R nodes for the variables. Note that the definition of the automorphism with respect to colours states that each vertex can only be mapped onto a vertex of the same colour.

For Method 4.2, we construct the following edge coloured graph with edge set initially empty $E = \emptyset$. For $i = \{0, \ldots, s\}$ where $s = |I| = |K| = |M|$ add an edge $v_{I_i}^{(r)}$ to $v_{K_i}^{(r)}$, i.e. from a vertex in the set that represents the constant element / variables to a vertex in the set of the objective function / constraints, with the relevant colour as shown in Figure 4.5.

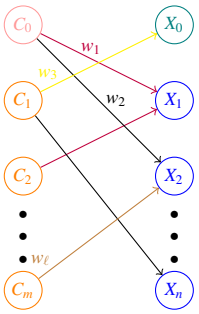


Figure 4.5: Weighted graph representation for Method 4.2.

For Method 4.1 the graph construction also accounts edges between nodes in the variable set to show the bilinear relations and loops for quadratic terms. Initially for graph $\mathbf{G} = (V, E, c)$ let $E = \emptyset$. For $i = \{0, \ldots, s\}$:

- If $I_i = J_i$ then $E = E \cup \{\{(v_{I_i}, v_{K_i})^r\} \cap \{(v_{I_i})^r\}\}$

- else for $I_i \neq J_i$ then $E = E \cup \{\{(v_{I_i}, v_{K_i})^r\} \cap \{(v_{J_i}, v_{K_i})^r\} \cap \{(v_{I_i}, v_{J_i})^r\}\}$

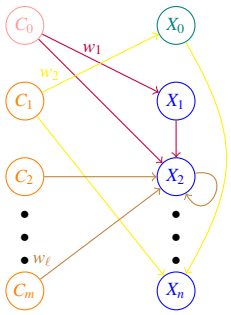


Figure 4.6: Weighted graph representation for Method 4.1.

4.5 Formulation Symmetry Detection via Binary Layered Graphs

To detect symmetry, we use software `nauty` (McKay and Piperno 2014). Ideally, each variable could be a vertex in the graph and each coefficient a label of an edge connecting the vertices involved using edge vertex coloured graphs in Section 4.4.2. But `nauty` (McKay and Piperno 2014) accepts only vertex coloured graphs, so we associate edge colours with layers in a graph. McKay and Piperno (2014) state that graphs similar to Section 4.4.2 and matrix representations in Section 4.4.1 are isomorphic to a general vertex coloured layered graph representation. According to this statement we describe how to illustrate an optimisation problem with *binary layered graph* (*BLG*) structures (McKay and Piperno (2014)). In this chapter, we convert the adjacency matrices in Section 4.4.1 into *binary layered graphs* (*BLG*) and generate the automorphism group of such graphs that projects the symmetry in the original optimisation problems.

McKay and Piperno (2014) explain how to convert a graph $\mathbf{G} = (V, E, c)$ with colouring $c : E \to \{0, \ldots, \ell - 1\}$ of ℓ colours into an ℓ - *layering graph*. First, replace each vertex $v_j \in V$ with a fixed connected graph of ℓ vertices $v_j^{(0)}, \ldots, v_j^{(\ell-1)}$. If an edge $(v_j, v_{j'})$ has colour r, add an edge from $v_j^{(r)}$ to $v_{j'}^{(r)}$. Finally, partition the vertices by the superscripts, $V_r = \{v_0^{(r)}, \ldots, v_{n-1}^{(r)}\}$.

4.5.1 Binary Layered Graph Representation

We use a binary representation to avoid many layers in **G** when the number of colours is large.

Definition 4.6 *Binary Layered Graph*

Let ℓ be the number of edge labels of **G**. *A BLG is an edge-labelled vertex-coloured graph* **B**. *Each vertex colour is associated with a binary representation. The number of layers of* **B** *is:*

$$L = \lceil \log_2(\ell + 1) \rceil \quad for \quad \ell \in \mathbb{Z} \tag{4.2}$$

Assign a unique positive integer $\mu(z)$ to each unique element z in vector **U**. The set $\{\mu(z) \,|\, z \in \mathbf{U}\}$ is a set of edge labels for **B**. For each $\mu(z)$ compute a binary representation.

$$z = c_{L-1}2^{L-1} + c_{L-2}2^{L-2} + \ldots + c_0 2^0, \quad \text{for} \quad c_t \in \{0,1\}\, t = \{0,\ldots,L-1\} \tag{4.3}$$

For nonzero c_t, the powers of 2 reveals which layers encode that value. If $c_t = 1$, add a new edge from $v_i^{(t)}$ to $v_j^{(t)}$ for every $c_t \in \{c_1,\ldots,c_{L-1}\}$. The form of a layered graph is shown in Figure 4.7.

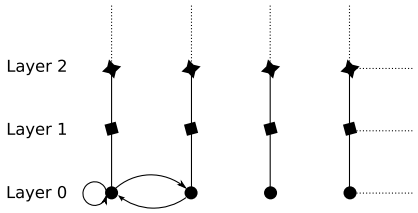


Figure 4.7: General form of a Binary Layered Graph.

In *QCQP* it is important to consider both nonlinear terms and how to incorporate the different variable coefficients in the graph representation. *BLG* structures can handle this situation with loops as described below. This section shows how to illustrate different mathematical problems as graphs.

Next we describe the different graphic illustrations of problems with a finite number of algebraic expressions.

4.5.2 Graph Structures

Section 4.4.1 presents two methods on how to associate matrices with optimisation problems *LQP* and *QCQP*.

For $i = \{1,\dots,n\}$, $k = \{1,\dots,m\}$, $\ell \in \mathbb{Z}$ where n is the number of variables, m is the number of constraints and ℓ the number of unique coefficients in each problem (P). The following graph representation skeletons are presented for $\mathbf{G}_P = (V_P, E_P)$. The vertex set consists of V_F set containing vertices associated with the objective function, V_C with the constraints and V_S with a constant and V_R the variables. Similar to Liberti (2012a), we define an equivalence relation $\sim$ on V_P as follows:

$$\begin{aligned}\forall u, v \in V_P\, u \sim v \implies & (u, v \in V_F \wedge \ell(u) = \ell(v)) \\ & \vee (u, v \in V_C \wedge \ell(u) = \ell(v)) \\ & \vee (u, v \in V_S \wedge \ell(u) = \ell(v)) \\ & \vee (u, v \in V_R \wedge \ell(u) = \ell(v))\end{aligned}$$

i.e. vertices on the same vertex set and layer are in the same partition and can be exchanged.

Graph 4.1 *represents linear problems (originally or after applying RLT) and matrix representation in Method 4.2.*

The number of layers $L = \lceil \log_2(\ell+1) \rceil + 1$. The total number of vertices is: $|V| = (\hat{n}+1)(L-1)+\hat{m}+1$. The vertex set consists of (layer 0) vertices that correspond to the objective function and the constraints of the problem. On every other layer, there are copies of these nodes as shown by the vertical lines. Then on the top layer there is one vertex for a constant element and vertices for each QCQP variable. From nodes in (layer 0) and its copies, we add edges with endpoints the nodes on the top layer, based on which variable is included on each constraint and what is the coefficient in front of this variable.

Graph 4.2 *represents quadratic (nonlinear) QCQP problems and matrix on Method 4.1.*

The graph consists of two different parts with number of layers $L = \lceil \log_2(\ell+1) \rceil + 2$; the vertices for the objective function and each constraint and layers of copies of these constraints (connected with vertical edges). In this part the horizontal edges encode the coefficients of the

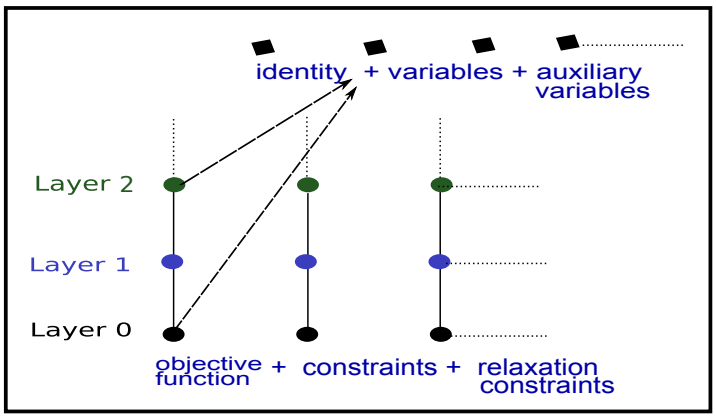


Figure 4.8: Illustration of **Graph 4.1**.

problem. The total number of vertices is $|V| = 2(n+1)+(m+1)(L-2)$. On the upper part as shown in Figure 4.7, there are vertices for a constant element and each variable and a layer of copies of variables (connected with vertical edges). On this layer the horizontal edges and loops distinguish the relations of linear and bilinear terms.

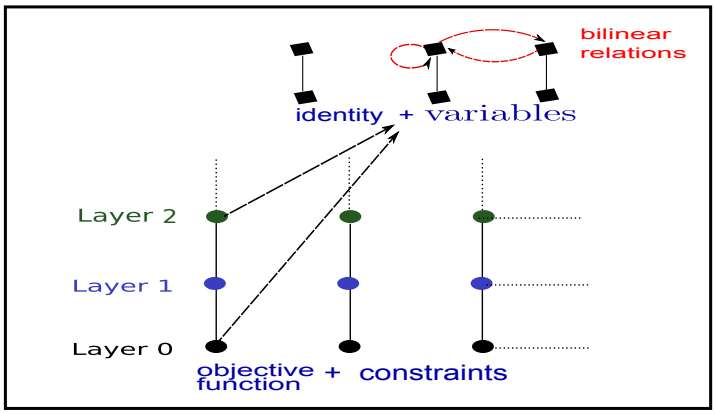


Figure 4.9: Illustration of **Graph 4.2**.

4.6 Computational Case

The following example incorporates all the steps and the algorithms proposed in this paper. We construct the binary labelled graph and then enter it into `nauty` through dreadnaut command lines which compute the formulation group of the original problem.

4.6.1 Numerical Example

In this part we consider the Example QP_1 of a *QCQP* problem from Subsection 4.4.1 and apply the two different graph representations as described in Subsection 4.5.2. Recall Example QP_1:

$$\begin{aligned} \max \quad & 3x_1 + 3x_4 + 2x_2x_3 && (c_0) \\ & x_2 + {x_1}^2 + 1 \leq 0 && (c_1) \\ & x_3 + x_4^2 + 1 \leq 0 && (c_2) \\ & x_2 + x_3 + 1 \leq 0 && (c_3) \\ & x_1, x_2, x_3, x_4 \in [0,1] \end{aligned}$$

The sparse matrix representation:

- $\mathbf{M_{QP_1}} = (3\,3\,2\,1\,1\,1\,1\,1\,1\,1\,1\,1)$
- $\mathbf{I_{QP_1}} = (0\,0\,2\,0\,0\,1\,0\,0\,4\,0\,0\,0)$
- $\mathbf{J_{QP_1}} = (1\,4\,3\,0\,2\,1\,0\,3\,4\,0\,2\,3)$
- $\mathbf{K_{QP_1}} = (0\,0\,0\,1\,1\,1\,2\,2\,2\,3\,3\,3)$

with vector of unique elements $\mathbf{U} = (1\,2\,3)$. There are three unique elements and $L = 2$ layers (see Equation 4.2) to represent the relation of the variables of this problem. The binary representation of each unique element is computed using Equation 4.3, e.g. $3 = 2^1 + 2^0$ indicates that there is an edge between vertices on layer zero and another edge between the same vertices on layer 1. Following the Graph 4.2 description, this graph consists of 4 layers and $|V| = 18$, one associated with a constant element and one with the objective function and the rest for the variables and constraints of the problem. The graph representation of QP_1 is shown in Figure 4.10. `Nauty` generates the following permutations: π = (1 2)(5 6)(9 12)(10 11)(14 17)(15 16); the automorphism group of the graph under which it remains invariant. The relevant enumeration distinguishes which permutations are applied on the constraints and which on the variables of the problem. We then reflect these information on the original problem and explain its symmetric properties. Permutations (1 2)(5 6), permute the constraints c_1, c_2 of Problem QP_1. Permutations (9 12)(10 11) are associated to the variables x_1, x_4 and x_2, x_3 with (14 17)(15, 16) their copies. Hence, the formulation group of problem refex1d is $\mathscr{G} = (x_1x_4)(x_2x_3)$.

Problem LPQ_1 is the relaxed form of the original Problem QP_1 after applying convex relaxation by introducing the auxiliary variables, $X_{23} = x_2x_3$, $X_{11} = x_1^2$, $X_{44} = x_4^2$ and adding the McCormick relaxation constraints.

$$
\begin{aligned}
\max \quad & 3x_1 + 2X_{23} + 3x_4 && (c_0)\\
\text{s.t.} \quad & X_{11} + x_2 + 1 \leq 0 && (c_1)\\
& x_3 + X_{44} + 1 \leq 0 && (c_2)\\
& x_2 + x_3 + 1 \leq 0 && (c_3)\\
& x_2 + x_3 - X_{23} - 1 \leq 0 && (c_4)\\
& X_{23} - x_2 \leq 0 && (c_5)\\
& X_{23} - x_3 \leq 0 && (c_6)\\
& 2x_1 - X_{11} - 1 \leq 0 && (c_7)\\
& X_{11} - x_1 \leq 0 && (c_8)\\
& 2x_4 - X_{44} - 1 \leq 0 && (c_9)\\
& X_{44} - x_4 \leq 0 && (c_{10})\\
& X_{23}, X_{11}, X_{44} \geq 0 &&\\
& x_1, x_2, x_3, x_4 \in [0,1] &&
\end{aligned}
$$

Similar to Problem QP_1 apply Method 4.2 described in Section 4.5. Observe that this problem also contains negative coefficients mapped to positive integers via function $\mu(z)$ e.g. $\mu(-1) = 4$ with binary representation $4 = 2^2$. The relevant graph is shown in Figure 4.11. `Nauty` generates (1 2)(5 6)(7 9)(8 10)(12 13)(16 17)(18 20)(19 21)(23 24)(27 28)(29 31)(30 32) (34 37)(35 36)(39 40) with specific permutations (34 37)(35 36)(39 40) to reveal the symmetric relations of variables $(x_1x_4)(x_2x_3)(X_{11}X_{44})$ the formulation group $\mathscr{G}$ of LPQ_1. The above results validate both Method 4.2 and Method 4.1 in Section 4.5 for representing an optimisation problem as a graph and then generate its symmetric properties. Figure 4.12 shows a *DAG* representation of Problem QP_1 as described in Section 4.3. The leaf nodes represent the variables and the coefficients, the intermediate nodes the operators and root nodes the plus signs that indicate the existence of a new constraint in this problem. Colours in the graph explain the vertex partitioning of the nodes that can be exchanged. The graph size in terms of the number of vertices and edges

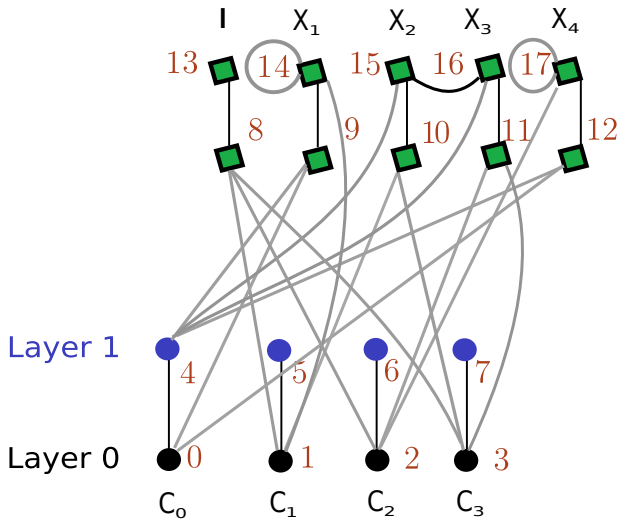


Figure 4.10: Illustration of Problem QP_1 using **Graph 4.2** representation.

is smaller to the size of the methods proposed in this chapter for this example and generates the same formulation group.

4.6.2 Comparison with Current Methods

We evaluate the trade-offs among the graph constructions in this paper and different graph constructions already in the literature. Regarding the graph transformation and its significant role in dealing with symmetry, Ostrowski et al. (2008) introduce the method "Orbital Branching" for combating symmetry. They illustrate each problem and its subproblems on each node of the tree as graphs and use `nauty` to compute the automorphism group and the orbits of the graph. The presence of many coefficients in a problem expand the difficulty of identifying its symmetric properties. Liberti (2012b) uses *Directed Acyclic Graphs* to represent any mathematical expression of MINLP and automatically generates the formulation group. A major advance of both methods is that they are easy to implement and *DAG* can capture the structure of any class of mathematical programming problem. This work proposes an alternative method that may be useful when working with problems with many coefficients of different values. Using the function that assigns integer values to the coefficients of a problem let us work not only with non $0-1$ coefficient but with any other value. Also the use of a logarithmic number of layers may reduce the number of nodes in the graph for problems with a large number of coef-

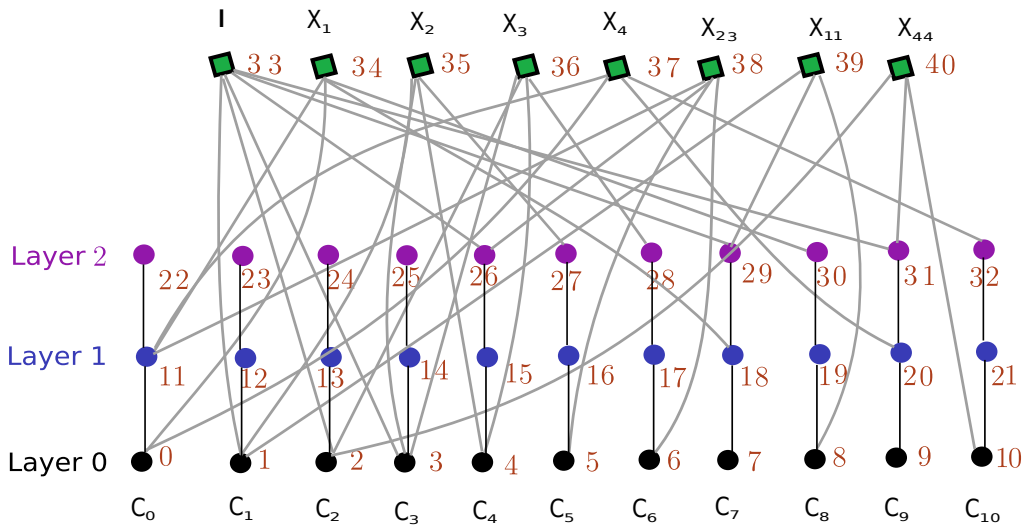


Figure 4.11: Illustration of Problem LQP_1 using **Graph 4.1** representation.

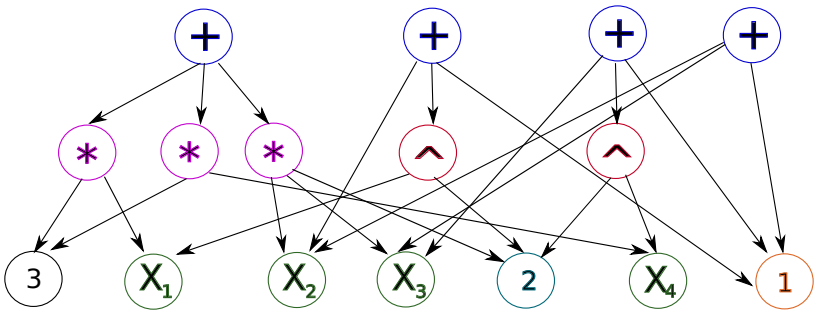


Figure 4.12: Illustration of Problem 1 using **DAG** representation.

ficients. For example, we are able to present 60 different coefficients in a graph with 6 layers. Another advantage is that we are able to capture the relation of bilinear terms in a way that it is unnecessary to create new nodes for every mathematical operation presented in the problem. Addition is the main operation on which the structure of the graph is based and multiplication is only presented with edges and loops. Subtraction is treated as a new coefficient together with the number that follows. The original form of *DAG* graphs without any simplification, provides important informations on the exact formulation of the original problem something which is not clear with our methods. *BLG* may be associated with problems that have the exact same symmetric structure but different formulation. This work though focuses on providing an alternative method to detect the symmetric structure of a problem and not solving the problem itself.

4.7 Conclusion

This work appraise the presence and significance of symmetry in optimisation problems. Symmetry representation and detection are the fundamental steps towards exploiting symmetry. We propose graph structures that may capture the symmetric properties of a problem in a coherent size.

Chapter 5

Detecting Symmetry and Understanding Complexities of the Minimum Number of Matches Problem in Heat Recovery Network Design

5.1 Introduction

Determining the minimum number of matches is the bottleneck of designing a heat recovery network using the sequential method. The minimum number of matches problem posits a set $H = \{1,2,\ldots,n\}$ of hot streams to be cooled and a set $C = \{1,2,\ldots,m\}$ of cold streams to be heated. Each stream is associated with an initial and a target temperature. The mathematical MILP formulations that follow the transportation and transshipment models are provided in Chapter 3. There are combinatorial and computational difficulties when we try to model and solve this problem. One source of problem complexity is the combinatorial explosion in the

possible number of matches of streams to enhance energy recovery. More precisely, Floudas (1995) observed that, in the MILP transshipment model, several solutions with different combinations of streams lead to the same objective function value. Under this interpretation these solutions are considered to be symmetric. When solving the problem using a tree search strategy such as the branch-and-bound algorithm, all these obstacles can cause exponentially large trees with long times to termination. Current state-of-the-art solvers do not recognise the presence of symmetry and cannot deal with moderately-sized problems when industry requires thousands of streams (Furman 2000, Chen et al. 2015a). In order to solve these problems, we first need to understand the aforementioned challenges in the special case of a single temperature interval and then reflect these to the bigger problem. Moreover, this subproblem is proved by Furman and Sahinidis (2001) to be $\mathscr{NP}$-hard even in the case of one temperature interval. The standard MILP formulations for the minimum number of matches in Chapter 3 contains big-M constraints, i.e. the on/off switches associated with weak continuous relaxations of MILP. Both optimisation-based heuristics and exact state-of-the-art methods for solving minimum number of matches problem are highly affected by the big-M parameter. Trivial methods for computing the big-M parameters are typically adopted, but Gundersen et al. (1997) propose a tighter way of computing the big-M parameters.

This chapter proceeds as follows: Section 5.2 presents the special case of the minimum number of matches in a single temperature interval problem. Section 5.3 analyses the problem structure and detects where symmetry and degeneracy arise. Section 5.4 discusses the packing nature of this problem. Then, Section 5.5 provides a novel $\mathscr{NP}$-hardness reduction to the bin packing problem and proposes a novel MILP formulation for the single temperature interval problem. Section 5.7 explores the maximum heat exchanged between the streams with match restrictions including the computation of tighter big-M parameters. These constraints are used in Section 5.8 to examine the computational limits of exact optimisation methods when solving instances of the minimum number of matches from test cases that are manually constructed in this thesis and the ones that currently exist in literature.

Table 5.1: HENS transshipment model symbols (Papoulias and Grossmann 1983). Regular expressions denote alternatives, e.g. expression $FCp_{[i,j]}$ represents FCp_i and FCp_j, the capacity of hot stream i and cold stream j, respectively.

Name	Units	Description
Sets		
HS, CS	–	Hot/Cold process streams
HU, CU	–	Hot/Cold utilities
H, C	–	Hot/Cold streams & utilities
$T = \{1, \cdots, k\}$	–	Temperature intervals
FCp_i, FCp_j	–	Heat capacities of HS, CS
Indices		
$i \in H = \{HS \cup HU\}$	–	Hot process stream/utility
$j \in C = \{CS \cup CU\}$	–	Cold process stream/utility
$t \in T$	–	Temperature interval
Parameters		
$FCp_{[i,j]}$	$[kW/K]$	Flowrate capacity
$T^{HS}_{\text{in},i}, T^{HS}_{\text{out},i}$	$[K]$	Inlet, outlet temperature of hot stream i
$T^{CS}_{\text{in},j}, T^{CS}_{\text{out},j}$	$[K]$	Inlet, outlet temperature of cold stream j
$T^{HU}_{\text{in},i}, T^{HU}_{\text{out},i}$	$[K]$	Inlet, outlet temperature of hot utility i
$T^{CU}_{\text{in},j}, T^{CU}_{\text{out},j}$	$[K]$	Inlet, outlet temperature of cold utility j
$\delta T_{[i,j]t}$	$[K]$	Temperature change at t
$\sigma i, t^{HU}$	$[kW]$	Heat load of HU entering t
$\delta^{CU}_{j,t}$	$[kW]$	Heat load of CU exiting t
$\sigma_{i,t}, \delta_{j,t}$	$[kW]$	Heat load at t
R_t	$[kW/K]$	Total heat residual exiting t
ΔR_t	$[kW]$	Heat residual difference at t
Variables		
$y_{i,j,t}$	$[0, 1]$	Existence of match (ij)
$q_{i,j,t}$	$[kW]$	Heat load of (ij) at t
$R_{i,t}$	$[kW/K]$	Heat residual of HS exiting t
$U_{i,j}$	$[kW]$	Upper bound of match (ij)

5.2 Single Temperature Interval

This section analyses the problem structure of the minimum number of matches problem in a single temperature interval. Initially consider the problem description and the MILP formulation similar to the transshipment model given in Chapter 3. The temperature change δT_t is assumed to be constant and the same for all streams in a fixed temperature interval. The heat loads $\sigma_{i,t}$ provided and $\delta_{j,t}$ required by the relevant subset of hot streams HS and cold streams CS, respectively, are:

$$\sigma_{i,s} = \delta T_s FCp_i \tag{5.1}$$

$$\delta_{j,t} = \delta T_t FCp_j \tag{5.2}$$

In this model, the hottest hot utility is in the top temperature interval and a cold utility in the bottom interval; utilities are treated as streams in intermediate intervals. Excess heat is transferred to the next interval via a heat residual. To decide the minimum utility consumption in order to balance the energy at each interval the LP based on the transshipment model is solved first. The cost in the original formulation in Section 3.4 in Chapter 3 as Floudas (1995) suggests is assumed to be $k_i^{HU} = k_j^{CU} = 1$ for both in order to meet the minimum consumption. A modified version for $t' \in TI$ is the following:

$$\min \quad \sigma_{i,t'}^{HU} + \delta_{j,t'}^{CU} + R_{t'} + R_{t'-1} \tag{5.3}$$

$$\text{s.t.} \quad R_{t'} - R_{t'-1} + \delta_{i,k} - \sigma_{j,1} = \sum_{i \in H} \sigma_{i,t'} - \sum_{j \in C} \delta_{j,t'} \qquad i \in HU, j \in CU \tag{5.4}$$

$$R_{t'} \geq 0 \tag{5.5}$$

$$\sigma_{i,t'} = \delta_{j,t'} = 0 \qquad i \in HU, j \in CU \tag{5.6}$$

$$\sigma_{i,1} > 0, \delta_{j,k} > 0 \qquad i \in HU, j \in CU \tag{5.7}$$

$$R_1 = R_k = 0 \tag{5.8}$$

Hence the LP model initially provides the utility duties of the system, $\sigma_{i,1}^{HU}, \delta_{j,k}^{CU}, R_t, R_{t-1}$. Figure 5.1 represents a transshipment model interval (Papoulias and Grossmann 1983, Floudas

1995). The overall energy balance in Figure 5.1 is given by:

$$R_t - R_{t-1} + \sum_{j \in CU} \delta_{j,t}^{CU} - \sum_{i \in HU} \sigma_{i,t}^{HU} = \sum_{i \in HS} \sigma_{i,t} - \sum_{j \in CS} \delta_{j,t} \tag{5.9}$$

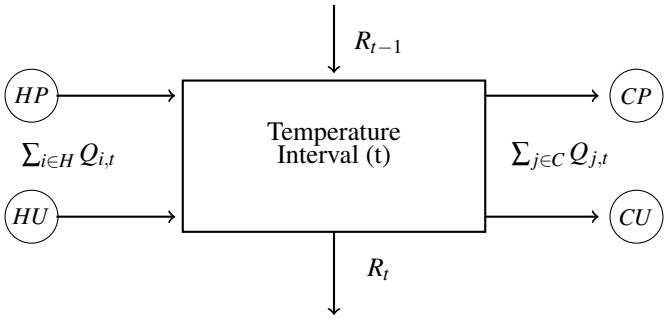


Figure 5.1: Heat balance around a temperature interval.

The objective is to minimise the matches of streams subject to thermodynamic constraints and prohibiting matches between hot utilities HU and cold utilities CU. The following formulation, illustrated in Figure 5.1, represents one fixed temperature interval $t = t'$:

$$\min \quad \sum_{i \in HS} \sum_{j \in CS} y_{i,j,t'} \tag{5.10}$$

$$\text{s.t.} \quad \sigma_{i,t'} = R_{i,t'} - R_{i,t'-1} + \sum_{j \in C} q_{i,j,t'}, \qquad i \in HS \tag{5.11}$$

$$\sigma_{i,t'}^{HU} = R_{i,t'} - R_{i,t'-1} + \sum_{j \in CS} q_{i,j,t'}, \qquad i \in HU \tag{5.12}$$

$$\delta_{j,t'} = \sum_{i \in H} q_{i,j,t'}, \qquad j \in CS \tag{5.13}$$

$$\delta_{j,t'}^{CU} = \sum_{i \in HS} q_{i,j,t'}, \qquad j \in CU \tag{5.14}$$

$$R_{t'} = \sum_{i \in H} R_{i,t'}, \tag{5.15}$$

$$q_{i,j,t'} = 0, \qquad i \in HU, j \in CU \tag{5.16}$$

$$q_{i,j,t'} \leq \min\{\sigma_{it'} + R_{i,t'-1}, \delta_{jt'}\} y_{i,j,t'}, \qquad i \in H,\ j \in C \tag{5.17}$$

$$R_{i,t'} \geq 0,\ q_{i,j,t'} \geq 0 \qquad i \in H,\ j \in C \tag{5.18}$$

$$y_{i,j,t'} \in \{0, 1\} \qquad i \in H,\ j \in C \tag{5.19}$$

5.3 Combinatorial Structure

Using the MILP transshipment formulation, this section investigates network topology and detects symmetry in HENS. Figure 5.2 is based on the transshipment model (Floudas 1995). Analogously to transferring a product from source to destination via intermediate intervals, the transshipment model transfers the heat from hot streams and utilities to the cold streams and utilities via temperature intervals. The temperature change is caused by matching the hot and cold streams and utilities at each interval, so, for two sets of 3 hot streams and 3 cold streams, there are $3 \cdot 3 = 9$ binary variables and in the worst case the MILP will require $2^9 = 512$ nodes. The

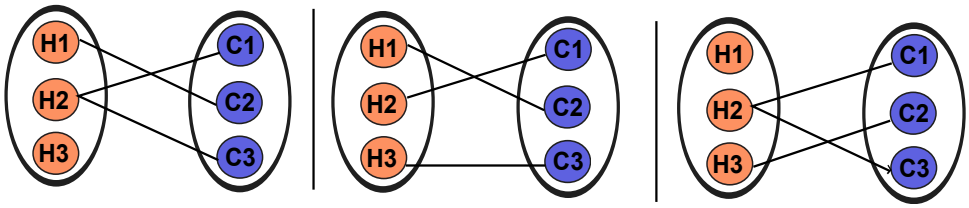


Figure 5.2: Possible configurations of hot/cold stream pairs.

MILP formulation exhibits combinatorial explosion of the possible number of configurations of pairs of hot and cold streams. For example, in Figure 5.2, C_1 can receive heat from 3 hot streams. At the same time C_2, C_3 no matter how C_1 matches, also require heat from one of the 3 choices. Hence there are $3^3 = 27$such configurations. Since $\delta_{j,t'} \neq 0 \implies \sum_{i \in HS} q_{i,j,t'} > 0$, $j \in CS$; i.e. each cold stream needs to match with at least one hot stream in order to satisfy the load requirements. In the worst case scenario of two sets of n hot streams and m cold streams, their match is restricted either as one to one or one to many. Hence the following Lemma holds:

Lemma 5.1 *There are n^m such configurations.*

Proof:

Possible configurations of n hot, m cold streams

$$= \underbrace{\binom{n}{1} \cdots \binom{n}{1}}_{m} = \frac{n!}{(n-1)!} \cdots \frac{n!}{(n-1)!} = \frac{n(n-1)!}{(n-1)!} \cdots n = n^m$$ ∎

We observe that when we only consider hot-to-cold matches in each transshipment model interval, n hot streams and m cold streams may generate $n \cdot m$ possible pairs. There may be many MILP transshipment model solutions with equivalent objective value (Gundersen and Grossmann 1990); we posit that this is due to symmetry.

The significant role of symmetry in optimisation models is described and defined by Margot (2010) and Liberti (2012a). When solving the problem using a tree search strategy such as branch-and-cut, symmetric solutions are unnecessary duplications that need not be investigated. Exploiting symmetry, e.g. via advanced branching strategies, may offer an important advantage for branch-and-cut (Ostrowski et al. 2011, Costa et al. 2013). Identifying problem symmetries is an important step towards expediting tree search algorithms because computationally classifying equivalence allows state-of-the-art solver software to omit symmetric solutions. But symmetry has not been characterised in several critically important process systems engineering applications such as heat exchanger network synthesis; neither do current MILP solvers detect or use symmetries for these energy efficiency problems. This section uses group theory to study the MILP transshipment model of heat exchanger network synthesis and identifies symmetry and degeneracy in the problem.

5.3.1 Symmetry in Heat Exchanger Networks

Applying the formulation and definitions on several cases and combinations of streams, leads to the statement that in a fixed interval in HEN there can appear local symmetries through the exchange of streams. These symmetries arise in the feasible set of solutions from the topology of the problem as several pairs based on the binary variable $y_{i,j,t}$ give the same number of matches. The trouble with the HENS problem is that there is both symmetry in the objective and the constraints. This yields to the observation that if solutions in a temperature interval have the same number of matches and change of residual $\Delta R_{t'} = R_{t'} - R_{t'-1}$ then they are related between them. The symmetry group of a HEN is defined as follows: For $i \in HS$, $j \in CS$ and $t = t'$ if $\sum_{i \in HS} \sum_{j \in CS} y_{i,j,t'} = \sum_{i \in HS} \sum_{j \in CS} y'_{ijt'}$ and $\Delta R_{t'} = \Delta R'_{t'}$ then $\exists \pi \in \Pi^n$ and $\pi' \in \Pi^m$ such that:

$$\mathscr{G}(\mathrm{HEN}(t'), \Delta R_{t'}) \cong \{\pi : HS \longrightarrow HS, \pi' : CS \longrightarrow CS\} \tag{5.20}$$

In the context of exploiting symmetry Liberti (2012a) and Costa et al. (2013) are the first who use algebra groups to explain the structure of optimisation problems and represent the symmetry. The properties of these groups are then used to generate symmetry breaking constraints which improve the time that is taken for the problems to be solved. In this section descriptions and

proofs specify under which cases streams and utilities are considered to be symmetric in a single temperature interval.

For each stream the heat flow $FCp_{[i,j]}$ is considered and Eq. (5.9) is rewritten.

$$\Delta R_{t'} = \delta T_{t'} \Big(\sum_{i \in HS} FCp_i - \sum_{j \in CS} FCp_j \Big) \tag{5.21}$$

If two hot or two cold streams have the same flowrate heat capacity then they are equivalent.

Lemma 5.2 *For hot streams $h_1, h_2 \in HS$ if $FCp_{h_1} = FCp_{h_2}$ then $\exists$ a permutation $\pi \in \Pi^n$ such that $\pi(h_1) = (h_2)$.*

Proof:

Let $FCp_h = FCp_{h_1} = FCp_{h_2}$, and since temperature interval is assumed to be constant then from Eq. (5.21)

$\Delta R_{t'} = \delta T_{t'} \Big(FCp_{h_1} + FCp_{h_2} - \sum_{j \in CS} FCp_j \Big) = \delta T_{t'} \Big(FCp_h + FCp_h - \sum_{j \in CS} FCp_j \Big)$ ∎

Lemma 5.3 *For cold streams $c_1, c_2 \in CS$ if $FCp_{c_1} = FCp_{c_2}$ then $\exists$ a permutation $\pi' \in \Pi^m$ such that $\pi'(c_1) = (c2)$.*

Proof:

Same as Lemma 5.2 for cold stream. ∎

Since the change of temperature $\delta T_{t'}$ is the same and constant it can be trivially claimed that the above results hold for the cases where hot streams and utilities or cold streams and utilities have the same heat load $\sigma_{i,t'}, \delta_{j,t'}$ and can be exchanged between them in the possible matches as shown in Figure 5.3.

For the cases where there exist more than one type of streams with the same heat capacities the idea of Ostrowski et al. (2015) for the unit commitment problem is contemplated in this part. He distinguishes the units with the same characteristic into classes and proves that the structure of the problem can enforce the branching strategies that he proposes when the problem is solved with the *B&B* algorithm.

Let W be the set of classes of equivalent hot streams and n_w the number of streams in each class.

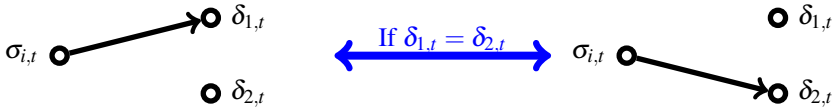


Figure 5.3: Configuration of symmetry in the minimum number of matches problem when there are streams of equal heat capacity.

Proposition 5.1 *For $t = t'$, given a set $HS^w = \{h_1, \cdots, h_{n_w}\} \subset HS$ with $FCp_i = FCp_{i'}$ the symmetry group of HS^w:*

$$\mathscr{G}(HS^w) \cong \{\pi \in \Pi^n | \pi : HS^w \longrightarrow HS^w\} \cong S_{n_w}$$

Proof:

Claim: there are $n!$ such permutations between all the elements of HS^w.

This can be shown as follows (Clark 1984):

(1) Assign $\pi(h_1)$ to one of the elements of HS^w: there are n such choices

Because π is bijective $\pi(h_1) \neq \pi(h_2)$

(2) Assign $\pi(h_2)$ to one of the remaining elements of $HS^w - \{\pi(h_1)\}$: there are $(n-1)$ such choices

...

(n) Assign $\pi(h_{n_w})$ to the only remaining element: there is only 1 such choice

Hence there are $n(n-1)\cdots 1 = n!$ such permutations under which HS^w is invariant. ■

Let Z be the set of classes of equivalent cold streams and n_z the number of streams in each class.

Proposition 5.2 *For $t = t'$, given a set $CS^z = \{c_1, \cdots, c_{m_z}\} \subset CS$ with $FCp_j = FCp_{j'}$ the symmetry group of CS^z:*

$$\mathscr{G}(CS^z) \cong \{\pi' \in \Pi^m | \pi' : CS^z \longrightarrow CS^z\} \cong S_{m_z}$$

Proof:

Follows Proposition 5.1 for cold streams. ■

Proposition 5.3 *Let $\mathscr{G}(HS^1), \mathscr{G}(HS^2), \cdots, \mathscr{G}(HS^w)$ be the sequence of finite groups $S_{n_1}, S_{n_2}, \cdots, S_{n_w}$.*

$$\mathscr{G}(HS) \cong \mathscr{G}(HS^1) \times \mathscr{G}(HS^2) \times \cdots \times \mathscr{G}(HS^w) \cong S_{n_1} \times \cdots \times S_{n_w}$$

Proof:

Follows a relevant Proof that is provided in (Liberti 2012a, Costa et al. 2013). ■

Proposition 5.4 *Let $\mathscr{G}(CS^1), \mathscr{G}(CS^2), \cdots, \mathscr{G}(CS^z)$ be the sequence of finite groups $S_{m_1}, S_{m_2}, \cdots, S_{m_z}$:*

$$\mathscr{G}(CS) \cong \mathscr{G}(CS^1) \times \mathscr{G}(CS^2) \times \cdots \times \mathscr{G}(CS^z) \cong S_{m_1} \times \cdots \times S_{m_z}$$

Proof:

Same as proof of Proposition 5.3 for cold streams. ■

Theorem 5.1 *For $t = t'$ with sets of classes of equivalent hot and cold streams $HS^w \subset HS$, $CS^z \subset CS$ the symmetry group that describes the relations of all streams in the interval is given by: $\mathscr{G}(HEN(t'), \Delta R_{t'}) \cong \mathscr{G}(HS) \times \mathscr{G}(CS)$*

Proof:

Follows from Proposition 5.3, 5.4 and the definition of the internal and external direct product of groups. ■

These proofs can also be trivially generalised for the cases where hot streams and utilities and cold streams and utilities have the same heat load from the Eq. (5.1) and Lemma 5.2, Lemma 5.3 which consist the bottleneck of the above results.

Furthermore, all the above results and proofs contemplate cases where the heat rate capacity flow is the same for all the hot and cold streams involved in the same temperature interval. In

the following examples we also examine some cases of symmetric instances with dissimilar flowrate heat capacities.

If we merge these two disjoint sets we define a new ordered set $FCp_{hc} = FCp_h \cup FCp_c$ and denote it as $(FCp_{hc}, \leq)$ as defined in Chapter 1. Let $\mathscr{F}_\omega$ be the set of feasible solutions with minimum number of matches (ij), i.e. all the solutions in $\mathscr{F}_\omega$ consist of the same number of matches. For any feasible solution with minimum number of matches in the interval $t = t' \in TI$ we observe the following symmetric cases. In all cases we assume that there are no one-to-one matches and subset sums.

Example 5.1 *Case: $min_{i \in HS} FCp_i > FCp_j$, $\forall j \in CS$ and $R_\omega \geq 0$ and $n = m$ then $\mathscr{G}(HEN(t')) \cong S_m$. Let $HS = \{h_1, h_2\}$ and $CS = \{c_1, c_2\}$ with $FCp_h = \{7,5\}$ and $FCp_c = \{4,2\}$. If we merge them in ascending order $FCp_{c_2} < FCp_{c_1} < FCp_{h_2} < FCp_{h_1}$ with $(FCp_{hc}, \leq) = \{6,5,3,1\}$. The minimum number of matches is 2 with the set of feasible solutions*

$$\mathscr{F}_\omega = \left\{ \begin{pmatrix} h_1 & c_1 \\ h_1 & c_2 \end{pmatrix} \begin{pmatrix} h_1 & c_1 \\ h_2 & c_2 \end{pmatrix} \begin{pmatrix} h_2 & c_1 \\ h_1 & c_2 \end{pmatrix} \right\}$$

for $\omega = 1, \cdots 3$, and $\delta FCp_\omega = \{1,6,6\}$. From definition of symmetry, we consider $\mathscr{F}_2$ symmetric to $\mathscr{F}_3$, i.e. $\exists a\, \pi' \in \Pi^m$, $\pi'(\mathscr{F}_2) = \mathscr{F}_3$ with $\pi' = (c_1 c_2)$ and leaves $\mathscr{F}_2$ invariant. Hence the symmetry group of this case is $S_2 \cong \langle (c_1 c_2) \rangle$.

e.g.

5.3.2 Degeneracy in Heat Exchanger Networks

We posit that detecting symmetry is particularly difficult because as stated above the initial form of the problem does not inherit symmetry. Also, the heat that is provided or required by the streams is a continuous variable which can be split into several temperature intervals. Following the interpretation of degeneracy in MILP described in Section 2.7, Chapter 2 we claim that the continuous nature of these variables causes the phenomenon of degeneracy in this problem. As shown in Figure 5.4, as long an the integer variables that decide the possible matches is set then there are multiple ways to distribute the required heat that lead to multiple equivalent solutions.

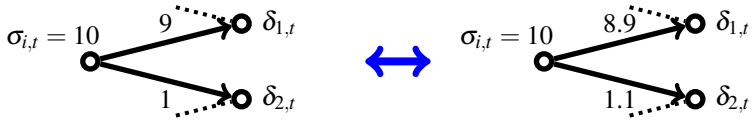


Figure 5.4: Degeneracy in the minimum number of matches problem.

5.3.3 Computational Test Case

The proofs of symmetry in HEN and the above observations demonstrated in a test case [1] of a transshipment model as formulated and implemented on GAMS 24.7.1 by Chen et al. (2015a). The model has been tested on a single 3.40 GHz Intel(R) Core(TM) i7-4770 CSU of a computer with 130 GB memory and running Linux. The MILP solver CPLEX 12.6 is used with optimality gap set to be 10^{-3} and absolute gap 0.99. The flow capacities of the streams that are used lie in a range that are closed to each other and considered as balanced streams. Authors report that tests with balanced streams show exponential increase in the number of nodes and the termination time, which is expected as there are much more combinations of pairs that can take place. From a CPLEX's feature "solnpool" several optimal solutions with the same objective value are obtained. The given data of the streams and how they take part at each of the three subnetworks in which the problem is solved are analysed.

The instance that is tested here is the $Transshipment\_V1\_5$. It consists of 5 hot streams and 5 cold streams and 2 hot utilities and 1 cold utility as initially provided and obtained from the LP transshipment model in the output analysis. The problems data are shown in Table 5.2 and ten solutions of the MILP model with objective value 24 are presented analytically in Figure 5.5.

Table 5.2: Data for Test Case.

Streams [i,j]	$FCp_{[i,j]}$ [kW/K]	σ_i, δ_j [kW]
5 HS	[1;2;1.5;1.7;2.5]	[280;440;345;442;500]
5 CS	[1.3;1.5;1.9;2.5;2.8]	[195;360;570;625;504]
2 HU	-	[110;195]
1 CU	-	[60]

The last subnetwork is isolated as the configuration of matches vary in each solution. It is

[1] http://minlp.org/library/problem/index.pHS?i=191&lib=MINLP

HP - CP	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3	N1	N2	N3
1.1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1.2	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1
1.3	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1.4	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0
1.5	1	0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
2.1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
2.2	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1
2.3	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	1	1
2.4	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
2.5	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0
3.1	1	1	1	1	1	0	1	1	0	1	1	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	1	1	1	0
3.2	0	0	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.3	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	1
3.4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3.5	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4.2	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	1	0	1	0	0	1	0	0	1	0	0	1	0
4.3	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1
4.4	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
4.5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
5.1	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	1	1	0	0	1	0	0	1	0	0	1	0	1
5.2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	1
5.3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5.4	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
5.5	0	1	1	0	1	1	0	1	1	0	1	1	0	1	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	0
HU - CP 1.5	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0
2.4	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
2.5	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0	0	1	0
HP - CU 1.1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
2.1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	1
3.1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
4.1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5.5: Matches of streams/utilities at each subnetwork.

observed that the load that is provided by $H1$ and $H3$ are the same and the load that is required by $C1$ and CU are also the same. What is interesting is that even if initially they have different heat capacities in this subnetwork following the proofs from previous section these streams can be exchanged. As illustrated in Figure 5.6 these permutations lead to other optimal solutions some of which appear in the set of these solutions.

What is crucial in the results and observations is that if we are able to represent the relation of the parameters of a HEN problem from the given data then potentially the duplications of the identical solutions can be eliminated when the problem is solved. At the same time though is important that if at least one such optimal solution is produced all the others can be generated from the symmetry group that is assigned to the problem. Hence their effect in the overall investment cost can be evaluated which is sequentially solved in the last part of the formulation,

that is not examined in this work.

5.4 Packing Nature of the Minimum Number of Matches Problem

Ceccon et al. (2016) uncover special structures of the pooling problem in MINLP problems. They claim that finding these named structures will allow branch-and-cut software solvers to apply generic algorithms and expedite performance for many mathematical optimisation problems. In a similar way, this section relates the minimum number of matches problem with packing problems. The thermodynamic constraints impose obstacles when we try to fit HENS into the standard formulations. Nevertheless, the next two chapters use variations of a simple packing representation to analyse its computational complexity, derive a novel MILP formulation for a single temperature interval and develop heuristics and approximation algorithms.

A feasible solution of the minimum number of matches problem can be viewed as a fractional packing of 2-dimensional items into 2-dimensional boxes as shown in Figure 5.7. Hot stream $i \in H$ is a rectangle of width FCp_i and and height spanning the interval $[T^{HS}_{\text{in},i}, T^{HS}_{\text{out},i}]$, while cold stream $j \in C$ is a rectangle of width FCp_j and height spanning the interval $[T^{CS}_{\text{in},j}, T^{CS}_{\text{out},j}]$. Notice that the positioning of the items and the boxes on the vertical temperature axis is important, while the positioning on the horizontal flow rate axis does not matter. Then, a feasible solution of the problem is a fractional packing of the items in the boxes so that no pair of items overlap in any box. The objective of this problem is to *minimise the number of item splittings*.

5.5 Computational Complexity: $\mathscr{NP}$-hardness reduction

The minimum number of matches problem is known to be strongly $\mathscr{NP}$-hard, even in the special case of a single temperature interval. Furman and Sahinidis (2004) propose an $\mathscr{NP}$-hardness reduction from the well-known 3-Partition problem, i.e. they show that the minimum number of matches problem has difficulty equivalent to the 3-Partition problem. This chapter presents an alternative $\mathscr{NP}$-hardness reduction from the bin packing problem.

Theorem 5.2 *There exists an $\mathcal{NP}$-hardness reduction from bin packing to the minimum number of matches problem with a single temperature interval.*

Proof:

Initially, define the decision version of bin packing. A bin packing instance consists of a set $B = \{1,2,\ldots,m\}$ of bins, each bin of capacity K, and a set $O = \{1,2,\ldots,n\}$ of objects, where object $i \in O$ has size $s_i \in (0,K]$. The goal is to determine whether there exist a feasible packing $O_1, O_2, \ldots, O_m$ of the objects into the bins, where $O_j \subseteq O$ is the subset of objects packed in bin $j \in B$. Each object is placed in exactly one bin, i.e. $\cup_{j=1}^{m} O_j = O$ and $O_j \cap O_{j'} = \emptyset$ for each $1 \leq j < j' \leq m$, and the total size of the objects in a bin do not exceed its capacity, i.e. $\sum_{i \in O_j} s_i \leq K$, for $j \in B$.

Consider an instance (O,n,B,m) of bin packing. Construct an instance of the minimum number of matches problem with a single temperature interval by setting $H = O$, $h_i = s_i$ for $i = 1,\ldots,n$, $C = B$ and $c_j = K$ for $j = 1,\ldots,m$. We claim that bin packing has a feasible solution if and only if the constructed minimum number of matches instance is feasible using exactly n matches.

To the first direction, consider a feasible packing $O_1,\ldots,O_m$. For each $i \in H$ and $j \in C$, we obtain a solution for the minimum number of matches instance by setting $q_{i,j} = h_i$ if $i \in O_j$, and $q_{i,j} = 0$, otherwise. By the constraints $\cup_{j=1}^{m} O_j = O$ and $O_j \cap O_{j'} = \emptyset$ for each $1 \leq j < j' \leq m$, there is exactly one $j \in B$ such that $i \in O_j$. Hence, the number of matches is $|\{(i,j) \in H \times C : q_{i,j} > 0\}| = n$ and $\sum_{j \in C} q_{i,j} = h_i$ for every $i \in H$. Since the capacity of bin $j \in B$ is not exceeded, we have that $\sum_{i \in O_j} s_i \leq K$, or equivalently $\sum_{i \in H} q_{i,j} \leq c_j$ for all $j \in C$. Thus, the obtained solution is feasible.

To the other direction, consider a feasible solution for the minimum number of matches instance. Obtain a feasible packing by placing object $i \in O$ in the bin j if and only if $q_{i,j} > 0$. Since the solution contains at most n matches and $h_i > 0$, for each $i \in H$, each hot stream $i \in H$ matches with exactly one cold stream $j \in C$ and it holds that $q_{i,j} = h_i$. That is, each object is placed in exactly one bin. Given that $\sum_{i \in H} q_{i,j} \leq c_j = K$, the bin capacity constraints are also satisfied. ■

This alternative setting of the minimum number of matches problem gives new insight into the packing nature of the problem. A major contribution of this thesis is to design efficient, greedy heuristics motivated by packing as presented in Chapter 6.

5.6 Novel Mixed-Integer Linear Programming Formulation for a Single Temperature Interval

In the single temperature interval problem, a feasible solution can be represented as a bipartite graph $G = (H \cup C, M)$ as shown in Figure 5.2 in which there is a node for each hot stream $i \in H$, a node for each cold stream $j \in C$ and the set $M \subseteq H \times C$ specifies the matches. We observe that an optimal solution whose graph G represents an optimal solution does not contain any cycle as shown in Figures 5.8 and 5.9. A connected graph without cycles is a *tree*, so G is a forest consisting of trees. The number v of edges in G, i.e. the number of matches, is related to the number ℓ of trees with the equality $v = n + m - \ell$. Since n and m are input parameters, minimising the number of matches in a single temperature interval is equivalent to finding a solution whose graph consists of a maximal number ℓ of trees.

We propose a novel MILP formulation for the single temperature interval problem. In an optimal solution without cycles, there can be at most $\min\{n,m\}$ trees. From a packing perspective, we assume that there are $\min\{n,m\}$ available bins and each stream is placed into exactly one bin. If a bin is non-empty, then its content corresponds to a tree of the graph. The objective is to find a feasible solution with a maximum number of bins. To formulate the problem as an MILP, we define the set $B = \{1,2,\ldots,\min\{n,m\}\}$ of available bins. Binary variable x_b is 0 if bin $b \in B$ is empty and 1, otherwise. A binary variable $w_{i,b}$ indicates whether hot stream $i \in H$ is placed into bin $b \in B$. Similarly, a binary variable $z_{j,b}$ specifies whether cold stream $j \in C$ is placed into bin $b \in B$. Then, the minimum number of matches problem can be formulated:

$$\max \sum_{b \in B} x_b \tag{5.22}$$

$$x_b \leq \sum_{i \in H} w_{i,b} \qquad b \in B \tag{5.23}$$

$$x_b \leq \sum_{j \in C} z_{j,b} \qquad b \in B \tag{5.24}$$

$$\sum_{b \in B} w_{i,b} = 1 \qquad i \in H \tag{5.25}$$

$$\sum_{b \in B} z_{j,b} = 1 \qquad j \in C \tag{5.26}$$

$$\sum_{i \in H} w_{i,b} \cdot h_i = \sum_{j \in C} z_{j,b} \cdot c_j \qquad b \in B \tag{5.27}$$

$$x_b, w_{i,b}, z_{j,b} \in \{0,1\} \qquad b \in B, i \in H, j \in C \tag{5.28}$$

Expression (5.22), the objective function, maximises the number of bins. Equations (5.23) and (5.24) ensure that a bin is used if there is at least one stream in it. Equations (5.25) and (5.26) enforce that each stream is assigned to exactly one bin. Finally, Eqs. (5.27) ensure the heat conservation of each bin. Note that, unlike the transportation and transshipment models, Eqs. (5.22)-(5.27) do not use a big-M parameter. This formulation is particularly useful in the next chapter where an approximation algorithm is proposed for solving each temperature interval individually. However, the aim of generalising this idea to multiple temperature intervals is still a challenge. The next section explores alternatively, methods of generating tighter big-M constraints in a single and multiple temperature intervals. This leads to the evaluation of the exact methods on solving the minimum number of matches using the proposed bounds.

5.7 Maximum Heat Computations with Match Restrictions

This section discusses the feasibility of HENS problem. We propose methods to compute the maximum heat that can be feasibly exchanged given a minimum number of matches instance. Such methods are used to reduce the value of big-M parameter $U_{i,j}$.

Section 5.7.1 is limited to a restricted subset of matches in a single temperature interval. Section 5.7.2 calculates the maximum heat that can be feasibly exchanged for the most general case of multiple temperature intervals.

5.7.1 Maximum Heat in a Single Temperature Interval

Given an instance of the single temperature interval problem and a subset M of matches, the maximum amount of heat that can be feasibly exchanged between the streams using only the matches in M can be computed by solving MaxHeatLP. For simplicity, MaxHeatLP drops

temperature interval indices for variables $q_{i,j}$.

$$\begin{aligned}
\max \quad & \sum_{(i,j)\in M} q_{i,j} \\
& \sum_{j\in C} q_{i,j} \le h_i && i \in H \\
& \sum_{i\in H} q_{i,j} \le c_j && j \in C \\
& q_{i,j} \ge 0 && i \in H, j \in C
\end{aligned} \tag{MaxHeatLP}$$

5.7.2 Multiple Temperature Intervals

Maximising the heat exchanged through a subset of matches across multiple temperature intervals can be solved with an LP that generalises MaxHeatLP. The generalised LP must satisfy the additional requirement that, after removing a maximum heat exchange, the remaining instance is feasible. Feasibility is achieved using residual capacity constraints which are essential for the efficiency of greedy packing heuristics (see Section 6.4.1).

Given a set M of matches, let $A(M)$ be the set of quadruples (i,s,j,t) such that a positive amount of heat can be feasibly transferred via the transportation arc with endpoints the nodes (i,s) and (j,t). The set $A(M)$ does not contain any quadruple (i,s,j,t) with: (i) $s > t$, (ii) $\sigma_{i,s} = 0$, (iii) $\delta_{j,t} = 0$, or (iv) $(i,j) \notin M$. Let $V^H(M)$ and $V^C(M)$ be the set of transportation vertices (i,s) and (j,t), respectively, that appear in $A(M)$. Similarly, given two fixed vertices $(i,s) \in V^H(M)$ and $(j,t) \in V^C(M)$, we define the sets $V^C_{i,s}(M)$ and $V^H_{j,t}(M)$ of their respective neighbours in $A(M)$.

Consider a temperature interval $u \in T$. We define by $A_u(M) \subseteq A(M)$ the subset of quadruples with $s \le u < t$, for $u \in T$. The total heat transferred via the arcs in $A_u(M)$ must be upper bounded by $R_u = \sum_{i=1}^{n} \sum_{s=1}^{u} \sigma_{i,s} - \sum_{j=1}^{m} \sum_{t=1}^{u} \delta_{j,t}$. Furthermore, $A(M)$ eliminates any quadruple (i,s,j,t) with $R_u = 0$, for some $s \le u < t$. Finally, we denote by $T(M)$ the subset of temperature intervals affected by the matches in M, i.e. if $u \in T(M)$, then there exists a quadruple $(i,s,j,t) \in A(M)$, with $s \le u < t$.

$$\max \sum_{(i,s,j,t)\in A(M)} q_{i,s,j,t} \tag{5.29}$$

$$\sum_{(j,t)\in V^C_{i,s}(M)} q_{i,s,j,t} \le \sigma_{i,s} \qquad (i,s) \in V^H(M) \tag{5.30}$$

$$\sum_{(i,s)\in V^H_{j,t}(M)} q_{i,s,j,t} \leq \delta_{j,t} \qquad (j,t) \in V^C(M) \tag{5.31}$$

$$\sum_{(i,s,j,t)\in A_u(M)} q_{i,s,j,t} \leq R_u \qquad u \in T(M) \tag{5.32}$$

$$q_{i,s,j,t} \geq 0 \qquad (i,s,j,t) \in A(M) \tag{5.33}$$

Expression (5.29) maximises the total exchanged heat by using only the matches in M. Constraints (5.30) and (5.31) ensure that each stream uses only part of its available heat. Constraints (5.32) enforce the heat residual capacities. Using this method of computing the maximum heat exchanged between any pair of streams and subsequently the big-M constraints, the next session evaluates the performance of exact methods for solving this problem.

5.8 Performance of Exact Methods for Solving the Minimum Number of Matches Problem

We evaluate exact methods using state-of-the-art commercial approaches. For each problem instance, CPLEX and Gurobi solve the transportation and transshipment models as formulated in Chapter 3, Section 3.5. The description, configuration and analysis of HENS follow Floudas (1995). This section addresses MILP transshipment models consisting of a set of hot process streams HS to be cooled and a set of cold process streams CS to be heated; each stream has an initial and target temperature and a heat capacity. There are also hot utilities HU and cold utilities CU with associated temperatures. The symbols representing the mathematical formulation are shown in Table 5.1.

5.8.1 System Specification and Benchmark Instances

All computations are run on an Intel Core i7-4790 CPU 3.60GHz with 15.6 GB RAM running 64-bit Ubuntu 14.04. CPLEX 12.6.3 and Gurobi 6.5.2 solve the minimum number of matches problem exactly. The mathematical optimisation models and heuristics are implemented in Python 2.7.6 and Pyomo 4.4.1 (Hart et al. 2011, 2012).

We use problem instances from two existing test sets (Furman and Sahinidis 2004, Chen et al.

2015a). We also generate a collection of larger test cases using work of Grossmann (2017). An instance of general heat exchanger network design consists of streams and utilities with inlet, outlet temperatures, flowrate heat capacities and other parameters.

The *Furman (2000) test set* consists of test cases from the engineering literature. Table 5.3 reports bibliographic information on the origin of these test cases. We manually digitize this data set and make it publicly available for the first time (Letsios et al. 2017). Table 5.3 lists the 26 problem instance names and information on their sizes. The total number streams and temperature intervals varies from 6 to 38 and from 5 to 32, respectively. Table 5.3 also lists the number of binary and continuous variables as well as the number of constraints in the transshipment MILP formulation.

The *Chen et al. (2015b,a) test set* consists of 10 problem instances. These instances are classified into two categories depending on whether they consist of balanced or unbalanced streams. Test cases with balanced streams have flowrate heat capacities in the same order of magnitude, while test cases with unbalanced streams have dissimilar flowrate heat capacities spanning several orders of magnitude. The sizes of these instances range from 10 to 42 streams and from 12 to 35 temperature intervals. Table 5.3 reports more information on the size of each test case.

The Grossmann (2017) test set is generated randomly. The inlet, outlet temperatures of these instances are fixed while the values of flowrate heat capacities are generated randomly with fixed seeds. This test set contains 12 moderately challenging problems (see Table 5.3) with a classification into balanced and unbalanced instances, similarly to the Chen et al. (2015b,a) test set. The smallest problem involves 27 streams and 23 temperature intervals while the largest one consists of 43 streams and 37 temperature intervals. *The Large Scale test set* is generated randomly. These instances have 80 hot streams, 80 cold streams, 1 hot utility and 1 cold utility. For each hot stream $i \in HS$, the inlet temperature $T_{\text{in},i}^{HS}$ is chosen uniformly at random in the interval $(30, 400]$. Then, the outlet temperature $T_{\text{out},i}^{HS}$ is selected uniformly at random in the interval $[30, T_{\text{in},i}^{HS})$. Analogously, for each cold stream $j \in CS$, the outlet temperature $T_{\text{out},j}^{CS}$ is chosen uniformly at random in the interval $(20, 400]$. Next, the inlet temperature $T_{\text{in},j}^{CS}$ is chosen uniformly at random in the interval $[20, T_{\text{out},j}^{CS})$. The flowrate heat capacities FCp_i and FCp_j of hot stream i and cold stream j are chosen as floating numbers with two decimal digits in the interval $[0, 15]$. The hot utility has inlet temperature $T_{\text{in}}^{HU} = 500$, outlet temperature $T_{\text{out}}^{HS} = 499$,

and cost $\kappa^{HU} = 80$. The cold utility has inlet temperature $T_{\text{in}}^{CU} = 20$, outlet temperature $T_{\text{out}}^{CU} = 21$, and cost $\kappa^{CU} = 20$. The minimum heat recovery approach temperature is $\Delta T_{\min} = 10$.

5.8.2 Experiments

Based on the difficulty of each test set, we set a time limit for each solver run as follows: (i) 1800 seconds for the Furman (2000) test set, (ii) 7200 seconds for the Chen et al. (2015b,a) test set, and (iii) 14400 seconds for the Grossmann (2017) and large scale test sets. In each solver run, we set absolute gap 0.99, relative gap 4%, and maximum number of threads 1.

Table 5.4 reports the best found objective value, CPU time and relative gap, for each solver run. Observe that state-of-the-art approaches cannot, in general, solve moderately-sized problems with 30-40 streams to global optimality. For example, none of the test cases in the Grossmann (2017) or large scale test sets is solved to global optimality within the specified time limit. Table 5.5 contains the results reported by Furman and Sahinidis (2004) using CPLEX 7.0 with 7 hour time limit. CPLEX 7.0 fails to solve 4 instances to global optimality. Interestingly, CPLEX 12.6.3 still cannot solve 3 of these 4 instances with a 1.5 hour timeout.

Theoretically, the transshipment MILP is better than the transportation MILP because the former has asymptotically fewer variables. This observation is validated experimentally with the exception of very few instances, e.g. `balanced10`, in which the transportation model computes a better solution within the time limit. CPLEX and Gurobi are comparable and neither dominates the other. Instances with balanced streams are harder to solve, which highlights the difficulty introduced by symmetry, see Kouyialis and Misener (2017).

The preceding numerical analysis refers to the extended transportation MILP. Table 5.7 compares solver performance to the reduced transportation MILP, i.e. a formulation removing redundant variables $q_{i,s,j,t}$ with $s > t$ and Equations (3.11). Note that modern versions of CPLEX and Gurobi show effectively no difference between the two formulations. Despite the various differences, the obtained results indicate that CPLEX and Gurobi are able to detect redundant variables and their performance on the two models is not substantially different.

These findings motivate the design of efficient heuristic methods and approximation algorithms with proven performance guarantees.

Test Case	Hot Streams	Cold Streams	Temp. Intervals	Binary Vars	Continuous Vars	Constraints	Ref
Furman and Sahinidis (2004) Test Set							
10sp-la1	5	6	9	30	315	134	Linnhoff. and Ahmad (1989)
10sp-ol1	5	7	8	35	320	136	Shenoy (1995)
10sp1	5	6	9	30	315	134	Pho and Lapidus (1973)
12sp1	10	3	13	30	520	209	Grossmann and Sargent (1978)
14sp1	7	8	14	56	882	273	Grossmann and Sargent (1978)
15sp-tkm	10	7	15	70	1200	335	Tantimuratha et al. (2000)
20sp1	10	11	20	110	2400	540	Grossmann and Sargent (1978)
22sp-ph	12	12	18	144	2808	588	Polley and Heggs (1999)
22sp1	12	12	17	144	2652	564	Miguel et al. (1998)
23sp1	11	13	19	143	2926	610	Mocsny and Govind (1984)
28sp-as1	17	13	15	221	3570	688	Ahmad and Smith (1989)
37sp-yfyv	21	17	32	357	12096	1594	Yu et al. (2000)
4sp1	3	3	5	9	60	42	Lee et al. (1970)
6sp-cf1	3	4	5	12	75	50	Ciric and Floudas (1989)
6sp-gg1	3	3	5	9	60	42	Gundersen and Grossmann (1990)
6sp1	3	4	6	12	90	57	Lee et al. (1970)
7sp-cm1	4	5	8	20	192	96	Colberg and Morari (1990)
7sp-s1	7	2	8	14	168	93	Shenoy (1995)
7sp-torw1	5	4	7	20	175	88	Trivedi et al. (1990)
7sp1	3	5	6	15	108	66	Masso and Rudd (1969)
7sp2	4	4	7	16	140	76	Masso and Rudd (1969)
7sp4	7	2	8	14	168	93	Dolan et al. (1990)
8sp-fs1	6	4	8	24	240	110	Farhanieh and Sunden (1990)
8sp1	5	5	8	25	240	110	Grossmann and Sargent (1978)
9sp-al1	5	6	9	30	315	134	Ahmad and Linnhoff (1989)
9sp-has1	6	5	9	30	324	135	Hall et al. (1990)
Chen et al. (2015b,a) Test Set							
balanced10	12	11	20	132	2880	604	
balanced12	14	13	23	182	4508	817	
balanced15	17	16	28	272	8092	1213	
balanced5	7	6	12	42	588	205	
balanced8	10	9	16	90	1600	404	
unbalanced10	12	11	20	132	2880	604	
unbalanced15	17	16	28	272	8092	1213	
unbalanced17	19	18	32	342	11552	1545	
unbalanced20	22	21	36	462	17424	2032	
unbalanced5	7	6	12	42	588	205	
Grossmann (2017) Test Set							
balanced12_random0	14	13	23	182	4508	817	
balanced12_random1	14	13	23	182	4508	817	
balanced12_random2	14	13	23	182	4508	817	
balanced15_random0	17	16	28	272	8092	1213	
balanced15_random1	17	16	28	272	8092	1213	
balanced15_random2	17	16	28	272	8092	1213	
unbalanced17_random0	19	18	32	342	11552	1545	
unbalanced17_random1	19	18	32	342	11552	1545	
unbalanced17_random2	19	18	32	342	11552	1545	
unbalanced20_random0	22	21	36	462	17424	2032	
unbalanced20_random1	22	21	36	462	17424	2032	
unbalanced20_random2	22	21	36	462	17424	2032	

Table 5.3: Problem sizes of the test cases. The number of variables and constraints are computed with respect to the transshipment model. All test cases are available online (Letsios et al. 2017).

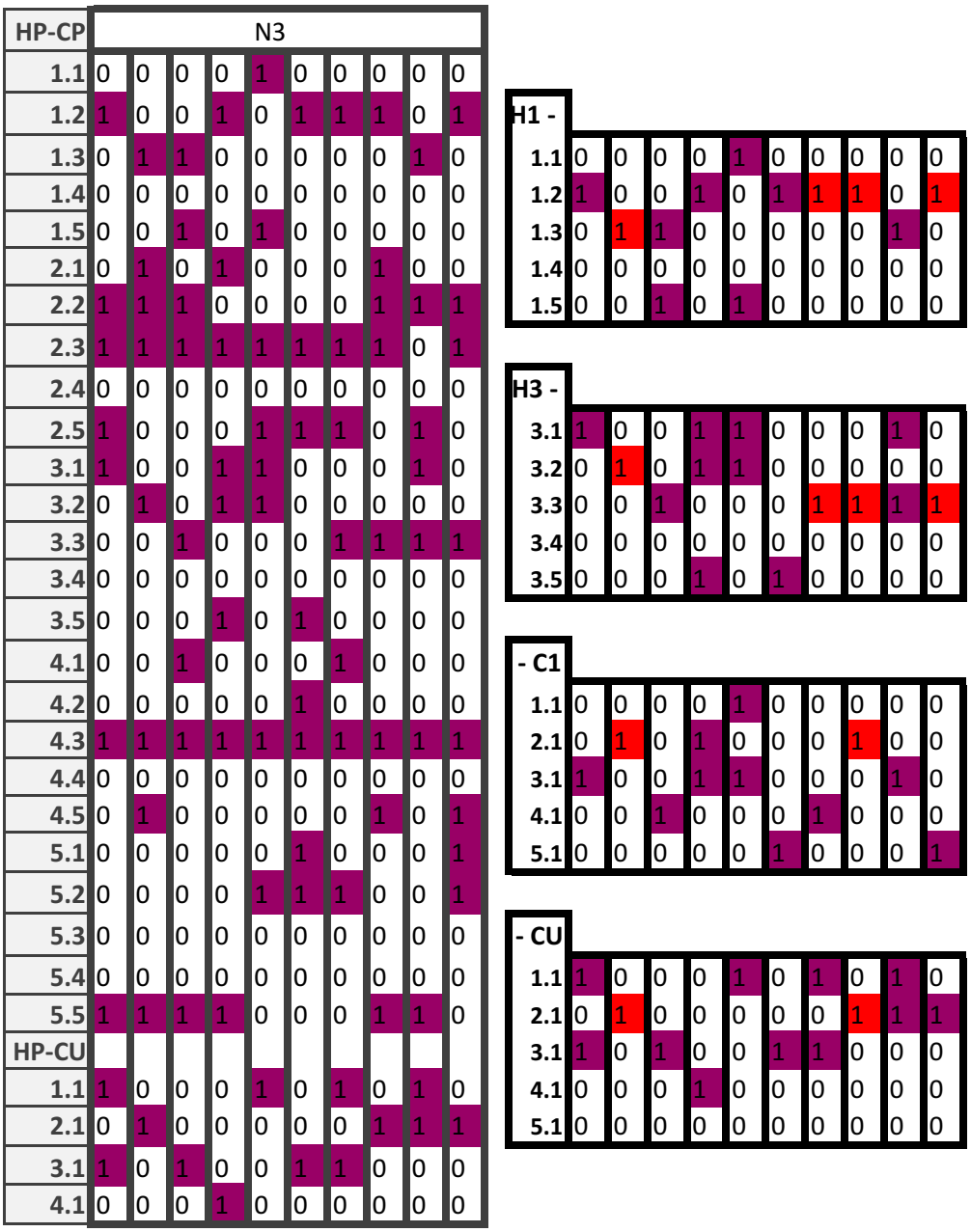


Figure 5.6: Matches of streams/utilities in subnetwork three.

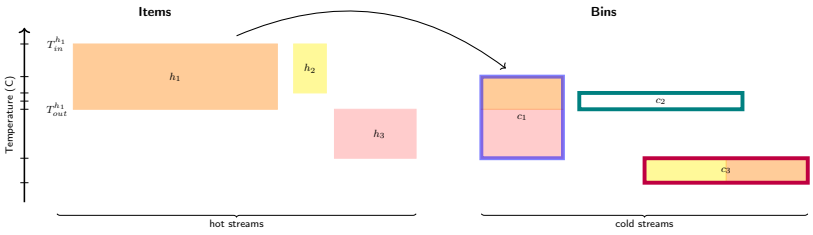


Figure 5.7: A packing representation of the minimum number of matches.

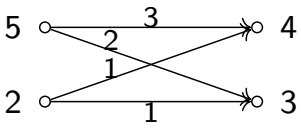


Figure 5.8: Cycle existence.

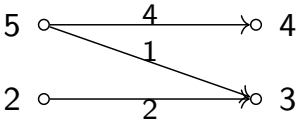


Figure 5.9: Eliminate any cycle via backtracking method.

Test Case	CPLEX Transportation			CPLEX Transshipment			Gurobi Transportation			Gurobi Transshipment		
	Value	Time (s)	Gap	Value	Time (s)	Gap	Value	Time (s)	Gap	Value	Time (s)	Gap
Furman and Sahinidis (2004) Test Set (30min time limit)												
10sp-la1	12	0.04		**12**	**0.03**		12	0.10		12	0.09	
10sp-ol1	14	0.06		**14**	**0.03**		14	0.13		14	0.11	
10sp1	10	0.51		**10**	**0.05**		10	0.24		10	0.13	
12sp1	12	0.08		**12**	**0.05**		12	0.16		12	0.11	
14sp1	14	145.50		**14**	**41.23**		14	170.45		14	126.71	
15sp-tkm	19	0.17		**19**	**0.07**		19	0.28		19	0.14	
20sp1	19	*	19%	19	*	19%	19	*	21%	**19**	*	**15%**
22sp-ph	26	0.25		**26**	**0.04**		26	0.44		26	0.13	
22sp1	25	*	12%	**25**	*	**11%**	25	*	12%	25	*	12%
23sp1	23	*	28%	23	*	28%	23	*	30%	**23**	*	**26%**
28sp-as1	30	0.19		**30**	**0.05**		30	0.44		30	0.12	
37sp-yfyv	36	54.80		36	7.36		36	20.40		**36**	**6.02**	
4sp1	5	0.03		**5**	**0.02**		5	0.10		5	0.08	
6sp-cf1	**6**	**0.03**		**6**	**0.03**		6	0.09		6	0.09	
6sp-gg1	**3**	**0.02**		**3**	**0.02**		3	0.09		3	0.08	
6sp1	6	0.03		**6**	**0.02**		6	0.30		6	0.09	
7sp-cm1	**10**	**0.02**		**10**	**0.02**		10	0.09		10	0.08	
7sp-s1	**10**	**0.02**		**10**	**0.02**		10	0.09		10	0.08	
7sp-torw1	10	0.03		**10**	**0.02**		10	0.10		10	0.09	
7sp1	**7**	**0.03**		7	0.04		7	0.09		7	0.08	
7sp2	7	0.05		**7**	**0.03**		7	0.09		7	0.09	
7sp4	8	0.03		**8**	**0.02**		8	0.10		8	0.08	
8sp-fs1	11	0.03		**11**	**0.02**		11	0.10		11	0.08	
8sp1	9	0.04		**9**	**0.03**		9	0.14		9	0.10	
9sp-al1	12	0.04		**12**	**0.03**		12	0.11		12	0.09	
9sp-has1	**13**	**0.04**		**13**	**0.04**		13	0.12		13	0.09	
Chen et al. (2015b,a) Test Set (2h time limit)												
balanced10	25	*	6%	24	1607.14		25	*	4%	**24**	**358.18**	
balanced12	30	*	16%	**28**	*	**7%**	29	*	13%	29	*	10%
balanced15	36	*	19%	37	*	17%	**35**	*	**17%**	36	*	16%
balanced5	14	0.27		14	0.20		14	0.43		14	0.23	
balanced8	20	180.84		20	69.16		20	997.01		20	248.08	
unbalanced10	25	36.24		25	7.45		25	46.81		**25**	**15.97**	
unbalanced15	36	*	8%	36	*	4%	36	*	8%	**36**	*	**4%**
unbalanced17	43	*	15%	43	*	11%	43	*	13%	**43**	*	**9%**
unbalanced20	55	*	22%	51	*	13%	51	*	17%	**50**	*	**10%**
unbalanced5	16	0.09		**16**	**0.05**		16	0.26		16	0.13	
Grossmann (2017) Test Set (4h time limit)												
balanced12_random0	29	*	13%	**28**	*	**7%**	29	*	13%	**28**	*	**7%**
balanced12_random1	29	*	13%	**29**	*	**9%**	30	*	13%	29	*	10%
balanced12_random2	30	*	16%	**29**	*	**10%**	29	*	10%	**29**	*	**10%**
balanced15_random0	36	*	18%	36	*	15%	35	*	14%	**36**	*	**13%**
balanced15_random1	36	*	18%	36	*	15%	35	*	17%	**35**	*	**11%**
balanced15_random2	36	*	17%	35	*	12%	36	*	16%	**35**	*	**11%**
unbalanced17_random0	44	*	16%	**43**	*	**9%**	43	*	13%	**43**	*	**9%**
unbalanced17_random1	44	*	16%	44	*	10%	44	*	15%	**43**	*	**6%**
unbalanced17_random2	43	*	13%	**43**	*	**9%**	43	*	13%	**43**	*	**9%**
unbalanced20_random0	51	*	16%	**51**	*	**12%**	52	*	19%	52	*	13%
unbalanced20_random1	52	*	18%	52	*	15%	52	*	19%	**51**	*	**11%**
unbalanced20_random2	51	*	16%	52	*	14%	52	*	19%	**50**	*	**10%**

Table 5.4: Computational results using exact solvers CPLEX 12.6.3 and Gurobi 6.5.2 with relative gap 4%. Relative gap is defined (best incumbent - best lower bound) / best incumbent and * indicates timeout. **Bold** values mark the best solver result. The transshipment formulation performs better than the transportation model: the transshipment model solves one additional problem (balanced10) and performs as well or better than the transportation model on 46 of the 48 test cases (with respect to time or gap closed). CPLEX solves the small models slightly faster than Gurobi while Gurobi closes more of the optimality gap for large problems. All exact method results are available online (Letsios et al. 2017).

Test Case	CPLEX			
	FS04		LKM17	
	Value	Time	Value	Time
10sp-la1	12	0.07	12	0.03
10sp-ol1	14	0.09	14	0.03
10sp1	10	2.20	10	0.05
12sp1	12	0.04	12	0.05
14sp1	14	33.76	14	41.23
15sp-tkm	19	0.70	19	0.07
20sp1	19	**	19	*
22sp-ph	26	1.84	26	0.04
22sp1	25	**	25	*
23sp1	23	**	23	*
28sp-as1	30	0.03	30	0.05
37sp-yfyv	36	**	36	7.36
4sp1	5	0.00	5	0.02
6sp-cf1	6	0.01	6	0.03
6sp-gg1	3	0.00	3	0.02
6sp1	6	0.00	6	0.02
7sp-cm1]	10	0.00	10	0.02
7sp-s1	10	0.00	10	0.02
7sp-torw1	10	0.03	10	0.02
7sp1	7	0.01	7	0.04
7sp2	7	0.04	7	0.03
7sp4	8	0.00	8	0.02
8sp-fs1	11	0.01	11	0.02
8sp1]	9	0.03	9	0.03
9sp-al1	12	0.03	12	0.03
9sp-has	13	0.03	13	0.04

Table 5.5: Comparison of our results (labelled LKM17) with the ones reported by Furman and Sahinidis (2004) (labelled FS04). The CPLEX comparison basically confirms that CPLEX has improved in the past 13 years: LKM17 use CPLEX 12.6.3 while FS04 use CPLEX 7.0. An * indicates 30min timeout while ** corresponds to a 7h timeout. All results are available online (Letsios et al. 2017).

Test Case	CPLEX Transshipment	
	Value	Time
large_scale0	175	*
large_scale1	219	*
large_scale2	239	*

Table 5.6: Upper bounds, i.e. feasible solutions, for large-scale instances computed by CPLEX 12.6.3 transshipment model with 4h timeout. Symbol * indicates timeout.

Test Case	CPLEX Transportation			CPLEX Reduced Transportation			Gurobi Transportation			Gurobi Reduced Transportation		
	Value	Time	Gap	Value	Time	Gap	Value	Time	Gap	Value	Time	Gap
Furman and Sahinidis (2004) Test Set(30min time limit)												
10sp-la1	12	0.04		12	0.05		12	0.10		12	0.13	
10sp-ol1	14	0.06		14	0.04		14	0.13		14	0.11	
10sp1	10	0.51		10	0.65		10	0.24		10	0.13	
12sp1	12	0.08		12	0.08		12	0.16		12	0.13	
14sp1	14	145.50		14	144.87		14	170.45		14	172.62	
15sp-tkm	19	0.17		19	0.14		19	0.28		19	0.20	
20sp1	19	*	19%	19	*	19%	19	*	21%	19	*	21%
22sp-ph	26	0.25		26	0.13		26	0.44		26	0.24	
22sp1	25	*	12%	25	*	12%	25	*	12%	25	*	12%
23sp1	23	*	28%	23	*	28%	23	*	30%	23	*	30%
28sp-as1	30	0.19		30	0.09		30	0.44		30	0.23	
37sp-yfyv	36	54.80		36	32.86		36	20.40		36	89.50	
4sp1	5	0.03		5	0.02		5	0.10		5	0.08	
6sp-cf1	6	0.03		6	0.02		6	0.09		6	0.08	
6sp-gg1	3	0.02		3	0.02		3	0.09		3	0.08	
6sp1	6	0.03		6	0.02		6	0.30		6	0.08	
7sp-cm1	10	0.02		10	0.02		10	0.09		10	0.09	
7sp-s1	10	0.02		10	0.02		10	0.09		10	0.12	
7sp-torw1	10	0.03		10	0.03		10	0.10		10	0.09	
7sp1	7	0.03		7	0.23		7	0.09		7	0.08	
7sp2	7	0.05		7	0.04		7	0.09		7	0.09	
7sp4	8	0.03		8	0.02		8	0.10		8	0.10	
8sp-fs1	11	0.03		11	0.05		11	0.10		11	0.10	
8sp1	9	0.04		9	0.07		9	0.14		9	0.13	
9sp-al1	12	0.04		12	0.03		12	0.11		12	0.10	
9sp-has1	13	0.04		13	0.04		13	0.12		13	0.10	
Chen et al. (2015b,a) Test Set (2h time limit)												
balanced10	25	*	6%	25	*	6%	25	*	4%	25	*	8%
balanced12	30	*	16%	30	*	16%	29	*	13%	29	*	13%
balanced15	36	*	19%	37	*	21%	35	*	17%	36	*	19%
balanced5	14	0.27		14	0.26		14	0.43		14	0.33	
balanced8	20	180.84		20	885.66		20	997.01		20	214.00	
unbalanced10	25	36.24		25	64.92		25	46.81		25	61.19	
unbalanced15	36	*	8%	36	*	9%	36	*	8%	36	*	8%
unbalanced17	43	*	15%	43	*	15%	43	*	13%	43	*	13%
unbalanced20	55	*	22%	53	*	21%	51	*	17%	53	*	20%
unbalanced5	16	0.09		16	0.08		16	0.26		16	0.22	
Grossmann (2017) Test Set (4h time limit)												
balanced12_random0	29	*	13%	28	*	8%	29	*	13%	29	*	13%
balanced12_random1	29	*	13%	29	*	11%	30	*	13%	29	*	10%
balanced12_random2	30	*	16%	29	*	13%	29	*	10%	28	*	7%
balanced15_random0	36	*	18%	36	*	18%	35	*	14%	36	*	19%
balanced15_random1	36	*	18%	36	*	18%	35	*	17%	35	*	17%
balanced15_random2	36	*	17%	37	*	19%	36	*	16%	36	*	19%
unbalanced17_random0	44	*	16%	43	*	14%	43	*	13%	43	*	13%
unbalanced17_random1	44	*	16%	44	*	15%	44	*	15%	44	*	15%
unbalanced17_random2	43	*	13%	43	*	14%	43	*	13%	44	*	13%
unbalanced20_random0	51	*	16%	51	*	16%	52	*	19%	52	*	19%
unbalanced20_random1	52	*	18%	52	*	18%	52	*	19%	52	*	17%
unbalanced20_random2	51	*	16%	51	*	17%	52	*	19%	53	*	20%

Table 5.7: Computational results using exact solvers CPLEX 12.6.3 and Gurobi 6.5.2 with relative gap 4% for solving the transportation and reduced transportation MILP models. The relative gap is (best incumbent - best lower bound) / best incumbent and * indicates timeout. All exact method results are available online in Letsios et al. (2017).

Chapter 6

Heuristics with Performance Guarantees for the Minimum Number of Matches Problem in Heat Recovery Network Design

This work is dedicated, with deepest respect, to the memory of Professor C. A. Floudas. Professor Floudas showed that, given many provably-strong solutions to the minimum number of matches problem, he could design effective heat recovery networks. So the diverse solutions generated by this manuscript directly improve Professor Floudas' method for automatically generating heat exchanger network configurations.

Current MILP solvers are not able to solve the minimum number of matches problem to global optimality for moderately-sized problems. Hence we shift our interest on developing heuristics with performance guarantees that provide near-optimal solutions in coherent time. These methods have guaranteed solution quality and efficient run-time bounds. Furman and Sahinidis (2004) propose a collection of approximation algorithms, for the minimum number of matches problem by exploiting the LP relaxation of an MILP formulation. They present a greedy ap-

proximation algorithm similar to the Cerda et al. (1983) northwest algorithm, which generalises a simple greedy algorithm for the single temperature interval problem that resembles to the tick-off heuristic in Linnhoff and Hindmarsh (1983). They present a unified worst-case analysis of their algorithms' performance guarantees and show a non-constant approximation ratio scaling with the number of temperature intervals and a constant performance guarantee for the single temperature interval problem. This chapter explores this challenging optimisation problem from a graph theoretic perspective and correlate it with other special optimisation problems such as cost flow network and packing problems. In the sequential method, many possible stream configurations are required to evaluate the minimum overall cost (Floudas 1995), so a complementary contribution of this work is a heuristic methodology for producing multiple solutions efficiently. We classify the heuristics based on their algorithmic nature into three categories: (i) relaxation rounding, (ii) water filling, and (iii) greedy packing. The relaxation rounding heuristics we consider are (i) Fractional LP Rounding (FLPR), (ii) Lagrangian Relaxation Rounding (LRR), and (iii) Covering Relaxation Rounding (CRR). The water-filling heuristics are (i)Water-Filling Greedy (WFG), and (ii)Water-Filling MILP (WFM). Finally, the greedy packing heuristics are (i) Largest Heat Match LP-based (LHMLP), (ii) Largest Heat Match Greedy (LHM), (iii) Largest Fraction Match (LFM), and (iv) Shortest Stream (SS). Major ingredients of these heuristics are adaptations of single temperature interval algorithms proposed in this section and mathematical models in Section 5. We use maximum heat LP in a single temperature interval and the extended maximum heat LP for a subset of matches on multiple temperature intervals.
This chapter proceeds as follows: Section 6.1 proposes an improved greedy approximation algorithm for the single temperature interval problem and a greedy algorithm computing maximum heat between two streams and their corresponding big-M parameter. Sections 6.2 - 6.4 present our heuristics for the minimum number of matches problem based on: (i) relaxation rounding, (ii) water filling, and (iii) greedy packing, respectively, as well as new theoretical performance guarantees. Section 6.5 evaluates experimentally the heuristics and discusses numerical results.

6.1 Approximation algorithms

Approximation algorithms have been developed for two problem classes relevant to process systems engineering: heat exchanger networks (Furman and Sahinidis 2004) and pooling (Dey

	Relaxation Rounding			Water Filling		Greedy Packing			
	FLPR (6.2.1)	LRR (6.2.2)	CRR (6.2.3)	WFM (6.3)	WFG (6.3)	LHM (6.4.2)	LFM (6.4.3)	LHM-LP (6.4.2)	SS (6.4.4)
Single Temperature Interval Problem									
MILP Model (5.6)				✓					
Approximation Algorithm (6.1.1)					✓				
Maximum Heat Computations									
Two Streams, Big-M Parameter (6.1.2)	✓	✓	✓			✓	✓		✓
Single Temperature Interval (5.7.1)				✓	✓				
Multiple Temperature Intervals (5.7.2)			✓					✓	

Table 6.1: Table indicating the single temperature interval problem and maximum heat with match restrictions components used by each heuristic. Each element is associated with a section number. This table may be used as a roadmap to the chapter.

Heuristic	Abbrev.	Section	Performance Guarantee	Running Time
Single Temperature Interval Problem				
Simple Greedy	SG	6.1.1	$2^{\dagger}$ (tight)	$O(nm)$
Improved Greedy	IG	6.1.1	1.5 (tight)	$O(nm)$
Relaxation Rounding Heuristics				
Fractional LP Rounding	FLPR	6.2.1	$O(k)^{\dagger}$, $O(U_{\max})$, $\Omega(n)$	1 LP
Lagrangian Relaxation Rounding	LRR	6.2.2		2 LPs
Covering Relaxation Rounding	CRR	6.2.3		$O(nm)$ ILPs
Water Filling Heuristics				
Water Filling MILP	WFM	6.3 & 5.6	$O(k)^{\dagger}$, $\Omega(k)$	$O(k)$ MILPs
Water Filling Greedy	WFG	6.3 & 6.1.1	$O(k)^{\dagger}$, $\Omega(k)$	$O(nmk)$
Greedy Packing Heuristics				
Largest Heat Match LP-based	LHM-LP	6.4.2	$O(\log n + \log(h_{\max}/\varepsilon))$	$O(n^2m^2)$ LPs
Largest Heat Match Greedy	LHM	6.4.2		$O(n^2m^2k)$
Largest Fraction Match	LFM	6.4.3		$O(n^2m^2k)$
Shortest Stream	SS	6.4.4		$O(nmk)$

Table 6.2: Performance guarantees for the minimum number of matches problem. The performance guarantees marked † are from Furman and Sahinidis (2004); all others are new to this manuscript.

and Gupte 2015). The literature and further explanations are provided in the introduction. Table 6.2 lists performance guarantees for the minimum number of matches problem; most are new to this manuscript.

6.1.1 Improved Approximation Algorithm for a Single Temperature Interval Problem

Furman and Sahinidis (2004) propose a greedy 2-approximation algorithm for the minimum number of matches problem in a single temperature interval. We show that their analysis is

Algorithm 1 Simple Greedy (SG), developed by Furman and Sahinidis (2004), is applicable to one temperature interval only.

1: Sort the streams so that $h_1 \geq h_2 \geq \ldots \geq h_n$ and $c_1 \geq c_2 \geq \ldots \geq c_m$.
2: Set $i = 1$ and $j = 1$.
3: **while** there is remaining heat load to be transferred **do**
4: Transfer $q_{i,j} = \min\{h_i, c_j\}$
5: Set $h_i = h_i - q_{i,j}$ and $c_j = c_j - q_{i,j}$
6: **if** $h_i = 0$, **then** set $i = i + 1$
7: **if** $c_j = 0$, **then** set $j = j + 1$
8: **end while**

tight. We also propose an improved, tight 1.5-approximation algorithm by prioritising matches with equal heat loads and exploiting graph theoretic properties.

The simple greedy (SG) algorithm considers the hot and the cold streams in non-increasing heat load order (Furman and Sahinidis 2004). Initially, the first hot stream is matched to the first cold stream and an amount $\min\{h_1, c_1\}$ of heat is transferred between them. Without loss of generality $h_1 > c_1$, which implies that an amount $h_1 - c_1$ of heat load remains to be transferred from h_1 to the remaining cold streams. Subsequently, the algorithm matches h_1 to c_2, by transferring $\min\{h_1 - c_1, c_2\}$ heat. The same procedure repeats with the other streams until all remaining heat load is transferred.

Furman and Sahinidis (2004) show that Algorithm SG is 2-approximate for one temperature interval. Our new result in Theorem 6.1 shows that this ratio is tight.

Lemma 6.1 concerns the structure of an optimal solution for the single temperature interval problem. It shows that the corresponding graph is acyclic and that the number of matches is related to the number of graph's connected components (trees), if arc directions are ignored.

Lemma 6.1 *Consider an instance H, C of the single temperature interval problem. For each optimal solution* $(\vec{y}^*, \vec{q}^*)$*, there exists an integer* $\ell^* \in [1, \min\{n, m\}]$ *s.t.*

- *if arc directions are ignored, the corresponding graph* $G(\vec{y}^*, \vec{q}^*)$ *is a forest consisting of* ℓ^* *trees, i.e. there are no cycles, and*
- $(\vec{y}^*, \vec{q}^*)$ *contains* $v^* = m + n - \ell^*$ *matches.*

Proof:

Assume that $G(\vec{y}^*, \vec{q}^*)$ contains a cycle, after removing arc directions. Moreover, let $M =$

$\{(i_1,j_1),(i_2,j_1),(i_2,j_2),(i_3,j_2),\ldots,(i_g,j_{g-1}),(i_g,j_g),(i_1,j_g)\}$ be a subset of matches forming a cycle. Denote by $q^*_{\min} = \min\{q^*_{i,j} : (i,j) \in M\}$ the minimum amount of heat transferred via a match in M. Without loss of generality, assume that $q^*_{i_1,j_1} = q^*_{\min}$. Starting from $(\vec{y}^*,\vec{q}^*)$, produce a feasible solution $(\vec{y},\vec{q})$ as follows. Set $q_{i_1,j_1} = 0$, $q_{i_e,j_e} = q^*_{i_e,j_e} - q^*_{\min}$ and $q_{i_e,j_{e-1}} = q^*_{i_e,j_{e-1}} + q^*_{\min}$, for $e = 2,\ldots,g$, as well as $q_{i_1,j_g} = q^*_{i_1,j_g} + q^*_{\min}$. The new solution $(\vec{y},\vec{q})$ is feasible and has a strictly smaller number of matches compared to $(\vec{y}^*,\vec{q}^*)$, which is a contradiction.

Since $G(\vec{y}^*,\vec{q}^*)$ does not contain a cycle, it must be a forest consisting of ℓ^* trees (which we call *bins* from a packing perspective). Let $B = \{1,\ldots,\ell^*\}$ be the set of these trees and M_b the subset of matches in tree $b \in B$. By definition, tree $b \in B$ contains $|M_b|$ matches (edges) and, therefore, $|M_b|+1$ streams (nodes). Furthermore, each stream appears in exactly one tree implying that $\sum_{b=1}^{\ell^*} |M_b| = n+m-\ell^*$. Thus, it holds that the number of matches in $(\vec{y}^*,\vec{q}^*)$ is equal to:

$$v = \sum_{b=1}^{\ell^*} |M_b| = n+m-\ell^*.$$

■

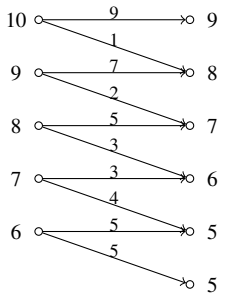
Theorem 6.1 *Algorithm SG achieves an approximation ratio of 2 for the single temperature interval problem and it is tight.*

Proof:

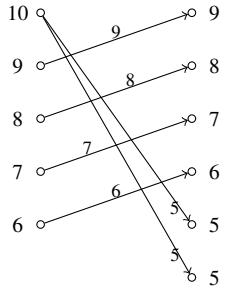
In the algorithm's solution, the number v of matches is equal to the number of steps that the algorithm performs. For each pair of streams $i \in H$ and $j \in C$ matched by the algorithm, at least one has zero remaining heat load exactly after they have been matched. Therefore, the number of steps is at most $v \leq n+m-1$. The optimal solution contains at least $v^* \geq \max\{n,m\}$. Hence, the algorithm is 2-approximate.

Consider a set of n hot streams with heat loads $h_i = 2n+1-i$ for $1 \leq i \leq n$ and $m = n+1$ cold streams with $c_j = 2n-j$, for $1 \leq j \leq m$. As shown in Figure 6.1 for the special case $n = 5$, the algorithm uses $2n$ matches while the optimal solution has $n+1$ matches. Hence, the 2 approximation ratio of Algorithm SG is asymptotically tight. ■

Algorithm IG improves Algorithm SG by: (i) matching the pairs of hot and cold streams with



(a) Algorithm SG: 10



(b) Optimal solution: 6

Figure 6.1: An instance showing the tightness of the 2 performance guarantee for Algorithm SG.

Algorithm 2 Improved Greedy (IG) is applicable to one temperature interval only.

1: **for** each pair of hot stream i and cold stream j s.t. $h_i = c_j$ **do**
2: Transfer h_i amount of heat load (also equal to c_j) between them and remove them.
3: **end for**
4: Run Algorithm SG with respect to the remaining streams.

equal heat loads and (ii) using the acyclic property in the graph representation of an optimal solution.

Lemma 6.2 formalises the benefit of matching stream pairs with equal heat loads and indicates the way of manipulating these matches in the analysis of Algorithm IG and the proof of Theorem 6.2.

Lemma 6.2 *Consider an instance (H,C) of the single temperature interval problem and suppose that there exists a pair of streams $i \in H$ and $j \in C$ such that $h_i = c_j$. Then,*

- *there exists an optimal solution $(\vec{y}^*, \vec{q}^*)$ s.t. $q^*_{i,j} = h_i$, i.e. i and j are matched together,*
- *any ρ-approximate solution for $(H \setminus \{i\}, C \setminus \{j\})$ is also ρ-approximate for (H,C) with the addition of match (i,j).*

Proof:

Consider an optimal solution $(\vec{y}^*, \vec{q}^*)$ in which i and j are not matched solely to each other. Suppose that i is matched with $j_1, j_2, \ldots, j_{m'}$ while j is matched with $i_1, i_2, \ldots, i_{n'}$. Without loss of generality, $q^*_{i,j} = 0$; the case $0 < q^*_{i,j} < h_i$ is treated similarly. Starting from $(\vec{y}^*, \vec{q}^*)$, we

obtain the slightly modified solution $(\vec{y},\vec{q})$ in which i is matched only with j. The c_j units of heat of $i_1, i_2, \ldots, i_{n'}$ originally transferred to j are now exchanged with $j_1, j_2, \ldots, j_{m'}$, which are no longer matched with i. The remaining solution is not modified. Analogously to the proof of Theorem 6.1, we show that there can be at most $n' + m' - 1$ new matches between the n' hot streams (i.e. $i_1, i_2, \ldots, i_{n'}$) and the m' cold streams (i.e. $j_1, j_2, \ldots, j_{m'}$) in $(\vec{y},\vec{q})$. By also taking into account the new match (i, j), we conclude that there exists always a solution in which i is only matched with j and has no more matches than $(\vec{y}^*,\vec{q}^*)$.

Consider an optimal solution $(\vec{y}^*,\vec{q}^*)$ for (H,C), in which there are v^* matches and i is matched only with j. An optimal solution for $(H \setminus \{i\}, C \setminus \{j\})$ contains $v^* - 1$ matches. Suppose that $(\vec{y},\vec{q})$ is the union of a ρ-approximate solution for $(H \setminus \{i\}, C \setminus \{j\})$ and the match (i, j). Let v be the number of matches in $(\vec{y},\vec{q})$. Clearly, $v - 1 \leq \rho \cdot (v^* - 1)$ which implies that $v \leq \rho \cdot v^*$, as $\rho \geq 1$. ∎

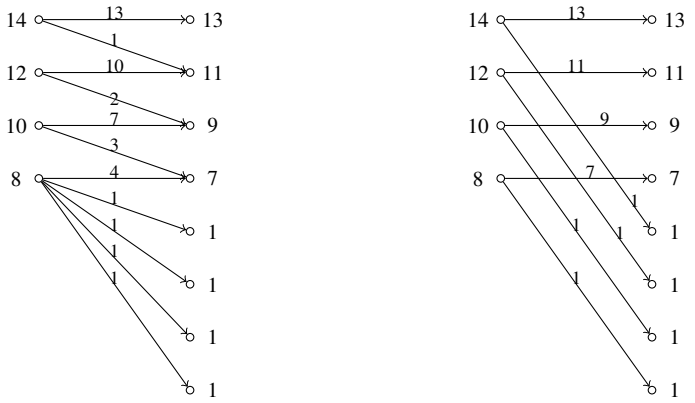
The following theorem shows a tight analysis for Algorithm IG.

Theorem 6.2 *Algorithm IG achieves an approximation ratio of 1.5 for the single temperature interval problem and it is tight.*

Proof:

By Theorem 6.1, Algorithm IG produces a solution $(\vec{y},\vec{q})$ with $v \leq n + m$ matches. Consider an optimal solution $(\vec{y}^*,\vec{q}^*)$. By Lemma 6.1, $(\vec{y}^*,\vec{q}^*)$ consists of ℓ^* trees and has $v^* = n + m - \ell^*$ matches. Due Lemma 6.2, we may assume that instance does not contain a pair of equal hot and cold streams. Hence, each tree in the optimal solution contains at least 3 streams, i.e. $\ell^* \leq (n+m)/3$. Thus, $v^* \geq (2/3)(n+m)$ and we conclude that $v \leq (3/2)v^*$.

For the tightness of our analysis, consider an instance of the problem with n hot streams, where $h_i = 4n - 2i$ for $i = 1, \ldots, n$, and $m = 2n$ cold streams such that $c_j = 4n - 2j - 1$ for $j = 1, \ldots, n$ and $c_j = 1$ for $j = n+1, \ldots, 2n$. Algorithm IG uses $3n$ matches, while the optimal solution uses $2n$ matches. Hence the $3/2$ approximation ratio of the algorithm is tight. Figures 6.2a and 6.2b show the special case with $n = 4$. ∎



(a) Algorithm IG: 11

(b) Optimal solution: 8

Figure 6.2: An instance showing the tightness of the 1.5 performance guarantee for Algorithm IG.

6.1.2 Greedy Algorithm for Big-M Parameter Computation

A common way of computing the big-M parameters is setting $U_{i,j} = \min\{h_i, c_j\}$ for each $i \in H$ and $j \in C$. Gundersen et al. (1997) propose a better method for calculating the big-M parameter. Our novel Greedy Algorithm MHG (Maximum Heat Greedy) obtains tighter $U_{i,j}$ bounds than either the trivial bounds or the Gundersen et al. (1997) bounds by exploiting the transshipment model structure.

Given hot stream i and cold stream j, Algorithm MHG computes the maximum amount of heat that can be feasibly exchanged between i and j in any feasible solution. Algorithm MHG is tight in the sense that there is always a feasible solution where streams i and j exchange exactly $U_{i,j}$ units of heat. Note that, in addition to $U_{i,j}$, the algorithm computes a value $q_{i,s,j,t}$ of the heat exchanged between each hot stream $i \in H$ in temperature interval $s \in T$ and each cold stream $j \in C$ in temperature interval $t \in T$, so that $\sum_{s,t \in T} q_{i,s,j,t} = U_{i,j}$. These $q_{i,s,j,t}$ values are required by greedy packing heuristics in Section 6.4.

Algorithm 3 is a pseudocode of Algorithm MHG. The correctness, i.e. the maximality of the heat exchanged between i and j, is a corollary of the well known maximum flow - minimum cut theorem. Initially, the procedure transfers the maximum amount of heat across the same temperature interval; $q_{i,u,s,u} = \min\{\sigma_{i,u}, \delta_{j,u}\}$ for each $u \in T$. The remaining heat is transferred greedily in a top down manner, with respect to the temperature intervals, by account-

Algorithm 3 Maximum Heat Greedy (MHG)

Input: Hot stream $i \in H$ and cold stream $j \in C$

$\vec{q} \leftarrow \vec{0}$
for $u = 1, 2, \ldots, k$ **do**
    $q_{i,u,j,u} \leftarrow \min\{\sigma_{i,u}, \delta_{j,u}\}$
    $\sigma_{i,u} \leftarrow \sigma_{i,u} - q_{i,u,j,u}$
    $\delta_{j,u} \leftarrow \delta_{j,u} - q_{i,u,j,u}$
end for
for $s = 1, 2, \ldots, k-1$ **do**
    for $t = s+1, s+2, \ldots, k$ **do**
        $q_{i,s,j,t} = \min\{\sigma_{i,s}, \delta_{j,t}, \min_{s \leq u \leq t-1}\{R_u\}\}$
        $\sigma_{i,s} \leftarrow \sigma_{i,s} - q_{i,s,j,t}$
        $\delta_{j,t} \leftarrow \delta_{j,t} - q_{i,s,j,t}$
        for $u = s, s+1, s+2, \ldots, t-1$ **do**
            $R_u \leftarrow R_u - q_{i,s,j,t}$
        end for
    end for
end for
Return $\vec{q}$

ing heat residual capacities. For each temperature interval $u \in T$, the heat residual capacity $R_u = \sum_{i=1}^{n} \sum_{s=1}^{u} \sigma_{i,s} - \sum_{j=1}^{m} \sum_{t=1}^{u} \delta_{j,t}$ imposes an upper bound on the amount of heat that may descend from temperature intervals $1, 2, \ldots, u$ to temperature intervals $u+1, u+2, \ldots, k$.

6.2 Relaxation Rounding Heuristics

This section investigates relaxation rounding heuristics for the minimum number of matches problem. Figure 6.3 shows the main steps in relaxation rounding. These heuristics begin by optimising an efficiently-solvable relaxation of the original MILP. The efficiently-solvable relaxation allows violation of certain constraints, so that the optimal solution(s) is (are) typically infeasible in the original MILP. The resulting infeasible solutions are subsequently rounded to feasible solutions for the original MILP. We consider 3 types of relaxations. Section 6.2.1 relaxes the integrality constraints and proposes fractional LP rounding. Section 6.2.2 relaxes the big-M constraints, i.e. Eqs. (3.10), and uses Lagrangian relaxation rounding. Section 6.2.3 relaxes the heat conservation equations, i.e. Eqs. (3.8)-(3.9), and takes an approach based on covering relaxations.

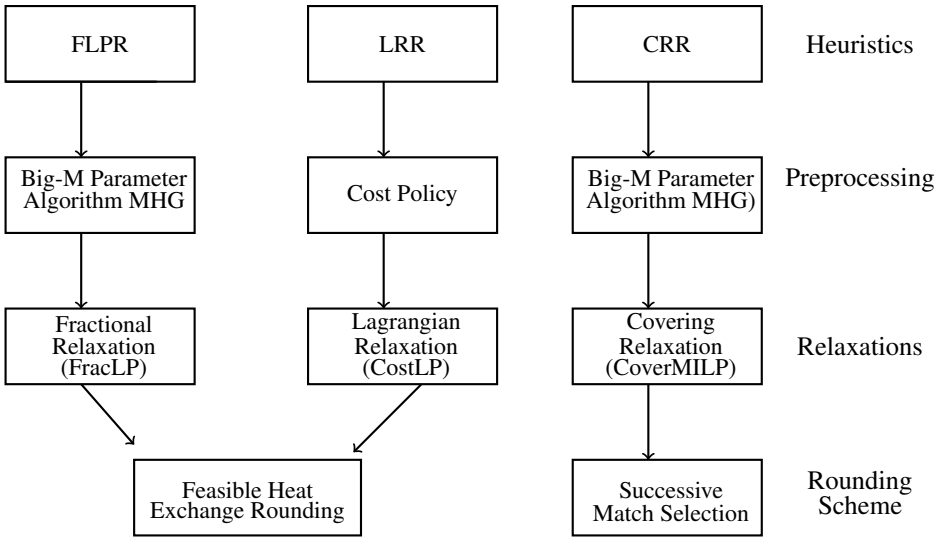


Figure 6.3: The main components of relaxation rounding heuristics are (i) a preprocessing step, (ii) a relaxation, and (iii) a rounding scheme. The preprocessing step constructs the relaxation. Fractional relaxation and covering relaxation require big-M parameter computations, while Lagrangian relaxation minimum cost LP requires cost calculations. FLPR and LRR compute a feasible heat exchange between all streams, i.e. values to variables $q_{i,s,j,t}$, by solving their respective relaxations and round the relaxed solutions according to Algorithm 4. Heuristic CRR adds matches incrementally until it ends up with a feasible solution. Feasibility is determined using the maximum heat LP in Section 5.7.2.

6.2.1 Fractional Linear Programming Rounding

The LP rounding heuristic, originally proposed by Furman and Sahinidis (2004), transforms an optimal fractional solution for the transportation MILP to a feasible integral solution. We show that the fractional LP can be solved efficiently via network flow techniques. We observe that, in the worst case, the heuristic produces a weak solution if it starts with an arbitrary optimal solution of the fractional LP. We derive a novel performance guarantee showing that the heuristic is efficient when the heat of each chosen match (i, j) is close to big-M parameter $U_{i,j}$, in the optimal fractional solution.

Consider the fractional LP obtained by replacing the integrality constraints $y_{i,j} \in \{0,1\}$ of the transportation MILP, i.e. Eqs. (3.7)-(3.12), with the constraints $0 \leq y_{i,j} \leq 1$, for each $i \in H$ and $j \in C$:

$$
\begin{aligned}
\min \sum_{i \in H} \sum_{j \in C} & y_{i,j} \\
\sum_{j \in C} \sum_{t \in T} & q_{i,s,j,t} = \sigma_{i,s} && i \in H, s \in T \\
\sum_{i \in H} \sum_{s \in T} & q_{i,s,j,t} = \delta_{j,t} && j \in C, t \in T \\
\sum_{s,t \in T} & q_{i,s,j,t} \leq U_{i,j} \cdot y_{i,j} && i \in H, j \in C \\
& q_{i,s,j,t} = 0 && i \in H, j \in C, s,t \in T : s \leq t \\
& 0 \leq y_{i,j} \leq 1,\ q_{i,s,j,t} \geq 0 && i \in H, j \in C,\ s,t \in T
\end{aligned}
\tag{FracLP}
$$

FracLP can be solved via minimum cost flow methods. Figure 6.4 illustrates a network N, i.e. a minimum cost flow problem instance, such that finding a minimum cost flow in N is equivalent to optimising the fractional LP. Network N is a layered graph with six layers of nodes: (i) a source node S, (ii) a node for each hot stream $i \in H$, (iii) a node for each pair (i,s) of hot stream $i \in H$ and temperature interval $s \in T$, (iv) a node for each pair (j,t) for each cold stream $j \in C$ and temperature interval $t \in T$, (v) a node for each cold stream $j \in C$, and (vi) a destination node D. We add: (i) the arc (S,i) with capacity h_i for each $i \in H$, (ii) the arc $(i,(i,s))$ with capacity $\sigma_{i,s}$ for each $i \in H$ and $s \in T$, (iii) the arc $((i,s),(j,t))$ with infinite capacity for each $i \in H$, $j \in C$ and $s,t \in T$, (iv) the arc $((j,t),j)$ with capacity $\delta_{j,t}$ for each $j \in H$ and $t \in T$, and (v) the

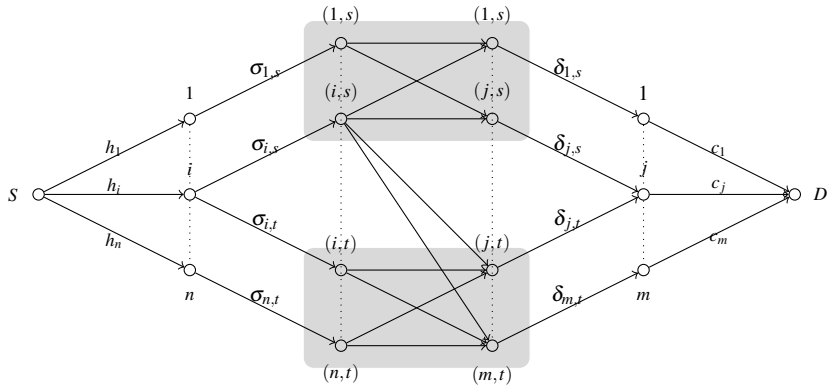


Figure 6.4: Minimum cost network flow formulation of FracLP. The heat is modelled as flow transferred from a source node S to a destination node D. All finite capacities are labelled above the corresponding arcs. The cost is incurred in each arc between node $(i,s) \in H \times T$ and node $(j,t) \in C \times T$ under the condition that heat flows to the same or a lower temperature interval.

Algorithm 4 Fractional LP Rounding (FLPR) (Furman and Sahinidis 2004)

1: $(\vec{y}^f, \vec{q}^f) \leftarrow FractionalLP(I)$
2: $\vec{q} \leftarrow \vec{q}^f$
3: **for** each $i \in H$ and $j \in C$ **do**
4: **if** $\sum_{s,t \in T} q_{i,s,j,t} > 0$ **then**
5: $y_{i,j} \leftarrow 1$
6: **else**
7: $y_{i,j} \leftarrow 0$
8: **end if**
9: **end for**
10: Return $(\vec{y}, \vec{q})$

arc (j,D) with capacity c_j for each $j \in C$. Each arc $((i,s),(j,t))$ has cost $1/U_{i,j}$ for $i \in H$, $j \in C$ and $s,t \in T$. Every other arc has zero cost. Any flow of cost $\sum_i h_i$ on network N is equivalent to a feasible solution for FracLP with the same cost and vice versa.

Furman and Sahinidis (2004) observe that any feasible solution of FracLP can be rounded to a feasible solution of the original problem via Algorithm 4, a simple greedy procedure that we call FLPR. Given a problem instance I, the procedure $FractionalLP(I)$ computes an optimal solution of FracLP. We denote by $(\vec{y}^f, \vec{q}^f)$ the optimal fractional solution.

An inherent drawback of the Furman and Sahinidis (2004) approach is the existence of optimal fractional solutions with unnecessary matches. Theorem 6.3 shows that Algorithm FLPR

performance is bad in the worst case, even for instances with a single temperature interval.

Theorem 6.3 *Algorithm FLPR is $\Omega(n)$-approximate.*

Proof:

We construct a minimum number of matches instance for which Algorithm FLPR produces a solution $\Omega(n)$ times far from the optimal solution. This instance consists of a single temperature interval and an equal number of hot and cold streams, i.e. $n = m$, with the same heat load $h_i = n$ and $c_j = n$, for each $i \in H$ and $j \in C$. Because of the single temperature interval, we ignore the temperature interval indices of the variables $\vec{q}$. In the optimal solution, each hot stream is matched with exactly one cold stream and there are $v^* = n$ matches in total. Given that there exist feasible solutions such that $q_{i,j} = n$, for every possible $i \in H$ and $j \in C$, the algorithm computes the upper bound $U_{i,j} = n$. In an optimal fractional solution, it holds that $q^f_{i,j} = 1$, for each $i \in H$ and $j \in C$. In this case, Algorithm FLPR sets $y_{i,j} = 1$ for each pair of streams $i \in H$, $j \in C$ and uses a total number of matches equal to $v = \sum_{i \in H} \sum_{j \in C} y_{i,j} = \Omega(n^2)$. Therefore, it is $\Omega(n)$-approximate. ∎

Consider an optimal fractional solution to FracLP and suppose that $M \subseteq H \times C$ is the set of pairs of streams exchanging a positive amount of heat. For each $(i, j) \in M$, denote by $L_{i,j}$ the heat exchanged between hot stream i and cold stream j. We define:

$$\phi(M) = \min_{(i,j) \in M} \left\{ \frac{L_{i,j}}{U_{i,j}} \right\}$$

as the *filling ratio*, which corresponds to the minimum portion of an upper bound $U_{i,j}$ filled with the heat $L_{i,j}$, for some match (i, j). Given an optimal fractional solution with filling ratio $\phi(M)$, Theorem 6.4 obtains a $1/\phi(M)$-approximation ratio for FLPR.

Theorem 6.4 *Given an optimal fractional solution with a set M of matches and filling ratio $\phi(M)$, FLPR produces a $(1/\phi(M))$-approximate integral solution.*

Proof:

We denote Algorithm FLPR's solution and the optimal fractional solution by $(\vec{y}, \vec{q})$ and $(\vec{y}^f, \vec{q}^f)$,

respectively. Moreover, suppose that $(\vec{y}^*, \vec{q}^*)$ is an optimal integral solution. Let $M \subseteq H \times C$ be the set of matched pairs of streams by the algorithm, i.e. $y_{i,j} = 1$, if $(i,j) \in M$, and $y_{i,j} = 0$, otherwise. Then, it holds that:

$$\begin{aligned}
\sum_{(i,j)\in M} y_{i,j} &= \sum_{(i,j)\in M} \frac{U_{i,j}}{L_{i,j}} \sum_{s,t\in T} \frac{q_{i,s,j,t}}{U_{i,j}} \\
&\leq \frac{1}{\phi(M)} \sum_{(i,j)\in M} \sum_{s,t\in T} \frac{q^f_{i,s,j,t}}{U_{i,j}} \\
&\leq \frac{1}{\phi(M)} \sum_{(i,j)\in M} y^f_{i,j} \\
&\leq \frac{1}{\phi(M)} \sum_{i\in H} \sum_{j\in C} y^*_{i,j}.
\end{aligned}$$

The first equality is obtained by using the fact that, for each $(i,j) \in M$, it holds that $y_{i,j} = \frac{U_{i,j}}{L_{i,j}} \frac{L_{i,j}}{U_{i,j}}$ and $L_{i,j} = \sum_{s,t\in T} q_{i,s,j,t}$. The first inequality is true by the definition of the filling ratio $\phi(M)$ and the fact that $\vec{q} = \vec{q}^f$. The second inequality holds by the big-M constraint of the fractional relaxation. The final inequality is valid due to the fact that the optimal fractional solution is a lower bound on the optimal integral solution. ∎

In the case where all heat supplies and demands are integers, the integrality of the minimum cost flow polytope and Theorem 6.4 imply that FLPR is $U_{\max}$-approximate, where $U_{\max} = \max_{(i,j)\in H\times C}\{U_{i,j}\}$ is the biggest big-M parameter. Because performance guarantee of FLPR scales with the big-M parameters $U_{i,j}$, we improve the heuristic performance by computing a small big-M parameter $U_{i,j}$ using Algorithm MHG in Section 6.1.2.

6.2.2 Lagrangian Relaxation Rounding

Furman and Sahinidis (2004) design efficient heuristics for the minimum number of matches problem by applying the method of Lagrangian relaxation and relaxing the big-M constraints. This approach generalises Algorithm FLPR by approximating the fractional cost of every possible match $(i,j) \in H \times C$ and solving an appropriate LP using these costs. We present the LP and revisit different ways of approximating the fractional match costs.

In a feasible solution, the fractional cost $\lambda_{i,j}$ of a match (i,j) is the cost incurred per unit of heat

transferred via (i, j). In particular,

$$\lambda_{i,j} = \begin{cases} 1/L_{i,j}, & \text{if } L_{i,j} > 0, \text{ and} \\ 0, & \text{if } L_{i,j} = 0 \end{cases}$$

where $L_{i,j}$ is the heat exchanged via (i, j). Then, the number of matches can be expressed as $\sum_{i,s,j,t} \lambda_{i,j} \cdot q_{i,s,j,t}$. Furman and Sahinidis (2004) propose a collection of heuristics computing a single cost value for each match (i, j) and constructing a minimum cost solution. This solution is rounded to a feasible integral solution equivalently to FLPR.

Given a cost vector $\vec{\lambda}$ of the matches, a minimum cost solution is obtained by solving:

$$\begin{aligned} \min & \sum_{i \in H} \sum_{j \in C} \sum_{s,t \in T} \lambda_{i,j} \cdot q_{i,s,j,t} \\ & \sum_{j \in C} \sum_{t \in T} q_{i,s,j,t} = \sigma_{i,s} && i \in H, s \in T \\ & \sum_{i \in H} \sum_{s \in T} q_{i,s,j,t} = \delta_{j,t} && j \in C, t \in T \\ & q_{i,s,j,t} \geq 0 && i \in H, j \in C, \ s,t \in T \end{aligned} \tag{CostLP}$$

A challenge in Lagrangian relaxation rounding is computing a cost $\lambda_{i,j}$ for each hot stream $i \in H$ and cold stream $j \in C$. We revisit and generalise policies for selecting costs.

6.2.2.0.1 Cost Policy 1 (Maximum Heat) Matches that exchange large amounts of heat incur low fractional cost. This observation motivates selecting $\lambda_{i,j} = 1/U_{i,j}$, for each $(i, j) \in H \times C$, where $U_{i,j}$ is an upper bound on the heat that can be feasibly exchanged between i and j. In this case, Lagrangian relaxation rounding is equivalent to FLPR (Algorithm 4).

6.2.2.0.2 Cost Policy 2 (Bounds on the Number of Matches) This cost selection policy uses lower bounds α_i and β_j on the number of matches of hot stream $i \in H$ and cold stream $j \in C$, respectively, in an optimal solution. Given such lower bounds, at least α_i cost is incurred for the h_i heat units of i and at least β_j cost is incurred for the c_j units of j. On average, each heat unit of i is exchanged with cost at least α_i/h_i and each heat unit of j is exchanged with cost at least β_j/c_j. So, the fractional cost of each match $(i, j) \in H \times C$ can be approximated by

setting $\lambda_{i,j} = \alpha_i/h_i$, $\lambda_{i,j} = \beta_j/c_j$ or $\lambda_{i,j} = \frac{1}{2}(\frac{\alpha_i}{h_i} + \frac{\beta_j}{c_j})$.

Furman and Sahinidis (2004) use lower bounds $\alpha_i = 1$ and $\beta_j = 1$, for each $i \in H$ and $j \in C$. We show that, for any choice of lower bounds α_i and β_j, this cost policy for selecting $\lambda_{i,j}$ is not effective. Even when α_i and β_j are tighter than 1, all feasible solutions of CostLP attain the same cost. Consider any feasible solution $(\vec{y}, \vec{q})$ and the fractional cost $\lambda_{i,j} = \alpha_i/h_i$ for each $(i, j) \in H \times C$. Then the cost of $(\vec{y}, \vec{q})$ in CostLP is:

$$\sum_{i \in H} \sum_{j \in C} \sum_{s,t \in T} \lambda_{i,j} \cdot q_{i,s,j,t} = \sum_{i \in H} \sum_{j \in C} \sum_{s,t \in T} \frac{\alpha_i}{h_i} \cdot q_{i,s,j,t} = \sum_{i \in H} \alpha_i.$$

Since every feasible solution in (CostLP) has cost $\sum_{i \in H} \alpha_i$, Lagrangian relaxation rounding returns an arbitrary solution. Similarly, if $\lambda_{i,j} = \beta_j/c_j$ for $(i, j) \in H \times C$, every feasible solution has cost $\sum_{j \in C} \beta_j$. If $\lambda_{i,j} = \frac{1}{2}(\frac{\alpha_i}{h_i} + \frac{\beta_j}{c_j})$, all feasible solutions have the same cost $1/2 \cdot (\sum_{i \in H} \alpha_i + \sum_{j \in C} \beta_j)$.

6.2.2.0.3 Cost Policy 3 (Existing Solution) This method of computing costs uses an existing solution. The main idea is to use the actual fractional costs for the solution's matches and a non-zero cost for every unmatched streams pair. A minimum cost solution with respect to these costs may improve the initial solution. Suppose that M is the set of matches in the initial solution and let $L_{i,j}$ be the heat exchanged via $(i, j) \in M$. Furthermore, let $U_{i,j}$ be an upper bound on the heat exchanged between i and j in any feasible solution. Then, a possible selection of costs is $\lambda_{i,j} = 1/L_{i,j}$ if $(i, j) \in M$, and $\lambda_{i,j} = 1/U_{i,j}$ otherwise.

6.2.3 Covering Relaxation Rounding

This section proposes a novel covering relaxation rounding heuristic for the minimum number of matches problem. The efficiency of Algorithm FLPR depends on lower bounding the unitary cost of the heat transferred via each match. The goal of the covering relaxation is to use these costs and lower bound the number of matches in a stream-to-stream to basis by relaxing heat conservation. The heuristic constructs a feasible integral solution by solving successively instances of the covering relaxation.

Consider a feasible MILP solution and suppose that M is the set of matches. For each hot stream

$i \in H$ and cold stream $j \in C$, denote by $C_i(M)$ and $H_j(M)$ the subsets of cold and hot streams matched with i and j, respectively, in M. Moreover, let $U_{i,j}$ be an upper bound on the heat that can be feasibly exchanged between $i \in H$ and $j \in C$. Since the solution is feasible, it must be true that $\sum_{j \in C_i(M)} U_{i,j} \geq h_i$ and $\sum_{i \in H_j(M)} U_{i,j} \geq c_j$. These inequalities are necessary, though not sufficient, feasibility conditions. By minimising the number of matches while ensuring these conditions, we obtain a covering relaxation:

$$
\begin{aligned}
\min \sum_{i \in H} \sum_{j \in C} & y_{i,j} \\
\sum_{j \in C} & y_{i,j} \cdot U_{i,j} \geq h_i && i \in H \\
\sum_{i \in H} & y_{i,j} \cdot U_{i,j} \geq c_j && j \in C \\
& y_{i,j} \in \{0,1\} && i \in H, j \in C
\end{aligned}
\tag{CoverMILP}
$$

In certain cases, the matches of an optimal solution to CoverMILP overlap well with the matches in a near-optimal solution for the original problem. Our new Covering Relaxation Rounding (CRR) heuristic for the minimum number of matches problem successively solves instances of the covering relaxation CoverMILP. The heuristic chooses new matches iteratively until it terminates with a feasible set M of matches. In the first iteration, Algorithm CRR constructs a feasible solution for the covering relaxation and adds the chosen matches in M. Then, Algorithm CRR computes the maximum heat that can be feasibly exchanged using the matches in M and stores the computed heat exchanges in $\vec{q}$. In the second iteration, the heuristic performs same steps with respect to the smaller updated instance $(\vec{\sigma}', \vec{\delta}')$, where $\sigma'_{i,s} = \sigma_{i,s} - \sum_{j,t} q_{i,s,j,t}$ and $\delta'_{j,t} = \delta_{j,t} - \sum_{i,s} q_{i,s,j,t}$. The heuristic terminates when all heat is exchanged.

Algorithm 5 is a pseudocode of heuristic CRR. Procedure $CoveringRelaxation(\vec{\sigma}, \vec{\delta})$ produces an optimal subset of matches for the instance of the covering relaxation in which the heat supplies and demands are specified by the vectors $\vec{\sigma}$ and $\vec{\delta}$, respectively. Procedure $MHLP(\vec{\sigma}, \vec{\delta}, M)$ (LP-based Maximum Heat) computes the maximum amount of heat that can be feasibly exchanged by using only the matches in M and is based on solving the LP in Section 5.7.2.

Algorithm 5 Covering Relaxation Rounding (CRR)

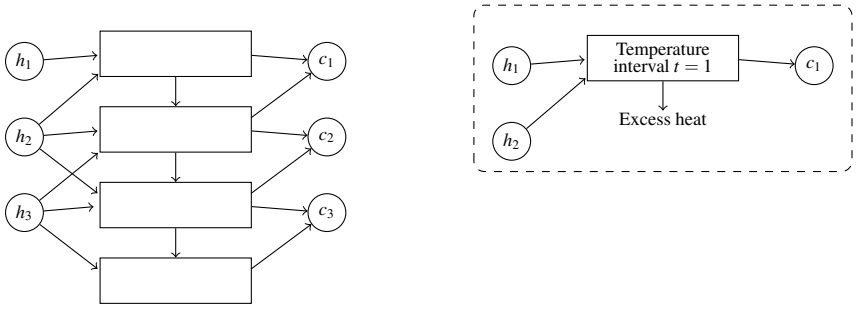
$M \leftarrow \emptyset$
$\vec{q} \leftarrow \vec{0}$
$r \leftarrow \sum_{i \in H} h_i$
while $r > 0$ **do**
    For each $i \in H$ and $s \in T$, set $\sigma'_{i,s} \leftarrow \sigma_{i,s} - \sum_{j \in C} \sum_{t \in T} q_{i,s,j,t}$
    For each $j \in C$ and $t \in T$, set $\delta'_{j,t} \leftarrow \delta_{j,t} - \sum_{i \in H} \sum_{s \in T} q_{i,s,j,t}$
    $M' \leftarrow CoveringRelaxation(\vec{\sigma}', \vec{\delta}')$
    $M \leftarrow M \cup M'$
    $\vec{q} \leftarrow MHLP(\vec{\sigma}, \vec{\delta}, M')$
    $r \leftarrow \sum_{i \in H} h_i - \sum_{i \in H} \sum_{j \in C} \sum_{s,t \in T} q_{i,s,j,t}$
end while

6.3 Water Filling Heuristics

This section introduces *water filling heuristics* for the minimum number of matches problem. These heuristics produce a solution iteratively by exchanging the heat in each temperature interval, in a top down manner. The water filling heuristics use, in each iteration, an efficient algorithm for the single temperature interval problem (see Section 6.3).

Figure 6.5 shows the main idea of a *water filling heuristic* for the minimum number of matches problem with multiple temperature intervals. The problem is solved iteratively in a top-down manner, from the highest to the lowest temperature interval. Each iteration produces a solution for one temperature interval. The main components of a water filling heuristic are: (i) a maximum heat procedure which reuses matches from previous iterations and (ii) an efficient single temperature interval algorithm.

Given a set M of matches and an instance $(\vec{\sigma}_t, \vec{\delta}_t)$ of the problem in the single temperature interval t, the procedure $MHS(\vec{\sigma}_t, \vec{\delta}_t, M)$ (Maximum Heat for Single temperature interval) computes the maximum heat that can be exchanged between the streams in t using only the matches in M. At a given temperature interval t, the MHS procedure solves the LP in Section 5.7.1. The procedure $SingleTemperatureInterval(\vec{\sigma}_t, \vec{\delta}_t)$ produces an efficient solution for the single temperature interval problem with a minimum number of matches and total heat to satisfy one cold stream. $SingleTemperatureInterval(\vec{\sigma}_t, \vec{\delta}_t)$ either: (i) solves the MILP exactly (Water Filling MILP-based or WFM) or (ii) applies the improved greedy approximation Algorithm IG in Section (Water Filling Greedy or WFG). Both water filling heuristics solve instances of the single temperature interval problem in which there is no heat conservation, i.e. the heat supplied by



(a) Top Down Temperature Interval Structure (b) Excess Heat Descending

Figure 6.5: A water filling heuristic computes a solution by exploiting the top down temperature interval structure and moving from the higher to the lower temperature interval. In each temperature interval t, the heuristic isolates the streams with positive heat at t, it matches them and descends the excess heat to the next interval which is sequentially solved.

the hot streams is greater or equal than the heat demanded by the cold streams. The exact WFM uses the MILP proposed in Eqs. (6.1) - (6.6). The greedy heuristic WFG adapts Algorithm IG by terminating when the entire heat demanded by the cold streams has been transferred. After addressing the single temperature interval, the excess heat descends to the next temperature interval. Algorithm 6 represents our water filling approach in pseudocode. Figure 6.6 shows the main components of water filling heuristics.

The reformulated MILP in Eqs. (6.1)-(6.6) solves the single temperature interval problem without heat conservation. It is similar to the MILP in Eqs. (5.22)-(5.28) with heat conservation, except that it does not contain constraints (5.23) while Equalities (5.25) and (5.27) become the inequalities (6.3) and (6.5). In the single temperature interval problem with (without) heat conservation, the total heat of hot streams is equal to (greater than or equal to) the demand of the cold streams. Each water filling algorithm step solves the single temperature interval problem without heat conservation. All heat demands of cold streams are satisfied and the excess heat

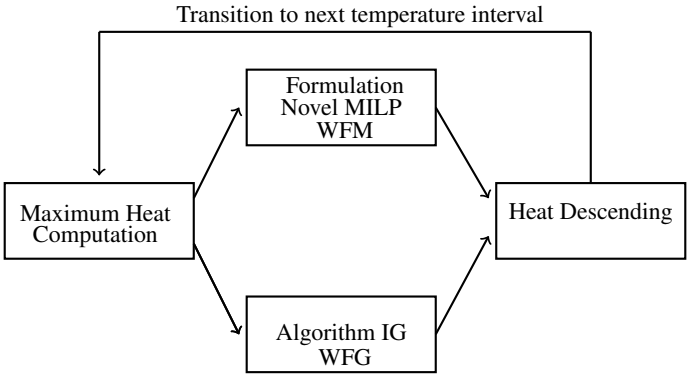


Figure 6.6: Water filling heuristics solve the temperature intervals serially in a top-down manner and keep composition feasible. The main components are (i) a maximum heat computation re-using higher temperature interval matches, (ii) a single temperature interval problem algorithm, and (iii) excess heat descending between consecutive temperature intervals. Heuristic WFM uses the MILP formulation in Section 5.6, Chapter 5 for solving the single temperature interval problem, while heuristic WFG uses the Section 6.3.

supply of hot streams descends to the subsequent temperature interval.

$$\max \sum_{b \in B} x_b \tag{6.1}$$

$$x_b \geq \sum_{j \in C} z_{j,b} \qquad b \in B \tag{6.2}$$

$$\sum_{b \in B} w_{i,b} \leq 1 \qquad i \in H \tag{6.3}$$

$$\sum_{b \in B} z_{j,b} = 1 \qquad j \in C \tag{6.4}$$

$$\sum_{i \in H} w_{i,b} \cdot h_i \geq \sum_{j \in C} z_{j,b} \cdot c_j \qquad b \in B \tag{6.5}$$

$$x_b, w_{i,b}, z_{j,b} \in \{0,1\} \qquad b \in B, i \in H, j \in C \tag{6.6}$$

Theorem 6.5 shows an asymptotically tight performance guarantee for water filling heuristics proportional to the number of temperature intervals. The positive performance guarantee implies the proof of Furman and Sahinidis (2004).

Theorem 6.5 *Algorithms WFG and WFM are $\Theta(k)$-approximate (i.e. both $O(k)$-approximate*

Algorithm 6 Water Filling (WF)

$M \leftarrow \emptyset$
$\vec{q} \leftarrow \vec{0}$
for $t = 1, 2, \ldots, k$ **do**
    if $t \neq 1$ **then**
        $\vec{q}\,' \leftarrow MHS(\vec{\sigma}_t, \vec{\delta}_t, M)$
        $\vec{q} \leftarrow \vec{q} + \vec{q}\,'$
        For each $i \in H$, set $\sigma_{i,t} \leftarrow \sigma_{i,t} - \sum_{j \in C} \sum_{t \in T} q'_{i,j,t}$
        For each $j \in C$, set $\delta_{j,t} \leftarrow \delta_{j,t} - \sum_{i \in H} \sum_{s \in T} q'_{i,j,t}$
    end if
    $(M', \vec{q}\,') \leftarrow SingleTemperatureInterval(\vec{\sigma}_t, \vec{\delta}_t)$
    $M \leftarrow M \cup M'$
    $\vec{q} \leftarrow \vec{q} + \vec{q}\,'$
    if $t \neq k$ **then**
        for $i \in H$ **do**
            $\vec{\sigma}_{i,t+1} \leftarrow \vec{\sigma}_{i,t+1} + (\vec{\sigma}_{i,t} - \sum_j q_{i,j,t})$ (excess heat descending)
        end for
    end if
end for

and $\Omega(k)$-approximate).

Proof:

A water filling algorithm solves an instance of the single temperature interval problem in each temperature interval $t = 1, \ldots, k$. This instance consists of at most n hot streams and at most m cold streams. By Theorem 6.1, algorithms WFG and WFM introduce at most $n+m$ new matches in each temperature interval and produce a solution with $v \leq k(n+m)$ matches. In the optimal solution, each hot and cold stream appears in at least one match which means that $v^* \geq \max\{n, m\}$ matches are chosen in total. So, $v \leq 2k \cdot v^*$.

On the negative side, we show a lower bound on the performance guarantee of algorithms WFG and WFM using the extension of the problem instance in Figure 6.7 with an equal number of hot streams, cold streams and temperature intervals, i.e. $m = n = k$. Each hot stream $i \in H$ has heat supply $\sigma_{i,s} \in \{0, 1\}$ and each cold stream $j \in C$ has heat demand $\delta_{j,t} \in \{0, 1\}$, for each $s, t \in T$. Hot stream i has unit heat in temperature intervals $\{1, \ldots, i\}$ and no supply elsewhere. Cold stream j demands unit heat in temperature intervals $\{j, \ldots, k\}$ and no demand elsewhere. In the optimal solution, hot stream i is matched with cold stream $j = n - i$ and there are $v^* = k$ matches in total. Algorithms WFG and WFM produce the same solution in which hot stream i is matched with cold streams $\{1, 2, \ldots, j\}$, where $j = i$, and there are $v = O(k^2)$ matches in total.

■

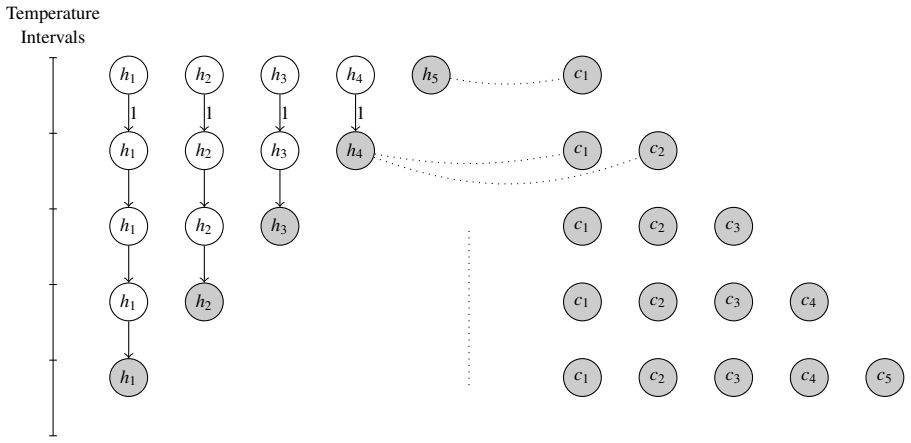


Figure 6.7: An instance showing (asymptotically) the tightness of the $O(k)$ performance guarantee for greedy packing heuristics. In this instance, it holds that $n = m = k$ and every heat supply and heat demand belongs to $\{0,1\}$ in each temperature interval. In the optimal solution, hot stream i is matched with cold stream $j = n - i$ and there are n matches. In the algorithm's solution, hot stream i is matched with cold streams $1,\ldots,m$ and there are $\Omega(n^2)$ matches in total.

6.4 Greedy Packing Heuristics

This section proposes greedy heuristics motivated by the packing nature of the minimum number of matches problem. Each greedy packing heuristic starts from an infeasible solution with zero heat transferred between the streams and iterates towards feasibility by greedily selecting matches. The two main ingredients of such a heuristic are: (i) a match selection policy and (ii) a heat exchange policy for transferring heat via the matches. Section 6.4.1 observes that a greedy heuristic has a poor worst-case performance if heat residual capacities are not considered. Sections 6.4.2 - 6.4.4 define formally the greedy heuristics: (i) Largest Heat Match First, (ii) Largest Fraction Match First, and (iii) Smallest Stream First. Figure 6.8 shows the main components of greedy packing heuristics.

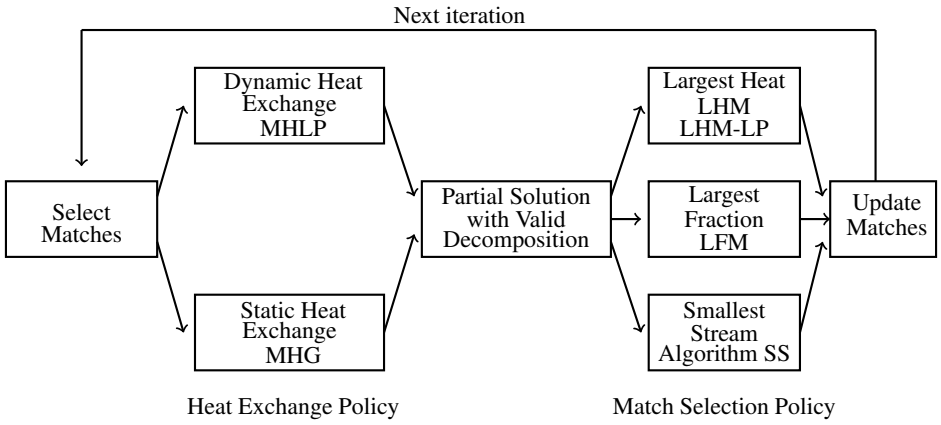


Figure 6.8: Greedy packing heuristics select matches iteratively one by one. The main components of greedy packing heuristics are (i) a heat exchange policy, and (ii) a match selection policy. Greedy packing heuristics apply these policies with respect to all unmatched stream pairs, in each iteration. Options for the heat exchange policy include *dynamic heat exchange*, which solves the Section 5.7.2 maximum heat LP, and *static heat exchange*, which uses the Section 6.1.2 greedy algorithm. Once the heat exchange policy has been applied for every unmatched pair of streams, a match selection policy chooses the new match, e.g. (i) with the largest heat (LHM), (ii) with the largest fraction (LFM), or (iii) of the shortest stream (SS).

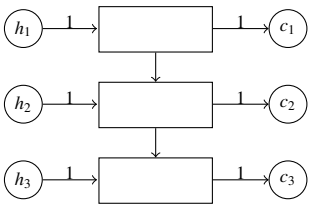


Figure 6.9: A bad example of a non monotonic heuristic. If a heuristic begins by matching h_1 with c_2 and h_2 with c_3, then many unnecessary matches might be required to end up with a feasible solution.

6.4.1 A Pathological Example and Heat Residual Capacities

A greedy match selection heuristic is efficient if it performs a small number of iterations and chooses matches exchanging large heat load in each iteration. Our greedy heuristics perform large moves towards feasibility by choosing good matches in terms of: (i) heat and (ii) stream fraction. An efficient greedy heuristic should also be monotonic in the sense that every chosen match achieves a strictly positive increase on the covered instance size.

The Figure 6.9 example shows a pathological behavior of greedy non-monotonic heuristics. The instance consists of 3 hot streams, 3 cold streams and 3 temperature intervals. Hot stream $i \in H$ has heat supply $\sigma_{i,s} = 1$ for $s = i$ and no supply in any other temperature interval. Cold stream $j \in C$ has heat demand $\delta_{j,t} = 1$ for $t = j$ and no demand in any other temperature interval. Consider the heuristic which selects a match that may exchange the maximum amount of heat in each iteration. The matches (h_1, c_2) and (h_2, c_3) consist the initial selections. In the subsequent iteration, no match increases the heat that can be feasibly exchanged between the streams and the heuristic chooses unnecessary matches.

A sufficient condition enforcing strictly monotonic behavior and avoiding the above pathology, is for each algorithm iteration to satisfy the heat residual capacities. As depicted in Figure 6.10, a greedy heuristic maintains a set M of selected matches together with a decomposition of the original instance I into two instances I^A and I^B. If $I = (H, C, T, \vec{\sigma}, \vec{\delta})$, then it holds that $I^A = (H, C, T, \vec{\sigma}^A, \vec{\delta}^A)$ and $I^B = (H, C, T, \vec{\sigma}^B, \vec{\delta}^B)$, where $\sigma = \vec{\sigma}^A + \vec{\sigma}^B$ and $\vec{\delta} = \vec{\delta}^A + \vec{\delta}^B$. The set M corresponds to a feasible solution for I^A and the instance I^B remains to be solved. In particular, I^A is obtained by computing a maximal amount of heat exchanged by using the

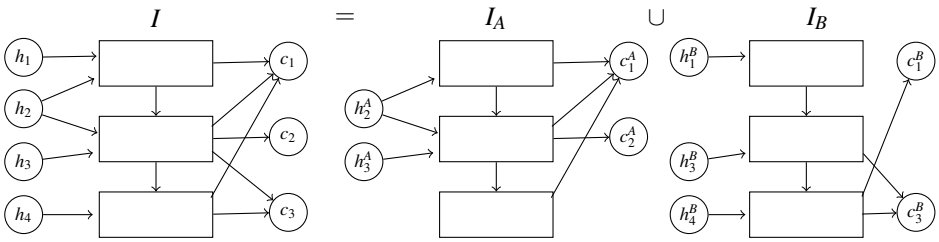


Figure 6.10: Decomposition of a greedy packing heuristic. The problem instance I is the union of the instance I_A already solved by the heuristic and the instance I_B that remains to be solved.

matches in M and I^B is the remaining part of I. Initially, I^A is empty and I^B is exactly the original instance I. A selection of a match increases the total heat exchanged in I^A and reduces it in I^B. A greedy heuristic is monotonic if I^B is feasible in each iteration. Furthermore, I^B is feasible if and only if I^A satisfies the heat residual capacities $R_u = \sum_{i\in H}\sum_{s=1}^{u}\sigma_{i,s} - \sum_{j\in C}\sum_{t=1}^{u}\delta_{j,t}$, for $u \in T$.

6.4.2 Largest Heat Match First

Our Largest Heat Match First heuristics arise from the idea that the matches should individually carry large amounts of heat in a near optimal solution. Suppose that Q_v is the maximum heat that may be transferred between the streams using only a number v of matches. Then, minimising the number of matches is expressed as $\min\{v : Q_v \geq \sum_{i=1}^{n} h_i\}$. This observation motivates the greedy packing heuristic which selects matches iteratively until it ends up with a feasible set M of matches exchanging $\sum_{i=1}^{n} h_i$ units of heat. In each iteration, the heuristic chooses a match maximising the additional heat exchanged. Our two variants of largest heat matches heuristics are: (i) LP-based Largest Heat Match (LHM-LP) and (ii) Greedy Largest Heat Match (LHM).

Heuristic LHM-LP uses the $MHLP(M)$ (LP-based Maximum Heat) procedure to compute the maximum heat that can be transferred between the streams using only the matches in the set M. This procedure is repeated $O(nm)$ times in each iteration, once for every candidate match, and solves an LP incorporating the proposed heat residual capacities. Algorithm 7 is an LHM-LP heuristic using the LP in Section 5.7.2. The algorithm maintains a set M of chosen matches and selects a new match (i', j') to maximise $MHLP(M \cup (i', j'))$.

Algorithm 7 Largest Heat Match First LP-based (LHM-LP)

1: $M \leftarrow \emptyset$
2: $r \leftarrow \sum_{i \in H} h_i$
3: **while** $r > 0$ **do**
4: $\quad (i', j') \leftarrow \arg\max_{(i,j) \in H \times C \setminus M} \{MHLP(M \cup \{(i,j)\})\}$
5: $\quad M \leftarrow M \cup \{(i', j')\}$
6: $\quad r \leftarrow \sum_{i \in H} h_i - MHLP(M)$
7: **end while**
8: Return M

Lemma 6.3 shows a condition ensuring the strict monotonicity of a greedy heuristic which decomposes any instance I into the instances I^A (already solved) and I^B (remaining to be solved) in each iteration (see Section 6.4.1).

Lemma 6.3 *A greedy heuristic is strictly monotonic if I^B is feasible in each iteration.*

Proof:

Given that I^A is of maximal heat (see Section 6.4.1), any match of M is redundant in any feasible solution of I^B. Since I^B is feasible, there exists a match in $H \times C \setminus M$ whose selection increases the amount of heat exchanged in I^A. ■

Lemma 6.4 states necessary and sufficient conditions for the feasibility of a minimum number of matches instance I. The first set of conditions ensures that heat always flows from the hot side to the same or lower temperature intervals on the cold side. The last condition enforces heat conservation.

Lemma 6.4 *An instance I of the minimum number of matches is feasible if and only if the following conditions hold.*

- *For each $u \in T \setminus \{k\}$, it is the case that $R_u \geq 0$, or equivalently*

$$\sum_{i \in H} \sum_{s=1}^{u} \sigma_{i,s} \geq \sum_{j \in C} \sum_{t=1}^{u} \delta_{j,t}$$

- *It holds that $R_k = 0$, or equivalently*

$$\sum_{i \in H} \sum_{s=1}^{k} \sigma_{i,s} = \sum_{j \in C} \sum_{t=1}^{k} \delta_{j,t}.$$

Proof:

To the first direction, a violation of a condition makes the task of constructing a feasible solution impossible. To the opposite direction, Algorithm MHG in Section 3 constructs a feasible solution for every instance satisfying the conditions; it suffices to consider all the hot and cold streams as one large hot and large cold stream, respectively. The single hot stream has heat load $\sum_{i\in H}\sigma_{i,s}$ in temperature interval $s \in T$ and the single cold stream has heat load $\sum_{j,t}\delta_{j,t}$ in temperature interval $t \in T$. ∎

Given a decomposition of an instance I into instances I^A and I^B, Lemma 6.5 shows that a careful construction of I^A respecting the proposed heat residual capacities in Section 6.4.1 implies that I^B is also feasible.

Lemma 6.5 *Consider a decomposition of a feasible instance I into the instances I^A and I^B. Let R, $\vec{R}^A$ and $\vec{R}^B$ be the corresponding heat residual capacities. If I^A is feasible and it holds that $R^A_u \leq R_u$ for each $u \in T$, then I^B is also feasible.*

Proof:

To show that the Lemma is true, it suffices to show that I^B satisfies the feasibility conditions of Lemma 6.4. Consider a temperature interval $u \in T \setminus \{k\}$. Then,

$$\begin{aligned} R^A_u \leq R_u &\Leftrightarrow \sum_{i\in H}\sum_{s=1}^{u}\sigma^A_{i,s} - \sum_{j\in C}\sum_{t=1}^{u}\delta^A_{j,t} \leq \sum_{i\in H}\sum_{s=1}^{u}\sigma_{i,s} - \sum_{j\in C}\sum_{t=1}^{u}\delta_{j,t} \\ &\Leftrightarrow \sum_{i\in H}\sum_{s=1}^{u}\left(\sigma_{i,s} - \sigma^A_{i,s}\right) \geq \sum_{j\in C}\sum_{t=1}^{u}\left(\delta_{j,t} - \delta^A_{j,t}\right) \\ &\Leftrightarrow \sum_{i\in H}\sum_{s=1}^{u}\sigma^B_{i,s} \geq \sum_{j\in C}\sum_{t=1}^{u}\delta^B_{j,t} \end{aligned}$$

In the same fashion, the fact that $R_k = R^A_k = 0$ implies that $R^B_k = 0$. Hence, I^B is feasible. ∎

Given a set M matches, the LP in Eqs. (6.7)-(6.11) maximises the total stream fraction that can be covered using only matches in M. It is similar to the LP in Eqs. (5.29)-(5.33) in Section 6.4.2, except that the maximum fraction objective function (6.7) replaces the maximum heat objective function (5.29).

$$\max \sum_{(i,s,j,t)\in A(M)} \left(\frac{q_{i,s,j,t}}{h_i} + \frac{q_{i,s,j,t}}{c_j} \right) \tag{6.7}$$

$$\sum_{(j,t)\in V^C_{i,s}(M)} q_{i,s,j,t} \leq \sigma_{i,s} \qquad (i,s) \in V^H(M) \tag{6.8}$$

$$\sum_{(i,s)\in V^H_{j,t}(M)} q_{i,s,j,t} \leq \delta_{j,t} \qquad (j,t) \in V^C(M) \tag{6.9}$$

$$\sum_{(i,s,j,t)\in A_u(M)} q_{i,s,j,t} \leq R_u \qquad u \in T \tag{6.10}$$

$$q_{i,s,j,t} \geq 0 \qquad (i,s,j,t) \in A(M) \tag{6.11}$$

The following theorem shows a performance guarantee for Algorithm LHM-LP using a standard packing argument.

Theorem 6.6 *Algorithm LHM-LP is $O(\log n + \log \frac{h_{\max}}{\varepsilon})$-approximate, where ε is the required precision.*

Proof:

Initially, we show an approximation ratio of $O(\log n + \log h_{\max})$ for the special case of the problem with integer parameters. Then, we generalise the result to decimal parameters.

We denote by v the number of the algorithm's matches and by v^* the number of matches in an optimal solution. To upper bound v, we assign unitary costs to the transferred heat in the algorithm's solution as follows. Algorithm LHM-LP constructs a feasible set M of matches greedily. At each iteration, LHM-LP selects a match whose addition in M maximises the heat that can be feasibly exchanged using the matches in M. For $\ell = 1, \ldots, v$, let M_ℓ be the set of selected matches at the end of the ℓ-th iteration and Q_ℓ be the maximum amount of heat the can be feasibly exchanged between all streams using exactly the matches in M_ℓ. Before the algorithm begins, $M_0 = \emptyset$ and $Q_0 = 0$. The extra amount of transferable heat with the addition of the ℓ-th chosen match is $E_\ell = Q_\ell - Q_{\ell-1}$, for $\ell = 1, \ldots, v$. We set the unitary cost to this part of the algorithm's total heat as $\kappa_\ell = \frac{1}{E_\ell}$. Then, the algorithm's number of matches can be expressed:

$$v = \sum_{\ell=1}^{v} \kappa_\ell \cdot E_\ell. \tag{6.12}$$

Let S_ℓ be the total remaining heat to be transferred when the ℓ-th iteration completes. Then, $S_0 = Q$ and $S_\ell = Q - Q_\ell$, for $\ell = 1, \ldots, v$, where $Q = \sum_{i=1}^{n} h_i$ is the total amount of heat. Note that $S_v = 0$ because the algorithm produces a feasible solution. Since the algorithm chooses the match that results in the highest increase of transferred heat in each iteration, it must be the case that $E_1 \geq \ldots \geq E_v$ or equivalently $\kappa_1 \leq \ldots \leq \kappa_v$. At the end of the ℓ-th iteration, the remaining heat can be transferred using at most v^* additional matches by selecting the remaining matches of an optimal solution. Using a simple average argument we get that $\kappa_\ell \leq \frac{v^*}{S_{\ell-1}}$, for each $\ell = 1, \ldots, v$. Thus, Eq. (6.12) implies:

$$v \leq \sum_{\ell=1}^{v} \left(\frac{v^*}{S_{\ell-1}} \right) \cdot E_\ell = \sum_{\ell=1}^{v} \left(\frac{E_\ell}{Q - Q_{\ell-1}} \right) \cdot v^*. \tag{6.13}$$

By the integrality of the minimum cost network flow polytope, each value E_ℓ is an integer, for $\ell = 1, \ldots, v$. Hence,

$$\frac{E_\ell}{Q - Q_{\ell-1}} = \sum_{e=1}^{E_\ell} \frac{1}{Q - Q_{\ell-1}} \leq \sum_{e=1}^{E_\ell} \frac{1}{Q - Q_{\ell-1} - e + 1}.$$

Given that $Q_\ell = Q_{\ell-1} + E_\ell$,

$$\frac{E_\ell}{Q - Q_{\ell-1}} \leq \sum_{e=Q_{\ell-1}}^{Q_\ell - 1} \frac{1}{Q - e}. \tag{6.14}$$

Inequalities (6.13) and (6.14) imply:

$$v \leq \left(\sum_{e=1}^{Q} \frac{1}{e} \right) \cdot v^*.$$

Using the asymptotic bound $\sum_{e=1}^{Q} \frac{1}{e} = O(\log Q)$ of harmonic series and the fact that $Q \leq n \cdot h_{max}$, we conclude that the algorithm is $O(\log n + \log h_{\max})$-approximate, where $h_{\max} = \max_{i \in H}\{h_i\}$ is the maximum heat of a hot stream.

Generalising to decimal parameters, the algorithm is $O(\log n + \log \frac{h_{\max}}{\varepsilon})$, where ε is the precision required for solving the problem instance. The reasoning is the same except that, instead of considering integer units, we consider ε units to extend inequality (6.14). ∎

LHM-LP heuristic is polynomial-time in the worst case. The i-th iteration solves $nm - i + 1$ LP instances which sums to solving a total of $\sum_{i=1}^{nm}(nm - i + 1) = O(n^2 m^2)$ LP instances in

Algorithm 8 Largest Heat Match First Greedy (LHM)

$M \leftarrow \emptyset$
$\vec{q} \leftarrow \vec{0}$
$r \leftarrow \sum_{i \in H} h_i$
while $r > 0$ **do**
    $(i', j', \vec{q}\,') \leftarrow \arg\max_{(i,j) \in H \times C \setminus M} \{MHG(\vec{\sigma}, \vec{\delta}, i, j)\}$
    $M \leftarrow M \cup \{(i', j')\}$
    $\vec{q} \leftarrow \vec{q} + \vec{q}\,'$
    For each $s \in T$, set $\sigma_{i',s} \leftarrow \sigma_{i',s} - \sum_{t \in T} q'_{i',s,j',t}$
    For each $t \in T$, set $\delta_{j',t} \leftarrow \delta_{j',t} - \sum_{s \in T} q'_{i',s,j',t}$
    $r \leftarrow r - \sum_{s,t \in T} q'_{i',s,j',t}$
end while
Return M

the worst case. However, for large instances, the algorithm is time consuming because of this iterative LP solving. So, we also propose an alternative, time-efficient greedy approach. The new heuristic version builds a solution by selecting matches and deciding the heat exchanges, without modifying them in subsequent iterations.

The new approach for implementing the heuristic, that we call LHM, requires the $MHG(\vec{\sigma}, \vec{\delta}, i, j)$ procedure. Given an instance $(\vec{\sigma}, \vec{\delta})$ of the problem, it computes the maximum heat that can be feasibly exchanged between hot stream $i \in H$ and cold stream $j \in C$, as defined in Section 6.1.2. The procedure also computes a corresponding value $q_{i,s,j,t}$ of heat exchanged between $i \in H$ in temperature interval $s \in T$ and $j \in C$ in temperature interval $t \in T$. LHM maintains a set M of currently chosen matches together with their respective vector $\vec{q}$ of heat exchanges. In each iteration, it selects the match (i', j') and heat exchanges q' between i' and j' so that the value $MHG(\vec{\sigma}, \vec{\delta}, i', j')$ is maximum. Algorithm 8 is a pseudocode of this heuristic.

6.4.3 Largest Fraction Match First

The heuristic *Largest Fraction Match First* (LFM) exploits the bipartite nature of the problem by employing matches which exchange large fractions of the stream heats. Consider a feasible solution with a set M of matches. Every match $(i, j) \in M$ covers a fraction $\sum_{s,t \in T} \frac{q_{i,s,j,t}}{h_i}$ of hot stream $i \in H$ and a fraction $\sum_{s,t \in T} \frac{q_{i,s,j,t}}{c_j}$ of cold stream $j \in C$. The total covered fraction of all streams is equal to $\sum_{(i,j) \in M} \sum_{s,t \in T} \left(\frac{q_{i,s,j,t}}{h_i} + \frac{q_{i,s,j,t}}{c_j} \right) = n + m$. Suppose that F_v is the maximum amount of total stream fraction that can be covered using no more than v matches. Then, minimising the number of matches is expressed as $\min\{v : F_v \geq n + m\}$. Based on this observation,

the main idea of LFM heuristic is to construct iteratively a feasible set of matches, by selecting the match covering the largest fraction of streams, in each iteration. That is, LFM prioritises proportional matches in a way that high heat hot streams are matched with high heat cold streams and low heat hot streams with low heat cold streams. In this sense, it generalises the idea of Algorithm IG for the single temperature interval problem (see Section 6.3), according to which it is beneficial to match streams of (roughly) equal heat.

An alternative that would be similar to LHM-LP is an LFM heuristic with an $MFLP(M)$ (LP-based Maximum Fraction) procedure computing the maximum fraction of streams that can be covered using only a given set M of matches. Like the LHM-LP heuristic, this procedure would be based on solving an LP, except that the objective function maximises the total stream fraction. The LFM heuristic can be also modified to attain more efficient running times using Algorithm MHG, as defined in Section 6.1.2. In each iteration, the heuristic selects the match (i, j) with the highest value $\frac{U'_{i,j}}{h_i} + \frac{U'_{i,j}}{c_j}$, where $U'_{i,j}$ is the maximum heat that can be feasibly exchanged between i and j in the remaining instance.

6.4.4 Smallest Stream Heuristic

Subsequently, we propose *Smallest Stream First* (SS) heuristic based on greedy match selection, which also incorporates stream priorities so that a stream is involved in a small number of matches. Let α_i and β_j be the number of matches of hot stream $i \in H$ and cold stream $j \in C$, respectively. Minimising the number of matches problem is expressed as $\min\{\sum_{i \in H} \alpha_i\}$, or equivalently $\min\{\sum_{j \in C} \beta_j\}$. Based on this observation, we investigate heuristics that specify a certain order of the hot streams and match them one by one, using individually a small number of matches. Such a heuristic requires: (i) a stream ordering strategy and (ii) a match selection strategy. To reduce the number of matches of small hot streams, heuristic SS uses the order $h_1 \leq h_2 \leq \ldots \leq h_n$.

In each iteration, the next stream is matched with a low number of cold streams using a greedy match selection strategy; we use greedy LHM heuristic. Observe that SS heuristic is more efficient in terms of running time compared to the other greedy packing heuristics, because it solves a subproblem with only one hot stream in each iteration. Algorithm 9 is a pseudocode of SS heuristic. Note that other variants of ordered stream heuristics may be obtained in a similar

Algorithm 9 Smallest Steam First (SS)

Sort the hot streams in non-decreasing order of their heat loads, i.e. $h_1 \leq h_2 \leq \ldots \leq h_n$.
$M \leftarrow \emptyset$
$\vec{q} \leftarrow \vec{0}$
for $i \in H$ **do**
    $r \leftarrow h_i$
    while $r > 0$ **do**
        $(i, j', \vec{q}\,') \leftarrow \arg\max_{j \in C}\{MHG(\vec{\sigma}, \vec{\delta}, i, j)\}$
        $M \leftarrow M \cup \{(i, j')\}$
        $\vec{q} \leftarrow \vec{q} + \vec{q}\,'$
        For each $s \in T$, set $\sigma_{i,s} \leftarrow \sigma_{i,s} - \sum_{t \in T} q'_{i,s,j',t}$
        For each $t \in T$, set $\delta_{j',t} \leftarrow \delta_{j',t} - \sum_{s \in T} q'_{i,s,j',t}$
        $r \leftarrow r - \sum_{s,t \in T} q'_{i',s,j',t}$
    end while
end for
Return M

way. The heuristic uses the MHG algorithm in Section 6.1.2.

6.5 Numerical Results

This section evaluates the proposed heuristics on three test sets. Section 6.5.1 provides information on system specifications and benchmark instances. Section 5.8 presents computational results of exact methods and shows that commercial, state-of-the-art approaches have difficult solving moderately-sized instances to global optimality. Section 6.5.2 & 6.5.3 evaluate experimentally the heuristic methods and compare the obtained results with those previously reported in the literature. All result tables are provided in 6.7. Letsios et al. (2017) provide test cases and source code for the paper's computational experiments.

6.5.1 System Specification and Benchmark Instances

All computations are run on an Intel Core i7-4790 CPU 3.60GHz with 15.6 GB RAM running 64-bit Ubuntu 14.04. CPLEX 12.6.3 and Gurobi 6.5.2 solve the minimum number of matches problem exactly. The mathematical optimisation models and heuristics are implemented in Python 2.7.6 and Pyomo 4.4.1 (Hart et al. 2011, 2012).

We use problem instances from two existing test sets (Furman and Sahinidis 2004, Chen et al.

2015a). We also generate two collections of larger test cases. The smaller of the two sets uses work of Grossmann (2017). The larger of the two sets was created using our own random generation method.

6.5.2 Heuristic Methods

We implement the proposed heuristics using Python and develop the LP models with Pyomo (Hart et al. 2011, 2012). We use CPLEX 12.6.3 with default settings to solve all LP models within the heuristic methods. Letsios et al. (2017) make the source code available. The following discussion covers the 48 problems with 43 streams or fewer. Section 6.5.3 discusses the 3 examples with 160 streams each.

The difficulty of solving the minimum number of matches problem to global optimality motivates the design of heuristic methods and approximation algorithms with proven performance guarantees. Initially we want to highlight the difficulty of solving these problems due to the existence of big-M constraints as shown in Table 6.3. The method proposed in this work to compute the big-M parameters improves the fractional relaxation results on all instances. Tables 6.4 and 6.5 contain the computed objective value and CPU times, respectively, of the heuristics for all test cases. For the challenging Chen et al. (2015b,a) and Grossmann (2017) test sets, heuristic LHM-LP always produces the best solution. The LHM-LP running time is significantly higher compared to all heuristics due to the iterative LP solving, despite the fact that it is guaranteed to be polynomial in the worst case. Alternatively, heuristic SS produces the second best heuristic result with very efficient running times in the Chen et al. (2015b,a) and Grossmann (2017) test sets. Figure 6.11 depicts the performance ratio of the proposed heuristics using a box and whisker plot, where the computed objective value is normalised with the one found by CPLEX for the transshipment MILP. Figure 6.12 shows a box and whisker plot of the CPU times of all heuristics in $\log_{10}$ scale normalised by the minimum CPU time for each test case. Figure 6.13 shows a line chart verifying that our greedy packing approach produces better solutions than the relaxation rounding and water filling ones.

Table 6.6 contains the heuristic results reported by Furman and Sahinidis (2004) and the ones obtained with our improved version of the FLPR, LRR, and WFG heuristics of Furman and Sahinidis (2004). Our versions of FLPR, LRR, and WFG perform better for the Furman and

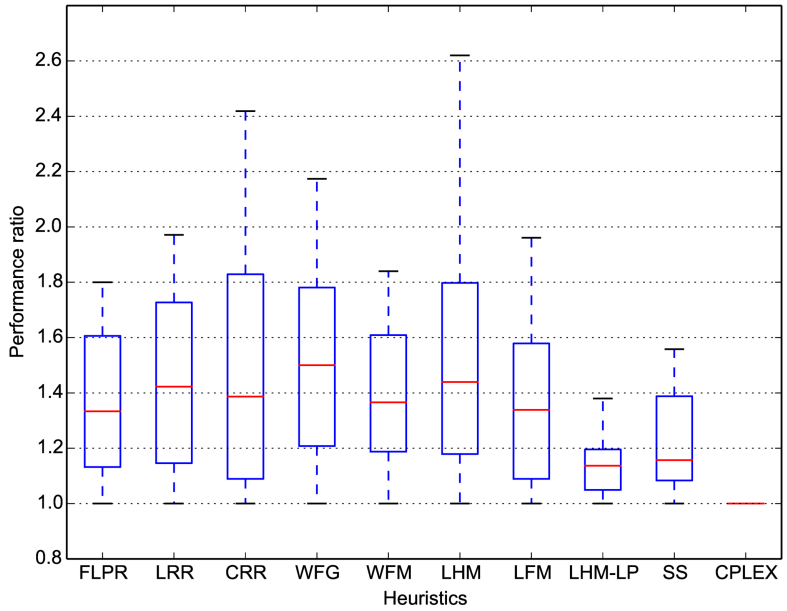


Figure 6.11: Box and whisker diagram of 48 heuristic performance ratios, i.e. computed solution / best known solution for the problems with 43 streams or fewer.

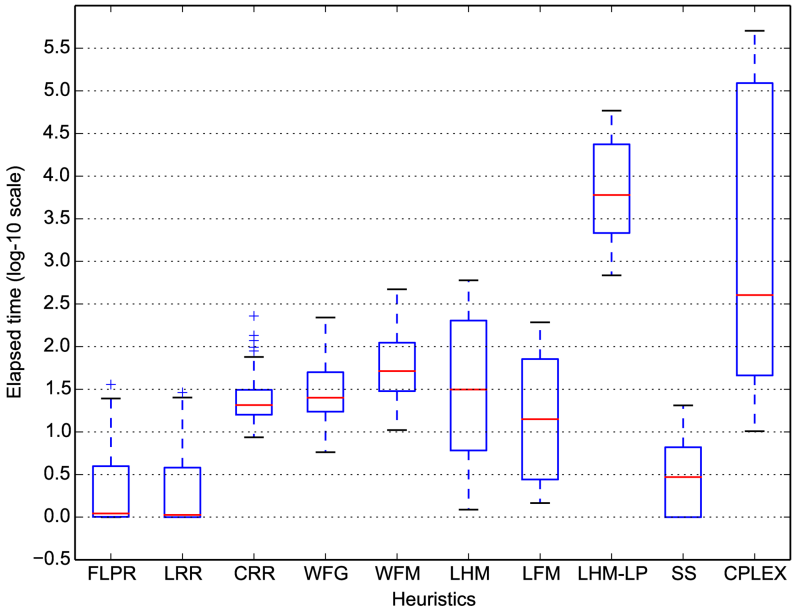


Figure 6.12: Box and whisker diagram of 48 CPU times ($\log_{10}$ scale) normalised by the minimum CPU time for each test case.

Sahinidis (2004) test set because of our new Algorithm MHG for tightening the big-M parameters. For example, out of the 26 instances, our version of FLPR performs strictly better than the

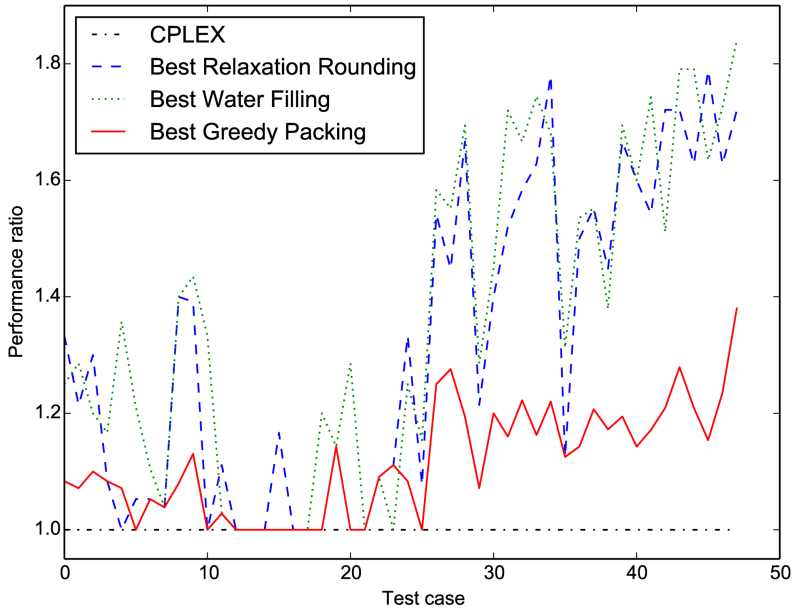


Figure 6.13: Line chart comparing the performance ratio, i.e. computed solution / best known solution, of the best computed result by heuristic methods: relaxation rounding, water filling, and greedy packing. This graph applies to the 48 problems with 43 streams or fewer.

Furman and Sahinidis (2004) version 20 times and worse only once (`10sp1`). To further explore the effect of the big-M parameter, Table 6.7 shows how different computations for the big-M parameter change the FLPR and LRR performance. Table 6.7 also demonstrates the importance of the big-M parameter on the transportation MILP fractional relaxation quality.

In particular, Table 6.7 compares the three big-M computation methods discussed in Section 6.1.2: (i) the trivial bounds, (ii) the Gundersen et al. (1997) method, and (iii) our greedy Algorithm MHG. Our greedy maximum heat algorithm dominates the other approaches for computing the big-M parameters. Algorithm MHG also outperforms the other two big-M computation methods by finding smaller feasible solutions via both Fractional LP Rounding and Lagrangian Relaxation Rounding. In the 48 test cases, Algorithm MHG produces the best FLPR and LRR feasible solutions in 46 and 43 test cases, respectively. Algorithm MHG is strictly best for 33 FLPR and 32 LRR test cases. Finally, Algorithm MHG achieves the tightest fractional MILP relaxation for all test instances.

Figure 6.11 and Table 6.4 show that our new CRR heuristic is competitive with the other relaxation rounding heuristics, performing as well or better than FLPR or LRR in 19 of the 48 test

cases and strictly outperforming both FLPR and LRR in 8 test cases. Although CRR solves a sequence of MILPs, Figure 6.12 and Table 6.5 show that its running time is efficient compared to the other relaxation rounding heuristics.

Our water filling heuristics are equivalent to or better than Furman and Sahinidis (2004) for 25 of their 26 test set instances (all except `7sp2`). In particular, our Algorithm WFG is strictly better than their WFG in 18 of 26 instances and is worse in just one. This improvement stems from the new 1.5-approximation algorithm for the single temperature interval problem (see Section 6.3). The novel Algorithm WFM is competitive with Algorithm WFG and produces equivalent or better feasible solutions for 37 of the 48 test cases. In particular, WFM has a better performance ratio than WFG (see Figure 6.11) and WFM is strictly better than WFG in all but 1 of the Grossmann (2017) instances. The strength of WFM highlights the importance of our new MILP formulation in Eqs. (5.22)-(5.28). At each iteration, WFM solves an MILP without big-M constraints and therefore has a running time in the same order of magnitude as its greedy counterpart WFG (see Figure 6.12).

In summary, our heuristics obtained via the relaxation rounding and water filling methods improve the corresponding ones proposed by Furman and Sahinidis (2004). Furthermore, greedy packing heuristics achieve even better results in more than 90% of the test cases.

6.5.3 Larger Scale Instances

Although CPLEX and Gurobi do not converge to global optimality for many of the Furman (2000), Chen et al. (2015b,a), and Grossmann (2017) instances, the solvers produce the best heuristic solutions in all test cases. But the literature instances are only moderately sized. We expect that the heuristic performance improves relative to the exact approaches as the problem sizes increase. Towards a more complete numerical analysis, we randomly generate 3 larger scale instances with 160 streams each.

For larger problems, the running time may be important to a design engineer (Linnhoff and Hindmarsh 1983). We apply the least time consuming heuristic of each type for solving the larger scale instances, i.e. apply relaxation rounding heuristic FLPR, water filling heuristic WFG, and greedy packing heuristic SS. We also solve the transshipment model using CPLEX

12.6.3 with a 4h timeout. The results are in Table 6.8.

For these instances, greedy packing SS computes a better solution than the relaxation rounding FLPR heuristic or the water filling WFG heuristic, but SS has larger running time. In instance `large-scale1`, greedy packing SS computes 218, a better solution than the CPLEX value 219. Moreover, the CPLEX heuristic spent the first 1hr of computation time at solution 257 (18% worse than the solution SS obtains in 10 minutes) and the next 2hr of computation time at solution 235 (8% worse than the solution SS obtains in 10 minutes). Any design engineer wishing to interact with the results would be frustrated by these times.

In instance `large-scale2`, CPLEX computes a slightly better solution (239) than the SS heuristic (242). But the good CPLEX solution is computed slightly before the 4h timeout. For more than 3.5hr, the best CPLEX heuristic is 273 (13% worse than the solution SS obtains in 10 minutes). Finally, in instance `large-scale0`, CPLEX computes a significantly better solution (175) than the SS heuristic (233). But CPLEX computes the good solution after 2h and the incumbent is similar to the greedy packing SS solution for the first 2 hours. These findings demonstrate that greedy packing approaches are particularly useful when transitioning to larger scale instances.

6.6 Conclusion

In his PhD thesis, Professor Floudas showed that, given a solution to the minimum number of matches problem, he could solve a nonlinear optimisation problem designing effective heat recovery networks. But the sequential HENS method cannot guarantee that promising minimum number of matches solutions will be optimal (or even feasible!) to Professor Floudas' nonlinear optimisation problem. Since the nonlinear optimisation problem is relatively easy to solve, we propose generating many good candidate solutions to the minimum number of matches problem. This manuscript develops nine heuristics with performance guarantees to the minimum number of matches problem. Beyond approximation algorithms, numerical analysis shows that our improved algorithms for the single temperature interval, our new way of computing tighter big-M parameters and the other enhancements, improve the performance of relaxation rounding and water filling heuristics.

6.7 Experimental Results

Test Case	Fractional Relaxation		
	Simple	GTA97	LKM17
Furman and Sahinidis (2004) Test Set			
10sp-la1	7.04	7.57	**8.35**
10sp-ol1	8.29	8.86	**9.94**
10sp1	7.11	7.24	**7.39**
12sp1	10.06	10.12	**10.26**
14sp1	8.79	8.92	**9.06**
15sp-tkm	11.01	11.47	**14.31**
20sp1	**11.75**	**11.75**	**11.75**
22sp-ph	20.15	20.89	**22.23**
22sp1	13.66	14.04	**15.86**
23sp1	13.31	**13.40**	**13.40**
28sp-as1	27.51	27.96	**28.45**
37sp-yfyv	31.96	31.93	**32.28**
4sp1	4.03	4.06	**4.25**
6sp-cf1	4.10	4.10	**4.18**
6sp-gg1	**3.00**	**3.00**	**3.00**
6sp1	**4.00**	**4.00**	**4.00**
7sp-cm	6.61	7.15	**8.40**
7sp-s1	7.83	7.83	**10.00**
7sp-torw1	5.68	5.84	**6.56**
7sp1	5.00	5.00	**5.01**
7sp2	**4.37**	**4.37**	**4.37**
7sp4	7.01	7.01	**7.11**
8sp-fs1	6.89	7.50	**8.69**
8sp1	6.15	6.22	**6.30**
9sp-al1	7.04	7.57	**8.35**
9sp-has1	6.91	7.14	**9.98**
Chen et al. (2015b,a) Test Set			
balanced10	13.51	13.92	**15.29**
balanced12	15.69	16.16	**17.48**
balanced15	18.84	19.31	**21.56**
balanced5	8.09	8.40	**8.95**
balanced8	11.54	11.88	**12.76**
unbalanced10	14.31	15.05	**16.96**
unbalanced15	19.62	20.59	**23.17**
unbalanced17	21.90	23.53	**27.48**
unbalanced20	25.89	27.72	**32.43**
unbalanced5	8.34	8.82	**10.93**
Grossmann (2017) Test Set			
balanced12_random0	15.76	16.21	**17.51**
balanced12_random1	15.67	16.06	**17.37**
balanced12_random2	15.67	16.14	**17.40**
balanced15_random0	18.59	19.19	**21.47**
balanced15_random1	18.86	19.38	**21.59**
balanced15_random2	18.73	19.41	**21.95**
unbalanced17_random0	22.48	23.97	**27.64**
unbalanced17_random1	22.43	23.89	**27.66**
unbalanced17_random2	21.99	23.61	**27.74**
unbalanced20_random0	26.02	27.91	**32.49**
unbalanced20_random1	26.01	27.74	**32.64**
unbalanced20_random2	25.68	27.67	**32.60**

Table 6.3: This table compares the effect of three different methods for computing the big-M parameter U_{ij}: (i) simple greedy, (ii) the Gundersen et al. (1997) (GTA97) method and (iii) our greedy Algorithm MHG (LKM17). LKM17 achieves the tightest fractional relaxation for all test instances.

Test Case	Relaxation Rounding			Water Filling		Greedy Packing				CPLEX
	FLPR	LRR	CRR	WFG	WFM	LHM	LFM	LHM-LP	SS	
Furman and Sahinidis (2004) Test Set										
10sp-la1	16	17	16	16	15	15	*13*	14	*13*	**12**
10sp-ol1	18	18	17	18	18	19	17	*15*	16	**14**
10sp1	18	15	13	15	12	17	15	*11*	12	**10**
12sp1	16	17	*13*	14	15	18	17	*13*	*13*	**12**
14sp1	**14**	20	18	21	19	16	16	15	16	**14**
15sp-tkm	20	22	20	23	25	21	22	**19**	21	**19**
20sp1	22	*20*	23	24	21	*20*	21	*20*	21	**19***
22sp-ph	*27*	28	28	*27*	28	37	*27*	*27*	*27*	**26**
22sp1	35	37	36	42	35	34	31	*27*	29	**25***
23sp1	32	32	40	50	33	32	32	*26*	*26*	**23***
28sp-as1	**30**	**30**	**30**	40	45	50	50	**30**	40	**30**
37sp-yfyv	44	40	45	*37*	43	55	46	40	*37*	**36**
4sp1	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**
6sp-cf1	**6**	**6**	**6**	**6**	7	**6**	**6**	**6**	**6**	**6**
6sp-gg1	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**	**3**
6sp1	8	7	9	9	**6**	**6**	**6**	**6**	**6**	**6**
7sp-cm1	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
7sp-s1	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
7sp-torw1	11	12	10	12	12	12	11	11	**10**	**10**
7sp1	*8*	10	11	10	*8*	9	*8*	*8*	*8*	**7**
7sp2	**7**	**7**	**7**	9	9	**7**	**7**	**7**	**7**	**7**
7sp4	**8**	**8**	**8**	**8**	**8**	10	**8**	**8**	**8**	**8**
8sp-fs1	13	13	*12*	*12*	*12*	15	*12*	14	*12*	**11**
8sp1	11	11	10	14	**9**	10	10	10	10	**9**
9sp-al1	16	17	16	16	15	15	*13*	14	*13*	**12**
9sp-has1	15	14	15	15	16	15	14	**13**	15	**13**
Chen et al. (2015b,a) Test Set										
balanced10	39	42	37	42	38	40	42	*30*	35	**24**
balanced12	42	48	53	48	45	48	41	*37*	41	**28***
balanced15	60	69	71	63	61	82	62	*43*	51	**37***
balanced5	18	17	18	18	19	20	18	*15*	19	**14**
balanced8	28	33	35	29	32	29	30	*24*	30	**20**
unbalanced10	38	46	43	46	43	42	35	*29*	33	**25**
unbalanced15	57	64	63	64	60	85	55	*44*	49	**36***
unbalanced17	70	78	73	79	75	86	67	*50*	57	**43***
unbalanced20	89	89	104	84	90	106	80	*61*	68	**51***
unbalanced5	19	20	*18*	21	22	19	*18*	*18*	*18*	**16**
Grossmann (2017) Test Set										
balanced12_random0	42	48	52	44	43	44	45	*32*	42	**28***
balanced12_random1	45	49	53	50	45	47	43	*35*	40	**29***
balanced12_random2	42	49	57	49	40	46	43	*34*	42	**29***
balanced15_random0	60	61	66	67	61	64	63	*43*	53	**36***
balanced15_random1	56	65	71	66	56	65	55	*40*	52	**36***
balanced15_random2	54	69	63	63	61	67	55	*41*	54	**35***
unbalanced17_random0	74	80	86	81	65	102	72	*52*	67	**43***
unbalanced17_random1	74	74	104	84	77	100	70	*55*	56	**44***
unbalanced17_random2	70	79	95	77	77	111	76	*52*	59	**43***
unbalanced20_random0	93	93	109	100	85	115	86	*60*	64	**51***
unbalanced20_random1	83	89	117	92	88	114	100	*63*	75	**52***
unbalanced20_random2	87	86	111	102	92	131	96	*69*	74	**52***

Table 6.4: Upper bounds, i.e. feasible solutions, computed by our heuristics and CPLEX 12.6.3 with time limit (i) 30min for the Furman and Sahinidis (2004) Test Set, (ii) 2h for the Chen et al. (2015b,a) test set, and (iii) 4h for the Grossmann (2017) test set. Symbol * indicates timeout. **Bold** values indicate the best computed value. *Italic* values indicate the best heuristic result. The proposed heuristics produce feasible as good as the exact solver for 13 of the 48 test cases. All heuristic results are available online (Letsios et al. 2017).

Test Case	Relaxation Rounding			Water Filling		Greedy Packing				CPLEX
	FLPR	LRR	CRR	WFG	WFM	LHM	LFM	LHM-LP	SS	
Furman and Sahinidis (2004) Test Set										
10sp-la1	0.01	0.01	0.10	0.14	0.28	0.02	0.01	7.76	< 0.01	**0.03**
10sp-ol1	0.01	0.01	0.07	0.10	0.21	0.02	0.01	9.83	< 0.01	**0.03**
10sp1	0.01	0.01	0.10	0.13	0.26	0.02	0.01	7.39	< 0.01	**0.05**
12sp1	0.01	0.01	0.10	0.25	0.40	0.04	0.02	9.79	< 0.01	**0.05**
14sp1	0.01	0.01	0.13	0.22	0.42	0.09	0.04	24.49	0.02	**41.23**
15sp-tkm	0.01	0.01	0.12	0.23	0.52	0.17	0.09	29.46	0.02	**0.07**
20sp1	0.01	0.01	0.29	0.37	0.60	0.52	0.24	64.63	0.05	*
22sp-ph	0.01	0.02	0.20	0.46	0.64	0.92	0.30	156.76	0.05	**0.04**
22sp1	0.02	0.02	0.32	0.39	0.60	0.76	0.34	144.77	0.06	*
23sp1	0.04	0.04	0.34	0.29	0.52	0.91	0.40	239.94	0.07	*
28sp-as1	0.01	0.01	0.09	0.31	0.57	1.26	0.38	227.06	0.05	**0.05**
37sp-yfyv	0.02	0.03	0.92	0.82	1.45	14.68	4.74	1435.94	0.50	**7.36**
4sp1	0.01	0.01	0.04	0.08	0.16	< 0.01	< 0.01	0.75	< 0.01	**0.02**
6sp-cf1	0.01	0.01	0.09	0.08	0.18	< 0.01	< 0.01	1.25	< 0.01	**0.03**
6sp-gg1	0.01	0.01	0.05	0.07	0.12	< 0.01	< 0.01	0.42	< 0.01	**0.02**
6sp1	0.01	0.01	0.07	0.07	0.14	< 0.01	< 0.01	1.36	< 0.01	**0.02**
7sp-cm1	0.01	0.01	0.05	0.10	0.24	0.01	< 0.01	3.51	< 0.01	**0.02**
7sp-s1	0.01	0.01	0.05	0.13	0.23	0.01	< 0.01	2.18	< 0.01	**0.02**
7sp-torw1	0.01	0.01	0.09	0.09	0.21	0.01	0.01	3.82	< 0.01	**0.02**
7sp1	0.01	0.01	0.09	0.08	0.17	< 0.01	< 0.01	2.10	< 0.01	**0.04**
7sp2	0.01	0.01	0.09	0.06	0.16	< 0.01	< 0.01	2.15	< 0.01	**0.03**
7sp4	0.01	0.01	0.04	0.13	0.27	< 0.01	< 0.01	1.61	< 0.01	**0.02**
8sp-fs1	0.01	0.01	0.09	0.15	0.27	0.01	< 0.01	5.84	< 0.01	**0.02**
8sp1	0.01	0.01	0.10	0.13	0.25	0.01	< 0.01	4.75	< 0.01	**0.03**
9sp-al1	0.01	0.01	0.09	0.13	0.28	0.02	0.01	8.18	< 0.01	**0.03**
9sp-has1	0.01	0.01	0.09	0.12	0.34	0.02	0.01	6.99	< 0.01	**0.04**
Chen et al. (2015b,a) Test Set										
balanced10	0.02	0.02	0.32	0.43	0.84	1.15	0.50	181.13	0.09	**1607.14**
balanced12	0.03	0.03	0.62	0.57	1.05	2.74	1.00	397.37	0.16	*
balanced15	0.05	0.05	0.83	0.76	1.23	10.96	3.45	1147.96	0.41	*
balanced5	0.01	0.01	0.11	0.25	0.50	0.05	0.03	13.64	0.01	**0.20**
balanced8	0.02	0.01	0.21	0.31	0.65	0.33	0.15	68.85	0.04	**69.16**
unbalanced10	0.03	0.02	0.30	0.47	0.72	1.19	0.50	173.88	0.08	**7.45**
unbalanced15	0.05	0.04	0.90	0.74	1.34	11.28	3.78	1185.72	0.39	*
unbalanced17	0.07	0.07	1.45	0.95	1.76	21.25	8.31	2742.15	0.71	*
unbalanced20	0.13	0.13	3.08	1.25	2.41	47.55	15.94	7154.64	1.34	*
unbalanced5	0.01	0.01	0.11	0.26	0.45	0.05	0.04	16.40	0.01	**0.05**
Grossmann (2017) Test Set										
balanced12_random0	0.03	0.03	0.46	0.59	1.00	2.51	1.15	351.10	0.17	*
balanced12_random1	0.03	0.03	0.46	0.59	1.11	2.74	0.99	398.26	0.16	*
balanced12_random2	0.03	0.03	0.46	0.60	0.88	2.61	0.95	382.57	0.17	*
balanced15_random0	0.04	0.05	0.83	0.76	1.49	8.87	4.13	1241.33	0.43	*
balanced15_random1	0.05	0.04	0.90	0.83	1.36	9.01	3.22	1041.37	0.42	*
balanced15_random2	0.05	0.05	0.90	0.82	1.53	9.26	3.43	1104.94	0.43	*
unbalanced17_random0	0.12	0.11	1.85	0.95	1.72	24.25	8.65	3689.80	0.80	*
unbalanced17_random1	0.12	0.12	1.82	0.79	1.54	24.08	8.32	4052.52	1.53	*
unbalanced17_random2	0.12	0.12	1.89	1.00	1.60	25.60	9.72	3471.80	0.73	*
unbalanced20_random0	0.18	0.17	3.23	1.22	2.18	50.67	18.37	8820.55	1.31	*
unbalanced20_random1	0.21	0.19	3.48	1.24	2.21	51.19	19.35	9613.90	1.40	*
unbalanced20_random2	0.23	0.22	3.17	1.28	2.33	56.47	17.93	11854.82	1.40	*

Table 6.5: CPU times of the heuristics and CPLEX 12.6.3 with time limit (i) 30min for the Furman and Sahinidis (2004) test set, (ii) 2h for the Chen et al. (2015b,a) test set, and (iii) 4h for the Grossmann (2017) test set. An * indicates timeout. All heuristic results are available online (Letsios et al. 2017).

Test Case	Relaxation Rounding					Water Filling		
	FS04		LKM17			FS04	LKM17	
	FLPR	LRR	FLPR	LRR	CRR	WFG	WFG	WFM
10sp-la1	21	19	16	17	16	22	16	15
10sp-ol1	22	17	18	18	17	23	18	18
10sp1	14	14	18	15	13	21	15	12
12sp1	17	18	16	17	13	18	14	15
14sp1	27	21	14	20	18	27	21	19
15sp-tkm	29	27	20	22	20	29	23	25
20sp1	24	25	22	20	23	25	24	21
22sp-ph	34	40	27	28	28	35	27	28
22sp1	41	42	35	37	36	54	42	35
23sp1	38	32	32	32	40	60	50	33
28sp-as1	41	45	30	30	30	43	40	45
37sp-yfyv	67	59	44	40	45	61	37	43
4sp1	5	6	5	5	5	5	5	5
6sp-cf1	6	6	6	6	6	7	6	7
6sp-gg1	3	3	3	3	3	3	3	3
6sp1	9	10	8	7	9	9	9	6
7sp-cm1	11	10	10	10	10	10	10	10
7sp-s1	10	10	10	10	10	10	10	10
7sp-torw1	14	15	11	12	10	13	12	12
7sp1	10	13	8	10	11	10	10	8
7sp2	8	7	7	7	7	8	9	9
7sp4	11	9	8	8	8	8	8	8
8sp-fs1	14	14	13	13	12	14	12	12
8sp1	11	13	11	11	10	14	14	9
9sp-al1	17	19	16	17	16	20	16	15
9sp-has1	16	14	15	14	15	18	15	16

Table 6.6: Comparison of our results (labelled LKM17) with the ones reported by Furman and Sahinidis (2004) (labelled FS04). The LKM17 heuristics FLPR, LRR, and WFG perform better than their FS04 counterparts because of our improved calculation of the big-M parameter U_{ij}.

Test Case	Fractional LP Rounding			Lagrangian Relaxation Rounding		
	Simple	GTA97	LKM17	Simple	GTA97	LKM17
Furman and Sahinidis (2004) Test Set						
10sp-la1	17	**15**	16	17	**16**	17
10sp-ol1	20	19	**18**	22	23	**18**
10sp1	**18**	**18**	**18**	**14**	17	15
12sp1	17	17	**16**	**17**	**17**	**17**
14sp1	27	20	**14**	22	**17**	20
15sp-tkm	27	27	**20**	25	25	**22**
20sp1	**22**	**22**	**22**	27	24	**20**
22sp-ph	31	30	**27**	32	30	**28**
22sp1	45	37	**35**	40	44	**37**
23sp1	40	37	**32**	35	33	**32**
28sp-as1	41	**30**	**30**	45	**30**	**30**
37sp-yfyv	56	53	**44**	42	42	**40**
4sp1	**5**	**5**	**5**	**5**	**5**	**5**
6sp-cf1	**6**	**6**	**6**	**6**	**6**	**6**
6sp-gg1	**3**	**3**	**3**	**3**	**3**	**3**
6sp1	8	**7**	8	8	**7**	**7**
7sp-cm1	11	**10**	**10**	**10**	**10**	**10**
7sp-s1	**10**	**10**	**10**	**10**	**10**	**10**
7sp-torw1	15	15	**11**	14	13	**12**
7sp1	10	10	**8**	9	9	**10**
7sp2	9	**7**	**7**	**7**	**7**	**7**
7sp4	11	11	**8**	9	9	**8**
8sp-fs1	15	**13**	**13**	14	**13**	**13**
8sp1	**11**	**11**	**11**	12	12	**11**
9sp-al1	17	**15**	16	17	**16**	17
9sp-has1	16	**15**	**15**	**14**	15	**14**
Chen et al. (2015b,a) Test Set						
balanced10	61	46	**39**	46	43	**42**
balanced12	71	52	**42**	57	56	**48**
balanced15	91	63	**60**	91	**69**	**69**
balanced5	26	19	**18**	20	22	**17**
balanced8	42	33	**28**	38	35	**33**
unbalanced10	52	49	**38**	53	54	**46**
unbalanced15	73	60	**57**	76	**62**	64
unbalanced17	96	78	**70**	101	84	**78**
unbalanced20	132	95	**89**	137	99	**89**
unbalanced5	23	22	**19**	21	23	**20**
Grossmann (2017) Test Set						
balanced12_random0	73	56	**42**	61	52	**48**
balanced12_random1	62	56	**45**	60	54	**49**
balanced12_random2	66	51	**42**	53	57	**49**
balanced15_random0	93	68	**60**	75	73	**61**
balanced15_random1	96	68	**56**	79	73	**65**
balanced15_random2	102	64	**54**	86	76	**69**
unbalanced17_random0	106	77	**74**	108	95	**80**
unbalanced17_random1	116	82	**74**	99	91	**74**
unbalanced17_random2	101	84	**70**	94	92	**79**
unbalanced20_random0	131	95	**93**	136	103	**93**
unbalanced20_random1	138	91	**83**	139	104	**89**
unbalanced20_random2	138	102	**87**	131	100	**86**

Table 6.7: This table compares the effect of three different methods for computing the big-M parameter U_{ij}: (i) simple greedy, (ii) the Gundersen et al. (1997) (GTA97) method and (iii) our greedy Algorithm MHG (LKM17). **Bold** values mark the best result for each of the heuristics. LKM17 outperforms the other two big-M computation methods by finding smaller feasible solutions via both Fractional LP Rounding and Lagrangian Relaxation Rounding.

Test Case	Relaxation Rounding FLPR		Water Filling WFG		Greedy Packing SS		CPLEX Transshipment	
	Value	**Time**	**Value**	**Time**	**Value**	**Time**	**Value**	**Time**
large_scale0	233	8.84	306	58.52	*233*	*642.94*	**175**	*
large_scale1	273	15.59	432	54.53	***218***	***652.00***	219	*
large_scale2	279	41.83	497	54.46	*242*	*670.32*	**239**	*

Table 6.8: Upper bounds, i.e. feasible solutions, for large-scale instances computed by the least time consuming heuristics of each type and CPLEX 12.6.3 transshipment model with 4h time-out. Symbol * indicates timeout. **Bold** marks the best upper bound. *Italic* marks the best heuristic result. In instance large_scale1, heuristic LFM computes the best heuristic result.

Chapter 7

Conclusion

This thesis: 1) identifies special structure properties of nonconvex optimisation problems which may provide advances on solving them via exact methods, 2) uses specific insights for the heat exchanger network synthesis (HENS) problem to propose methods to deal with large-scale instances via both exact and approximation methods.

7.1 Contributions

7.1.1 Data Structures for Representing Symmetry in Quadratically Constrained Quadratic Programs

This work appraises the significant role of symmetry in optimisation problems. Symmetry representation and detection are the fundamental steps towards exploiting symmetry. This work provides a review on methods that currently exist in literature for detecting symmetry. This chapter introduces a novel symmetry detection methodology; Initially we associate quadratic and linear optimisation problems with matrices. Then we construct binary layered graphs that encode information from the matrices and capture the structure of the original problem. The algorithm implementation provided in the software package `nauty` by McKay and Piperno (2014) is associated with a search tree and determines the automorphism group of a problem and whether two graphs are isomorphic. In this work `nauty` reads these graphs and generates

important symmetric information of the original problem. The generators of this group can be projected to the variables and the constraints of the given mathematical program. A computational case corroborates the proposed methods and the conclusion are discussed.

7.1.2 Detecting Symmetry and Understanding Complexities of the Minimum Number of Matches Problem in Heat Recovery Network Design

This chapter studies the complexities that arise in designing heat exchanger networks by investigating the minimum number of matches problem. This work explores for the first time, the symmetric structure of the HENS problem. Initially we use group theory to study the MILP transshipment model for a single temperature interval. We identify types of symmetry arising in the problem. We also use parameters in the optimisation problem, e.g. temperature and heat capacities of each stream, to classify special cases with many equivalent optimal solutions and define degeneracy. Computational results from an online test case corroborate the proofs. Degeneracy is also defined in this problem providing more information on the complexities that may speed down the performance of current state-of-the-art solvers. Exploring further the computational complexities that arise when formulating this problem, this chapter introduces a novel $\mathscr{NP}$-hardness reduction of the minimum number of matches problem from the bin packing problem which also reveals the packing nature of this problems. By exploiting this structure, we propose a novel MILP formulation for the one temperature interval. We further explore the maximum heat exchanged between the streams with match restrictions including the computation of tighter big-M parameters. We evaluate and report the computational performance of the exact methods using state-of-the-art commercial approaches in three test cases of instances in the engineering literature; (i) we manually digitise the Furman (2000) instances, (ii) process the Chen et al. (2015a, 2015b) existing instances in the literature, and (iii) randomly generate the Grossmann (2017) instances with fixed seeds.

7.1.3 Heuristics with Performance Guarantees for the Minimum Number of Matches Problem in Heat Recovery Network Design

This chapter develops new heuristics and provably efficient approximation algorithms for the minimum number of matches problem; (i) relaxation rounding, (ii) water filling, and (iii) greedy packing. In heat exchanger network design, many possible stream configurations are required to evaluate the minimum overall cost, so a complementary contribution of this work is a heuristic methodology for producing multiple solutions efficiently. This work shows that existing linear programming (LP) rounding has poor worst case performance. We also prove a new positive performance guarantee for LP rounding indicating an improved worst-case performance with tighter big-M parameters. We also develop alternative relaxation rounding heuristics based on Lagrangian relaxation and a new covering relaxation. We improve water filling heuristics by developing novel, efficient ways for solving the single temperature interval problem. Using graph theoretic properties, we propose an improved, greedy approximation algorithm which prioritises stream matches with equal heat loads, for the single temperature interval problem. With appropriate LP, we further improve water filling heuristics by reusing in each iteration matches selected in previous iterations. We show that the temperature interval dependent performance guarantee is asymptotically tight for water filling heuristics. Finally, we develop a new greedy approach for designing efficient minimum number of matches heuristics motivated by the packing nature of the problem. Greedy packing requires feasibility conditions inspired by pinch point decompositions. Using feasibility conditions, we derive an LP that selects matches carrying a large amount of heat and incurring low unitary cost for exchanging heat. Our main greedy packing heuristic selects matches greedily by solving LP instances. Using a standard packing argument, we obtain a new logarithmic performance guarantee. Despite its polynomial worst-case running time, our heuristic is experimentally time-consuming due to the repeated LP solving. So, we propose three other greedy packing heuristic variants which improve the running time at the price of solution quality. The numerical results show that our novel greedy packing heuristics dominate relaxation rounding and water filling ones in the majority of test cases that currently exist in literature and the ones generated in this thesis.

7.2 Future Work

7.2.1 Exploit Symmetry in Heat Exchanger Networks

Further computational results are required to validate the robustness of the methods proposed in this chapter which is the extension of this work. This thesis focuses on detecting rather than eliminating or breaking symmetry in optimisation problems. These detection algorithms may be integrated into branching or cutting plane methods that exploit symmetry. There are two main strategies for breaking symmetry; the static and dynamic method and several techniques classified for each as presented by (Margot 2010). More precisely static strategy is related with adding constraints to the formulation prior to solving it. It is considered to be easier and ideally some or all symmetric global optima become infeasible except one. On the other hand dynamic strategies as isomorphism pruning and orbital branching (Margot 2002, Ostrowski et al. 2011) are the most efficient for exploitation of symmetry as the *B&B* is modified and recognises the symmetries while solving the problem.

7.2.2 Constraints Generation for Heat Exchanger Network Synthesis

Initially we abbreviate the idea of defining and adding static symmetric constraints as (Liberti 2012a) Liberti and Ostrowski (2014), Dias and Liberti (2015) suggest to the formulation of the problem. These constraints can be generated using the symmetry groups proposed in Chapter 4. As an advantage of other general symmetry breaking constraints that have been proposed we take into account the special structure of heat exchanger network problem that is extensively studied. Costa et al. (2013) propose a similar idea for the Kissing Number Problem and other Point Packing Problems. They use static constraints to reformulate the problem and they prove that the narrowed problem contains at least one feasible optima and then solve the problem by using spatial *B&B* (Smith et al. 2001). One idea which follows the above papers is to derive constrains and restrict the values of binary variables on whether there is a match or not between two streams. For example if two hot streams are symmetric then we can force (weight) one of them to match with a cold streams. Hence we eliminate the symmetric possibility of matching with the other hot stream.

7.2.3 Mathematical Reformulation of Heat Exchanger Network

A general framework of how to reformulate mathematical programs by adding static symmetry breaking constraints and ensuring that at least one global optimum is preserved is presented by Liberti (2009) with detailed proofs. For HENS in (Chen et al. 2015a) a set of dissagregated models are tested with reformulation of the upper bound constraint; the heat load that is exchanged when two streams match. Moreover Chen et al. (2015a) provide formulations with weight for preferences and restrictive choices. The following formulation with binary variables can be tested and improved:

$y_{i,j}$	binary variable indicating whether hot stream i is matched with cold stream j
$y_{i,j'}$	binary variable indicating whether hot stream i is matched with cold stream j'
$y_{i,j,j'}$	binary variable indicating whether hot stream i is matched with both cold stream j and j'

For $y_{i,j,j'} = y_{i,j} \cdot y_{i,j'}$

$y_{i,j}$	$y_{i,j'}$	$y_{i,j,j'}$
0	0	0
0	1	0
1	0	0
1	1	1

$$\min \sum_{i\in H}\sum_{j\in C} y_{i,j}$$

$$\sum_{j\in C}\sum_{u\in TI} q_{i,t,j,u} = Q^H_{i,t} \qquad i\in H, t\in TI$$

$$\sum_{i\in H}\sum_{t\in TI} q_{i,t,j,u} = Q^C_{j,u} \qquad j\in C, u\in TI$$

$$\sum_{t\in TI}\sum_{u\in TI} q_{i,t,j,u} \le U_{i,j}\cdot y_{i,j} \qquad i\in H, j\in C$$

$$q_{i,t,j,u} = 0 \qquad t>u,\ t,u\in TI$$

$$\sum_{t\in TI}\sum_{u\in TI} q_{i,t,j,u} + \sum_{t\in TI}\sum_{u\in TI} q_{i,t,j',u}$$

$$\le U_{i,j}\cdot y_{i,j} + U_{i,j'}\cdot y_{i,j'} + [U_{i,j,j'} - U_{i,j} - U_{i,j'}]\cdot y_{i,j,j'} \quad i\in H, j, j'\in C, j'>j$$

$$y_{i,j} + y_{i,j'} \le y_{i,j,j'} + 1 \qquad i\in H, j, j'\in C, j'>j$$

$$y_{i,j,j'} \le y_{i,j} \qquad i\in H, j, j'\in C, j'>j$$

$$y_{i,j,j'} \le y_{i,j'} \qquad i\in H, j, j'\in C, j'>j$$

$$y_{i,j}, y_{i,j'}, y_{i,j,j'} \in \{0,1\}, q_{i,t,j,u}\ge 0 \qquad i\in H, j\in C,\ t,u\in TI$$

7.2.4 Disjunctive Programming

Another interesting idea which can be applied to problems with linear/nonlinear models, $0-1$ and continuous decision variables is using disjunctive programming. There is an extensive literature on this class of optimisation models by Grossmann (2002), Balas (2010), Grossmann and Ruiz (2012) and others. A case where the problem is reformulated using the above approach is the floor layout problem (Huchette et al. 2017) based on the constraint that the boxes that located in a floor cannot overlap between them. More precisely in an optimisation problem all the constraints need to be satisfied or in other words the feasible set of solutions can be considered as the intersection of the feasible sets of each constraint. Hence disjunctive programming transforms the intersection to a union relation between the constraints which relaxes the feasible space. For the formulation of HEN in Chapter 3, any decision binary variable consists on the equation:

$$0 \le Q_{ij} \le U_{ij} y_{ijt},\ i\in HS,\ j\in CS, t\in TI \tag{7.1}$$

which can be reformulated by using the basic steps of disjunctive programming in the following way. Suppose that two cold streams say $j, \hat{j} \in \text{CS}$ require the same load (are symmetric in the way that has already defined for HEN synthesis). In a conjunctive normal form the two constraints would intersect as:

$$0 \leq Q_{ij} \leq U_{ij} y_{ijt} \cap 0 \leq Q_{i\hat{j}} \leq U_{i\hat{j}} y_{i\hat{j}t} \tag{7.2}$$

Then there are 4 possible cases on how they can match with any hot stream $i \in HS$; either none of them matches or one of them or both of them. Hence a disjunctive transformation could be:

$$\begin{pmatrix} y_{ijt} = 0 \\ y_{i\hat{j}t} = 0 \\ Q_{ij} = 0 \\ Q_{i\hat{j}} = 0 \end{pmatrix} \cup \begin{pmatrix} y_{ijt} = 1 \\ y_{i\hat{j}t} = 0 \\ Q_{ij} \leq U_{ij} \\ Q_{i\hat{j}} = 0 \end{pmatrix} \cup \begin{pmatrix} y_{ijt} = 0 \\ y_{i\hat{j}t} = 1 \\ Q_{ij} = 0 \\ Q_{i\hat{j}} \leq U_{i\hat{j}} y_{i\hat{j}t} \end{pmatrix} \cup \begin{pmatrix} y_{ijt} = 1 \\ y_{i\hat{j}t} = 1 \\ Q_{ij} = \leq U_{ij} \\ Q_{i\hat{j}} \leq U_{i\hat{j}} y_{i\hat{j}t} \end{pmatrix} \tag{7.3}$$

The general idea is that we can benefit from such a reformulation by using this LP relaxation approach to generate bounds. Further to the reformulation there are some important questions raised in this section. First of all how close to the optimal solution is the LP relaxation and for the General Disjunctive Programming what is the best MILP formulation (i.e. big-M or convex hull).

Bibliography

T. Achterberg. SCIP: solving constraint integer programs. *Math Program Comput*, 1(1), 2009.

C.S. Adjiman, S. Dallwig, C.A. Floudas, and A. Neumaier. A global optimization method, sBB, for general twice-differentiable constrained NLPs I. theoretical advances. *Comput Chem Eng*, 22(9): 1137 – 1158, 1998.

S. Ahmad and B. Linnhoff. SUPERTARGETING: Different process structures for different economics. *J. Energy Res Tech*, 111(3):131–136, 1989.

S. Ahmad and R. Smith. Targets and design for minimum number of shells in heat exchanger networks. *AIChE J*, 67(5):481 – 494, 1989.

J. Alemany, P. Komarnicki, J. Linand, and F. Magnago. Exploiting symmetry in unit commitment solutions for a large-scale electricity market. In *Electric Power Systems Research*, volume 140. Elsevier, 2016.

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

K. M. Anstreicher. Semidefinite programming versus the reformulation-linearization technique for non-convex quadratically constrained quadratic programming. *J Glob Optim*, 43(2):471–484, 2009.

M. A. Armstrong. *Groups and Symmetry*. Undergraduate Texts in Mathematics. Springer, 1988.

S. Arora and B. Barak, editors. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

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

E. Balas. Disjunctive programming. In M. Jünger, M. T. Liebling, D. Naddef, L. G. Nemhauser, R. W. Pulleyblank, G. Reinelt, G. Rinaldi, and A. L. Wolsey, editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 283–340. Springer Berlin Heidelberg, 2010.

E. Balas, S. Ceria, G. Cornuéjols, and N. Natraj. Gomory cuts revisited. *Oper Res Lett*, 19(1):1–9, 1996.

R. C. Baliban, J. A. Elia, R. Misener, and C. A. Floudas. Global optimization of a MINLP process synthesis model for thermochemical based conversion of hybrid coal, biomass, and natural gas to liquid fuels. *Comput Chem Eng*, 42:64 – 86, 2012.

N. V. Bancheva, B.B. Ivanov, N. Shah, and C.C. Pantelides. Heat exchanger network design for multipurpose batch plants. *Comput Chem Eng*, 20(8):989 – 1001, 1996.

P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wachter. Branching and bounds tightening techniques for non-convex MINLP. *Optim Meth Soft*, 24(4-5):597–634, 2009a.

P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optim Meth Soft*, 24(4-5):597–634, 2009b.

T. Berthold and M.E. Pfetsch. Detecting orbitopal symmetries. In *Proceedings of the Annual International Conference of the German Operations Research Society (GOR)*, pages 433–438. Springer Berlin Heidelberg, 2009.

D. P. Bertsekas, A. Nedi, and A. E. Ozdaglar, editors. *Convex analysis and optimization*. Athena Scientific, 2003.

L. T. Biegler, I. E. Grossmann, and A. W. Westerberg, editors. *Systematic methods of chemical process design*. Prentice-Hall international series in the physical and chemical engineering sciences. Prentice Hall PTR, 1997.

R. Bödi, K. Herr, and M. Joswig. Algorithms for highly symmetric linear and integer programs. *Math Program*, 137(1):65–90, 2013.

F. Boukouvala, R. Misener, and C. A. Floudas. Global optimization advances in mixed-integer nonlinear programming, MINLP, and constrained derivative-free optimization, CDFO. *Europ J of Oper Res*, 252(3):701 – 727, 2016.

D. Bremner, M. S. Sikirić, D. V. Pasechnik, T. Rehn, and A. Schürmann. Computing symmetry groups of polyhedra. *LMS J Comp Math*, 17(1):565–581, 2014.

P. Brucker. *Scheduling Algorithms*. Springer-Verlag, 3 edition, 2001.

M. R. Bussieck and A. Pruessner. Mixed-integer nonlinear programming. *SIAG/OPT Newsletter: Views & News*, 14(1):19–22, 2003.

P. J. Cameron. *Permutation Groups*. London Mathematical Society Student Texts. Cambridge uni., 1999.

A. Caprara and M. Locatelli. Global optimization problems and domain reduction strategies. *Math Program*, 125:123–137, 09 2010.

P. M. Castro, B. Custodio, and H. A. Matos. Optimal scheduling of single stage batch plants with direct heat integration. *Comput Chem Eng*, 82:172 – 185, 2015.

F. Ceccon, G. Kouyialis, and R. Misener. Using functional programming to recognize named structure in an optimization problem: Application to pooling. *AIChE J*, 62(9):3085–3095, 2016.

J. Cerda and A.W. Westerberg. Synthesizing heat exchanger networks having restricted stream/stream matches using transportation problem formulations. *Chem Eng Sci*, 38(10):1723–1740, 1983.

J. Cerda, A.W. Westerberg, D. Mason, and B. Linnhoff. Minimum utility usage in heat exchanger network synthesis: A transportation problem. *Chem Eng Sci*, 38(3):373–387, 1983.

A. Charnes. Optimality and Degeneracy in Linear Programming. *Econometrica*, 20(2):160–170, 1952.

G. Chartrand, editor. *Introductory Graph Theory*. Dover Books on Mathematics Series. Dover, 1977.

J.J.J. Chen. Comments on improvements on a replacement for the logarithmic mean. *Chem Eng Sci*, 42 (10):2488 – 2489, 1987.

Y. Chen, I. E. Grossmann, and D. C. Miller. Computational strategies for large-scale MILP transshipment models for heat exchanger network synthesis. *Comput Chem Eng*, 82:68–83, 2015a.

Y. Chen, I.E. Grossmann, and D. Miller. Large-scale MILP transshipment models for heat exchanger network synthesis, Modification of: 23:10:51, March 28 2015b. Available from CyberInfrastructure for MINLP [`www.minlp.org`, a collaboration of Carnegie Mellon University and IBM Research] at: `www.minlp.org/library/problem/index.php?i=191`.

A. R. Ciric and C. A. Floudas. Heat exchanger network synthesis without decomposition. *Comput Chem Eng*, 15(6):385–396, 1991.

A.R. Ciric and C.A. Floudas. A retrofit approach for heat exchanger networks. *Comput Chem Eng*, 13(6): 703 – 715, 1989.

A. Clark, editor. *Elements of Abstract Algebra*. Dover Books in Mathematics. Courier Corporation, 1984.

R. D. Colberg and M. Morari. Area and capital cost targets for heat exchanger network synthesis with constrained matches and unequal heat transfer coefficients. *Comput Chem Eng*, 14(1):1 – 22, 1990.

R. W. Conway, W. L. Maxwell, and L. W. MillerW. *Theory of Scheduling*. Dover Books on Computer Science. Dover Publications, 2012.

A. Costa, P. Hansen, and L. Liberti. On the impact of symmetry-breaking constraints on spatial branch-and-bound for circle packing in a square. *Disc Appl Math*, 161(1):96–106, 2013.

J. M. Crawford, M. L. Ginsberg, E.M. Luks, and A. Roy. Symmetry-breaking predicates for search problems. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, pages 148–159. Morgan Kaufmann Publishers Inc., 1996.

S. S. Dey and A. Gupte. Analysis of MILP techniques for the pooling problem. *Oper Res*, 63(2):412–427, 2015.

G. Dias and L. Liberti. Orbital independence in symmetric mathematical programs. In *Proceedings of the Combinatorial Optimization and Applications: 9th International Conference, COCOA 2015, TX, USA, 2015.*, pages 467–480. Springer International Publishing, 2015.

W. B. Dolan, P. T. Cummings, and M. D. Le Van. Algorithmic efficiency of simulated annealing for heat exchanger network design. *Comput Chem Eng*, 14(10):1039–1050, 1990.

M. Elf, G. Gutwenger, M. Jünger, and G. Rinaldi. Branch-and-cut algorithms for combinatorial optimization and their implementation in ABACUS. In *Computational Combinatorial Optimization, Optimal or Provably Near-Optimal Solutions*, pages 157–222. Springer-Verlag, 2001.

I. Elhallaoui, A. Metrane, G. Desaulniers, and F. Soumis. An improved primal simplex algorithm for degenerate linear programs. *INFORMS*, 23(4):569–577, 2011.

J. A. Elia, R. C. Baliban, and C. A. Floudas. 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*, 49(16):7371–7388, 2010.

M. Escobar and J. O. Trierweiler. Optimal heat exchanger network synthesis: A case study comparison. *Appl Therm Eng*, 51(1-2):801–826, 2013.

European Commission. An EU strategy on heating & cooling, 2016. Brussels, COM 51.

Y. Faenza and V. Kaibel. Extended formulations for packing and partitioning orbitopes. *Math Oper Res*, 34(3):686–697, 2009.

B. Farhanieh and B. Sunden. Analysis of an existing heat exchanger network and effects of heat pump installations. *Heat Rec Syst and CHP*, 10(3):285–296, 1990.

M. Fischetti, A. Lodi, M. Monaci, D. Salvagnin, and A. Tramontani. Improving branch-and-cut performance by random sampling. *Math Program Comp*, 8(1):113–132, 2016.

M. Fischetti, L. Liberti, D. Salvagnin, and T. Walsh. Orbital shrinking: Theory and applications. *Disc. Appl. Math.*, 222:109–123, 2017.

C. A. Floudas, editor. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications Topics in Chemical Engineering*. Topics in Chemical Engineering. Oxford University Press, New York, 1995.

C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1):3–38, 2008. ISSN 1573-2916.

C. A. Floudas and I. E. Grossmann. Synthesis of flexible heat exchanger networks with uncertain flowrates and temperatures. *Comput Chem Eng*, 11(4):319 – 336, 1987.

C. A. Floudas, A. R. Ciric, and I. E. Grossmann. Automatic synthesis of optimum heat exchanger network configurations. *AIChE J*, 32(2):276–290, 1986.

C. A. Floudas, J. A. Elia, and R. C. Baliban. Hybrid and single feedstock energy processes for liquid transportation fuels: A critical review. *Comput Chem Eng*, 41:24 – 51, 2012.

C.A. Floudas, I.G. Akrotirianakis, S. Caratzoulas, C. A. Meyer, and J. Kallrath. Global optimization in the 21st century: Advances and challenges. *Comput Chem Eng*, 29(6):1185 – 1202, 2005.

E. J. Friedman. Fundamental domains for integer programs with symmetries. In A. W. M. Dress, V. Xu, and B. Zhu, editors, *COCOA*, Lecture Notes in Computer Science, pages 146–153. Springer, 2007.

K. C. Furman. *Analytical investigations in heat exchanger network synthesis*. PhD thesis, University of Illinois, 2000.

K. C. Furman and N. V. Sahinidis. Computational complexity of heat exchanger network synthesis. *Comput Chem Eng*, 25(9):1371–1390, 2001.

K. C. Furman and N. V. Sahinidis. A critical review and annotated bibliography for heat exchanger network synthesis in the 20*th* century. *Ind Eng Chem Res*, 41(10):2335–2370, 2002.

K. C. Furman and N. V. Sahinidis. Approximation algorithms for the minimum number of matches problem in heat exchanger network synthesis. *Ind Eng Chem Res*, 43(14):3554–3565, 2004.

T. Gal. Degeneracy problems in mathematical programming and degeneracy graphs. *ORiON*, 6(1), 2003.

I. P. Gent and B. M. Smith. Symmetry breaking in constraint programming. In W. Horn, editor, *Proceedings of the ECAI-2000,14th European Conference on Artificial Intelligence, 2000*, pages 599–603. IOS Press, 2000.

I. P. Gent, T. Kelsey, S. A. Linton, I. McDonald, I. Miguel, and B. M. Smith. Conditional symmetry breaking. In P. van Beek, editor, *Proceedings of the Principles and Practice of Constraint Programming - CP 2005: 11th International Conference, Spain, 2005.* Springer Berlin Heidelberg, 2005.

K. George and M. R. Osborne. On degeneracy in linear programming and related problems. *Ann of Oper Res*, 46(2):343–359, 1993.

A. Ghoniem and H. D. Sherali. Defeating symmetry in combinatorial optimization via objective perturbations and hierarchical constraints. *IIE Transactions*, 43(8):575–588, 2011.

A. Gibbons, editor. *Algorithmic Graph Theory*. Cambridge University Press, 1985.

A. M. Gleixner, T. Berthold, B. Müller, and S. Weltge. Three enhancements for optimization-based bound tightening. *J Glob Optim*, 67(4):731 – 757, 2017.

R. E. Gomory. An algorithm for integer solutions to linear programs. In R. L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*. McGraw-Hill, New York, 1963.

I. Grossmann. Personal communication, 2017.

I. E. Grossmann. Mixed-integer nonlinear programming techniques for the synthesis of engineering systems. *Res Eng Des*, 1(3):205–228, 1990.

I. E. Grossmann and J. P. Ruiz. Generalized disjunctive programming: A framework for formulation and alternative algorithms for MINLP optimization. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 93–115. Springer New York, New York, NY, 2012.

I. E. Grossmann and R. W. H. Sargent. Optimum design of heat exchanger networks. *Comput Chem Eng*, 2(1):1 – 7, 1978.

I.E. Grossmann. Review of nonlinear mixed-integer and disjunctive programming techniques. *Optimization and Engineering*, 3(3):227–252, 2002.

T. Gundersen and I.E. Grossmann. Improved optimization strategies for automated heat exchanger network synthesis through physical insights. *Comput Chem Eng*, 14(9):925 – 944, 1990.

T. Gundersen and L. Naess. The synthesis of cost optimal heat exchanger networks. *Comput Chem Eng*, 12(6):503 – 530, 1988.

T. Gundersen, P. Traedal, and A. H. Ahmady. Improved sequential strategy for the synthesis of near-optimal heat exchanger networks. *Comput Chem Eng*, 21:S59 – S64, 1997.

S. G. Hall, S. Ahmad, and R. Smith. Capital cost targets for heat exchanger networks comprising mixed materials of construction, pressure ratings and exchanger types. *Comput Chem Eng*, 14(3):319–335, 1990.

W. E. Hart, C. Laird, J.P. Watson, and D. L. Woodruff. Pyomo: modeling and solving mathematical programs in python. *Math Program Comp*, 3(3):219–260, 2011.

W. E. Hart, C. Laird, J.P. Watson, and D. L. Woodruff, editors. *Pyomo–optimization modeling in python*, volume 67. Springer Science & Business Media, 2012.

M. M. F. Hasan, G. Jayaraman, I. A. Karimi, and H. E. Alfadala. Synthesis of heat exchanger networks with nonisothermal phase changes. *AIChE J*, 56(4):930–945, 2010.

K. Herr, T. Rehn, and A. Schürmann. Exploiting symmetry in integer convex optimization using core points. *Oper Res Lett*, 41(3):298–304, 2013.

C. Hojny and M. E. Pfetsch. Symmetry handling via symmetry breaking polytopes. In *13th Cologne Twente Workshop on Graphs and Combinatorial Optimization, Istanbul, Turkey, 2015. proceedings*, pages 63–66, 2015.

J. N. Hooker. A hybrid method for the planning and scheduling. *Constraints*, 10(4):385–401, 2005.

J. Huchette, S. S. Dey, and J. P. Vielma. Strong mixed-integer formulations for the floor layout problem. *INFOR: Information Systems and Operational Research*, pages 1–42, 2017.

M. G. Ierapetritou and C. A. Floudas. Effective continuous-time formulation for short-term scheduling: I. multipurpose batch processes. *Ind Eng Che Res*, 37:4341–4359, 1998.

V. Kaibel, M. Peinhardt, and M. E. Pfetsch. Orbitopal fixing. *Disc Optim*, 8(4):595–610, 2011.

J. Kallrath. Cutting circles and polygons from area-minimizing rectangles. *J Glob Optim*, 43:299–328, 2009.

N. Katoh, A. Shioura, and T. Ibaraki. Resource allocation problems. In P. M. Pardalos, D. Z. Du, and R. L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 2897–2988. Springer, 2013.

B. Knueven, J. Ostrowski, and S. Pokutta. Detecting almost symmetries of graphs. *Math Program Comp.*, 2017.

O. T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, R. E. Bixby, E. Danna, G. Gamrath, A. M. Gleixner, S. Heinz, A. Lodi, H. Mittelmann, T. Ralphs, D. Salvagnin, D. E. Steffy, and K. Wolter. MIPLIB 2010: Mixed integer programming library version 5. *Math Program Comput*, 3(2):103–163, 2011.

Q. Kong and N. Shah. Development of an optimization-based framework for simultaneous process synthesis and heat integration. *Ind Eng Che Res*, 56(17):5000–5013, 2017.

B. O. Koopman. The optimum distribution of effort. *J of the Oper Res Soc of America*, 1(2):52–63, 1953.

G. Kouyialis and R. Misener. Detecting symmetry in designing heat exchanger networks. In *Proceedings of the International Conference of Foundations of Computer-Aided Process Operations - FOCAPO/CPC 2017*, Arizona, 2017.

S. Kucherenko, P. Belotti, L. Liberti, and N. Maculan. New formulations for the kissing number problem. *Disc. App. Math.*, 155(14):1837 – 1841, 2007.

A. H. Land and A. G. Doig. An automatic method for solving discrete programming problems. *ECONOMETRICA*, 28(3):497–520, 1960.

K. F. Lee, A. H. Masso, and D. F. Rudd. Branch and bound synthesis of integrated process designs. *Ind Eng Chem Fund*, 9(1):48 – 58, 1970.

D. Letsios and R. Misener. Exact Lexicographic Scheduling and Approximate Rescheduling. *ArXiv e-prints*, 2018.

D. Letsios, G. Kouyialis, and R. Misener. Source code, 2017. `https://github.com/dimletsios/min\_matches\_heuristics`.

D. Letsios, G. Kouyialis, and R. Misener. Heuristics with performance guarantees for the minimum number of matches problem in heat recovery network design. *Comp Chem Eng*, 113:57 – 85, 2018.

L. Liberti. Reformulation and convex relaxation techniques for global optimization. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 2(3):255–258, 2004.

L. Liberti. Writing global optimization software. In L. Liberti and N.Maculanr, editors, *Global Optimization: from Theory to Implementation*, pages 211–262. Springer Berlin, 2006.

L. Liberti. Automatic generation of symmetry-breaking constraints. In B. Yang, D. Z. Du, and C. A. Wang, editors, *Combinatorial Optimization and Applications*, pages 328–338, Berlin, Heidelberg, 2008. Springer.

L. Liberti. *Reformulations in Mathematical Programming: A Computational Approach*, pages 153–234. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

L. Liberti. Symmetry in mathematical programming. In J. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 263–283. Springer New York, 2012a.

L. Liberti. Reformulations in mathematical programming: automatic symmetry detection and exploitation. *Math Program*, 131(1):273–304, 2012b.

L. Liberti. Undecidability and hardness in Mixed-Integer Nonlinear Programming. *RAIRO - Oper Res*, 2018.

L. Liberti and J. Ostrowski. Stabilizer-based symmetry breaking constraints for mathematical programs. *J Glob Optim*, 60(2):183–194, 2014.

L. Liberti, S. Cafieri, and D. Savourey. The Reformulation-Optimization Software Engine. In K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, editors, *Mathematical Software - ICMS 2010*, pages 303 – 314. Springer Berlin Heidelberg, 2010.

B. Linnhoff. and S. Ahmad. SUPERTARGETING: optimum synthesis of energy management systems. *J. Energy Res Tech*, 111(3):121 – 130, 1989.

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

B. Linnhoff and E. Hindmarsh. The pinch design method for heat exchanger networks. *Chem Eng Sci*, 38 (5):745 – 763, 1983.

A. Lodi and A. Tramontani. *Performance Variability in Mixed-Integer Programming*, pages 1–12. INFORMS, 2013.

I. Lotero, F. Trespalacios, I. E. Grossmann, D. J. Papageorgiou, and M-S Cheon. An MILP-MINLP decomposition method for the global optimization of a source based model of the multiperiod blending problem. *Comput Chem Eng*, 87:13 – 35, 2016.

C. T. Maravelias. Mixed-time representation for state-task network models. *Ind Eng Chem Res*, 44(24): 9129–9145, 2005.

C. T. Maravelias and I. E. Grossmann. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Comp Chem Eng*, 28(10):1921 – 1949, 2004.

F. Margot. Pruning by isomorphism in branch-and-cut. *Math Program*, 94(1):71–90, 2002.

F. Margot. Small covering designs by branch-and-cut. *Math Program*, 94(2):207–220, 2003a.

F. Margot. Exploiting orbits in symmetric ILP. *Math Program*, 98(1):3–21, 2003b.

F. Margot. Symmetric ILP: Coloring and small integers. *Disc Optim*, 4(1):40–62, 2007. Mixed Integer Programming.

F. Margot. Symmetry in integer linear programming. In *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 647–686. Springer Berlin Heidelberg, 2010.

A. H. Masso and D. F. Rudd. The synthesis of system designs. ii. heuristic structuring. *AIChE J*, 15(1): 10–17, 1969.

G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part i - convex underestimating problems. *Math Program*, 10(1):147–175, 1976.

B. D. McKay and A. Piperno. Practical graph isomorphism, ii. *J. Symb. Comp.*, 60:94–112, 2014.

J. B. Miguel, R. Pham, and V. Manousiouthakis. On the state space approach to mass/heat exchanger network design. *Chem Eng Sci*, 53(14):2595 – 2621, 1998.

R. Misener and C. A. Floudas. ANTIGONE: Algorithms for coNTinuous / Integer Global Optimization of Nonlinear Equations. *J Glob Optim*, 59(2):503–526, 2014.

R. Misener, J. B. Smadbeck, and C. A. Floudas. Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2. *Optim Meth Soft*, 30(1):215–249, 2015.

M. Mistry and R. Misener. Optimising heat exchanger network synthesis using convexity properties of the logarithmic mean temperature difference. *Comput Chem Eng*, 94:1–17, 2016.

D. Mocsny and R. Govind. Decomposition strategy for the synthesis of minimum-unit heat exchanger networks. *AIChE J*, 30(5):853 – 856, 1984.

A. M. Niziolek, O. Onel, M. M. F. Hasan, and C. A. Floudas. Municipal solid waste to liquid transportation fuels part ii: Process synthesis and global optimization strategies. *Comput Chem Eng*, 74:184 – 203, 2015.

R. Oberdieck, N. A. Diangelakis, I. Nascu, M. M. Papathanasiou, M. Sun, S. Avraamidou, and E. N. Pistikopoulos. On multi-parametric programming and its applications in process systems engineering. *Chem Eng Res Des*, 116:61 – 82, 2016.

J. Omer, M. Towhidi, and F. Soumis. The positive edge pricing rule for the dual simplex. *Comput. Oper. Res.*, 61:135–142, 2015.

J. Ostrowski, J. Linderoth, F. Rossi, and S. Smriglio. Constraint orbital branching. In *Proceedings of the*

Integer Programming and Combinatorial Optimization: 13th International Conference, IPCO 2008 Bertinoro, Italy, 2008., pages 225–239. Springer-Verlag, 2008.

J. Ostrowski, A. Vannelli, and M. F. Anjos. Symmetry in scheduling problems. In *Cahier du GERAD G-2010-69, GERAD,QC, Canada*, 2010.

J. Ostrowski, J. Linderoth, F. Rossi, and S. Smiriglio. Orbital branching. *Math Program*, 126(1):147–178, 2011.

J. Ostrowski, M. F. Anjos, and A. Vannelli. Modified orbital branching for structured symmetry with an application to unit commitment. *Math Program*, 150(1):99–129, 2015.

C. H. Papadimitriou, editor. *Computational Complexity*. Addison-Wesley, 1994.

L. G. Papageorgiou, N. Shah, and C. C. Pantelides. Optimal scheduling of heat-integrated multipurpose plants. *Ind Eng Che Res*, 33(12):3168–3186, 1994.

K. P. Papalexandri and E. N. Pistikopoulos. A multiperiod MINLP model for the synthesis of flexible heat and mass exchange networks. *Comput Chem Eng*, 18(11):1125 – 1139, 1994.

S. A. Papoulias and I. E. Grossmann. A structural optimization approach in process synthesis ii. *Comput Chem Eng*, 7(6):707–721, 1983.

W.R. Paterson. A replacement for the logarithmic mean. *Chem Eng Sci*, 39(11):1635 – 1636, 1984.

M. E. Pfetsch and T. Rehn. A computational comparison of symmetry handling methods for mixed integer programs. technical report. Technical report, Optimization Online, 2015.

T. K. Pho and L. Lapidus. Topics in computer-aided design: Part ii. synthesis of optimal heat exchanger networks by tree searching algorithms. *AIChE J*, 19(6):1182 – 1189, 1973.

G. T. Polley and P. J. Heggs. Don't let the pinch pinch you. *Chem Eng Program*, 95(12):27 – 36, 1999.

J. F. Puget. Automatic detection of variable and value symmetries. In P. van Beek, editor, *Proceedings of the Principles and Practice of Constraint Programming - CP 2005: 11th International Conference, Spain, 2005.*, pages 475–489. Springer Berlin Heidelberg, 2005.

A. Qualizza, P. Belotti, and F. Margot. Linear programming relaxations of quadratically constrained quadratic programs. In L. Lee and S. Leyffer, editors, *Mixed Integer Nonlinear Programming*, pages 407–426, New York, NY, 2012. Springer New York.

A. Ramani and I. L. Markov. Automatically exploiting symmetries in constraint programming. In B. V. Faltings, A. Petcu, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints*, pages 98–112. Springer-Verlag, 2005.

A. Ramani, F. Aloul, I. Markov, and K. A. Sakallah. Breaking instance-independent symmetries in exact graph coloring. *JAIR*, 1:324– 329, 03 2006.

D. J. S. Robinson, editor. *A Course in the Theory of Groups*. Graduate Texts in Mathematics. Springer, 1996.

K. Roos, T. Terlaky, and J-P. Vial. Theory and algorithms for linear optimization - an interior point approach. In *Wiley-Interscience series in discrete mathematics and optimization*, 1998.

N. V. Sahinidis. BARON: A general purpose global optimization software package. *J Glob Optim*, 8(2): 201–205, 1996.

D. Salvagnin. A dominance procedure for integer programming. Master's thesis, University of Padova, 2005.

M. R. Shelton and I. E. Grossmann. Optimal synthesis of integrated refrigeration systems-i: Mixed-integer programming model. *Comput Chem Eng*, 10(5):445–459, 1986.

U. V. Shenoy, editor. *Heat exchanger network synthesis: process optimization by energy and resource analysis*. Gulf Professional Publishing, 1995.

H. D. Sherali and J. C. Smith. Improving discrete model representations via symmetry considerations. *Manag. Sci.*, 47(10):1396–1407, 2001.

A. Skjäl, R. Misener, T. Westerlund, and C. A. Floudas. A generalization of the classical źbb convex underestimation via diagonal and nondiagonal quadratic terms. *J Optim Theory Appl*, 154(2):462–490, 2012.

E. M. B. Smith and C. C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Comput. Chem. Eng.*, 23(4):457–478, 1999.

E.M.B. Smith, C.C Pantelides, and G.V Reklaitis. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimization of nonconvex MINLP's. *Comput Chem Eng*, 25:1399–1401, 2001.

R. Smith. *State of the art in process integration*, volume 20, pages 1337–1345. Elsevier Science Ltd, 15-16 edition, 10 2000.

R T. Sousa, S Liu, L G. Papageorgiou, and N Shah. Global supply chain planning for pharmaceuticals. *Chem Eng Res Des*, 89(11):2396 – 2409, 2011.

E. Specht. `http://www.packomania.com/`, 2018.

H. Stefansson, S. Sigmarsdottir, P. Jensson, and N. Shah. Discrete and continuous time representations and mathematical models for large production scheduling problems: A case study from the pharmaceutical industry. *Europ J of Oper Res*, 215(2):383 – 392, 2011.

L. M. Stephen, T. Fischer, T. Gally, G. Gamrath, A. Gleixner, R. L. Gottwald, G. Hendel, T. Koch, M. E. Lübbecke, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, S Schenker,

R. Schwarz, F. Serrano, Y. Shinano, D. Weninger, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 4.0. Technical Report 17-12, ZIB, Berlin, 2017.

L. Tantimuratha, A. C. Kokossis, and F. U. Müller. The heat exchanger network design as a paradigm of technology integration. *Appl Therm Eng*, 20(15):1589 – 1605, 2000.

M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Math Program*, 103(2):225 – 249, 2005.

G. A. Tijssen and G. Sierksma. Balinski - tucker simplex tableaus: Dimensions, degeneracy degrees, and interior points of optimal faces. *Math Program*, 81(3):349–372, 1998.

K. K. Trivedi, B. K. O'Neill, J. R. Roach, and R. M. Wood. Systematic energy relaxation in mer heat exchanger networks. *Comput Chem Eng*, 14(6):601 – 611, 1990.

V. V. Vazirani, editor. *Approximation Algorithms*. Springer, 2001.

D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

K. L. Yee and N. Shah. Improving the efficiency of discrete time scheduling formulation. *Comput Chem Eng*, 22, 1998.

T. F. Yee and I. E. Grossmann. Simultaneous optimization models for heat integration II. Heat exchanger network synthesis. *Comput Chem Eng*, 14(10):1165–1184, 1990.

L. Youdong and S. Linus. The global solver in the LINDO API. *Optim Meth Soft*, 24(4-5):657–668, 2009.

H. Yu, H. Fang, P. Yao, and Y. Yuan. A combined genetic algorithm/simulated annealing algorithm for large scale system energy integration. *Comput Chem Eng*, 24(8):2023 – 2035, 2000.

A. Zanette, M. Fischetti, and E. Balas. Lexicography and degeneracy: can a pure cutting plane algorithm work? *Math Program*, 130(1):153–176, 2011.

X. G. Zhao, B. K. O'Neill, J. R. Roach, and R. M. Wood. Heat integration for batch processes. *Chem Eng Res Des*, 76(6):700 – 710, 1998.