2018

Mixed Integer Conic Optimization and its Applications

Mohammad Shahabsafa
Lehigh University

Mixed Integer Conic Optimization and Its Applications

by

Mohammad Shahabsafa

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Industrial and Systems Engineering

Lehigh University
January 2019

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Mohammad Shahabsafa

Mixed Integer Conic Optimization and Its Applications

Date

Tamás Terlaky, Dissertation Director, Chair

Accepted Date

Committee Members

Dr. Tamás Terlaky, Committee chair

Dr. Julio C. Góez

Dr. Joaquim R. R. A. Martins

Dr. Martin Takáč

Dr. Natasha Vermaak

Dr. Luis F. Zuluaga

Acknowledgement

I would like to sincerely thank my advisor, Prof. Tamás Terlaky, for his patience, guidance, and inspiration throughout my PhD studies. I wish to express my gratitude for the invaluable opportunities that he provided for me during the last five years. My thanks also go to the members of my dissertation committee for providing valuable comments on my dissertation.

I would like to thank my wife, Sajedeh, whose love, kindness, and support were crucial to overcome the difficulties of the PhD life. I would also like to thank my parents, my brother, and my in-laws for their endless sacrifice and selfless support.

I would like to thank Prof. Luis F. Zuluaga from the ISE Department, Lehigh University and Prof. Joaquim R. R. A. Martins from the Aerospace Engieering Department, University of Michigan, whose insight and support were crucial in the structural design optimization project. I would also like to thank Ali Mohammad-Nezhad and Weiming Lei, PhD students from the ISE Department, Lehigh, and Sicheng He from the Aerospace Engineering Department, University of Michigan for their contributions in the project.

I gratefully acknowledge the contributions of Pennsylvania Department of Corrections personnel for their valuable support of the inmate assignment and inmate transportation projects. I specifically thank Kristofer Bret Bucklen, director of Bureau of Planning, Research, and Statistics (PRS) and William F. Nicklow, director of the Office of the Population Management (OPM), for their leadership and invaluable insight in the projects. I wish to express my gratitude to Jessica Campbell from the PRS for her tireless, strong support. I acknowledge Jennifer Hendricks and Tanya Brandt, from the OPM, for their invaluable efforts in testing the IADSS and patiently providing invaluable and insightful feedback. I am also grateful to Profs. Louis J. Plebani and George R. Wilson from the ISE Department, Lehigh for their contributions in the project. Last but not least, I acknowledge the efforts and contributions of Chaitanya Gudapati and Anshul Sharma, former M.Sc. students of the ISE Department at Lehigh, who had significant contributions on the inmate assignment and inmate transportation projects.

Contents

List of Tables

List of Figures

Abstract

In this dissertation, we present our work on the theory and applications of Mixed Integer Linear Optimization (MILO) and Mixed Integer Second Order Cone Optimization (MISOCO). The dissertation is separated in three parts.

In the first part, we focus on the theory of MISOCO. We develop a methodology to efficiently identify the cases of Disjunctive Conic Cuts (DCCs) that do not tighten the description of the feasible set of the MISOCO problem. We introduce the concept of pathological disjunctions for general mixed integer conic optimization problems, and use the concept of pathological disjunctions to identify redundant DCCs for MISOCO problems.

In the second part, we study truss design problems. We propose various mathematical models for minimum-weight discrete truss design problems, considering force balance equations, Hooke's law, displacement bounds, yield stress, and Euler buckling constraints, while only discrete cross-sectional areas of the bars are allowed. Additionally, we propose a novel solution methodology to solve the resulting discrete truss design problems. Numerical results indicate that, compared to directly using a commercial solver to solve the problems, the new solution methodology is remarkably faster and results in significantly better solutions.

In the third part, we focus on two critical problems that every correctional system faces on a daily basis, namely, the Inmate Assignment Problem (IAP) and Inmate

Transportation Problem (ITP). The IAP concerns the assignment of inmates to correctional institutions and scheduling of their treatment programs. We present the Inmate Assignment Decision Support System (IADSS) that is developed for the Pennsylvania Department of Corrections (PADoC) and has assisted the PADoC to significantly improve the inmate assignment process. The core of the IADSS is a multi-objective MILO model which is developed with the goal to simultaneously assign inmates to correctional institutions and schedule their treatment programs, while considering various factors, rules, and criteria of the assignment. Implementation of the IADSS at the PADoC has resulted in substantial monetary savings, security enhancement for the correctional institutions, and improvement of public safety. Additionally, we present our work on the ITP. We propose a multi-objective MILO model for the ITP and demonstrate the effectiveness of the model in reducing the costs of the inmate transportation process.

Chapter 1

Introduction

1.1 Background

A Mixed Integer Conic Optimization (MICO) problem is to minimize a linear function over the intersection of a closed pointed convex cone and a set of affine constraints, where a subset of the variables are constrained to be integer. A MICO problem is defined as

$$\begin{aligned} \min \quad & \langle c, x \rangle \\ \text{s.t.} \quad & A(x) = b, \\ & x \in \mathcal{K}, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned}$$

where $c \in \mathbb{R}^n$, $\langle c, x \rangle$ denotes the inner product of vectors c and x, $A(\cdot)$ is a linear map from $\mathbb{R}^n$ to $\mathbb{R}^m$, $b \in \mathbb{R}^m$, $p \in \mathbb{R}$, and $\mathcal{K}$ is a closed pointed convex cone.

In this dissertation, we focus on two important classes of MICO problems, namely, Mixed Integer Linear Optimization (MILO) and Mixed Integer Second Order Cone Optimization (MISOCO) problems.

A MILO problem is to minimize a linear function over a polyhedral set, where a subset of the variables are constrained to be integer. A MILO problem can be defined as

$$\begin{aligned} \min \quad & c^T x, \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$, and $p \in \mathbb{R}$. Numerous problems can mathematically be formulated as a MILO model, including but not limited to network design problems [Bertsimas and Tsitsiklis, 1997], the assignment problem [Flood, 1953, Kuhn, 1955, Votaw and Orden, 1952], the traveling salesman problem [Cook, 2012, Dantzig et al., 1954], the knapsack problem [Dantzig, 1957], and crew scheduling problems [Arabeyre et al., 1969, Caprara et al., 1998].

Gomory [1958, 1960b, 1963] proposed a *cutting plane algorithm* which was the first finitely-terminating algorithm to solve pure integer linear optimization problems with bounded feasible sets and obtain the global optimal solution. Land and Doig [1960] were the first to propose the *Branch and Bound* (B&B) algorithm to solve MILO problems. Dakin [1965] proposed the B&B algorithm to solve a general mixed integer optimization problem given that the optimal solution of the continuous relaxations of the problem at the nodes of the B&B tree can be computed. A variant of the B&B algorithm is the branch and cut algorithm where cuts are added in solving the subproblems at the nodes of the B&B tree [Crowder et al., 1983, Van Roy and Wolsey, 1987].

A MISOCO problem is to minimize a linear function over the intersection of an affine space and a Cartesian product of second order cones. A MISOCO problem can be defined as

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \in \mathcal{L}, \\ & x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}, \end{aligned} \tag{1.1}$$

where $c \in \mathbb{R}^n,\ A \in \mathbb{R}^{m\times n},\ b \in \mathbb{R}^m$, $\mathcal{L} = \mathcal{L}^{n_1} \times \mathcal{L}^{n_2} \times \cdots \times \mathcal{L}^{n_k}$ is the Cartesian product of the second order cones, $\mathcal{L}^{n_i} = \left\{ x^i = (x^i_1, x^i_2, \ldots, x^i_{n_i}) \in \mathbb{R}^{n_i} \mid x^i_1 \geq \|x^i_{2:n_i}\|_2 \right\}$, and

$$x = \left\{ \begin{pmatrix} x^1 \\ x^2 \\ \vdots \\ x^k \end{pmatrix} : x^i \in \mathbb{R}^{n_i},\ i = 1, \ldots, k \right\}.$$

A variety of problems can mathematically be formulated as a MISOCO problem. Aktürk et al. [2014] propose a MISOCO model for airline recovery optimization with the goal to investigate the trade-off between flight delays and cost of recovery. Bertsimas and Shioda [2009] develop a MISOCO model for cardinality constrained portfolio optimization problems, and propose a branch and bound based algorithm to solve the problem using its special structure. Jabr et al. [2012] propose a MISOCO model for the minimum loss distribution network reconfiguration problem. In recent years, optimization packages including CPLEX [IBM Knowledge Center, 2017], GuRoBi [Gurobi Optimization Inc., 2016], and MOSEK [MOSEK, 2017] have added the capability of solving MISOCO problems, which has led to broader application of MISOCO in tackling complex problems arising in a variety of industries.

1.2 Dissertation overview

In this dissertation, we present our work on the theory and applications of MILO and MISOCO. We focus on the theory of MISOCO in Chapter 2 and we study the applications of MILO in Chapters 3, 4, 5, and 6.

In Chapter 2, we study disjunctive conic cuts (DCCs) for MISOCO problems. The main goal of Chapter 2 is to develop a methodology to efficiently identify the cases of DCCs that do not tighten the description of the feasible set of the MISOCO problem. We introduce the concept of pathological disjunctions for general mixed integer conic optimization problems, and use the concept of pathological disjunctions to identify redundant DCCs for MISOCO problems.

In Chapters 3 and 4, we study truss design problems. In Chapter 3, we develop novel mathematical models for minimum-weight discrete truss design problems, considering force balance equations, Hooke's law, displacement bounds, yield stress, and Euler buckling constraints, while only discrete cross-sectional areas of the bars are allowed. In Chapter 4, we propose a novel solution methodology to solve the resulting discrete truss design problems. Numerical results indicate that, compared to directly using a commercial solver to solve the problems, the new solution methodology is remarkably faster and results in significantly better solutions.

In Chapter 5, we present our pioneering work on a critical problem that every correctional system faces on a daily basis. We study the problem of inmate assignment to correctional institutions and scheduling of their treatment programs. We develop a novel multi-objective MILO model with the goal to simultaneously assign inmates to correctional institutions and schedule their treatment programs, while considering various factors, rules, and criteria of the assignment. We have developed a decision support system for the Pennsylvania Department of Corrections (PADoC), which helps the PADoC to significantly improve the inmate assignment process. The core of the system is the MILO model which is developed for the simultaneous assignment of inmates and scheduling their treatment programs. Implementation of the system at the PADoC has resulted in substantial monetary savings, security enhancement for the correctional institutions, and improvement of the public safety.

In Chapter 6, we present our work on the inmate transportation problem (ITP). We formally define the ITP and propose a multi-objective MILO model for the problem. Then we present the numerical results to demonstrate the effectiveness of the model in reducing the costs of the inmate transportation process.

Finally, we present the summary and conclusions of the dissertation in Chapter 7, and elaborate the future research that can be built upon the dissertation.

1.3 Publications and Accomplishments

The results and accomplishments of this dissertation including the papers, technical reports and the awards are as follows:

- Chapter 2:

  - Shahabsafa M., Góez J. C., and Terlky T., (2018) On pathological disjunctions and redundant disjunctive conic cuts, *Operations Research Letters*, 46(5):500-504, doi: `https://doi.org/10.1016/j.orl.2018.07.004`
  - Shahabsafa M., Góez J. C., and Terlaky T. (2018) Supplement to the paper: on pathological disjunctions and redundant disjunctive conic cuts, Technical report, ISE Department, Lehigh University, URL: `https://ise.lehigh.edu/sites/ise.lehigh.edu/files/18T_007.pdf`

- Chapters 3 and 4:

  - Shahabsafa M., Mohammad-Nezhad A., Terlaky T., Zuluaga L. F., He S., Hwang J. T., Martins J. R. R. A. (2018) A novel approach to discrete truss design problems using mixed integer neighborhood search. *Structural and Multidisciplinary Optimization*, 58(6):2411-2429, doi: `https://doi.org/10.1007/s00158-018-2099-8`.
  - Shahabsafa M., Lei W., Terlaky T., Zuluaga L. F., He S., Martins J. R. R. A. (2018) The application of the NS-MILO algorithm to multi-scenario truss design problems. *Working Paper*
  - Shahabsafa M., Lei W., Terlaky T., Zuluaga L. F., He S., Martins J. R. R. A. (2018) On the lower bound of truss design problems. *Working Paper*

- Chapter 5:

  - The work was honored to win the INFORMS 2017 Daniel H. Wagner Prize.
    `https://www.informs.org/Recognizing-Excellence/INFORMS-Prizes/Daniel-H.-Wagner-Prize-for-Excellence-in-Operations-Research-Practice`
  - Shahabsafa M., Terlaky T., Gudapati C., Sharma A., Plebani L., Wilson G., Bucklen K. B. (2018) The Inmate Assignment and Scheduling Problem and its Application in the PA Department of Correction, *Interfaces*, 48(5):467-483, doi: `https://doi.org/10.1287/inte.2018.0962`
  - Santos P., Shahabsafa M., Terlaky T. (2018) Optimization saves state prison system millions, *ORMS Today*, 45(3):34-38, doi: `https://doi.org/10.1287/orms.2018.03.10`

- Terlaky T., Shahabsafa M., Plebani L., Wilson G., Sharma A., Gudapati C. (2018) Assigning inmates to correctional institutions and programs, *IFORS Newsletter*, 12(3)2:5, URL: http://ifors.org/september-2018-issue/

- Chapter 6:

- Sharma A, Shahabsafa M, Terlaky T. (2018) The Inmate transportation Problem and its Application in the PA Department of Correction, Accepted in final form for publication in H. Yang, and R. Qiu (Eds.), *Advances in Service Science Proceedings of the 2018 INFORMS International Conference on Service Science*, Springer

1.4 Notation

Following is the list of the notation conventions used in the dissertation.

$\mathbb{R}$	Set of real numbers
$\mathbb{R}_+$	Set of non-negative real numbers
$\mathbb{Z}$	Set of integer numbers
$i, j, k, \ldots$	indices are denoted with lower case letters
$\lvert \cdot \rvert$	Absolute value of a scalar
$\mathcal{A}, \mathcal{B}$	Sets are denoted with calligraphic letters
A, B, K	Matrices are denoted with capital letters
c, q	Parameter vectors are denoted with lower case letters
x, x^i, σ	Decision variable vectors are denoted with lower case letters
$K \succeq 0$	Positive semi-definite matrices
$K \succ 0$	Positive definite matrices
$\mathcal{L}, \mathcal{L}^{n_i}$	Second order cones
$\lVert \cdot \rVert$	Euclidean norm of a vector
$\mathcal{A} \bigvee \mathcal{B}$	The disjunction of sets $\mathcal{A}$ and $\mathcal{B}$
$\operatorname{diag}(\cdot)$	The diagonal matrix of a vector
$\operatorname{lin}(\cdot)$	Lineality space of a set
$\operatorname{conv}(\cdot)$	Convex hull of a set
$\operatorname{proj}_{\mathcal{X}}(\mathcal{F})$	The orthogonal projection of set $\mathcal{F}$ on subspace $\mathcal{X}$

Chapter 2

Disjunctive Conic Cuts

In this chapter we work on identifying the cases where Disjunctive Conic Cuts (DCCs) do not strengthen the formulation of MISOCO problems. In Section 2.1, we overview the literature of adding nonlinear cuts for MISOCO problems. In Section 2.2, we present a brief overview of the derivation of DCCs for the MISOCO problems. In Section 2.3, we define pathological disjunctions and propose results on how to identify redundant DCCs. In Section 2.4, we explore some common instances of redundant DCCs for MISOCO problems. We present numerical results to demonstrate the effect of adding redundant DCCs to a MISOCO problem in Section 2.5. Finally, Section 2.6 presents our conclusions.

2.1 Introduction

A General MISOCO problem is defined by (1.1). In principle, one can solve a MISOCO problem exactly using a branch and cut methodology. One of the key elements of branch and cut methodology is the derivation of effective and efficient cuts to strengthen the formulation. Studies have shown performance improvements that one can obtain in a branch and cut framework using various cuts including Chvátal-Gomory cuts, mixed-integer rounding cuts, lift-and-project cuts, and split cuts [see e.g., Nemhauser and Wolsey, 1990]. Most of the work on cut generation has been focused on obtaining valid linear inequalities. However, the possibility of generating nonlinear cuts for MISOCO problems have recently received significant attention in the optimization community. Stubbs and Mehrotra [1999] extended Balas et al. [1993] lift and project procedure to 0-1 mixed integer convex optimization problems. They derive valid inequalities for mixed integer problems by solving a convex optimization sub-problem. Çezik and Iyengar [2005] derive convex cuts for mixed 0-1 conic optimization problems. They consider the linear cone, the second-order cone, and the cone of positive semidefinite matrices and extend a variety of techniques, used in generating cuts for MILO problems such as Gomory cuts [Gomory, 1960a] and lift and project cuts.

For MISOCO problems in particular, Atamtürk and Narayanan [2010, 2011] extended the idea of mixed integer rounding cuts developed by Nemhauser and Wolsey [1990]. They reformulated a SOC in terms of two-dimensional polyhedral SOCs and designed a rounding procedure to derive conic cuts for the original MISOCO problem. Kılınç-Karzan and Yıldız [2014] consider a two-term disjunction on a SOC and derive closed-form convex inequalities describing the convex hull of the intersection of the disjunction with the cone. They characterize the cases where one SOC inequality is enough to describe the mentioned convex hull. Drewes [2009] presents lift-and-project based linear and convex quadratic cuts for MISOCO problems. Dadush et al. [2011] extend the idea of split cuts [Nemhauser and Wolsey, 1990] for a full dimensional ellipsoid. They consider parallel disjunctions on ellipsoids and derive a conic cut which describes the convex hull of the ellipsoid intersected with the disjunctive set. Andersen and Jensen [2013] extend the idea of intersection cuts [Balas, 1971] to mixed integer conic quadratic sets. Modaresi et al. [2015] explains the relationship between mixed integer rounding cuts [Atamtürk and Narayanan, 2010, 2011] and split cuts, [Dadush et al., 2011] and discusses the trade-off between computational cost of adding the split cuts and strength of the formulation resulted from adding them to the model.

Belotti et al. [2013, 2015, 2017] consider a disjunction on a general MISOCO problem and generate a class of cuts called *Disjunctive Conic Cuts (DCCs)* and *Disjunctive Cylindrical Cuts (DCyC)*. The DCCs and DCyCs describe the convex hull of the intersection of the disjunction with the feasible set of the continuous relaxation of a MISOCO problem. In other words, the intersection of the DCC with the feasible set of the continuous relaxation of the MISOCO problem is equal to the convex hull of the intersection of the disjunction with the feasible set of the continuous relaxation problem. In this chapter we use the approach proposed by Belotti et al. [2017] and provide tests to improve the derivation of effective DCCs. The aim of the tests is to save computational time and effort by identifying cases when the DCCs do not provide further tightening of the formulation at hand.

2.2 DCCs for MISOCO problems

We provide an overview of the derivation of DCCs for the MISOCO problem (1.1) and a parallel disjunction. In this section, we assume that $\mathcal{L}$, defined in model (1.1), is a single SOC.

We use the approach presented by Belotti et al. [2017] to rewrite the feasible set of the continuous relaxation of (1.1) in terms of the null space of the affine constraints. Let $\mathcal{C} = \mathcal{L} \cap \{x \mid Ax = b\}$, and recall from Belotti et al. [2017] that $\{x \in \mathbb{R}^n \mid Ax = b\} = \{x \in \mathbb{R}^n \mid \exists w \in \mathbb{R}^\ell, x = x^0 + Hw\}$, where $x^0 \in \mathbb{R}^n$, $Ax_0 = b$, and we assume that $\{x|Ax = b\}$ is non-empty. Here, the columns of $H \in \mathbb{R}^{n\times\ell}$ form a basis for the null space of A, and $\ell = n - m$ is the dimension of the null space.

Hence, $\mathcal{C}$ in the null space of matrix A may be represented as

$$\hat{\mathcal{C}} = \{w \in \mathbb{R}^\ell \mid w^\top P w + 2p^\top w + \rho \le 0,\ x^0 + H_1 w \ge 0\}, \tag{2.1}$$

where H_1 is the first row of H, $P \in \mathbb{R}^{\ell\times\ell}$ is a symmetric matrix, $p \in \mathbb{R}^\ell$, and $\rho \in \mathbb{R}$. We denote the quadratic set (2.1) by the triplet (P, p, ρ). Notice that if a set or parameter is defined both in $\mathbb{R}^n$ and the reduced null space $\mathbb{R}^\ell$, then the one defined in the null space is indicated by adding a "ˆ" to the set or parameter. It is proved [Góez, 2013, Lemma 4.2] that the matrix P in (2.1) has at most one non-positive eigenvalue. That restricts the possible shapes of $\hat{\mathcal{C}}$ to ellipsoids, paraboloids, hyperboloids of two sheets, or SOCs.

Consider now a parallel disjunction of the form

$$\mathcal{A} = \{x \in \mathbb{R}^n \mid a^\top x \le \alpha\} \bigvee \mathcal{B} = \{x \in \mathbb{R}^n \mid a^\top x \ge \beta\}, \tag{2.2}$$

where $a \in \mathbb{R}^n$, $\alpha, \beta \in \mathbb{R}$, and $\alpha < \beta$. We can rewrite the disjunction in the null space of the affine constraints as follows

$$\hat{\mathcal{A}} = \{w \in \mathbb{R}^\ell \mid \hat{a}^\top w \le \hat{\alpha}\} \bigvee \hat{\mathcal{B}} = \{w \in \mathbb{R}^\ell \mid \hat{a}^\top w \ge \hat{\beta}\}, \tag{2.3}$$

where $\hat{a} \in \mathbb{R}^\ell$, $\alpha, \beta \in \mathbb{R}$, $\hat{a} = H^\top a$, $\hat{\alpha} = \alpha - a^\top x_0$, and $\hat{\beta} = \beta - a^\top x_0$. We assume that $\hat{\mathcal{C}} \cap \hat{\mathcal{A}} \neq \emptyset$ and $\hat{\mathcal{C}} \cap \hat{\mathcal{B}} \neq \emptyset$, and we may assume w.l.o.g. that $\|\hat{a}\|_2 = 1$ and $\hat{\alpha} < \hat{\beta}$. Using this disjunction we now recall the main elements of the derivation of the DCC; for a detailed derivation see Belotti et al. [2013, 2015], Góez [2013]. The key result in that procedure is the existence of a uni-parametric family of quadratic sets defined as follows

$$\mathcal{Q}(\tau) = \left\{w \in \mathbb{R}^\ell \mid w^\top P(\tau) w + 2p(\tau)^\top w + \rho(\tau) \le 0\right\}, \tag{2.4}$$

where $P(\tau) = P + \tau\hat{a}\hat{a}^\top$, $p(\tau) = p - \frac{\tau}{2}(\alpha + \beta)\hat{a}^\top$, and $\rho(\tau) = \rho + \tau\alpha\beta$. Observe that $Q(\tau)$ is not necessarily convex. It is shown [Belotti et al., 2013] that the DCC is given by one of the roots of the equation $p(\tau)^\top P(\tau)^{-1}p(\tau) - \rho(\tau)$, which is a quadratic function of τ. That result provides an explicit formula for obtaining DCCs. However, before adding a DCC it is important to know if it actually tightens the formulation, or not. For that purpose, in this chapter we provide a set of tests to verify the effectiveness of a DCC.

2.3 Redundant DCCs and DCyCs

For the classification of redundant DCCs in this section, we consider parallel disjunctions as defined in Eqs (2.2) and (2.3). We use the results of Belotti et al. [2017], which ensures that if a DCC cuts a feasible point from (2.1), then it also cuts a feasible point from (1.1). Hence, we focus in this section on sets of the form presented in (2.1). Now, when we intersect a disjunction of the form (2.3) with the set $\hat{\mathcal{C}}$ as defined in (2.1) and derive a DCC, we show that in some instances it does not cut off any part of the feasible region. That implies that the derived DCC is redundant, negatively impacting the effectiveness of branch and cut procedures.

Definition 2.1 (Pathological disjunction)**.** *Let $\mathcal{X} \in \mathbb{R}^n$ be a closed convex set, and consider the disjunction $\mathcal{A} \cup \mathcal{B}$ as defined in (2.2). If $\operatorname{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{X}$, then disjunction $\mathcal{A} \cup \mathcal{B}$ is pathological for the set $\mathcal{X}$.*

Notice that if disjunction $\mathcal{A} \cup \mathcal{B}$ is pathological for the closed convex set $\mathcal{X}$, then every convex disjunctive cut using $\mathcal{A} \cup \mathcal{B}$ will be redundant, since all possible cuts must include $\operatorname{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B}))$. Now, suppose that the DCC exists for set $\mathcal{X}$ and disjunction $\mathcal{A} \cup \mathcal{B}$. In this case, the DCC, which is the tightest possible cut, will be redundant as well.

In this section, we explore pathological disjunctions, and thus redundant DCCs and DCyCs for MISOCO problems. The redundant cases are first defined for a general convex set, and then they are presented for MISOCO problems.

2.3.1 Redundant DCCs

We first consider sets resulting from the intersection of a closed pointed convex cone and a disjunctive set. In particular, we are interested in the instances when the vertex of the cone is in one of the halfspaces defining the disjunctive set. In this situation we have the following result.

Theorem 2.1 (Conic pathological disjunction)**.** *Let $\mathcal{K} \subseteq \mathbb{R}^n$ be a closed pointed convex cone with vertex v, and consider two half spaces $\mathcal{A} = \{x \in \mathbb{R}^n \mid a^\top x \leq \alpha\}$ and $\mathcal{B} = \{x \in \mathbb{R}^n \mid a^\top x \geq \beta\}$, such that $\alpha < \beta$, $\mathcal{K} \cap \mathcal{A} \neq \emptyset$ and $\mathcal{K} \cap \mathcal{B} \neq \emptyset$. If $v \in \mathcal{A} \cup \mathcal{B}$, then* $\operatorname{conv}(\mathcal{K} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{K}$.

Proof. As $\mathcal{K}$ is convex, we have $\operatorname{conv}(\mathcal{K} \cap (\mathcal{A} \cup \mathcal{B})) \subseteq \mathcal{K}$. Thus, we only need to prove that $\mathcal{K} \subseteq \operatorname{conv}(\mathcal{K} \cap (\mathcal{A} \cup \mathcal{B}))$. Let $x \in \mathcal{K}$ be given, then we need to show that $x \in \operatorname{conv}(\mathcal{K} \cap (\mathcal{A} \cup \mathcal{B}))$. If $x \in \mathcal{A} \cup \mathcal{B}$, then $x \in \operatorname{conv}(\mathcal{K} \cap (\mathcal{A} \cup \mathcal{B}))$. Now suppose $x \notin \mathcal{A} \cup \mathcal{B}$, i.e, $\alpha < a^\top x < \beta$. We know that the vertex v of the cone is in one of the disjunctive half spaces. Without loss of generality, we may assume that $a^\top v \leq \alpha$. Let $r = x - v$. Vector r is in fact a ray of the cone $\mathcal{K}$, since $x \in \mathcal{K}$. As $a^\top v \leq \alpha$ and $\alpha < a^\top x$, we have $a^\top r > 0$. We know that $\beta - a^\top x > 0$, therefore, there exists a $\gamma > 0$ such that $\gamma a^\top r = \beta - a^\top x$, and we obtain that $a^\top(x + \gamma r) = \beta$. Let $\bar{x} = x + \gamma r$. As $x = v + r$, we have

$$\bar{x} = v + (1 + \gamma)r.$$

Vector r is a ray of the cone $\mathcal{K}$, so we have $\bar{x} \in \mathcal{K}$. As $v \in \mathcal{A}$ and $\bar{x} \in \mathcal{B}$, we can conclude that $v, \bar{x} \in \mathcal{K} \cap (\mathcal{A} \cup \mathcal{B})$. Let $\eta = \frac{\gamma}{1+\gamma}$, then we have

$$x = \eta v + (1-\eta)\bar{x},$$

and we obtain that $x \in \operatorname{conv}(\mathcal{K} \cap (\mathcal{A} \cup \mathcal{B}))$, which completes the proof.

$\square$

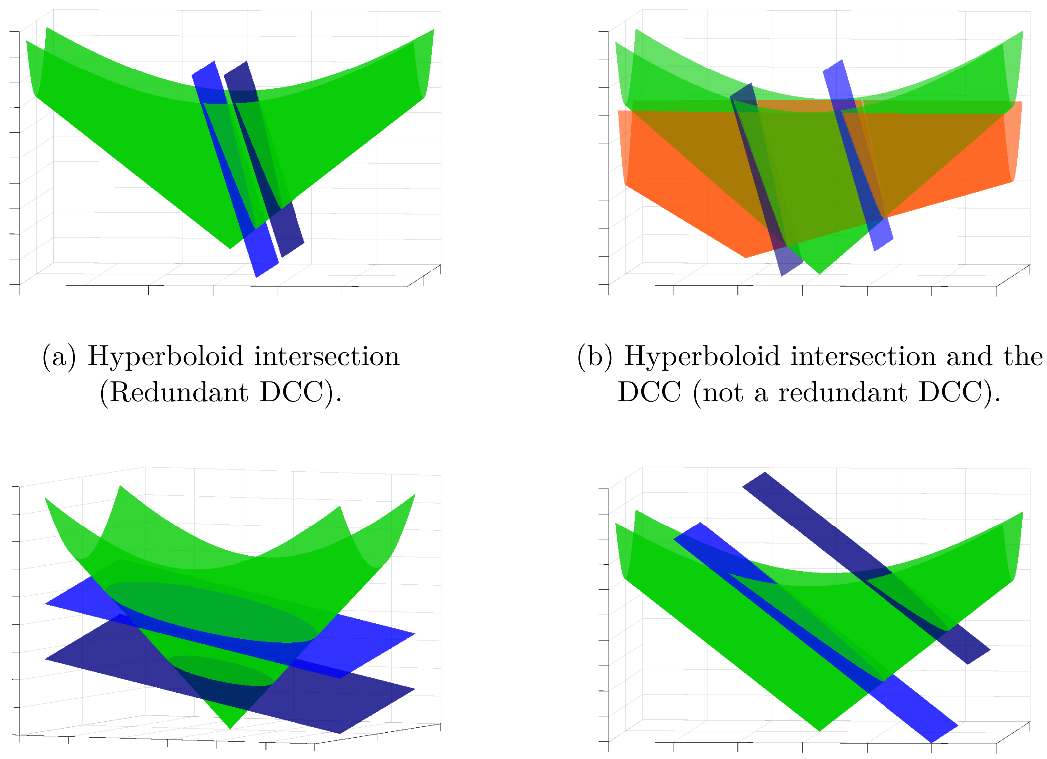
Notice that the result of Theorem 2.1 holds for any general closed pointed convex cone $\mathcal{K}$. However, to identify the redundant DCCs for MISOCO problems, in this section we focus on the special case where $\mathcal{K}$ is a SOC. We use Theorem 2.1 to characterize conic redundant DCCs for MISOCO problems.

Corollary 2.1 (Redundant DCCs for MISOCO)**.** *If the set $\hat{\mathcal{C}}$, as defined in* (2.1)*, is a cone and its vertex is in one of the disjunctive halfspaces $\hat{\mathcal{A}}$ or $\hat{\mathcal{B}}$, as defined in* (2.3)*, then the DCC is equal to $\hat{\mathcal{C}}$.*

The main consequence of Corollary 2.1 is that in this case a DCC does not cut off any part of the feasible set. This redundancy is illustrated in Figure 2.1. In Figures 2.1a and 2.1b, the intersections of the disjunctive hyperplanes with cone $\mathcal{C}$ are hyperboloids. Figure 2.1a is a redundant DCC, since the vertex of the cone is in one of the disjunctive half spaces; while in Figure 2.1b the DCC is not redundant. Figures 2.1c and 2.1d are other instances of the conic redundant DCCs, where the intersection of the cone with the hyperplanes are respectively an ellipsoid and a paraboloid.

Corollary 2.2 (Identification of a redundant DCC for MISOCO)**.** *If the following two conditions are satisfied for the set $\hat{\mathcal{C}}$ defined in* (2.1)*, and the disjunctive set defined in* (2.3)*, then we have a redundant DCC:*

- *the matrix P has exactly $n-1$ positive eigenvalues and one negative eigenvalue, and $p^\top P^{-1} p - \rho = 0$;*


(a) Hyperboloid intersection (Redundant DCC).

(b) Hyperboloid intersection and the DCC (not a redundant DCC).

(c) Ellipsoid intersection (Redundant DCC).

(d) Paraboloid intersection (Redundant DCC).

Figure 2.1: Illustration of the conic redundant DCCs.

- *the vertex of the cone* $v = P^{-1}p$ *satisfies either* $\hat{a}^\top v \geq \hat{\beta}$, *or* $\hat{a}^\top v \leq \hat{\alpha}$.

The first condition of Corollary 2.2 ensures that set $\hat{\mathcal{C}}$, defined in (2.1) is a cone and the second condition ensures that the set $\hat{\mathcal{C}}$ and disjunction (2.3) result in a redundant DCC.

2.3.2 Redundant DCyCs

We now consider sets resulting from the intersection of a closed convex cylinder and a disjunctive set. We first need to formally define a cylinder.

Definition 2.2. *(Lineality space [Rockafellar, 1997]) Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a closed convex set. The lineality space of $\mathcal{X}$ is defined as*

$$\mathrm{lin}(\mathcal{X}) = \{d \mid x + \alpha d \in \mathcal{X},\ \forall x \in \mathcal{X},\ \forall \alpha \in \mathbb{R}\}.$$

Lemma 2.1. *(Decomposition of a convex set [Rockafellar, 1997]) Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a nonempty closed convex set. Then we have*

$$\mathcal{X} = \mathcal{S} + \mathrm{lin}(\mathcal{X}),$$

where $\mathcal{S} = \mathcal{X} \cap \mathrm{lin}(\mathcal{X})^\perp$, and $\mathrm{lin}(\mathcal{X})^\perp$ is the orthogonal complement of $\mathrm{lin}(\mathcal{X})$.

Definition 2.3. *(Convex cylinder) Let $\mathcal{X}$ be a nonempty closed convex set, If $\mathrm{lin}(\mathcal{X}) \neq \{0\}$, then set $\mathcal{X}$ is a cylinder, and the set $\mathcal{S} = \mathcal{X} \cap \mathrm{lin}(\mathcal{X})^\perp$ is a base of the cylinder.*

The following theorem formalizes the cylindrical pathological disjunction.

Theorem 2.2 (Cylindrical pathological disjunction)**.** *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a closed convex cylinder, and consider two half spaces $\mathcal{A}$ and $\mathcal{B}$, defined in (2.2), such that $\alpha < \beta$, $\mathcal{X} \cap \mathcal{A} \neq \emptyset$, $\mathcal{X} \cap \mathcal{B} \neq \emptyset$. If $a \not\perp \mathrm{lin}(\mathcal{X})$, then $\mathrm{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{X}$.*

Proof. As $\mathcal{X}$ is convex it follows that $\mathrm{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})) \subseteq \mathcal{X}$. Thus, to complete the proof we need to show that $\mathcal{X} \subseteq \mathrm{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B}))$. The proof goes by contradiction. Assume to the contrary that there exists an $\bar{x} \in \mathcal{X}$ such that $\bar{x} \notin \mathrm{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B}))$. Then, we have that $\bar{x} \notin \mathcal{A} \cup \mathcal{B}$, and we obtain $\alpha < a^\top \bar{x} < \beta$.

We know that $a = \mathrm{proj}_{\mathrm{lin}(\mathcal{X})}(a) + \mathrm{proj}_{\mathrm{lin}(\mathcal{X})^\perp}(a)$, where $\mathrm{proj}_{\mathrm{lin}(\mathcal{X})}(a)$ and $\mathrm{proj}_{\mathrm{lin}(\mathcal{X})^\perp}(a)$ denote the orthogonal projections of vector a to the subspaces $\mathrm{lin}(\mathcal{X})$ and $\mathrm{lin}(\mathcal{X})^\perp$, respectively. As $a \notin \mathrm{lin}(\mathcal{X})^\perp$, we have $\mathrm{proj}_{\mathrm{lin}(\mathcal{X})}(a) \neq 0$. Let d be defined as

$$d = \mathrm{sign}\left(a^\top \mathrm{proj}_{\mathrm{lin}(\mathcal{X})}(a)\right) \mathrm{proj}_{\mathrm{lin}(\mathcal{X})}(a).$$

Thus we obtain $a^\top d > 0$. As $a^\top \bar{x} - \alpha > 0$, there exists $\gamma_\alpha > 0$ such that $\gamma_\alpha a^\top d \geq a^\top \bar{x} - \alpha$, i.e., we have $a^\top(\bar{x} - \gamma_\alpha d) \leq \alpha$. Let $\bar{x}_1 = \bar{x} - \gamma_\alpha d$, then $\bar{x}_\alpha \in \mathcal{A}$. Similarly, there exists $\gamma_\beta > 0$ such that $\gamma_\beta a^\top d \geq \beta - a^\top \bar{x}$, i.e., we have $a^\top(\bar{x} + \gamma_\beta d) \geq \beta$. Let $\bar{x}_\beta = \bar{x} + \gamma_\beta d$, then $\bar{x}_\beta \in \mathcal{B}$. Additionally, we have $\bar{x}_\alpha, \bar{x}_\beta \in \mathcal{X}$, since $\bar{x} \in \mathcal{X}$ and $d \in \text{lin}(\mathcal{X})$. Hence, we obtain that $\bar{x}_\alpha, \bar{x}_\beta \in \mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})$. Now, let $\eta = \frac{\gamma_\beta}{\gamma_\beta + \gamma_\alpha}$, then we have

$$\bar{x} = \eta \bar{x}_\alpha + (1 - \eta)\bar{x}_\beta.$$

Therefore, $\bar{x} \in \text{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B}))$, which is a contradiction. This proves that $\text{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{X}$. $\square$

From Theorem 2.2 we obtain the following result for the special case where the lineality space of cylinder $\mathcal{X}$ is one-dimensional.

Remark 2.1. *In Theorem 2.2, assume that $\mathcal{X}$ is a closed convex cylinder such that* $\dim(\text{lin}(\mathcal{X})) = 1$, *i.e.,* $\text{lin}(\mathcal{X}) = \{\alpha d \mid \alpha \in \mathbb{R}\}$. *If $a^\top d \neq 0$, then* $\text{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{X}$.

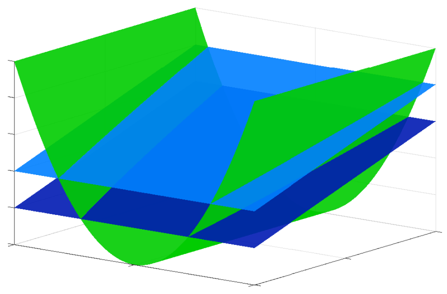
Notice that the results of Theorem 2.2 and Remark 2.1 hold for a general closed convex cylinder $\mathcal{X}$. However, to identify the redundant DCyCs for MISOCO problems, in this section we focus on the special case where $\mathcal{X}$ is a cylinder defined by a quadratic constraint.

We use Theorem 2.2 and Remark 2.1 to characterize the redundant DCyCs for MISOCO problems.

Corollary 2.3 (Redundant DCyC for MISOCO)**.** *Let $\hat{\mathcal{C}}$, as defined in (2.1), be a cylinder, and consider the two half spaces defined in (2.3). If $\hat{a} \not\perp \text{lin}(\hat{\mathcal{C}})$, then the DCyC is equal to $\hat{\mathcal{C}}$.*

Now, consider the special case of Corollary 2.3 where $\text{lin}(\hat{\mathcal{C}})$ is one-dimensional and defined as $\text{lin}(\hat{\mathcal{C}}) = \{\alpha d \mid \alpha \in \mathbb{R}\}$. Then, the condition $\hat{a} \not\in \text{lin}(\hat{\mathcal{C}})$ simplifies to

$\hat{a}^\top d \neq 0$. The redundant DCyC is illustrated in Figure 2.2a, where the DCyC is equal to the cylindrical set. However, in Figure 2.2b, the DCyC is not equal to the original cylinder. In that case we may derive a DCyC that does tighten the original cylinder.


(a) A cylindrical redundant DCyC.

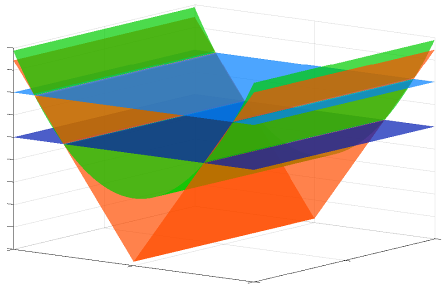

(b) Not a cylindrical redundant DCyC.

Figure 2.2: Illustration of the cylindrical redundant DCyC.

Corollary 2.4. (Identification of a redundant DCyC for MISOCO) *Consider the set $\hat{\mathcal{C}}$, as defined in (2.1), and a disjunction as defined in (2.3). We have a cylindrical redundant DCyC if the following two conditions are satisfied:*

- *System $\begin{bmatrix} P & p \end{bmatrix}^\top d = 0$, for $d \neq 0$, has a solution.*
- *System $\begin{bmatrix} P & p \end{bmatrix} y = \hat{a}$, for $y \in \mathbb{R}^{\ell+1}$, does not have a solution.*

Proof. From $\begin{bmatrix} P & p \end{bmatrix}^\top d = 0$, we have $Pd = 0$ and $p^\top d = 0$. So for all $w \in \hat{\mathcal{C}}$, we have

$$d^\top P d\alpha^2 + 2d^\top (Pw + p)\alpha = 0. \tag{2.5}$$

We know, for all $w \in \hat{\mathcal{C}}$, that $w^\top P w + 2p^\top w + \rho \leq 0$. So from (2.5), we have

$$(w + \alpha d)^\top P(w + \alpha d) + 2p^\top (w + \alpha d) + \rho \leq 0, \quad \forall \alpha \in \mathbb{R}.$$

Hence, $d \in \text{lin}(\hat{\mathcal{C}})$. As $d \neq 0$, we conclude that $\hat{\mathcal{C}}$ is a cylinder.

Let col$(\cdot)$, row$(\cdot)$, and null$(\cdot)$ denote respectively the column space, row space, and null space of a matrix. If system $[P\ p]y = \hat{a}$, for $y \in \mathbb{R}^{\ell+1}$, does not have a solution, then $\hat{a} \notin \text{col}([P\ p])$; thus, $\hat{a} \notin \text{row}([P\ p]^\top)$. Therefore, there exists a $d_0 \neq 0$ such that $d_0 \in \text{null}([P\ p]^\top)$ and $\hat{a}^\top d_0 \neq 0$. Hence, we obtain that $\hat{a} \not\perp \text{lin}(\hat{\mathcal{C}})$. From Corollary 2.3, we can conclude that this is a redundant DCyC. □

Remark 2.2. *The redundancy of a DCyC is independent of the base of the cylinder.*

Notice in Corollary 2.4 that the first condition ensures that set $\hat{\mathcal{C}}$, defined in (2.1), is a cylinder and the second condition ensures that $\hat{C}$ and disjunction (2.3) define a redundant DCyC.

Corollary 2.4 is in fact a sufficient condition to identify the cylindrical pathological disjunction, as defined in Theorem 2.2, for a MISOCO problem. In Corollary 2.5, we provide a necessary and sufficient condition to identify when the normal vector of the disjunctive hyperplanes is orthogonal to the lineality space of the cylinder. The difference between Corollaries 2.4 and 2.5 is that the former considers a condition in the null space of the affine constraints of the MISOCO problem, while the latter one is defined in the original space of the decision variables of the MISOCO problem. The following lemma is needed to prove Corollary 2.5.

Lemma 2.2 (Lineality space of intersection of two convex sets [Bertsekas, 2009])**.** *If $\mathcal{X}_1$ and $\mathcal{X}_2$ are convex sets such that $\mathcal{X}_1 \cap \mathcal{X}_2 \neq \emptyset$, then* $\text{lin}(\mathcal{X}_1 \cap \mathcal{X}_2) = \text{lin}(\mathcal{X}_1) \cap \text{lin}(\mathcal{X}_2)$*.*

Corollary 2.5. (Identification of a redundant DCyC for MISOCO) *Consider the MISOCO problem (1.1) and disjunction (2.2). We may assume w.l.o.g. that we derive the DCyC for $\mathcal{L}_1 \in \mathbb{R}^{n_1}$. Condition $a \not\perp \text{lin}(\mathcal{X})$ holds if and only if*

$$\begin{pmatrix} A \\ a^\top \end{pmatrix} \begin{pmatrix} 0_{n_1} \\ x \end{pmatrix} = \begin{pmatrix} 0_m \\ 1 \end{pmatrix}, \tag{2.6}$$

where $x \in \mathbb{R}^{n-n_1}$.

Proof. Let $\mathcal{K}^{\mathcal{L}_1}$ be the cone $\mathcal{L}_1$ lifted to $\mathbb{R}^n$. So we have $\mathcal{K}^{\mathcal{L}_1} = \{(x^c, x^r) \in \mathbb{R}^{n_1} \times \mathbb{R}^{n-n_1} | x^c \in \mathcal{L}_1\}$. We know that $\text{lin}(\mathcal{L}_1) = \{0\}$, hence, $\text{lin}(\mathcal{K}^{\mathcal{L}_1}) = \{(0_{n_1}, x^r) \mid x^r \in \mathbb{R}^{n-n_1}\}$. Let $\mathcal{C} = \mathcal{K}^{\mathcal{L}_1} \cap \{x \in \mathbb{R}^n \mid Ax = b\}$. We know that $\text{lin}(\{x \in \mathbb{R}^n \mid Ax = b\}) = \text{null}(A)$. So from Lemma 2.2, we have $\text{lin}(\mathcal{C}) = \text{lin}(\mathcal{K}^{\mathcal{L}_1}) \cap \text{null}(A)$, and we can conclude that

$$\text{lin}(\mathcal{C}) = \left\{ \begin{pmatrix} 0_{n_1} \\ x^r \end{pmatrix} \middle| \; x^r \in \mathbb{R}^{n-n_1}, \; A \begin{pmatrix} 0_{n_1} \\ x^r \end{pmatrix} = 0 \right\}. \tag{2.7}$$

From Theorem 2.2, we know that if $a \not\perp \text{lin}(\mathcal{C})$, then we have a cylindrical redundant DCyC. Condition $a \not\perp \text{lin}(\mathcal{C})$ holds if and only if there exists $\bar{x} \in \text{lin}(\mathcal{C})$ such that $a^\top \bar{x} \neq 0$. As $\text{lin}(\mathcal{C})$ is a subspace, w.l.o.g. we can acquire $a^\top \bar{x} = 1$ for $a \not\perp \text{lin}(\mathcal{C})$. Combining this condition with equation (2.7), we can conclude that $a \not\perp \text{lin}(\mathcal{C})$ if and only if system $\begin{pmatrix} A \\ a^\top \end{pmatrix} \begin{pmatrix} 0_{n_1} \\ x^r \end{pmatrix} = \begin{pmatrix} 0_m \\ 1 \end{pmatrix}$ has a solution, which completes the proof.

□

2.4 Discussion

The conic and cylindrical pathological disjunctions, presented in Sections 2.3.1 and 2.3.2 respectively, form the fundamental cases for analyzing the redundant DCCs and DCyCs. In this section, we explore some instances where one can find the redundant cases embedded in more complex configurations. The cases presented in this section are built on the quadratic set $\hat{\mathcal{C}}$, as defined in (2.1), and a disjunction defined in (2.3), such that $\alpha < \beta$, $\mathcal{X} \cap \mathcal{A} \neq \emptyset$, $\mathcal{X} \cap \mathcal{B} \neq \emptyset$.

2.4.1 Conic cylinders

In this section, we define a conic cylinder, and we consider sets resulting from the intersection of a conic cylinder and a disjunctive set. Then, we formalize a special case of redundant DCyCs.

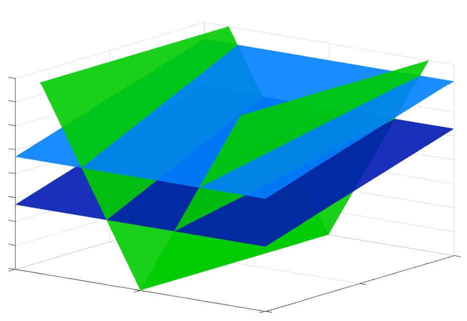
Definition 2.4. *(Conic cylinder) Let $\mathcal{X}$ be a closed convex set, and let $\mathcal{K} = \mathcal{X} \cap \operatorname{lin}(\mathcal{X})^{\perp}$. Set $\mathcal{X}$ is a conic cylinder if $\mathcal{K}$ is a convex cone, and $\operatorname{lin}(\mathcal{X}) \neq \{0\}$.*

The following corollary formalizes a special case where the DCyC is redundant.

Corollary 2.6. *Let $\mathcal{X} \subseteq \mathbb{R}^n$ be a closed convex cylinder such that $\mathcal{X} = \operatorname{lin}(\mathcal{X}) + \mathcal{K}$, and $\mathcal{K} = \mathcal{X} \cap \operatorname{lin}(\mathcal{X})^{\perp}$. Suppose that $\mathcal{K}$ is a convex cone with vertex v. Let $\mathcal{A} = \{x \in \mathbb{R}^n \mid a^\top x \leq \alpha\}$ and $\mathcal{B} = \{x \in \mathbb{R}^n \mid a^\top x \geq \beta\}$ such that $\alpha < \beta$, $\mathcal{X} \cap \mathcal{A} \neq \emptyset$, $\mathcal{X} \cap \mathcal{B} \neq \emptyset$, and $a \perp \operatorname{lin}(\mathcal{X})$. If $v \in \mathcal{A} \cup \mathcal{B}$, then $\operatorname{conv}(\mathcal{X} \cap (\mathcal{A} \cup \mathcal{B})) = \mathcal{X}$.*

Proof. The proof is analogous to that of Theorem 2.1. □

Notice in Corollary 2.6 that if $a \not\perp \operatorname{lin}(\mathcal{X})$, then from Corollary 2.3, we know that we have a redundant DCyC. However, in Corollary 2.6 we assume that $a \perp \operatorname{lin}(\mathcal{X})$, and derive another case of pathological disjunction and redundant DCyC. Figure 2.3 illustrates this result showing two different cases of redundant DCyCs. In Figure 2.3a we have a case where $a \not\perp \operatorname{lin}(\mathcal{X})$, which complies with Corollary 2.3, thus we have a redundant DCyC. In Figure 2.3b we have a case where $a \perp \operatorname{lin}(\mathcal{X})$, so it does not satisfy the conditions of Corollary 2.3. However, it complies with Corollary 2.6, thus we have a redundant DCyC. The classification of Figure 2.3b may be obtained noting that the base of the cylinder is a convex cone and its vertex is in one of the half spaces defining the disjunction. Henceforth, the original cylinder configures a redundant DCyC.


(a) A cylindrical redundant DCyC.

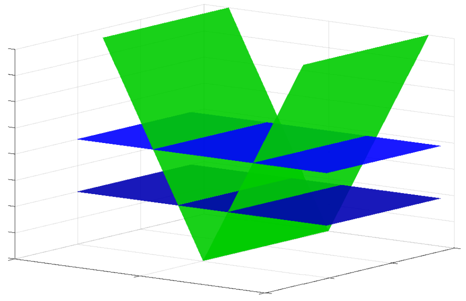

(b) A conic redundant DCC case for the base of the cylinder.

Figure 2.3: Illustration of redundant cases for conic cylinders.

2.4.2 Branching on a higher dimensional subspace

In some cases one may have to make a split disjunction on an integer variable that does not appear in the quadratic set $\mathcal{C} \in \mathbb{R}^n$. This is in fact one common instance of redundant DCyCs, where the cylinder is given by $\{(\xi, x) \in \mathbb{R} \times \mathbb{R}^n \mid (0, x) + (\xi, 0), x \in \mathcal{C}\}$, and we want to make a disjunction on the variable ξ. In this case, the disjunction is pathological and we have a redundant DCyC. This case is illustrated in Figure 2.4, where the quadratic set defines a cylinder with an ellipsoid base defined in the space of (x_1, x_2), and we make a disjunction on variable ξ.

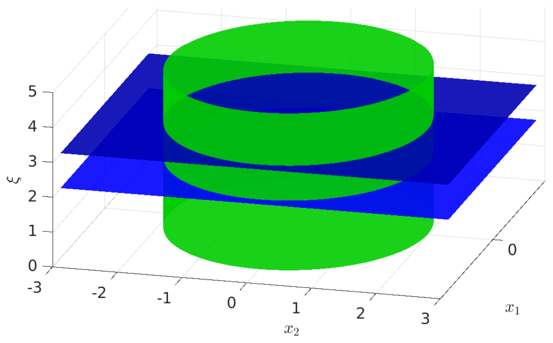


Figure 2.4: A redundant DCyC.

2.4.3 Eliminating pathology by branching

Suppose that deriving the DCC for the set $\hat{\mathcal{C}} \cap (\hat{\mathcal{A}} \cup \hat{\mathcal{B}})$, as defined in (2.1) and (2.3), results in a redundant cut. This situation does not necessarily render the DCC approach useless. In particular, further down the branch and bound tree, effective DCCs may be generated.

Figure 2.5 illustrates this case. Suppose that $\hat{\mathcal{C}}$ is a cone, with a vertex v, and one wants to make a disjunction on the binary variable x_2. Notice that the vertex of the cone is in one of the disjunctive half spaces in Figure 2.5. Then, the DCC will be equal to $\hat{\mathcal{C}}$, which is a redundant DCC. In this case, one can branch on the binary variable x_2 to obtain new quadratic sets in each branch. Consider first the branch $x_2 = 0$, the new quadratic set is obtained from the intersection of $\hat{\mathcal{C}}$ with the hyperplane $x_2 = 0$, which defines a two-dimensional SOC. In this branch, making a disjunction on the binary variable x_1 again leads to a redundant DCC. Now consider the branch $x_2 = 1$; the new quadratic set is obtained from the intersection of $\hat{\mathcal{C}}$ with the hyperplane $x_2 = 1$, which is one branch of a hyperboloid. Considering a disjunction on the binary variable x_1, one can derive a useful DCC in this branch.

The case presented in this section shows how one can identify opportunities down the branch and bound tree for using DCCs to improve the performance of a solver. This is useful to complement existing branching rules [Achterberg et al., 2005] to define new rules capable of exploiting the structure of a MISOCO problem, which calls for further research in this area.

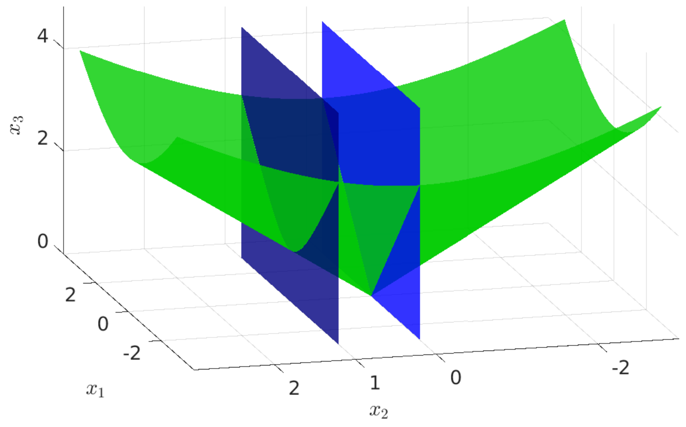


Figure 2.5: An instance of the conic redundant DCC.

2.5 Numerical Experiments

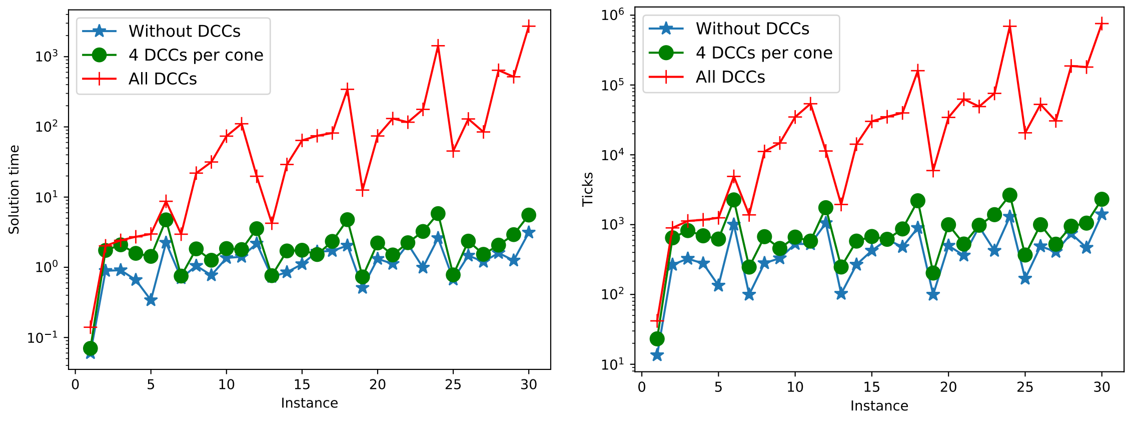
The main problem of failing to recognize the redundant DCCs and DCyCs is that current solvers do not yet have the prepossessing capabilities to recognize those redundancies. As a result, their performance, when solving a MISOCO problem after adding redundant DCCs or DCyCs, may suffer leading to significant increase in solution time. In this section we demonstrate this phenomena by solving two problem sets. First, we consider portfolio optimization problems with higher moment coherent risk (HMCR) measures presented by Vinel and Krokhmal [2014]. Second, we consider two conic formulations for service system design problems with congestion as presented in Góez and Anjos [2018]. We derive DCCs using the method presented by Belotti et al. [2017], and all the DCCs derived satisfy the conditions of Corollaries 2.1 and 2.2, thus all are redundant. We use CPLEX 12.8 with the default parameters to solve all the problems. For measuring the performance we use the wall-clock time and the deterministic time measured in *ticks* provided by CPLEX [IBM Knowledge Center, 2017].

2.5.1 A portfolio optimization problem

We consider the 4th moment coherent risk measure and use the method proposed by Ben-Tal and Nemirovski [2001] to reformulate the 4th-order cone optimization problem [see Vinel and Krokhmal, 2014, Eq. 49] as a second order cone optimization problem. We also consider the round-lot constraints, which represent a real-life policy that assets can be purchased only in *lots* of shares. Thus, the portfolio optimization problem with HMCR measures and round-lot constraints may be formulated as a MISOCO problem. The DCCs are added at the root node for 30 portfolio optimization problems with different number of assets and different number of scenarios. We use three different strategies: first we solve each problem without DCCs, second we derive 4 DCCs for each SOC, and finally we derive all possible DCCs for all the SOCs. In Figures 2.6a and 2.6b, the wall-clock solution time and number of ticks is reported for the three different approaches in solving the 30 portfolio optimization problems. For most of the 30 instances, the solution time and ticks lines for the strategy without adding redundant DCCs are below the line for strategy where 4 redundant DCCs are added for each SOC. Additionally, the solution time and ticks without adding DCCs is significantly less than that of the cases where all possible DCCs are added.

2.5.2 A service system design problem with congestion

For this problem class we consider formulations (MISOCO 1) and (MISOCO 4) and the instances presented in Góez and Anjos [2018]. For each of these instances we have ℓ client locations and m facility locations, where $\ell \in \{10, 20, 30\}$, and $m \in \{50, 100, 150, 200\}$. For formulation (MISOCO 1) the dimension of the SOCs is given by the number of client locations plus two, and the total number of cones is given by the locations available to open. Hence, for these formulations the potential

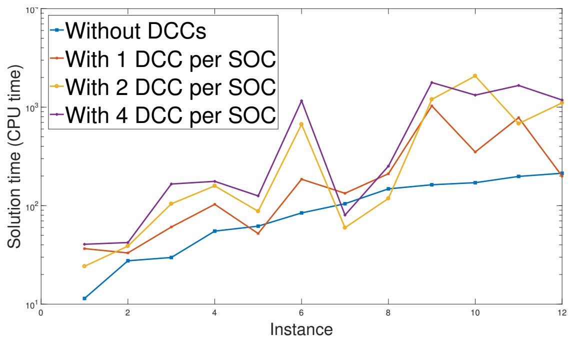


(a) Solution time. (b) Ticks.

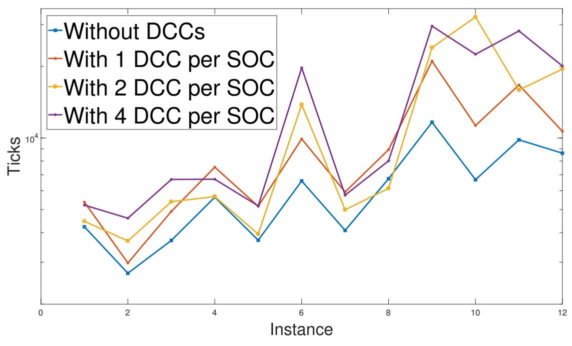
Figure 2.6: Numerical results for portfolio optimization problems.

number of DCCs one could add per cone is ℓ. For formulation (MISOCO 4) the number of SOCs is equal to ℓm, but all the cones are three dimensional, and one can derive only one DCC per cone. In this instance the dimension of the cones plays a role in its difficulty, in particular formulation (MISOCO 1) has higher dimensional cones than formulation (MISOCO 4). We observe that (MISOCO 1) needs more computational time, while the formulations are equivalent. We set a CPU time limit of 3600 seconds. Figures 2.7a and 2.7b show the results of the experiments with (MISOCO 1), where we add up to four DCCs per cone. There we list the results only for the instances that were solved within the time limit, which were 12 in total. Same behavior can be observed as with the portfolio problems in Section 2.5.1, i.e., the larger the number of redundant DCCs is, the larger the solution time. Figures 2.7c and 2.7d show the results of the (MISOCO 4) experiments. Here we manage to solve 139 instances, and for each instance we add one DCC per cone for all the cones, which accounts for all the possible DCCs. We see in Figure 2.7c that the results are more mixed than in the previous two experiments, showing that the

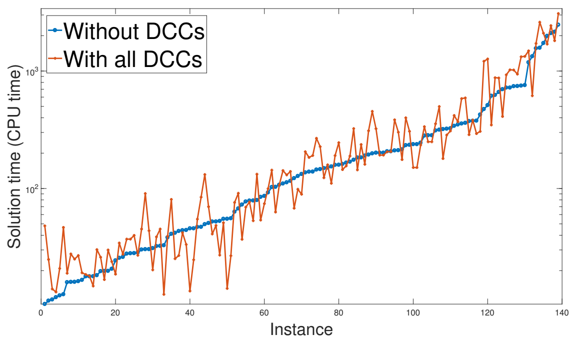
CPU solution time does not always worsen when the DCCs are added. Nonetheless, the solution time increases in 60 percent of the instances when DCCs are added. The negative effect of adding redundant DCCs to (MISOCO 4) formulation is more clear in Figure 2.7d, where we can see that the line showing the deterministic time taken to solve the instances without the DCCs is more consistently below the line for the instances with DCCs. To be more specific, the CPLEX deterministic solution time increased in 87 percent of the instances when DCCs were added.



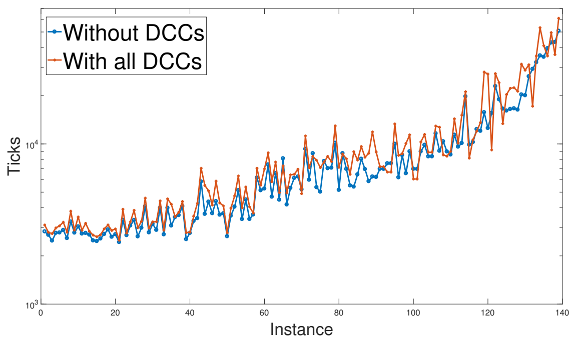
(a) Solution time (MISOCO 1).



(b) Ticks (MISOCO 1).



(c) Solution time (MISOCO 4).



(d) Ticks (MISOCO 4).

Figure 2.7: Numerical results of the portfolio optimization problem.

2.6 Conclusions

In this chapter we presented two fundamental pathological disjunctions, which help to identify redundant DCCs and DCyCs for MISOCO problems. We know that if the DCC is redundant, then any other disjunctive cut based on the same disjunction will be redundant, since the DCC describes the convex hull of the disjunctive set, and thus is the tightest possible disjunctive cut that can be obtained. We have also shown how the two fundamental cases are the building blocks of more complex instances. We illustrated that by analyzing some instances in Section 2.4, and showing how the instances are combinations of the two fundamental cases considered in this study.

Efficient implementation of branch and conic cut (BCC) algorithms for MISOCO requires the identification of pathological disjunctions. In a BCC framework, it is important to keep under control the growth of the problem. For that reason, identifying whether a DCC is redundant before adding it to the formulation is essential to obtain an efficient implementation of this methodology. Otherwise, as was shown in Section 2.5, the solution time can be adversely affected with the addition of redundant DCCs. Thus, this work highlights both limitations of and opportunities for efficient implementation of BCCs.

Chapter 3

Truss Design Problem: Modeling and Analysis

In this chapter we present mathematical optimization models for truss design problems under various assumptions. We start with literature review in Section 3.1. In Section 3.2, we review several mathematical optimization models for the truss design problem assuming that the cross-sectional areas of the bars are continuous and we present some characteristics of the feasible set of the truss design problem. In Section 3.3, we develop mathematical optimization models which provide lower bounds for the optimal objective value of the continuous truss design problem. Then, in Section 3.4, we propose mathematical optimization models for the discrete design problem, where the cross-sectional areas of the bars are discrete. We develop models for truss sizing problems and extend the models to account for multi-scenario design problems. We end the chapter by developing models for truss topology design and sizing optimization problems.

3.1 Introduction

The truss design problem is concerned with the optimal selection of geometry, topology and sizing of a structural system (see, e.g., the review paper by Bendsøe et al. [1994]). In a truss structure, some nodes are fixed at a point, while the others are free. The external force on the nodes results in structure deformation that induces internal forces on the bars. The internal forces in turn balance the external force on the free nodes. We aim to minimize the total weight of the truss. The design variables are the cross-sectional areas while the unknown internal forces, nodal displacements, and the bars' stresses are referred to as state variables, and are uniquely specified if the cross-sectional areas of the bars are determined.

The first use of numerical optimization for the truss design problem is by Dorn et al. [1964]. They used the ground structure approach, in which the optimal structure is a subset of a set of bars defined prior to solving the problem. They considered the single-load minimum weight truss design problem. Achtziger et al. [1992] considered the truss design problem and developed linear and quadratic optimization models using displacement variables only, with the goal to minimize compliance. Bendsøe and Ben-Tal [1993] considered the problem of minimizing the compliance for a given volume of the material in a truss, where the mathematical model is formulated in terms of the nodal displacements and bar cross-sectional areas. They developed a steepest descent algorithm to solve the problem.

Consider a truss design problem with the objective to minimize the total weight of a given structure, and assume that the cross-sectional areas of the bars are continuous decision variables. If we only impose force balance equations, Hooke's law, and bounds on the stress as the constraints, then all the bars are fully stressed in the optimal solution, and the model can be reformulated as a linear optimization problem. However, adding the Euler buckling constraints makes the problem non-convex and thus harder to solve. Achtziger [1999a] proposed an optimization model for the minimum weight truss design problem taking into account yield stress and Euler buckling constraints, but not considering the kinematic compatibility and the stress-strain relation. He then developed a sequential linear programming algorithm which generates feasible solutions for the problem [Achtziger, 1999b].

One of the frequent restrictions in practice, due to manufacturability and economic reasons, is that the cross-sectional areas of the bars cannot take an arbitrary value; instead they only take values from a predefined finite set. The areas of the truss elements are discrete because the bars are manufactured in fixed sizes. These restrictions also appear in other structural design problems, such as the design of shell element structures when dealing with laminated composites. Achtziger and Stolpe [2007a,b,c] considered the minimum compliance truss design problem with bounds on the volume of the truss, where the cross-sectional areas of the bars only take values from a discrete set. They proposed a mixed integer nonlinear optimization model and used a branch-and-bound algorithm to find the global optimum of the problem. They solved continuous relaxations of the problem to obtain lower bounds for the optimal objective value. However, they did not consider the bounds on stress and the Euler buckling constraints in the design problem. Stolpe [2007] considered the minimum compliance problem with constraints on the displacements and total volume of the structure. He proposed mixed integer linear optimization (MILO) and mixed integer quadratic optimization reformulations using the techniques presented by Petersen [1971] and by Glover [1975, 1984]. Rasmussen and Stolpe [2008] used a branch-and-cut approach to solve the MILO formulation of the minimum weight truss design problem, taking into account the stress and displacement constraints. However, they did not consider the buckling constraints in the model. They solved a 2D L-shaped truss problem with 54 bars and 108 binary variables, and a 3D cantilever truss with 40 bars and 160 binary variables to global optimality. Mela [2014] investigated the minimum weight truss design problem, where he assumed that the cross-sectional areas of the bars are discrete. He formulated and solved a MILO model taking into account the Euler buckling and kinematic stability constraints. Mela [2014] solved a 2D truss tower with 209 bars, 110 of which were overlapping members. Additionally, he solved a 2D L-shaped truss with 160 bars, 23 of which were overlapping. Stolpe [2004] suggested a mixed integer non-convex mathematical model for the minimum weight truss design problem with displacements, stress, and cross-sectional areas as variables, considering bounds on stress and cross-sectional areas, as well as Euler buckling constraints. Then, he used a branch-and-bound framework to obtain the global optimum of the truss problem. The largest instances solved to global optimality included 25 bars.

There are various other engineering design problems and the MILO formulations of a truss structure can be generalized to other structures, e.g., Mellaert et al. [2017] develop MILO formulations for frame structures with various engineering constraints.

3.2 Continuous truss design problem

In this section, we assume that the cross-sectional areas of the bars are continuous decision variables. In general, the truss design problem can be formulated as a nonlinear non-convex optimization problem [Stolpe, 2004]. We present three equivalent mathematical models for the continuous truss design problem.

3.2.1 Preliminary models

Let m be the number of bars in the truss, $\mathcal{I} = \{1, \dots, m\}$, and n be the number of degrees of freedom, which is

$$n = (\#\text{ of nodes} - \#\text{ of fixed nodes}) \times \text{dim. of the space.}$$

Let $x \in \mathbb{R}^m$ denote the cross-sectional areas of the bars, and $q \in \mathbb{R}^m$ the vector of the internal forces on the bars. The stress in bar $i \in \mathcal{I}$ is

$$\sigma_i = \begin{cases} \dfrac{q_i}{x_i} & \text{if} \quad x_i > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

The vector of external forces exerted on the nodes is denoted by $f \in \mathbb{R}^n$. The equilibrium between the internal forces and the external forces applied to the free nodes is maintained as a result of the *force balance* equation [De Klerk et al., 1995]

$$Rq = f, \tag{3.2}$$

where $R \in \mathbb{R}^{n \times m}$ is the topology matrix associated with the design problem. The i^{th} column of R, denoted by r_i, is the vector representing the topology of the i^{th} bar in the truss for all $i \in \mathcal{I}$.

Let $u \in \mathbb{R}^n$ denote the displacement vector of the nodes. Additionally, let $l \in \mathbb{R}^m$ and $\xi \in \mathbb{R}^m$ denote the length and elongation of the bars, respectively. The elongation of the bars depends on the displacement vector u as follows:

$$\xi = R^T u. \tag{3.3}$$

We assume that the elongation of the bars compared to the length of the bars is small. Let E_i, for $i \in \mathcal{I}$, denote the Young's modulus of bar i. We can restate the internal force q_i as a function of cross-sectional area x_i and elongation ξ_i. This relationship is governed by Hooke's law. For all $i \in \mathcal{I}$, we have

$$q_i = E_i \frac{\xi_i}{l_i} x_i. \tag{3.4}$$

Let $\lambda_i = E_i/l_i$ for $i = 1, \cdots, m$, and let $\Lambda = \mathrm{diag}(\lambda)$, where $\mathrm{diag}(.)$ is the diagonal matrix of the corresponding vector. Additionally, Let X denote the diagonal matrix of x, i.e., $X = \mathrm{diag}(x)$. We can rewrite equations (3.4) in the following matrix form:

$$q = \Lambda X \xi. \tag{3.5}$$

Let matrix $K_i \in \mathbb{R}^{n \times n}$, for $i \in \mathcal{I}$, be the contribution of bar $i \in \mathcal{I}$ to the global stiffness matrix, defined as

$$K_i = \lambda_i r_i {r_i}^T. \tag{3.6}$$

Obviously K_i is a positive semidefinite matrix. Further, the stiffness matrix of the truss is obtained by assembling the bar stiffness matrices as follows

$$K(x) = \sum_{i=1}^{m} x_i K_i. \tag{3.7}$$

From (3.6) and (3.7), we can rewrite matrix K as

$$K(x) = R \Lambda X R^T. \tag{3.8}$$

We may assume that the truss structure is stable, and hence $K(x)$ is positive definite, denoted as $K(x) \succ 0$. As mentioned below, this property is maintained by positive lower bounds on the bars' cross-sectional areas. From Equations (3.2), (3.3), (3.5), and (3.8), we can derive the following relationship:

$$\begin{aligned} f &= Rq = R\left(\Lambda X \xi\right) = R\Lambda X R^T u \\ &= K(x)u. \end{aligned} \tag{3.9}$$

From Equation (3.9) we can deduce that if $K(x)$ is positive definite for a given cross-sectional area x, then nodal displacement u is uniquely specified and so are ξ, σ, and q. This fact is the reason that cross-sectional area x is referred to as design variable, while u, ξ, σ, and q are called the state variables of the problem.

Lower and upper bounds are enforced for the bar stresses. Let $\sigma^{\min},\ \sigma^{\max} \in \mathbb{R}^m$ be the given lower and upper bounds on the bar stress, respectively. The bounds $\sigma^{\min}$ and $\sigma^{\max}$ are actually bounds on the compression and tension of the bars, respectively. Therefore, we have $\sigma^{\max} > 0$ and $\sigma^{\min} < 0$. Lower and upper bounds $u^{\min}$ and $u^{\max}$ are considered on the nodal displacements of the structure as well. Moreover, we consider lower and upper bounds on the cross-sectional areas of the bars, namely, $x^{\min},\ x^{\max} \in \mathbb{R}^m$. We consider the truss sizing problem, where $x_i^{\min} > 0$ for all $i \in \mathcal{I}$.

Furthermore, we consider the Euler buckling constraints, which are defined as

$$\sigma_i \geq \sigma_i^E, \qquad i \in \mathcal{I},$$

where σ_i^E is the Euler buckling stress for bar $i \in \mathcal{I}$. We assume that both ends of all the bars are pinned. Then, the Euler buckling stress for bar $i \in \mathcal{I}$ with a circular cross section is

$$\sigma_i^E = -\frac{\pi^2 E_i}{4\left(\frac{l_i}{\tau_i}\right)^2}, \tag{3.10}$$

where τ_i is the radius of bar $i \in \mathcal{I}$. If we define $\gamma_i = \pi E_i / 4 l_i^2$, then $\sigma_i^E = -\gamma_i x_i$, and the Euler buckling constraints can be written as

$$\sigma_i + \gamma_i x_i \geq 0, \qquad i \in \mathcal{I}. \tag{3.11}$$

Using Equation (3.1), we can also write the Euler buckling constraints as

$$q_i + \gamma_i x_i^2 \geq 0, \qquad i \in \mathcal{I}. \tag{3.12}$$

Constraint (3.11) can be generalized to the cases where the discrete choice of the bar sizes corresponds to similar cross-sectional shapes.

For example, suppose we have a discrete choice set composed of similar rectangles: $h_i/b_i = \alpha$, for $i \in \mathcal{I}$, where b_i and h_i denote the width and height for the ith choice, respectively, and α is a constant. Then the Euler buckling constraints are written as

$$\sigma_i + \gamma_i' x_i \geq 0, \quad \gamma_i' = \frac{\alpha \pi^2 E}{12 l_i^2}, \qquad i \in \mathcal{I}. \tag{3.13}$$

Notice that neither Hooke's law nor the Euler buckling constraints (3.12) are convex. This results in the following non-convex quadratic optimization formulation of the truss sizing problem:

$$\begin{array}{lllll}
\min & \rho l^T x & & & \\
\text{s.t.} & Rq & = f, & & \\
& q_i - \frac{E_i}{l_i}(r_i{}^T u) x_i & = 0, & i \in \mathcal{I}, & \\
& q_i + \gamma_i x_i^2 & \geq 0, & i \in \mathcal{I}, & \qquad (\mathrm{P}_1) \\
& u^{\min} \leq u & \leq u^{\max}, & & \\
& \sigma_i^{\min} x_i \leq q_i & \leq \sigma_i^{\max} x_i, & i \in \mathcal{I}, & \\
& x_i^{\min} \leq x_i & \leq x_i^{\max}, & i \in \mathcal{I}, &
\end{array}$$

where ρ is the density of the bar material. We may assume without loss of generality that all bars have the same density.

Model (P$_1$) has m non-convex equalities, and m non-convex inequalities. Each of the m non-convex equalities, has n bilinear terms. However, the Hooke's law constraint—the second constraint in (P$_1$)—can be used to derive a new formulation in terms of the cross-sectional areas $x \in \mathbb{R}^m$ and the nodal displacements $u \in \mathbb{R}^n$ as

$$\begin{aligned}
\min \quad & \rho l^T x \\
\text{s.t.} \quad & K(x)u = f, \\
& \frac{E_i}{l_i}(r_i{}^T u) = \sigma_i, && i \in \mathcal{I}, \\
& \sigma_i + \gamma_i x_i \geq 0, && i \in \mathcal{I}, \\
& u^{\min} \leq u \leq u^{\max}, \\
& \sigma_i^{\min} \leq \sigma_i \leq \sigma_i^{\max}, && i \in \mathcal{I}, \\
& x_i^{\min} \leq x_i \leq x_i^{\max}, && i \in \mathcal{I},
\end{aligned} \tag{P$_2$}$$

where all the nonlinearity of the problem is encapsulated in n non-convex equalities, that is the $K(x)u = f$ constraints, which include mn bilinear terms. However, we can decrease the number of bilinear terms by adding the internal forces $q \in \mathbb{R}^m$ and

Equation (3.1) back to the model. By doing so, problem (P$_1$) can be reformulated as

$$
\begin{aligned}
\min \quad & \rho l^T x && && \\
\text{s.t.} \quad & Rq && = f, && \\
& \sigma_i - \frac{E_i}{l_i} {r_i}^T u && = 0, && i \in \mathcal{I}, \\
& q_i - \sigma_i x_i && = 0, && i \in \mathcal{I}, \\
& \sigma_i + \gamma_i x_i && \geq 0, && i \in \mathcal{I}, \\
& u^{\min} \leq \quad u && \leq u^{\max}, && \\
& \sigma_i^{\min} \leq \quad \sigma_i && \leq \sigma_i^{\max} && i \in \mathcal{I}, \\
& x_i^{\min} \leq \quad x_i && \leq x_i^{\max} && i \in \mathcal{I}.
\end{aligned}
\tag{P$_3$}
$$

This model has m non-convex equalities, each of which has only one bilinear term. Consequently, model (P$_3$) is simpler than models (P$_1$) and (P$_2$). We utilize model (P$_3$) in Section 3.4.1 to derive mathematical optimization models for truss sizing problems when the cross-sectional areas are discrete.

3.2.2 Multi-scenario truss design problem

In Section 3.2.1, it was assumed that we have one external force scenario. In this section we extend model (P$_3$) to multi-scenario truss sizing problems. Let $\mathcal{K}$ be the set of the external force scenarios, and let ℓ be the number of the external force scenarios. For each scenario we have an external force and the nodal displacements and the stress on the bars are calculated based on the cross-sectional areas of the bars. Let f^k, u^k, ξ^k, q^k, and σ^k, for $k \in \mathcal{K}$, be the external force, nodal displacement, bar elongation, internal force, and stress on the bars in scenario k, respectively. It is worth emphasizing that the cross-sectional areas of the bars are the design variables for the structure, and thus are common for all the scenarios. In fact, we aim to minimize the weight of the structure which withholds all the considered external force scenarios. Model (P$_3$) can be extended to account for multi-scenario external forces as follows:

$$\begin{array}{lllr}
\min & \rho l^T x & & \\
\text{s.t.} & Rq^k & = f^k, & k \in \mathcal{K}, \\
& \sigma_i^k - \dfrac{E_i}{l_i} {r_i}^T u^k & = 0, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& q_i^k - \sigma_i^k x_i & = 0, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& \sigma_i^k + \gamma_i x_i & \geq 0, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& u^{\min} \leq \quad u^k & \leq u^{\max}, & k \in \mathcal{K}, \\
& \sigma_i^{\min} \leq \quad \sigma_i^k & \leq \sigma_i^{\max}, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& x_i^{\min} \leq \quad x_i & \leq x_i^{\max}, & i \in \mathcal{I}.
\end{array} \tag{3.14}$$

Models (P$_1$), (P$_2$) can similarly be extended to the multi-scenario design problem.

3.2.3 Some characteristics of the feasible set of the truss design problem

In this section, we explore two characteristics of the feasible set of a multi-scenario truss design problem. In Section 3.2.3.1, we present a theorem on the feasibility of a ray which starts from a feasible solution, and in Section 3.2.3.2, we present a theorem on adding an external force scenario which is a convex combination of the current external force scenarios.

Let $\mathcal{F}$ denote the set of feasible solutions of problem (3.14), and let $\mathcal{X}$ denote the m-dimensional subspace of the cross-sectional areas of the bars. Additionally, let $\text{proj}_{\mathcal{X}}(\mathcal{F})$ denote the orthogonal projection of the feasible set $\mathcal{F}$ on $\mathcal{X}$, the subspace of the cross-sectional areas.

3.2.3.1 Feasibility along rays

In this section, we show that if cross-sectional area $x \in \mathbb{R}^m$ is feasible for problem (3.14), then ray αx, for $\alpha > 1$, to the boundary of the feasible set will be feasible.

Theorem 3.1. *Suppose $x \in \text{proj}_{\mathcal{X}}(\mathcal{F})$. If $\alpha > 1$ and $\alpha x \leq x^{\max}$, then $\alpha x \in \text{proj}_{\mathcal{X}}(\mathcal{F})$.*

Proof. As $x \in \text{proj}_{\mathcal{X}}(\mathcal{F})$, there exists (u, q, σ) such that $(x, u, q, \sigma) \in \mathcal{F}$. Let $\bar{x} = \alpha x$, and let $(\bar{u}, \bar{\sigma}, \bar{q})$ be defined as follows:

$$\begin{aligned}
\bar{u}^k &= \frac{1}{\alpha}\, u^k, \quad \forall k = 1, \cdots, \ell, \\
\bar{\sigma}^k &= \frac{1}{\alpha}\, \sigma^k, \quad \forall k = 1, \cdots, \ell, \\
\bar{q}^k &= q^k, \qquad \forall k = 1, \cdots, \ell.
\end{aligned} \tag{3.15}$$

From (3.7), we know that $K(\bar{x}) = \alpha K(x)$. So we have

$$\begin{aligned}
\bar{u}^k &= \frac{1}{\alpha}\, u^k = \frac{1}{\alpha} K(x)^{-1} f^k \\
&= \frac{1}{\alpha} \left(\frac{1}{\alpha} K(\bar{x}) \right)^{-1} f^k \\
&= K(\bar{x})^{-1} f^k.
\end{aligned} \tag{3.16}$$

We know that $\bar{x} \leq x^{\max}$. Similar to (3.16), we can show that $(\bar{x}, \bar{u}, \bar{\sigma}, \bar{q})$ satisfies all the constraints of problem (3.14). Thus, we can conclude that $\bar{x} \in \text{proj}_{\mathcal{X}}(\mathcal{F})$. □

We use the characteristic, highlighted by Theorem 3.1, in developing a solution methodology for discrete truss sizing problems in Chapter 4.

3.2.3.2 Convex hull of the external force scenarios

Convex combination of external forces is studied in the shakedown analysis and optimal shakedown design of elsto-plastic trusses under multi-parameter static loading, see, e.g., Giambanco and Palizzolo [1995], Kaliszky and Lógó [2002], and Atkočiūnas et al. [2008]. However, to the best of our knowledge, this concept is not studied in the optimal design of the trusses that are limited to have elastic behavior while considering force balance equations, Hooke's law, Euler buckling constraints, yield stress, and displacement bounds.

In this section, we show that a truss structure that copes with a set of external force scenarios, also copes with any convex combination of the external force scenarios. To do so, we prove that if we have a multi-scenario truss design problem and add a scenario which is a convex combination of the current external forces, then the feasible set of the problem does not change.

Theorem 3.2. *Suppose we have the multi-scenario truss design problem* (3.14) *with feasible set $\mathcal{F}$. Let $\tilde{f}$ be a convex combination of the current external force scenarios, and let $\tilde{\mathcal{F}}$ be the feasible set of the problem with the external force $\tilde{f}$ added as a new scenario. The following holds*

$$\operatorname{proj}_{\mathcal{X}}(\tilde{\mathcal{F}}) = \operatorname{proj}_{\mathcal{X}}(\mathcal{F}).$$

Proof. If $\mathcal{F} = \emptyset$, then $\tilde{\mathcal{F}} = \emptyset$, and the theorem holds. Suppose $\tilde{\mathcal{F}} \neq \emptyset$ and let $\mathcal{F}_{\mathcal{X}}$ and $\tilde{\mathcal{F}}_{\mathcal{X}}$ denote the orthogonal projection of $\mathcal{F}$ and $\tilde{\mathcal{F}}$ on the subspace of the cross-sectional areas $\mathcal{X}$, respectively, i.e., $\mathcal{F}_{\mathcal{X}} = \operatorname{proj}_{\mathcal{X}}(\mathcal{F})$ and $\tilde{\mathcal{F}}_{\mathcal{X}} = \operatorname{proj}_{\mathcal{X}}(\tilde{\mathcal{F}})$. Suppose we have $\tilde{\mathcal{K}} = \mathcal{K} \cup \{\tilde{f}\}$, where the external force $\tilde{f}$ is a convex combination of the current external forces in the set S, defined as

$$\tilde{f} = \sum_{k\in\mathcal{K}} \lambda_k f^k, \quad \sum_{k\in\mathcal{K}} \lambda_k = 1, \quad \lambda_k \geq 0. \tag{3.17}$$

We know that $\tilde{\mathcal{F}}_{\mathcal{X}} \subseteq \mathcal{F}_{\mathcal{X}}$, since $\tilde{\mathcal{K}} \supset \mathcal{K}$. So we need only to prove that $\mathcal{F}_{\mathcal{X}} \subseteq \tilde{\mathcal{F}}_{\mathcal{X}}$. Let $\bar{x} \in \mathcal{F}_{\mathcal{X}}$ be given. We need to prove that $\bar{x} \in \tilde{\mathcal{F}}_{\mathcal{X}}$. As $\bar{x} \in \mathcal{F}_{\mathcal{X}}$, there exists $(\bar{u}, \bar{q}, \bar{\sigma})$ such that $(\bar{x}, \bar{u}, \bar{q}, \bar{\sigma}) \in \mathcal{F}$, where

$$\begin{aligned} \bar{q} &= (\bar{q}^1, \ldots, \bar{q}^\ell), \quad \bar{q}^k \in \mathbb{R}^m, \quad k = 1, \ldots, \ell, \\ \bar{u} &= (\bar{u}^1, \ldots, \bar{u}^\ell), \quad \bar{u}^k \in \mathbb{R}^n, \quad k = 1, \ldots, \ell, \\ \bar{\sigma} &= (\bar{\sigma}^1, \ldots, \bar{\sigma}^\ell), \quad \bar{\sigma}^k \in \mathbb{R}^m, \quad k = 1, \ldots, \ell. \end{aligned} \tag{3.18}$$

Let $(\tilde{u}, \tilde{q}, \tilde{\sigma})$ be defined as

$$\begin{aligned} \tilde{q} &= \sum_{k\in\mathcal{K}} \lambda_k q^k, \\ \tilde{u} &= \sum_{k\in\mathcal{K}} \lambda_k u^k, \\ \tilde{\sigma} &= \sum_{k\in\mathcal{K}} \lambda_k \sigma^k, \end{aligned} \tag{3.19}$$

where $\lambda = (\lambda_1, \ldots, \lambda_\ell)$ is the same coefficient vector used to generate the external force $\tilde{f}$ in (3.17). Then we have

$$\begin{aligned} R\tilde{q} &= R\left(\sum_{i\in S} \lambda_k q^k\right) \\ &= \sum_{k\in\mathcal{K}} \lambda_k \left(Rq^k\right) \\ &= \sum_{k\in\mathcal{K}} \lambda_k f^k \\ &= \tilde{f}. \end{aligned} \tag{3.20}$$

Equation (3.20) holds because the force balance equations are linear constraints of the decision variables, and thus are convex. In fact, when x is given, all the constraints of problem (3.14) become linear and so convex. As $\lambda \geq 0$, we can similarly prove that all the other constraints of (3.14) hold. Thus, we can conclude that $(\bar{x}, \tilde{u}, \tilde{q}, \tilde{\sigma})$, as defined in (3.19), satisfies all the constraints of problem (3.14) for the external force scenario (3.17).

□

Therefore, if we generate a new external force, which is a convex combination of the current external forces, its addition to the problem does not change the feasible set. Thus, we can conclude that it will not change the optimal solution. We can use this result to reduce the number of scenarios in a multi-scenario truss design problem by eliminating the scenarios that are in the convex hull of the rest of the scenarios.

3.3 Lower bound of the continuous models

In this section, we present two methodologies which provide a lower bound for the optimal objective value of the continuous truss design problem. One is the well-known McCormick relaxation, and the other is a relaxation introduced by Bienstock and Munoz [2014].

3.3.1 McCormick relaxation

McCormick envelopes [McCormick, 1976] are used in nonlinear and mixed integer nonlinear optimization problems with polynomial terms. McCormick envelopes provide a convex relaxation of non-convex problems, and thus, alleviate the difficulty caused by the non-convexity at the cost of adding new constraints and variables. For a minimization problem, the optimal objective value to the relaxation provides a lower bound for the optimal value of the original problem.

In this section, we derive the standard McCormick relaxation for the non-convex model (P_3). The feasible region of this model can be relaxed by using McCormick envelopes. Specifically, we derive McCormick envelope for the sets $\mathcal{T}_i$, for $i \in \mathcal{I}$, defined as:

$$\mathcal{T}_i := \{(q_i, x_i, \sigma_i) \mid q_i = x_i\sigma_i, \quad x_i^{\min} \leq x_i \leq x_i^{\max},\ \sigma_i^{\min} \leq \sigma_i \leq \sigma_i^{\max}\} \tag{3.21}$$

The following inequalities form the McCormick envelope for the set $\mathcal{T}_i$:

$$\begin{aligned}
&(x_i - x_i^{\min})(\sigma_i - \sigma_i^{\min}) \geq 0,\\
&(x_i^{\max} - x_i)(\sigma_i^{\max} - \sigma_i) \geq 0,\\
&(x_i - x_i^{\min})(\sigma_i^{\max} - \sigma_i) \geq 0,\\
&(x_i^{\max} - x_i)(\sigma_i - \sigma_i^{\min}) \geq 0.
\end{aligned} \tag{3.22}$$

Constraints (3.22) can be rewritten as

$$\begin{aligned}
&q_i \geq x_i^{\min}\sigma_i + x_i\sigma_i^{\min} - x_i^{\min}\sigma_i^{\min},\\
&q_i \geq x_i^{\max}\sigma_i + x_i\sigma_i^{\max} - x_i^{\max}\sigma_i^{\max},\\
&q_i \leq x_i^{\max}\sigma_i + x_i\sigma_i^{\min} - x_i^{\max}\sigma_i^{\min},\\
&q_i \leq x_i\sigma_i^{\max} + x_i^{\min}\sigma_i - x_i^{\min}\sigma_i^{\max}.
\end{aligned} \tag{3.23}$$

Quality of the McCormick relaxation highly depends on the bounds of the variables x and σ. Over the box $[x_i^{\min}, x_i^{\max}] \times [\sigma_i^{\min}, \sigma_i^{\max}]$, the first two inequalities of (3.23) define the lower approximation as a convex envelop of $q_i = x_i\sigma_i$, while the second two inequalities of (3.23) define its upper approximation as a concave envelop. Al-Khayyal and Falk [1983] prove that inequalities (3.23) define the convex hull of the set $\mathcal{T}_i$, as defined in (3.21).

The standard McCormick envelopes provide the following convex relaxation of model (P$_3$):

$$
\begin{aligned}
\min \quad & \rho l^T x \\
\text{s.t.} \quad & Rq = f, \\
& \sigma_i = \frac{E_i}{l_i} {r_i}^T u, && i \in \mathcal{I}, \\
& \sigma_i + \gamma_i x_i \geq 0, && i \in \mathcal{I}, \\
& q_i \geq x_i^{\min} \sigma_i + x_i \sigma_i^{\min} - x_i^{\min} \sigma_i^{\min}, && i \in \mathcal{I}, \\
& q_i \geq x_i^{\max} \sigma_i + x_i \sigma_i^{\max} - x_i^{\max} \sigma_i^{\max}, && i \in \mathcal{I}, \\
& q_i \leq x_i^{\max} \sigma_i + x_i \sigma_i^{\min} - x_i^{\max} \sigma_i^{\min}, && i \in \mathcal{I}, \\
& q_i \leq x_i \sigma_i^{\max} + x_i^{\min} \sigma_i - x_i^{\min} \sigma_i^{\max}, && i \in \mathcal{I}, \\
& \sigma_i^{\min} \leq \ \sigma_i \leq \sigma_i^{\max}, && i \in \mathcal{I}, \\
& x_i^{\min} \leq \ x_i \leq x_i^{\max}, && i \in \mathcal{I}, \\
& u^{\min} \leq \ u \ \leq u^{\max}.
\end{aligned}
\tag{3.24}
$$

As problem (3.24) is a relaxation of (P$_3$), its optimal objective value provides a lower bound for the optimal value of problem (P$_3$). Note that McCormick relaxations can similarly be derived for models (P$_1$) and (P$_2$).

It is worth mentioning that an arbitrarily tight MILO relaxation of the non-convex model (P$_3$) can be obtained by using piecewise McCormick relaxation, where the domain of one of the continuous variables of the bilinear terms is partitioned into disjoint regions and each region is identified by a binary variable (see, e.g., Bergamini et al. [2005], Gounaris et al. [2009], Karuppiah and Grossmann [2006], Tawarmalani and Sahinidis [2004]). The finer the partition is the tighter the relaxation becomes.

3.3.2 Relaxation based on binary expansion

In this section we propose an approximate MILO formulation of the problem (P_3), presented in page 42. We use the method by Bienstock and Munoz [2014], where binary expansion of a continuous variable is utilized to approximate bilinear terms of continuous variables. We have the following non-convex constraints in problem (P_3).

$$q_i - x_i\sigma_i = 0 \qquad i \in \mathcal{I} \tag{3.25}$$

To provide a linear approximation for the multiplication of the two continuous variables, we first define the following linear transformations

$$\begin{aligned} \bar{\sigma}_i &= \frac{\sigma_i - \sigma_i^{\min}}{\sigma_i^{\max} - \sigma_i^{\min}}, \\ \bar{x}_i &= \frac{x_i}{x_i^{\max} - x_i^{\min}}. \end{aligned} \tag{3.26}$$

From Equations (3.26), we have $0 \leq \bar{\sigma}_i \leq 1$ and $0 \leq \bar{x}_i \leq 1$. Let $\Delta\sigma_i = \sigma_i^{\max} - \sigma_i^{\min}$ and $\Delta x_i = x_i^{\max} - x_i^{\min}$. Then we have

$$\begin{aligned} \sigma_i &= \Delta\sigma_i\bar{\sigma}_i + \sigma_i^{\min}, \\ x_i &= \Delta x_i\bar{x}_i + x_i^{\min}. \end{aligned} \tag{3.27}$$

From Equations (3.27), we can rewrite constraints (3.25) as

$$q_i - \left(\Delta\sigma_i\Delta x_i\bar{\sigma}_i\bar{x}_i + \Delta\sigma_i x_i^{\min}\bar{\sigma}_i + \Delta x_i\sigma_i^{\min}\bar{x}_i + \sigma_i^{\min}x_i^{\min}\right) = 0 \qquad i \in \mathcal{I} \tag{3.28}$$

We need to linearize the bilinear term $\bar{\sigma}_i\bar{x}_i$ in constraints (3.28). We can write $\bar{x}_i$ as

$$\bar{x}_i = \sum_{j=1}^{\tau} 2^{-j}y_{ij} + \eta_i, \tag{3.29}$$

where $y_{ij} \in \{0,1\}$, η_i is the error of the binary approximation (3.29), $0 \leq \eta_i \leq 2^{-\tau}$, and $\tau \in \mathbb{N}$. So we have

$$\sum_{j=1}^{\tau} 2^{-j} y_{ij} \quad \leq \quad \bar{x}_i \quad \leq \quad \sum_{j=1}^{\tau} 2^{-j} y_{ij} + 2^{-\tau}. \tag{3.30}$$

From inequalities (3.30), we have

$$\sum_{j=1}^{\tau} 2^{-j} y_{ij} \bar{\sigma}_i \quad \leq \quad \bar{x}_i \bar{\sigma}_i \quad \leq \quad \sum_{j=1}^{\tau} 2^{-j} y_{ij} \bar{\sigma}_i + 2^{-\tau} \bar{\sigma}_i. \tag{3.31}$$

Let $w_{ij} = y_{ij}\bar{\sigma}_i$, which can be enforced by the following inequalities

$$\begin{aligned} w_{ij} &\leq \min\left(\bar{\sigma}_i, y_{ij}\right), \\ w_{ij} &\geq \max\left(\bar{\sigma}_i + y_{ij} - 1, 0\right). \end{aligned} \tag{3.32}$$

From inequalities (3.32), we can conclude that if $y_{ij} = 0$, then $w_{ij} = 0$, and if $y_{ij} = 1$, then $w_{ij} = \bar{\sigma}_i$. Now Let $\xi_i = \bar{x}_i \bar{\sigma}_i$, for $i \in \mathcal{I}$, and let $\mathcal{T} = \{1, \cdots, \tau\}$. Then the following set of constraints provides an approximation for the constraints (3.28)

$$\begin{array}{ll} \sigma_i = \Delta\sigma_i \bar{\sigma}_i + \sigma_i^{\min}, & i \in \mathcal{I}, \\ x_i = \Delta x_i \bar{x}_i + x_i^{\min}, & i \in \mathcal{I}, \\ q_i - \left(\Delta\sigma_i \Delta x_i \xi_i + \Delta\sigma_i x_i^{\min} \bar{\sigma}_i + \Delta x_i \sigma_i^{\min} \bar{x}_i + \sigma_i^{\min} x_i^{\min}\right) = 0, & i \in \mathcal{I}, \\ \sum_{j=1}^{\tau} 2^{-j} y_{ij} \quad \leq \quad \bar{x}_i \quad \leq \quad \sum_{j=1}^{\tau} 2^{-j} y_{ij} + 2^{-\tau}, & i \in \mathcal{I}, \\ \sum_{j=1}^{\tau} 2^{-j} w_{ij} \quad \leq \quad \xi_i \quad \leq \quad \sum_{j=1}^{\tau} 2^{-j} w_{ij} + 2^{-\tau} \bar{\sigma}_i, & i \in \mathcal{I}, \\ w_{ij} \leq \min\left(\bar{\sigma}_i, y_{ij}\right), & i \in \mathcal{I},\ j \in \mathcal{T}, \\ w_{ij} \geq \max\left(\bar{\sigma}_i + y_{ij} - 1, 0\right), & i \in \mathcal{I},\ j \in \mathcal{T}, \\ y_{ij} \in \{0,1\} & i \in \mathcal{I},\ j \in \mathcal{T}. \end{array} \tag{3.33}$$

We can replace constraints (3.25) with the set of constraints (3.33) in model (P_3) as follows

$$
\begin{array}{lll}
\min & \rho l^T x & \\
\text{s.t.} & Rq = f, & \\
& \sigma_i - \dfrac{E_i}{l_i} {r_i}^T u = 0, & i \in \mathcal{I}, \\
& \sigma_i + \gamma_i x_i \geq 0, & i \in \mathcal{I}, \\
& \sigma_i = \Delta\sigma_i \bar{\sigma}_i + \sigma_i^{\min}, & i \in \mathcal{I}, \\
& x_i = \Delta x_i \bar{x}_i + x_i^{\min}, & i \in \mathcal{I}, \\
& q_i - (\Delta\sigma_i \Delta x_i \xi_i + \Delta\sigma_i x_i^{\min} \bar{\sigma}_i + \Delta x_i \sigma_i^{\min} \bar{x}_i + \sigma_i^{\min} x_i^{\min}) = 0, & i \in \mathcal{I}, \\
& \sum_{j=1}^{\tau} 2^{-j} y_{ij} \quad \leq \quad \bar{x}_i \quad \leq \quad \sum_{j=1}^{\tau} 2^{-j} y_{ij} + 2^{-\tau}, & i \in \mathcal{I}, \\
& \sum_{j=1}^{\tau} 2^{-j} w_{ij} \quad \leq \quad \xi_i \quad \leq \quad \sum_{j=1}^{\tau} 2^{-j} w_{ij} + 2^{-\tau} \bar{\sigma}_i, & i \in \mathcal{I}, \\
& w_{ij} \leq \min\left(\bar{\sigma}_i, y_{ij}\right), & i \in \mathcal{I},\ j \in \mathcal{T}, \\
& w_{ij} \geq \max\left(\bar{\sigma}_i + y_{ij} - 1, 0\right), & i \in \mathcal{I},\ j \in \mathcal{T}, \\
& u^{\min} \leq \ u \ \leq u^{\max}, & \\
& \sigma_i^{\min} \leq \ \sigma_i \leq \sigma_i^{\max} & i \in \mathcal{I}, \\
& x_i^{\min} \leq \ x_i \leq x_i^{\max} & i \in \mathcal{I}, \\
& y_{ij} \in \{0, 1\} & i \in \mathcal{I},\ j \in \mathcal{T}.
\end{array}
\tag{3.34}
$$

Model (3.34) is in fact a relaxation of model (P_3) presented in page 42; thus, its optimal objective value provides a lower bound for the optimal objective value of model (P_3).

3.4 Discrete truss design problem

In problems (P_1), (P_2), and (P_3), presented in pages 40 and 42, we assume that the cross-sectional areas are continuous decision variables. In reality, however, the cross-sectional areas are frequently chosen from a discrete set, corresponding to standard pre-manufactured bars [Achtziger and Stolpe, 2006, 2007a, Cerveira et al., 2009]. Thus, x_i for $i \in \mathcal{I}$ takes values from the finite set

$$\mathcal{S}_i = \{s_{i1}, s_{i2}, \ldots, s_{ip_i}\}, \tag{3.35}$$

where $0 < s_{i1} < s_{i2} < \ldots < s_{ip_i}$. Let $\mathcal{J}_i = \{1, \ldots, p_i\}$ denote the set of indices of the discrete values in the set $\mathcal{S}_i$.

3.4.1 Sizing optimization

We propose two discrete modeling approaches for the discrete set (3.35), which are referred to as the *basic discrete model* and the *incremental discrete model*.

3.4.1.1 Basic discrete model

The cross-sectional areas of the bars in the basic discrete model are formulated as choice constraints as follows:

$$\begin{aligned}
&x_i = \sum_{j=1}^{p_i} s_{ij} z_{ij}, && i \in \mathcal{I}, \\
&\sum_{j=1}^{p_i} z_{ij} = 1, && i \in \mathcal{I},\ j \in \mathcal{J}_i, \\
&z_{ij} \in \{0,1\}, && i \in \mathcal{I},\ j \in \mathcal{J}_i.
\end{aligned} \tag{3.36}$$

In the basic model, binary variables represent the choice from the discrete set of the cross-sectional areas. If $z_{ij} = 1$ in the basic discrete model, then $x_i = s_{ij}$, and

$z_{i\bar{j}} = 0$ for $\bar{j} \neq j$.

Next, we explain the derivation of the MILO formulation for problem (P$_3$), presented in page 42, considering the choice constraints (3.36). This idea can be used to reformulate problems (P$_1$) and (P$_2$) analogously. We start by substituting the choice constraints (3.36) in the constraint $q_i - \sigma_i x_i = 0$ for all $i \in \mathcal{I}$. We have

$$q_i - \sigma_i \left(\sum_{j=1}^{p_i} s_{ij} z_{ij} \right) = 0, \ i \in \mathcal{I}, \tag{3.37}$$

where, for all $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$, we have the multiplication of a binary variable and a bounded continuous variable, i.e., $\sigma_i z_{ij}$. Using the idea introduced by Petersen [1971], see also Glover [1975, 1984], we can linearize constraints (3.37) by introducing auxiliary variables $\psi_{ij} = \sigma_i z_{ij}$, for $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$, and adding the following constraints:

$$\begin{aligned} z_{ij}\sigma_i^{\min} &\leq \psi_{ij} \leq z_{ij}\sigma_i^{\max}, \\ \sigma_i - \sigma_i^{\max}(1 - z_{ij}) &\leq \psi_{ij} \leq \sigma_i - \sigma_i^{\min}(1 - z_{ij}). \end{aligned} \tag{3.38}$$

Then, using constraints (3.36) and (3.38), the discrete version of problem (P$_3$), presented in page 42, can be reformulated to obtain the following MILO problem:

$$\begin{array}{lllll}
\min & \sum_{i\in\mathcal{I}} \rho l_i x_i & & & \\
\text{s.t.} & Rq & & = f, & \\
& x_i - \sum_{j=1}^{p_i} s_{ij} z_{ij} & & = 0, & i \in \mathcal{I} \\
& \sigma_i - \frac{E_i}{l_i} {r_i}^T u & & = 0, & i \in \mathcal{I}, \\
& q_i - \sum_{j=1}^{p_i} s_{ij} \psi_{ij} & & = 0, & i \in \mathcal{I}, \\
& \sigma_i + \gamma_i x_i & & \geq 0, & i \in \mathcal{I}, \\
& \sigma_i^{\min} & \leq \sigma_i & \leq \sigma_i^{\max}, & i \in \mathcal{I}, \\
& u^{\min} & \leq u & \leq u^{\max}, & \\
& z_{ij}\sigma_i^{\min} & \leq \psi_{ij} & \leq z_{ij}\sigma_i^{\max}, & i \in \mathcal{I},\ j \in \mathcal{J}_i, \\
& \sigma_i - \sigma_i^{\max}(1 - z_{ij}) & \leq \psi_{ij} & \leq \sigma_i - \sigma_i^{\min}(1 - z_{ij}), & i \in \mathcal{I},\ j \in \mathcal{J}_i, \\
& \sum_{j=1}^{p_i} z_{ij} & & = 1, & i \in \mathcal{I}, \\
& z_{ij} & & \in \{0, 1\}, & i \in \mathcal{I},\ j \in \mathcal{J}_i.
\end{array} \tag{3.39}$$

MILO model (3.39) is referred to as the *basic discrete model* in the sequel.

3.4.1.2 Incremental discrete model

Let $\delta_{ij} := s_{i,j+1} - s_{ij}$, for $i \in \mathcal{I}$ and $j \in \bar{\mathcal{J}}_i$, where $\bar{\mathcal{J}}_i = \{1, 2, \ldots, p_i - 1\}$. The incremental representation of choosing discrete values can be modeled as

$$\begin{array}{ll}
x_i = s_{i1} + \sum_{j=1}^{p_i - 1} \delta_{ij} z_{ij}, & i \in \mathcal{I}, \\
z_{ij} \geq z_{i,j+1}, & i \in \mathcal{I},\ j \in \bar{\bar{\mathcal{J}}}_i, \\
z_{ij} \in \{0, 1\} & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i,
\end{array} \tag{3.40}$$

where $\bar{\bar{\mathcal{J}}}_i = \{1, 2, \ldots, p_i - 2\}$. Binary variables z_{ij}, for $i \in \mathcal{I}$ and $j \in \bar{\mathcal{J}}_i$, represent the increments of the cross-sectional areas of the bars in constraints (3.40). If $z_{ij} = 1$, then $x_i \geq s_{ij}$ and $z_{i\bar{j}} = 1$ for $\bar{j} < j$. If, in addition, $z_{i,j+1} = 0$, then $z_{i\bar{j}} = 0$ for $\bar{j} > j+1$, and $x_i = s_{i,j+1}$.

In a similar manner, as in Section 3.4.1.1, we can substitute constraints (3.40) in model (P_3) and utilize (3.38) to linearize the bilinear terms. All these transform the discrete version of model (P_3) into the following MILO model:

$$
\begin{array}{llll}
\min & \displaystyle\sum_{i\in\mathcal{I}} \rho l_i x_i & & \\
\text{s.t.} & Rq & = f, & \\
& x_i - s_{i1} - \displaystyle\sum_{j=1}^{p_i-1} \delta_{ij} z_{ij} & = 0, & i \in \mathcal{I}, \\
& \sigma_i - \dfrac{E_i}{l_i} {r_i}^T u & = 0, & i \in \mathcal{I}, \\
& q_i - s_{i1}\sigma_i - \displaystyle\sum_{j=1}^{p_i-1} \delta_{ij} \psi_{ij} & = 0, & i \in \mathcal{I}, \\
& \sigma_i + \gamma_i x_i & \geq 0, & i \in \mathcal{I}, \\
& \sigma_i^{\min} \qquad\qquad \leq \sigma_i & \leq \sigma_i^{\max}, & i \in \mathcal{I}, \\
& u^{\min} \qquad\qquad \leq \ u & \leq u^{\max}, & \\
& z_{ij}\sigma_i^{\min} \qquad\quad \leq \psi_{ij} & \leq z_{ij}\sigma_i^{\max}, & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i, \\
& \sigma_i - \sigma_i^{\max}(1 - z_{ij}) \leq \psi_{ij} & \leq \sigma_i - \sigma_i^{\min}(1 - z_{ij}), & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i, \\
& z_{ij} & \geq z_{i,j+1}, & i \in \mathcal{I},\ k \in \bar{\bar{\mathcal{J}}}_i, \\
& z_{ij} & \in \{0,1\}, & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i.
\end{array}
\tag{3.41}
$$

Model (3.41) in the sequel is referred to as the *incremental discrete model*. Note that the incremental model has one less binary variable for each bar. In Section **??**, we compare the basic and incremental models and see that the incremental model is a better modeling approach when solving the discrete truss design problem.

3.4.1.3 Reformulating the MILO models

In this section, we propose a set of valid inequalities that can be used to obtain MILO reformulations of models (3.39) and (3.41) that can be solved faster. In particular, these valid inequalities can replace the discrete version of the Euler buckling constraints (3.11). Next, we derive these valid inequalities for the incremental discrete model (3.41). The derivation of the valid inequalities for model (3.39) can be done in a similar fashion.

Using formulation (3.40), the Euler buckling constraints (3.11) can be rewritten as

$$\sigma_i + \gamma_i \left(s_{i1} + \sum_{j=1}^{p_i-1} \delta_{ij} z_{ij} \right) \geq 0, \;\; i \in \mathcal{I}. \tag{3.42}$$

For each bar $i \in \mathcal{I}$, the Euler buckling constraint enforces an inequality on the binary variables z_{ij} for $j = 1, \ldots, p_i - 1$. We replace the Euler buckling constraint (3.42) by p_i constraints, each of which is formulated by using only one binary variable.

Theorem 3.3. *Considering the incremental model (3.41), the Euler buckling constraints (3.42) can be written as:*

$$\begin{aligned} \sigma_i + \gamma_i s_{ij}(1 - z_{ij}) - \sigma_i^{\min} z_{ij} &\geq 0, \qquad j = 1, \ldots, p_i - 1, \\ \sigma_i + \gamma_i s_{i,p_i} z_{i,p_i-1} - \sigma_i^{\min}(1 - z_{i,p_i-1}) &\geq 0. \end{aligned} \tag{3.43}$$

Proof. Suppose $x_i = s_{ik}$ for $k = 1 \ldots, p_i - 1$ and $i = 1, \ldots, m$. Then the Euler buckling constraint (3.42) reduces to

$$\sigma_i + \gamma_i s_{ik} \geq 0. \tag{3.44}$$

From the incremental formulation (3.40) we know that $z_{i1} = \ldots = z_{i,k-1} = 1$, and $z_{ik} = z_{i,k+1} = \ldots = z_{i,p_i-1} = 0$. Additionally, the set of constraints (3.43) is equivalent to

$$\begin{aligned} \sigma_i + \gamma_i s_{ij} &\geq 0, \qquad j = k, \ldots, p_i - 1, \\ \sigma_i - \sigma_i^{\min} &\geq 0. \end{aligned} \tag{3.45}$$

By the inequalities $\sigma_i - \sigma_i^{\min} \geq 0$, the constraints in (3.45) can be written as

$$\sigma_i \geq \max_{j=k,\ldots,p_i-1} \{-\gamma_i s_{ij}\}. \tag{3.46}$$

This constraint can be written as $\sigma_i \geq -\gamma_i s_{ik}$, which is equivalent to the Euler buckling constraint (3.44). Now, suppose that $x_i = s_{ip_i}$ for $i = 1, \ldots, m$. In this case, $z_{i1} = \ldots = z_{i,p_i-1} = 1$, and the set of constraints (3.43) is equivalent to

$$\begin{aligned} \sigma_i - \sigma_i^{\min} &\geq 0, \\ \sigma_i + \gamma_i s_{i,p_i} &\geq 0, \end{aligned}$$

which is equivalent to the Euler buckling constraint (3.42).

□

The reason to replace the Euler buckling constraint by p_i constraints is that it helps to decompose the Euler buckling constraints for the binary variables z_{ij}, $j = 1 \ldots, p_i - 1$. As it can be seen from constraint (3.42), the original Euler buckling constraint is written in terms of all the $p_i - 1$ binary variables corresponding to bar i, while in the reformulation (3.43), each constraint is in terms of only one binary variable. That is the reason that the reformulation would be more effective if p_i is large.

In Section 4.6, we demonstrate that replacing the Euler buckling constraints with constraints (3.43) in truss instances with large discrete sets can, in most cases, decrease the solution time by more than 20% on average. The reason to replace the Euler buckling constraint by p_i constraints is that it helps to decompose the Euler buckling constraints for the binary variables z_{ij}, $j = 1 \ldots, p_i - 1$. As it can be seen from constraint (3.42), the original Euler buckling constraint is written in terms of all the $p_i - 1$ binary variables corresponding to bar i, while in reformulation (3.43), each constraint is in terms of only one binary variable. That is the reason that the reformulation would be more effective if p_i is large.

3.4.2 Multi-scenario truss sizing optimization

In Section 3.4.1, we presented single scenario MILO models for the discrete truss sizing optimization problems. In the models presented in Sections 3.4.1, single scenario truss design problems were considered. In this section, we extend those models to multi-scenario discrete truss design problems, which were introduced in Section 3.2.2. As it was explained in Section 3.2.2, the cross-sectional areas of the bars are the design variables for the structure, and are common in all the external force scenarios. The solution of the multi-scenario truss design problem must withhold all the considered external force scenarios. The incremental discrete truss sizing optimization model (3.41) can be extended to account for the multiple external force scenarios as follows:

$$
\begin{array}{lllll}
\min & \sum_{i \in \mathcal{I}} \rho l_i x_i & & & \\
\text{s.t.} & R q^k & & = f^k, & k \in \mathcal{K}, \\
& x_i - s_{i1} - \sum_{j=1}^{p_i-1} \delta_{ij} z_{ij} & & = 0, & i \in \mathcal{I}, \\
& \sigma_i^k - \frac{E_i}{l_i} {r_i}^T u^k & & = 0, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& q_i^k - s_{i1}\sigma_i^k - \sum_{j=1}^{p_i-1} \delta_{ij} \psi_{ij} & & = 0, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& \sigma_i^k + \gamma_i x_i & & \geq 0, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& \sigma_i^{\min} & \leq \sigma_i^k & \leq \sigma_i^{\max}, & i \in \mathcal{I}, k \in \mathcal{K}, \\
& u^{\min} & \leq \ u^k & \leq u^{\max}, & k \in \mathcal{K}, \\
& z_{ij}\sigma_i^{\min} & \leq \psi_{ij}^k & \leq z_{ij}\sigma_i^{\max}, & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i, k \in \mathcal{K}, \\
& \sigma_i^k - \sigma_i^{\max}(1 - z_{ij}) & \leq \psi_{ij}^k & \leq \sigma_i^k - \sigma_i^{\min}(1 - z_{ij}), & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i, k \in \mathcal{K}, \\
& z_{ij} & & \geq z_{i,j+1}, & i \in \mathcal{I},\ j \in \bar{\bar{\mathcal{J}}}_i, \\
& z_{ij} & & \in \{0, 1\}, & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i,
\end{array}
\tag{3.47}
$$

In a similar manner, we can extend the basic discrete truss sizing optimization model (3.39) for multiple external force scenarios.

3.4.3 Topology design and sizing optimization

In Section 3.4.1, we developed the basic and incremental models for the discrete truss sizing optimization problem, where the cross-sectional areas of all the bars are strictly greater than zero. In this section, based on models (3.39) and (3.41) we present mathematical optimization models for the discrete truss topology design and sizing optimization (TDSO) problem, where bars can have zero cross-sectional areas. In other words, in truss topology and sizing optimization, we let the bars vanish in the final structure.

Let $\bar{\mathcal{S}}_i = \{0\} \cup \mathcal{S}_i$, where $\mathcal{S}_i$ is the set of non zeros cross-sectional areas corresponding to bar i, as defined in (3.35). The cross-sectional area of bar i, for $i \in \mathcal{I}$, takes values from the set $\bar{\mathcal{S}}_i$ in the discrete truss topology design and sizing optimization problem. We need to introduce new decision variables to incorporate zero cross-sectional areas. Let y_i, for $i \in \mathcal{I}$, be defined as

$$y_i = \begin{cases} 1 & \text{if } x_i > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Let σ_{ij} for $i \in \mathcal{I}$ and $j \in \mathcal{J}_i$ be defined as

$$\sigma_{ij} = \begin{cases} E_i \frac{\xi_i}{l_i} & \text{if } x_i = s_{ij}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.48}$$

Variable σ_{ij} represents the stress on bar i if its cross-sectional area is equal to s_{ij}; otherwise, σ_{ij} is zero. So we have

$$\sigma_i = \sum_{j \in \mathcal{J}_i} \sigma_{ij}.$$

Additionally, let σ_i^{d} be defined as

$$\sigma_i^{\mathrm{d}} = \begin{cases} E_i \frac{\xi_i}{l_i} & \text{if } x_i = 0, \\ 0, & \text{otherwise.} \end{cases} \tag{3.49}$$

Variable σ_i^{d}, for $i \in \mathcal{I}$, is a dummy variable and is equal to zero if bar i takes a non-zero cross-sectional area. However, if $x_i = 0$, then σ_i^{d} takes a non-zero value. Then, from (3.48) and (3.49), we have

$$E_i \frac{\xi_i}{l_i} - \left(\sum_{j=1}^{p_i} \sigma_{ij} + \sigma_i^{\mathrm{d}} \right) = 0. \tag{3.50}$$

As it can be seen in Equation (3.50), by introducing variable σ_i^{d}, variables σ_{ij}, for $j \in \mathcal{J}_i$, can all be equal to zero.

3.4.3.1 Basic model for TDSO

The choice constraints, as defined in (3.36), need to be slightly modified to account for topology optimization.

$$\begin{aligned}
&x_i = \sum_{j=1}^{p_i} s_{ij} z_{ij}, && i \in \mathcal{I}, \\
&\sum_{j=1}^{p_i} z_{ij} = y_i, && i \in \mathcal{I},\ j \in \mathcal{J}_i, \\
&z_{ij} \in \{0,1\} && i \in \mathcal{I}, j \in \mathcal{J}_i.
\end{aligned} \tag{3.51}$$

If $y_i = 1$, then (3.51) reduces to (3.36), while $y_i = 0$ enforces $x_i = 0$ and bar i vanishes from the structure. In addition, to enforce equalities (3.48) and (3.49), the following set of constraints are needed:

$$\begin{aligned}
(1-y_i)\underline{\sigma}_i^{\mathrm{d}} \ &\leq\ \sigma_i^{\mathrm{d}} \ \leq\ (1-y_i)\overline{\sigma}_i^{\mathrm{d}}, && i \in \mathcal{I} \\
\max\left(-\gamma_i s_{ij}, \sigma_i^{\min}\right) z_{ij} \ &\leq\ \sigma_{ij} \ \leq\ \sigma_i^{\max} z_{ij}, && i \in \mathcal{I},\ j \in \mathcal{J}_i,
\end{aligned} \tag{3.52}$$

where

$$\begin{aligned}
\underline{\sigma}_i^{\mathrm{d}} &= \tfrac{E_i}{l_i}\left(\sum_{v|r_{iv}<0} r_{iv} u_v^{\max} + \sum_{v|r_{iv}>0} r_{iv} u_v^{\min}\right), \\
\overline{\sigma}_i^{\mathrm{d}} &= \tfrac{E_i}{l_i}\left(\sum_{v|r_{iv}<0} r_{iv} u_v^{\min} + \sum_{v|r_{iv}>0} r_{iv} u_v^{\max}\right).
\end{aligned}$$

Note that the Euler buckling constraints are incorporated in the set of constraints (3.52) as well. The basic MILO model for topology design and sizing optimization is defined as

$$
\begin{array}{llll}
\min & \sum_{i\in\mathcal{I}} \rho l_i x_i & & \\
\text{s.t.} & Rq & = f, & \\
& R^T u & = \xi, & \\
& x_i - \sum_{k=1}^{p_i} s_{ik} z_{ik} & = 0, & i \in \mathcal{I} \\
& E_i \dfrac{\xi_i}{l_i} - \left(\sum_{j=1}^{p_i} \sigma_{ij} + \sigma_i^{\mathrm{d}} \right) & = 0, & i \in \mathcal{I}, \\
& q_i - \sum_{j=1}^{p_i} s_{ij}\sigma_{ij} & = 0, & i \in \mathcal{I}, \\
& \sum_{j=1}^{p_i} z_{ij} & = y_i, & i \in \mathcal{I}, \\
& \max\left(-\gamma_i s_{ij}, \sigma_i^{\min}\right) z_{ij} & \le \sigma_{ij} \le \sigma_i^{\max} z_{ij} & i \in \mathcal{I},\ j \in \mathcal{J}_i, \\
& u^{\min} & \le u \le u^{\max} & \\
& (1-y_i)\underline{\sigma}_i^{\mathrm{d}} & \le \sigma_i^{\mathrm{d}} \le (1-y_i)\overline{\sigma}_i^{\mathrm{d}}, & i \in \mathcal{I} \\
& y_i & \in \{0,1\} & i \in \mathcal{I}, \\
& z_{ij} & \in \{0,1\}, & i \in \mathcal{I},\ j \in \mathcal{J}_i.
\end{array}
\tag{3.53}
$$

If $y_i = 1$, for $i \in \mathcal{I}$, then model (3.53) reduces to a truss sizing model.

3.4.3.2 Incremental model for TDSO

In the incremental model of the TDSO problem, the cross-sectional area of bar i is defined as

$$
\begin{array}{ll}
x_i = s_{i1} y_i + \sum_{j=1}^{p_i - 1} \delta_{ij} z_{ij}, & i \in \mathcal{I}, \\
y_i \ge z_{i1} & i \in \mathcal{I}, \\
z_{ij} \ge z_{i,j+1}, & i \in \mathcal{I},\ j \in \bar{\bar{\mathcal{J}}}_i, \\
z_{ij} \in \{0,1\} & i \in \mathcal{I},\ j \in \bar{\mathcal{J}}_i.
\end{array}
\tag{3.54}
$$

To enforce equalities (3.48) and (3.49), the following set of constraints are needed:

$$\begin{aligned}
&\max\left(-\gamma_i s_{i1}, \sigma_i^{\min}\right)(y_i - z_{i1}) && \leq \sigma_{i1} \leq \sigma_i^{\max}\left(y_i - z_{i1}\right), && i \in \mathcal{I},\\
&\max\left(-\gamma_i s_{ij}, \sigma_i^{\min}\right)(z_{i,j-1} - z_{ij}) && \leq \sigma_{ij} \leq \sigma_i^{\max}\left(z_{i,j-1} - z_{ij}\right), && i \in \mathcal{I}, 2 \leq j \leq p_i - 1,\\
&\max\left(-\gamma_i s_{i,p_i}, \sigma_i^{\min}\right) z_{i,p_i-1} && \leq \sigma_{ip_i} \leq \sigma_i^{\max} z_{i,p_i-1}, && i \in \mathcal{I},\\
&(1-y_i)\underline{\sigma}_i^{\mathrm{d}} && \leq \sigma_i^{\mathrm{d}} \leq (1-y_i)\overline{\sigma}_i^{\mathrm{d}}, && i \in \mathcal{I},
\end{aligned} \tag{3.55}$$

From constraints (3.55), we can see that if $y_i = 0$, then $\sigma_{ij} = 0$, for $j \in \mathcal{J}_i$, and $\underline{\sigma}_i^{\mathrm{d}} \leq \sigma_i^{\mathrm{d}} \leq \overline{\sigma}_i^{\mathrm{d}}$. If $y_i = 1$, which implies that $x_i \neq 0$, then $\sigma_i^{\mathrm{d}} = 0$. In addition, if $z_{ij} = 1$ and $z_{i,j+1} = 0$, then $\sigma_{i\bar{j}} = 0$, for $\bar{j} \neq j+1$ and $\max\left(-\gamma_i s_{ij}, \sigma_i^{\min}\right) \leq \sigma_{ij} \leq \sigma_i^{\max}$. The incremental model for the TDSO problem is as follows:

$$\begin{aligned}
\min\quad & \sum_{i\in\mathcal{I}} \rho l_i x_i \\
\text{s.t.}\quad & Rq && = f,\\
& R^T u && = \xi,\\
& x_i - s_{i1} y_i - \sum_{j=1}^{p_i-1} \delta_{ij} z_{ij} && = 0, && i \in \mathcal{I}\\
& E_i \frac{\xi_i}{l_i} - \left(\sum_{j=1}^{p_i} \sigma_{ij} + \sigma_i^{\mathrm{d}}\right) && = 0, && i \in \mathcal{I},\\
& q_i - \sum_{j=1}^{p_i} s_{ij}\sigma_{ij} && = 0, && i \in \mathcal{I},\\
& u^{\min} && \leq u \leq u^{\max},\\
& \max\left(-\gamma_i s_{i1}, \sigma_i^{\min}\right)(y_i - z_{i1}) && \leq \sigma_{i1} \leq \sigma_i^{\max}\left(y_i - z_{i1}\right), && i \in \mathcal{I},\\
& \max\left(-\gamma_i s_{ij}, \sigma_i^{\min}\right)(z_{i,j-1} - z_{ij}) && \leq \sigma_{ij} \leq \sigma_i^{\max}\left(z_{i,j-1} - z_{ij}\right), && i \in \mathcal{I}, 2 \leq j \leq p_i - 1\\
& \max\left(-\gamma_i s_{i,p_i}, \sigma_i^{\min}\right) z_{i,p_i-1} && \leq \sigma_{ip_i} \leq \sigma_i^{\max} z_{i,p_i-1}, && i \in \mathcal{I},\\
& (1-y_i)\underline{\sigma}_i^{\mathrm{d}} && \leq \sigma_i^{\mathrm{d}} \leq (1-y_i)\overline{\sigma}_i^{\mathrm{d}}, && i \in \mathcal{I},\\
& y_i && \geq z_{i1}, && i \in \mathcal{I},\\
& z_{ij} && \geq z_{i,j+1}, && i \in \mathcal{I},\ j \in \bar{\bar{P}}_i\\
& y_i && \in \{0,1\}, && i \in \mathcal{I},\\
& z_{ij} && \in \{0,1\}, && i \in \mathcal{I},\ j \in \bar{P}_i.
\end{aligned} \tag{3.56}$$

It is worth mentioning that, similar to model (3.47), models (3.53) and (3.56) can be extended to account for the multi-scenario discrete TDSO problem.

3.5 Conclusions

In this chapter, we reviewed various mathematical optimization models for the continuous truss sizing problems and presented two important characteristics of the feasible set of the problem. Then we presented the basic and incremental MILO models for discrete truss sizing problems and extended the models to multi-scenario external force problems. At the end, we extended the MILO models to discrete truss topology design and sizing optimization problems.

Chapter 4

Truss Design Problem: Solution Methodology

In Chapter 3, we introduced various mathematical models for the truss design problem under different assumptions. In this chapter we introduce a novel solution methodology to provide high-quality solutions in a reasonable time for the discrete truss design problems, and demonstrate the efficiency of the methodology by extensive numerical experiments. We begin in Section 4.1 by giving a literature review of the solution methodologies that are used to solve discrete truss design problems. In Section 4.2, we propose the Neighborhood Search Mixed Integer Linear Optimization (NS-MILO) algorithm to solve discrete truss sizing problems. In Section 4.3 we introduce the truss structures that are used to benchmark the mathematical models and our solution methodology. In Section 4.4, we present the numerical results on the relaxation model (3.34), and in Section 4.5, we compare the performance of the basic and incremental models (3.39) and (3.41). In Section 4.6, we present the numerical results on strengthening the Euler buckling constraints. Then, in

Section 4.7, we demonstrate the efficiency of the NS-MILO algorithm through extensive computational experiments. Finally, we close the chapter by presenting our conclusions in Section 4.8.

4.1 Introduction and literature review

Several meta-heuristic methods have been used to generate good quality solutions for truss design optimization problems. Genetic algorithms yield the most common meta-heuristics which are historically used to provide good solutions to truss design problems [Hajela and Lee, 1995, Kaveh and Kalatjari, 2004, Rajeev and Krishnamoorthy, 1992, Wu and Chow, 1995]. Ant colony optimization [Bland, 2001, Camp and Bichon, 2004a, Kaveh et al., 2008] and particle swarm optimization methods [Li et al., 2009, Zeng and Li, 2012] have also been widely used to solve truss design problems.

Other methods including simulated annealing [Kripka, 2004], artificial bee colony optimization [Sonmez, 2011, Stolpe, 2011], mine blast algorithm [Sadollah et al., 2012, 2015], colliding bodies optimization [Kaveh and Ghazaan, 2015, Kaveh and Mahdavi, 2014]), and harmony search [SeokLee and Geem, 2004] have also been used. Stolpe [2016] provides a review on truss design problems with discrete cross-sectional areas. He presented various models and different methods, including global optimization methods, heuristics, and meta-heuristics to solve discrete truss design problems. In the conclusions, he also stated the need for more publicly available benchmarking problems, which we address herein by contributing three new scalable problem sets.

Mladenović and Hansen [1997] developed a variable neighborhood search (VNS) algorithm to solve discrete optimization problems. A VNS algorithm solves the problem iteratively over a neighborhood structure. At each iteration, local search methods are used to find the optimum in the neighborhood. Several general-purpose and problem-specific variants have been developed [Hanafi, 2016, Hansen and Mladenović, 2003, Lazić, 2010]. Svanberg and Werme [2005] used a neighborhood search method to solve the topology optimization problem. However, they consider a limited neighborhood, where "*two different designs are neighbors if they differ in only one single element*". Thus, the complexity of solving the associated subproblems to proven optimality is $\mathcal{O}(n)$, where n is the number of elements in the structure. In another article, Svanberg and Werme [2007] consider a M-neighborhood, where the number of the design variables that can simultaneously change is limited to M at each iteration ($M = 1, 2, 4$). The complexity of the associated subproblem on a M-neighborhood is $\mathcal{O}\left(n^M\right)$ for $M = 1, 2, 4$.

The subproblems of the neighborhood search MILO (NS-MILO) algorithm that we propose here are defined over exponentially large neighborhoods, which in turn decreases the likelihood of getting stuck in a local optimum. As our experiments illustrate, the trade-off between solving more complex subproblems and the time needed to obtain near-optimal solutions for large-scale truss design problems is advantageous. Additionally, we do not solve the subproblems to optimality in the NS-MILO approach, but rather, we stop the solution process of the subproblems as soon as a solution better than the current best solution is found. This is one of the reasons that enables the NS-MILO approach to scale well as the size of the problem increases.

The minimum weight discrete truss design problem considering Hooke's law, bounds on the stress, and Euler buckling constraints has only been solved for small-scale problems. The main contribution of the work presented herein is developing an efficient solution methodology to approximately solve large-scale discrete truss design problems with more than 12,000 binary variables.

First, we consider two well-known instances from the literature: the 10-bar truss and the 72-bar truss problems [Haftka and Gürdal, 2012] to compare the NS-MILO approach with other algorithms used in the literature to solve these instances. In addition, we introduce three different scalable truss design problem sets, namely, 2D cantilever trusses, 3D cantilever trusses, and truss models of an airplane wing with the goal to demonstrate how the NS-MILO approach scales as the size of the problem grows. These three new problems are available online[1].

The computational experiments are conducted on a workstation with Dual Intel Xeon® CPU E5-2630 @ 2.20 GHz (20 cores) and 64 GB of RAM. We use Gurobi 7.0.2 (2016) to solve the MILO models. Gurobi has the capability of using multiple threads in solving a MILO problem and it uses a branch and cut algorithm. The optimality gap threshold is set to 0.1%, i.e., when the gap between the best found solution and the lower bound is less than 0.1%, Gurobi stops and returns the best solution found up to that point. We set Gurobi to use 16 threads when solving all the problems and subproblems in this section.

4.2 The NS-MILO Approach

We now present a methodology for solving basic discrete model (3.39) and incremental discrete model (3.41). For ease of presentation in what follows, we refer to the problems with *full discrete sets* as the *original problems*, .

By way of experimentation, it turns out that MILO solvers are not able to solve to optimality even small-sized 3D instances of the truss design problem [Stolpe, 2016]. The results, presented in Table 4.1 for the optimization of 3D cantilever trusses (see Section 4.3.4) by Gurobi 7.0.2, confirm Stolpe's observation that current methodology is not able to solve even moderate-size discrete truss design problems to proven optimality. Note that the relative optimality gap threshold is 0.1%, and

[1] `https://github.com/shahabsafa/truss-data.git`

the cardinality of the discrete set of cross-sectional areas for all the bars is 41. The solver is terminated after 24 hours of CPU time for all but the smallest problem with optimality gap well over %20.

Table 4.1: Solutions of 3D-truss sizing instances obtained directly using Gurobi.

# bars	# bin. var.	Time(s)	Weight (kg)	Opt. Gap
20	800	247.55	9.75	0.07%
40	1600	86400.02	21.24	25.76%
60	2400	86400.03	24.82	30.03%
80	3200	86400.04	31.22	32.43%
100	4000	86400.05	32.04	34.36%

To generate high-quality solutions in a reasonable time, we present a new solution methodology, referred to as Neighborhood Search MILO (NS-MILO). The NS-MILO approach explores MILO subproblems which are defined over the feasible set of the original problem.

In existing neighborhood search algorithms used to solve the truss design problems, see e.g., Hanafi [2016], Hansen and Mladenović [2003], Lazić [2010], Mladenović and Hansen [1997], Svanberg and Werme [2005, 2007], the subproblems are defined on a small neighborhood of the incumbent solution so that those subproblems are polynomially solvable. This, in turn, may result in small local improvements, since the subproblems explore only a small neighborhood for a better solution. However, the neighborhoods of the subproblems in the NS-MILO approach are significantly larger, such that the number of the feasible solutions of the subproblems grows exponentially as the number of bars in the truss increases. Therefore, the subproblems become NP-hard, i.e., the time complexity to solve them to global optimality grows exponentially as the size of the problem grows. However, we do not solve the subproblems to proven optimality in the NS-MILO approach. Thus, this approach enables us to explore a significantly larger neighborhood for a better solution, which decreases the likelihood of getting stuck in a local optimum.

In existing neighborhood search algorithms, see e.g., Hanafi [2016], Hansen and Mladenović [2003], Lazić [2010], Mladenović and Hansen [1997], Svanberg and Werme [2005, 2007], the small neighborhood subproblems are solved to optimality, while in the NS-MILO approach, we do not solve the NP-hard subproblems to global optimality. Observe, that the computational resources needed to prove optimality of the subproblems can be better spent towards solving the original problem. Motivated by this observation, we stop solving the subproblems as soon as a solution better than the current best solution is found, and define the next subproblem in the neighborhood of the improved solution. Nowadays, MILO solvers, such as Gurobi (2016), have extremely powerful methods to improve the best integer feasible solution found, which helps to significantly reduce the time needed to improve the integer feasible solution of the subproblems of the NS-MILO approach.

The fact that the NS-MILO approach explores significantly larger neighborhoods and does not prove global optimality for the subproblems makes it possible to provide high-quality solutions to large-scale truss design problems.

The NS-MILO methodology is based on sequentially exploring MILO subproblems where the feasible set of each subproblem is a subset of the feasible set of the original MILO problem. The set of the discrete values of each bar at each subproblem is a subset of the full discrete set of that bar. In fact, the discrete values are chosen from the neighborhood of a given feasible solution. Hence, due to its reduced size, each subproblem is easier to tackle than the original problem. The MILO subproblems are denoted by $\text{MILO}_k(\boldsymbol{x})$. Specifically, $\text{MILO}_k(\boldsymbol{x})$ is the MILO formulation of a subproblem of the discrete truss design problem, where the cardinality of the discrete set for each bar is at most k, and the discrete set is generated from the assignment of the bar cross-sectional areas $\boldsymbol{x} \in \mathbb{R}^n$. Note that the complexity of solving a MILO_k subproblem is $\mathcal{O}(m^k)$. Let $\hat{\mathcal{S}}_i$ be the discrete set of bar

$i \in I$ in subproblem MILO$_k(\boldsymbol{x})$. We attempt to solve three different kinds of MILO subproblems:

1. MILO$_2(\boldsymbol{x})$, for $\boldsymbol{x} \in \mathbb{R}^n$, is the MILO formulation of the discrete design problem, where the cardinality of the discrete set of cross-sectional areas for each bar is equal to two. The set $\hat{\mathcal{S}}_i$ for a MILO$_2(\boldsymbol{x})$ is defined by
$$\hat{\mathcal{S}}_i := \begin{cases} \{s_{i1}, s_{i2}\}, & \text{if } \ x_i < s_{i1}, \\ \{s_{ik}, s_{i,k+1}\}, & \text{if } \ s_{ik} \le x_i < s_{i,k+1}, \\ \{s_{i,p_i-1}, s_{ip_i}\}, & \text{if } \ x_i \ge s_{ip_i}. \end{cases}$$

2. MILO$_3(\boldsymbol{x})$ for $x_i \in \mathcal{S}_i, \ i \in I$, is the MILO formulation of the discrete design problem where the cardinality of the discrete set for each bar is at most three. Suppose that $x_i = s_{ik}$. Then the set $\hat{\mathcal{S}}_i$ for a MILO$_3(\boldsymbol{x})$ is defined by
$$\hat{\mathcal{S}}_i := \begin{cases} \{s_{i1}, s_{i2}\}, & \text{if } \ k = 1, \\ \{s_{i,k-1}, s_{i,k}, s_{i,k+1}\}, & \text{if } \ 2 \le k \le p_i - 1, \\ \{s_{i,p_i-1}, s_{i,p_i}\}, & \text{if } \ k = p_i. \end{cases}$$

3. MILO$_5(\boldsymbol{x})$ for $x_i \in \mathcal{S}_i, \ i \in I$, is the MILO formulation of the discrete design problem where the cardinality of the discrete set for each bar is at most five. Suppose that $x_i = s_{ik}$. Then the set $\hat{\mathcal{S}}_i$ for a MILO$_5(\boldsymbol{x})$ is defined as follows
$$\hat{\mathcal{S}}_i = \begin{cases} \{s_{i1}, s_{i2}, s_{i3}\}, & \text{if } \ k = 1, \\ \{s_{i1}, s_{i2}, s_{i3}, s_{i4}\}, & \text{if } \ k = 2, \\ \{s_{i,k-2}, s_{i,k-1}, s_{i,k}, s_{i,k+1}, s_{i,k+2}\}, & \text{if } \ 3 \le k \le p_i - 2, \\ \{s_{i,p_i-2}, s_{i,p_i-1}, s_{i,p_i}, s_{i,p_i+1}\}, & \text{if } \ k = p_i - 1, \\ \{s_{i,p_i-2}, s_{i,p_i-1}, s_{i,p_i}\}, & \text{if } \ k = p_i. \end{cases}$$

Suppose that the original discrete set of the bar $i \in I$ is defined by the finite set (3.35). We start the NS-MILO approach by obtaining a high-quality feasible so- lution for the continuous model (P$_3$) using a nonlinear optimization (NLO) solver. The local optimal solution of the model (P$_3$) is denoted by $\boldsymbol{x}^{\mathbf{0}}$. Then we attempt to solve a sequence of MILO$_2$ subproblems. We start by attempting to solve MILO$_2$($\alpha\boldsymbol{x}^{\mathbf{0}}$) with $\alpha = 1$ using a MILO solver considering a predefined limit on the solution time. If an integer feasible solution is not found within the time limit, we increase α by 0.05, and again try to solve MILO$_2$($\alpha\boldsymbol{x}^{\mathbf{0}}$). We continue solving MILO$_2$($\alpha\boldsymbol{x}^{\mathbf{0}}$) subproblems until an integer feasible solution is found.

Next, we generate MILO$_3$($\hat{\boldsymbol{x}}$), where $\hat{\boldsymbol{x}}$ is the best integer feasible solution obtained from MILO$_2$($\alpha\boldsymbol{x}^{\mathbf{0}}$). We run the MILO solver to solve MILO$_3$($\hat{\boldsymbol{x}}$) until a solution better than the current best solution is found. The improved solution to MILO$_3$($\hat{\boldsymbol{x}}$) is assigned to $\hat{\boldsymbol{x}}$. As soon as a better solution is found, we stop the solver, and use the improved solution $\hat{\boldsymbol{x}}$ to generate the next MILO$_3$ subproblem. We continue with MILO$_3$($\hat{\boldsymbol{x}}$) subproblems until the objective function does not improve.

Afterwards, we attempt to solve MILO$_5$($\hat{\boldsymbol{x}}$), and similarly solve MILO$_5$($\hat{\boldsymbol{x}}$) until a better solution than the current best solution is found. We stop the solver as soon as a better solution is found, and use the improved solution to generate the next MILO$_5$ subproblem. We continue with MILO$_5$($\hat{\boldsymbol{x}}$) subproblems until the objective function does not improve.

Note that MILO$_3$ and MILO$_5$ subproblems are not solved to optimality, since MILO solvers spend a significant portion of time to reduce the optimality gap and ultimately proving the optimality of the best integer solution obtained. However, proving that a solution is optimal for a subproblem is not the best way to allocate computational resources in order to solve the original discrete problem. Therefore, we stop the solver as soon as a better feasible solution is found, and use that feasible solution to generate the next subproblem.

The approach described above to generate and attempt to solve MILO subproblems sequentially is in fact a moving-neighborhood search, where we search for better integer feasible solutions in the neighborhood of the best solution found so far. This neighborhood search approach for truss design problems is summarized in Algorithm 1. In this algorithm, BestSol(.) returns the best solution of the corresponding problem in the given time budget, while FindSol(.) returns a better solution as soon as it finds one, or returns the solution that was previously found. If FindSol(P) returns the previously found solution, it indicates that either that solution is optimal for the subproblem, or the time limit of solving subproblem (P) is reached. In $(\hat{x}, \hat{\eta}) :=$ BestSol(P) and $(\hat{x}, \hat{\eta}) :=$ FindSol(P), $\hat{x}$ is the solution that is returned and $\hat{\eta}$ is the weight of the structure for solution $\hat{x}$.

Algorithm 1 The NS-MILO approach for single scenario truss sizing problems

```
1: $x^0 :=$ local optimal solution of the continuous model
2: $\alpha := 1$
3: **repeat**
4:     $(\hat{x}, \hat{\eta}) := \text{BestSol}(\text{MILO}_2(\alpha x^0))$
5:     $\alpha := \alpha + 0.05$
6: **until** $\text{MILO}_2(\overline{x})$ is feasible
7: **repeat**
8:     $\eta_{\text{curr}} := \hat{\eta}$
9:     $(\hat{x}, \hat{\eta}) := \text{FindSol}(\text{MILO}_3(\hat{x}))$
10: **until** $\hat{\eta} = \eta_{\text{curr}}$
11: **repeat**
12:     $\eta_{\text{curr}} := \hat{\eta}$
13:     $(\hat{x}, \hat{\eta}) := \text{FindSol}(\text{MILO}_5(\hat{x}))$
14: **until** $\hat{\eta} = \eta_{\text{curr}}$
15: **return** $\hat{x}$
```

Attempting to solve MILO$_k$ subproblems can be done for bigger values of k $(k = 7, 9, \ldots)$. However, in our experiments, considering these subproblems did not help to significantly improve the solution, when one considers the time that was spent on trying to solve those larger neighborhood subproblems.

4.3 Truss Problems

In this section, we introduce the truss problems that are used in the numerical experiments to evaluate the mathematical models and the solution methodology. We consider two classical truss problems, namely, the 10-bar truss and the 72-bar truss to validate our NS-MILO approach. Additionally, we introduce three scalable truss problem sets that are used to evaluate the performance of the NS-MILO approach as the size of the problem grows.

4.3.1 The 10-bar truss

The 10-bar truss [Haftka and Gürdal, 2012], shown in Figure 4.1, is frequently used as a benchmark example. The external force f on nodes 2 and 4 is equal to 444,800 N (10^5 lb), and the material properties are listed in Table 4.2. Additionally, the displacement bound on the y-direction of the nodes 1 and 2 is ± 2.0 in. The discrete set of potential cross-sectional areas is listed in Appendix A.

Table 4.2: Aluminum alloy material properties used for 10-bar and 72-bar problems.

Property	Value
ρ	$0.1\,\text{lbm/in}^3$
E	10^7 psi
σ_Y	25000 psi

4.3.2 The 72-bar truss

The 72-bar truss problem [Haftka and Gürdal, 2012] shown in Figure 4.2 is another common benchmark. The bar material properties are listed in Table 4.2. Additionally, the displacement bound on the x and y direction of the nodes 1, 2, 3, and 4 is ± 0.25 in. We have two load cases. In load case one, the external force is only

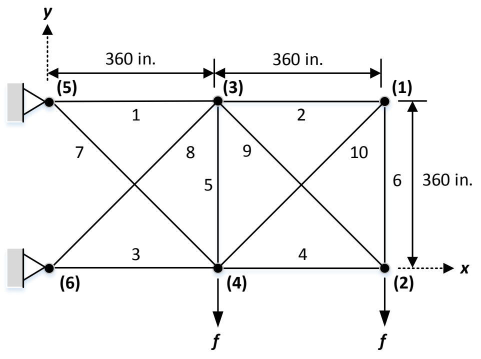


Figure 4.1: The 10-bar truss.

exerted on node 1, with value $f_x = 5000$ lbf, $f_y = 5000$ lbf, and $f_z = -5000$ lbf. In load case two, the external force is exerted on the z direction of the nodes 1, 2, 3, and 4 with value $f_z = -5000$ lbf. The discrete set of the cross-sectional areas is defined in Appendix A.

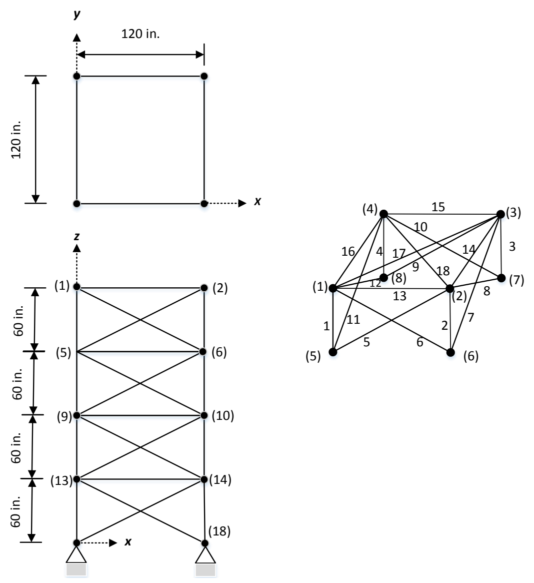


Figure 4.2: The 72-bar truss.

4.3.3 Scalable 2D cantilever truss problems

The 2D cantilever problem set is made scalable by varying the number of blocks, where each block has five bars. A 2D cantilever instance with 3 blocks is illustrated in Figure 4.3.

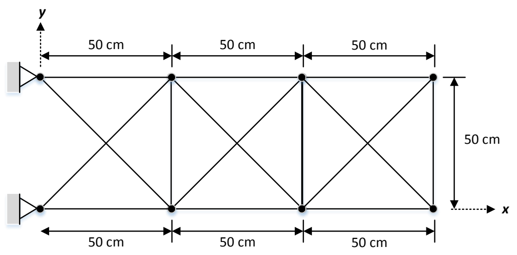


Figure 4.3: The 2D cantilever problem instance with 3 blocks.

The material properties are listed in Table 4.3. The yield stress corresponds to the yield strength of an aluminum alloy with a 50% safety margin. The external force is generated randomly at each node from a given interval using a uniform distribution. Let $f_0 = 1.25 \times 10^5/n_b$ N, where n_b is the number of the blocks of the cantilever problem. For all the bottom nodes, the y coordinate of the force randomly takes values in the interval $[-f_0, 0]$, and the x coordinate of the force takes value in the interval $[-f_0/10, f_0/10]$. For the top nodes, the y and x force coordinates take value in the intervals $[-f_0/10, 0]$ and $[-f_0/100, f_0/100]$, respectively. Hence, the dominant coordinate of the force at each node is the y direction with a negative sign. The average of the force on the bottom nodes is 10 times bigger than that of the top nodes. The ground structures and the external forces of the 2D cantilever trusses are available online[2]. The bars can take 41 different cross-sectional areas, which are listed in Appendix A.

Displacement bounds are considered for the two nodes of the tip of the cantilever

[2]https://github.com/shahabsafa/truss-data.git

Table 4.3: Aluminum alloy material properties used for the 2D and 3D cantilever problems.

Property	Value
ρ	$2.7\,\mathrm{kg/m^3}$
E	$69\,\mathrm{GPa}$
σ_Y	$172.36\,\mathrm{MPa}$

problem in the x and y directions, and are presented in Table 4.4. Note that the displacement bounds vary with the number of the blocks n_b.

Table 4.4: Displacement bounds for the 2D cantilever problem.

n_b	Bounds (cm)	n_b	Bounds (cm)	n_b	Bounds (cm)
1	0.1	8	6.4	36	129.6
2	0.4	12	14.4	40	160.0
3	0.9	16	25.6	44	193.6
4	1.6	20	40.0	48	230.4
5	2.5	24	57.6	52	270.4
6	3.6	28	78.4	56	313.6
7	4.9	32	102.4	60	360.0

4.3.4 Scalable 3D cantilever truss problems

The 3D cantilever problem set is an extension of the 2D cantilever problem set. Now each block is a cube, and all the diagonals of the five faces are included. Additionally, two of the main diagonals of each cube are included. This adds up to 20 bars per block. The 3D cantilever instance with 3 blocks is illustrated in Figure 4.4.

The material properties are listed in Table 4.3. Similar to the 2D cantilever problem, the force is generated randomly at each node from a given interval using a uniform distribution. Let $f_0 = 1.2 \times 10^6/n_b$ N, where n_b is the number of the blocks of the cantilever instance. For all the bottom nodes, the y coordinate of the force randomly takes values in the interval $[-f_0, 0]$, and the x and z force coordinates

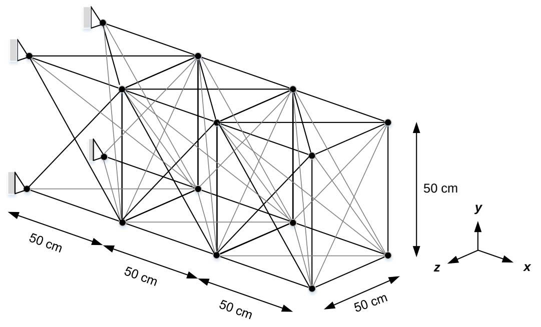


Figure 4.4: 3D cantilever instance with 3 blocks.

randomly take values in the interval $[-f_0/10, f_0/10]$. For the bottom nodes, the y, x, and z force coordinates take values in the intervals $[-f_0/10, 0]$, $[-f_0/100, f_0/100]$, and $[-f_0/100, f_0/100]$ respectively. Hence, the dominant coordinate of the force at each node is the y direction with a negative sign. The average of the force on the bottom nodes is 10 times bigger than that of the top nodes. The ground structures and the external forces of the 3D cantilever trusses are available online[3]. Additionally, the bars can take 41 different cross-sectional areas in the range $[0.25, 85]$ cm^2, which are listed in Appendix A.

Displacement bounds are considered for the four nodes at the tip of the cantilever truss on the x, y, and z direction, and are presented in Table 4.5. Similar to the 2D cantilever instances, the displacement bounds vary with the number of the blocks n_b.

4.3.5 Wing truss problems

We now consider a 3D wing modeled with bars. The truss layout is generated based on the undeformed common research model (uCRM) geometry [Brooks et al., 2018], and is shown in Figure 4.5. For the aerodynamic load, we assume an elliptical

[3]https://github.com/shahabsafa/truss-data.git

Table 4.5: Displacement bounds of the 3D cantilevers.

n_b	Bounds (cm)	n_b	Bounds (cm)	n_b	Bounds (cm)
1	0.3	6	3.6	11	12.1
2	0.4	7	4.9	12	14.4
3	0.9	8	6.4	13	16.9
4	1.6	9	8.1	14	19.6
5	2.5	10	10.0	15	22.5

distribution in the spanwise direction and a uniform distribution in the chordwise direction. The total load of one wing is set to be one half of the maximum takeoff weight of the uCRM. For simplicity, we also assume that the aerodynamic load is not affected by structural deformation, i.e., the aeroelastic effect is neglected in this study. The ground structures and the external forces of the wing trusses are available online[4]. The bars can take 44 different cross-sectional areas in the range $[0.25, 1200]$ cm^2, which are listed in Appendix A.

The material properties are listed in Table 4.6. While it would be possible to consider bounds on the displacements, we do not do that here, because in practical aircraft and wing design, stress and buckling constraints are sufficient to achieve feasible designs from the structural point of view.

Table 4.6: Aluminum alloy material properties for the wing problem.

Property	Value
ρ	$2.7\,\mathrm{kg/m^3}$
E	$69\,\mathrm{GPa}$
σ_Y	$270\,\mathrm{MPa}$

[4]https://github.com/shahabsafa/truss-data.git

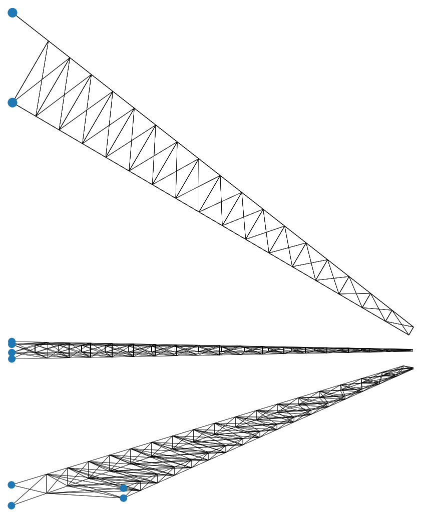


Figure 4.5: Wing truss problem instance with 315 bars.

4.4 Lower bound of the continuous model

In Section 3.3, we presented the MILO relaxation (3.34) which provides a lower bound for the optimal objective value of the continuous model (P_3), presented in page 42. In this section we aim to evaluate the relaxation model.

In Figure 4.6, the objective function of the relaxation model (3.33) for the 10-bar truss is plotted for different values of τ. Additionally, the objective value of the solution obtained from the non-linear optimization solver IPOPT [Wächter and Biegler, 2006] is plotted. We know that the objective value of model (3.33) and the objective value of the solution obtained by IPOPT provide a lower bound and an upper bound for the optimal objective value of problem (P_3), respectively. As we can see in Figure 4.6, the upper bound and lower bound of the optimal objective value for the 10-bar truss converge as τ increases. So we can conclude that the solution obtained by IPOPT is a global optimal solution and the lower bound obtained by MILO relaxation (3.34) converges to the optimal objective value of the problem as τ increases.

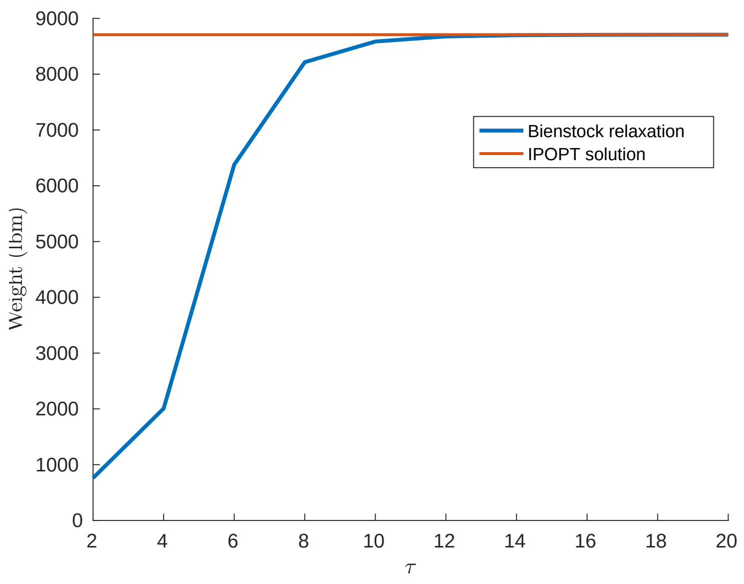


Figure 4.6: Comparison of the solution provided by the IPOPT solver and the solution of the relaxation model (3.34) with different values of τ for the 10-bar truss.

Let x^1 and x^2 be, respectively, the solution of model (P_3) obtained by IPOPT and the solution of the MILO relaxation model (3.34) for the 10-bar truss. Solutions x^1 and x^2 are reported in Table 4.7. From the table, we have $\|x^1\|_2 = 101.08$ and $\|x^1 - x^2\|_2 = 0.71$, which indicate that not only the optimal objective value of the MILO relaxation (3.34) converges to the one obtained by IPOPT for the 10-bar truss, but also the solution of the MILO relaxation (3.34) converges to the solution of model (P_3), as τ increases.

In Table 4.8, the results of the MILO relaxation (3.34) are presented for the 2D cantilever truss with 5 bars, 10-bar truss, and the 3D cantilever truss with 20 bars. The time budget for all the MILO relaxation problems is set to 8 hrs. As we can see in the table, the MILO relaxation, as τ increases, provides a tight lower bound for the optimal objective value of the 2D cantilever truss with 5 bars, 10-bar truss,

Table 4.7: Cross-sectional areas (in^2) and the weight (lbm) of the solution of the 10-bar truss obtained by IPOPT and by MILO relaxation (3.34) with $\tau = 16$.

Bar	Model (P_3)	Model (3.34)
1	17.68	17.12
2	0.10	0.10
3	57.09	57.09
4	40.62	40.62
5	0.10	0.10
6	0.10	0.10
7	7.42	7.34
8	69.16	69.15
9	12.55	12.98
10	0.10	0.10
W	8707.56	8704.40

and 3D cantilever truss with 20 bars. In fact, the gap between the lower bound of MILO relaxation (3.34) and the objective value of the solution provided by IPOPT is less than %1 and %0.1 for the 2D cantilever truss with 5 bars and the 10-bar truss, respectively.

Table 4.8: Comparison of the weight (lbm) and solution time (s) of the solution obtained by IPOPT with the solution of the MILO relaxation (3.34).

Problem type	# bars	Model	τ	Objective value	Lower bound	Time
2D cantilever	5	(P_3)-IPOPT	-	5.461	-	0.06
		(3.34)	12	5.352	5.352	402.15
			14	5.434	5.433	3,606.65
10-bar truss	10	(P_3)-IPOPT	-	8,707.558	-	0.16
		(3.34)	12	8,676.732	8,676.048	23.41
			14	8,698.829	8,698.008	62.76
			16	8,704.407	8,704.326	201.57
3D cantilever	20	(P_3)-IPOPT	-	20.182	-	0.28
		(3.34)	12	19.945	19.653	28,800.00
			14	20.126	19.755	28,800.00
			16	20.168	19.637	28,800.00

When optimality is proved for MILO relaxation (3.34), the optimal solution of the relaxation problem provides a lower bound for the optimal objective value of the continuous truss design problem. However, when the MILO relaxation (3.34) is not solved to optimality, then the actual solution does not provide a lower bound. Instead, the lower bound of the MILO relaxation provides a lower bound for the optimal solution of the continuous design problem. Notice in Table 4.8, that the solution time of the MILO relaxation (3.34) increases rapidly as τ increases or size of the problem grows. In other words current MILO solvers such as Gurobi [Gurobi Optimization Inc., 2016] or CPLEX [IBM Knowledge Center, 2017] are not capable of solving large scale MILO relaxation (3.34). Thus, the bound that can be computed for large scale truss design problems in a reasonable time is not as tight as the ones in Table 4.8.

4.5 Basic versus incremental model

Although the number of binary variables of the incremental model (3.41) is not significantly less than that of the basic model (3.39), the incremental model is solved significantly faster than the basic model. In Figure 4.7, the objective function improvements of the basic and incremental models are plotted for the 2D cantilever instance with 5 blocks. As we can see, the incremental model stops at $t = 64$ s, while the basic model stops at $t = 256$ s. Additionally, the incremental model finds the optimal solution at $t = 31$ s, while the basic model finds the optimal solution about 7 times slower at $t = 221$ s. Therefore, the incremental model is faster in proving optimality, and it finds the optimal solution significantly faster than the basic model for the 2D cantilever truss with 5 blocks.

In Table 4.9, the 2D and 3D cantilever trusses that are solved to global optimality in 24 hours with either the basic or incremental model are reported. The incremental model is significantly faster than the basic model in all the trusses reported in Table 4.9. As a result, we use the incremental model through the rest of the chapter.

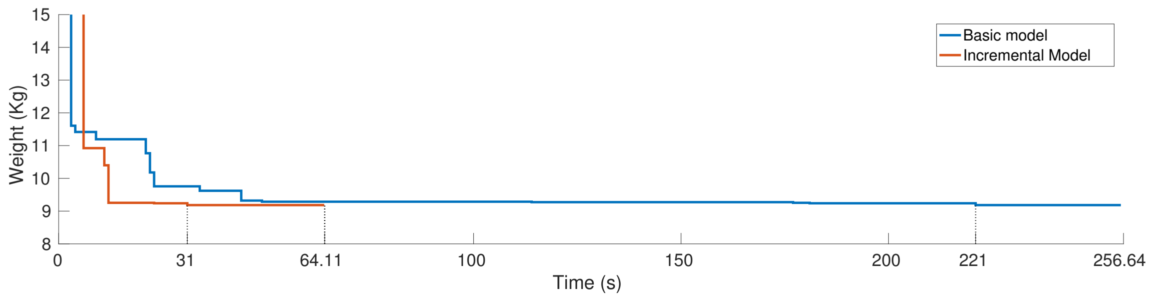


Figure 4.7: Comparison of the basic model and the incremental model for the 2D cantilever problem instance with 5 blocks.

The intuition behind the better performance of the incremental discrete model is that 1) In the incremental discrete model, fixing one of the binary variables to either zero or one, due to inequality constraints (3.40), automatically fixes the value of a high number of binary variables; 2) This same set of inequality constraints (3.40) involving the binary variables can be effectively used by MILO solvers to generate extra cuts that help to solve the MILO model efficiently.

Table 4.9: Solution times (s) and weights (kg) for the basic and incremental models.

	n_b	m	Basic model			Incremental model		
			bin. var	weight	time	bin var	weight	time
2D	1	5	205	2.55	0.93	200	2.55	0.89
	2	10	410	5.68	27.92	400	5.68	4.33
	3	15	615	6.87	240.42	600	6.87	24.17
	4	20	820	9.64	3883.97	800	9.64	174.54
	5	25	1025	9.18	256.64	1000	9.18	64.11
	6	30	1230	13.76	86400.00	1200	13.76	9891.51
3D	1	20	820	9.75	37275.40	800	9.75	247.55

4.6 Strengthening the Euler buckling constraints

Next, we numerically examine the effect of replacing the Euler buckling constraint introduced in (3.11) with constraints (3.43) introduced in Theorem 3.3. The impact of this replacement is demonstrated in Table 4.10, where t_1, and t_2 denote the solution time of the model with constraints (3.11), and the solution time of the model with constraints (3.43), respectively. As we can see in Table 4.10, replacing the Euler buckling constraints (3.11) with constraints (3.43), reduces the solution time in 5 out of 7 instances. Specifically, for the largest 2D cantilever truss and the 3D cantilever truss with 20 bars, the solution time decreases by 10% and 19% respectively.

Table 4.10: Impact of introducing the buckling constraints (3.43) on the solution time (s).

	n_b	m	Euler Const. (3.11)		Constr. (3.43)		$\frac{t_2}{t_1}$
			weight (kg)	t_1	weight (kg)	t_2	
2D	1	5	2.55	0.89	2.55	0.56	0.63
	2	10	5.68	4.33	5.68	3.16	0.73
	3	15	6.87	24.17	6.87	21.36	0.88
	4	20	9.64	174.54	9.64	190.51	1.09
	5	25	9.18	64.11	9.18	110.07	1.71
	6	30	13.76	9891.51	13.76	8891.16	0.90
3D	1	20	9.75	247.55	9.75	201.47	0.81

Replacing the Euler buckling constraints (3.11) with constraints (3.43) is useful for the truss design problems where we have a large discrete set for the cross-sectional areas. However, if the size of the discrete set is less than 5, the replacement does not help.

4.7 Numerical results with NS-MILO

In this section, we compare the NS-MILO approach with attempting to solve the original MILO problem directly by Gurobi. We refer to the latter as the *full-MILO approach*. In all the experiments, the incremental model is used.

To evaluate the NS-MILO approach, we consider the well-known 10-bar and 72-bar instances [Haftka and Gürdal, 2012] and compare the solutions obtained by NS-MILO with the ones obtained by other approaches used to solve those instances. Additionally, we attempt to solve the 2D and 3D cantilever instances with 20 to 300 bars, and the wing instances with 81 to 315 bars to see how the NS-MILO approach scales as the size of the problem grows. Results of the single-scenario and multi-scenario truss sizing problems are presented in Sections 4.7.1 and 4.7.2, respectively.

In Tables 4.13 – 4.15 and 4.18 – 4.20, m, w_f, and t_f denote the number of bars, the weight of the solution generated by the full-MILO approach, and the time to obtain that solution, respectively. Parameters n_s, w_n, and t_n denote the number of MILO subproblems, the weight of the solution, and the solution time of the NS-MILO approach, respectively. Additionally, in Tables 4.13, 4.14, 4.18, and 4.19, n_b is the number of blocks of the cantilever instances, and in Table 4.15, w_c, and t_c denote the weight of the solution of the continuous design problem, and the time to obtain that solution, respectively.

4.7.1 Single-scenario results

In this section, we compare the NS-MILO approach with the full-MILO approach for the single-scenario truss sizing problems. The maximum solution time of the full-MILO approach is set to 24 hrs for all the instances that are solved in this section. The solution time limits for the MILO$_2$, MILO$_3$, and MILO$_5$ subproblems are set to 300, 1500, and 2500 seconds respectively, except for the wing instances with more than 250 bars, where the solution time limits for MILO$_2$, MILO$_3$, and MILO$_5$ subproblems are set to 300, 3000, and 6000 seconds respectively. That means that our experiments compare the quality of the solution generated by the full-MILO approach after 24 hrs, unless an optimal solution was obtained earlier, with the one provided by the NS-MILO approach in the time settings mentioned above.

Results of the 10-bar truss are presented in Section 4.7.1.1. Additionally, results of the cantilever and wing trusses are presented in Sections 4.7.1.2 and 4.7.1.3, respectively.

4.7.1.1 10-bar truss

Results of the 10-bar truss without Euler buckling constraints are presented in Table 4.11. As we can see, the solution of the continuous model matches that of Haftka and Gürdal [2012], and the solution of the discrete model is identical to that of Cai and Thierauf [1993], Mahfouz [1999], Camp and Bichon [2004b], Camp [2007], Barbosa et al. [2008], Sonmez [2011], and Camp and Farshchin [2014]. The full-MILO approach has solved the problem to global optimality, thus the solution obtained by the full-MILO approach is the global optimal solution. As the solution provided by the NS-MILO approach is equal to that of the full-MILO, we can conclude that the solution provided by the NS-MILO approach is also the global optimal solution of the problem for the 10-bar instance without the Euler buckling constraints.

In Table 4.12 the solution of the continuous and discrete truss sizing problem with Euler buckling constraints is presented for the 10-bar instance. Petrovic et al. [2017] state that "*there is no research found which gives buckling constrained results for 10 bar trusses*". Yet, they considered only the continuous truss design problem with Euler buckling constraints for the 10-bar instance. Although there are some

Table 4.11: Cross-sectional areas (in^2) and the weights (lbm) for the 10-bar truss problem solutions without Euler buckling constraints.

Vars.	Continuous		Discrete	
	Haftka and Gürdal [2012]	Model (P_3)	Full-MILO	NS-MILO
1	30.52	30.52	33.50	33.50
2	0.10	0.10	1.62	1.62
3	23.20	23.20	22.90	22.90
4	15.22	15.22	14.20	14.20
5	0.10	0.10	1.62	1.62
6	0.55	0.55	1.62	1.62
7	7.46	7.46	7.97	7.97
8	21.04	21.04	22.90	22.90
9	21.53	21.53	22.00	22.00
10	0.10	0.10	1.62	1.62
W	5060.85	5060.60	5490.74	5490.74

articles that consider some kind of buckling constraints [Ho-Huu et al., 2016, Rahami et al., 2008, Rajeev and Krishnamoorthy, 1997, Wu and Chow, 1995], they do not consider the buckling constraints (3.11) or (3.12). In fact, the solutions provided by these authors do not satisfy the buckling constraints (3.11) or (3.12). Regardless, the full-MILO approach finds the global optimal solution in this small instance. Thus, to check the NS-MILO approach, it is enough to compare the solution obtained by the NS-MILO approach with the one provided by the full-MILO approach. As we can see in Table 4.12, the weight of the solution generated by the NS-MILO approach is within 0.1% of that of the global optimal solution obtained by the full-MILO approach.

4.7.1.2 Single-scenario 2D and 3D cantilever trusses

Results for the single-scenario 2D and 3D cantilever trusses are presented in Tables 4.13 and 4.14, respectively. Among the 2D and 3D cantilever trusses, as it can be seen in Tables 4.13 and 4.14, within the 24 hr time limit of the full-MILO

Table 4.12: Cross-sectional areas (in^2) and the weight (lbm) of the solution of the 10-bar truss with Euler buckling constraints.

Vars.	Continuous		Discrete	
	Petrovic et al. [2017]	Model (P$_3$)	Full-MILO	NS-MILO
1	11.56	17.68	18.80	18.80
2	8.17	0.10	1.62	1.62
3	65.90	57.09	52.50	52.50
4	24.38	40.62	42.50	42.50
5	0.11	0.10	1.62	1.62
6	9.46	0.10	1.80	1.62
7	26.27	7.42	4.80	4.97
8	41.50	69.16	80.00	80.00
9	4.31	12.55	14.20	14.20
10	54.64	0.10	1.62	1.62
W	10492.80	8707.56	9400.97	9403.15

approach, only the 2D and 3D trusses with 20 bars are solved to proven optimality. Comparing the solutions obtained by the full-MILO approach for the instances that are solved to optimality with those of the NS-MILO approach indicates that the NS-MILO approach has found the global optimal solution for all those instances.

From the results shown in Tables 4.13 and 4.14, it is clear that the NS-MILO approach for the 2D and 3D cantilever instances is significantly faster, at least 10 times, than the full-MILO approach. For all the 2D truss instances, the difference between the solutions obtained by the full-MILO and NS-MILO approaches is less than 0.1%. Therefore, we can say that the NS-MILO approach is able to find equally good solutions for the 2D cantilever instances, but significantly faster than the full-MILO approach.

In all the fifteen 3D cantilever trusses, the weight of the solution obtained by the NS-MILO approach is equal to, or lower than, the weight of the solution obtained from the full-MILO approach. For instance, the weight of the solution obtained by the full-MILO approach for the 3D instances with 300 bars is 69% more than that

Table 4.13: Weights (kg) and the solution times (s) for the 2D cantilever problem instances using the NS-MILO approach.

n_b	m	Full-MILO		NS-MILO			t_f/t_n
		w_f	t_f	n_s	w_n	t_n	
4	20	9.64	155.44	7	9.64	0.56	277.57
8	40	21.42	86400.00	5	21.36	8.83	9784.82
12	60	33.52	86400.16	8	33.53	541.09	159.67
16	80	49.51	86400.21	6	49.39	287.82	159.68
20	100	68.85	86400.28	11	68.88	2390.69	36.14
24	120	80.93	86400.15	10	80.92	2543.93	33.96
28	140	110.78	86400.14	13	110.83	3035.35	28.46
32	160	152.81	86400.07	14	152.78	5287.82	16.34
36	180	179.96	86400.14	15	179.81	2712.94	31.85
40	200	219.12	86400.17	8	219.25	1691.19	51.09
44	220	274.88	86400.17	11	275.51	4273.62	20.22
48	240	324.42	86400.19	18	324.88	5027.54	17.18
52	260	382.51	86400.14	15	382.77	6414.20	13.47
56	280	437.11	86400.22	15	437.20	5741.47	15.05
60	300	452.46	86400.11	20	452.70	6588.58	13.11

of the solution provided by the NS-MILO approach.

4.7.1.3 Single-scenario wing truss problems

As it can be seen in Table 4.15, none of the discrete problems of the wing trusses were solved to optimality in 24 hrs using the full-MILO approach. We let the wing instances run longer than 24 hrs, however, for the wing instance with 81 bars, after 48 hrs, more than 64 GB of memory was used by the solver to store the nodes of the branch and bound tree, which renders the full-MILO approach inefficient for times beyond 48 hrs. This is because once the memory limit is reached, MILO solvers use the hard drive to store the branch and bound tree information, which due to the time spent on writing and accessing the hard drive, results in an extremely slow process for the solver.

Comparing the best solution provided by the full-MILO approach with the one

Table 4.14: Weights (kg) and the solution times (s) for the 3D cantilever problem instances using the NS-MILO approach.

n_b	m	Full-MILO		NS-MILO			w_f/w_n
		w_f	t_f	n_s	w_n	t_n	
1	20	9.75	247.55	4	9.75	0.51	1.00
2	40	21.24	86400.02	10	21.18	1559.91	1.00
3	60	24.82	86400.02	7	24.75	2542.68	1.00
4	80	31.22	86400.03	5	31.18	4500.35	1.00
5	100	32.04	86400.04	15	31.43	4637.38	1.02
6	120	59.34	86400.05	30	59.22	4300.05	1.00
7	140	85.50	86400.06	5	78.46	4593.84	1.09
8	160	99.64	86400.03	15	89.55	5668.90	1.11
9	180	123.99	86400.05	14	113.33	5026.34	1.09
10	200	158.74	86400.06	8	122.31	4518.66	1.30
11	220	227.43	86400.03	8	144.21	3939.17	1.58
12	240	255.32	86400.04	7	172.02	4981.69	1.48
13	260	244.69	86400.12	16	183.73	6321.59	1.33
14	280	375.48	86400.07	7	222.61	5254.40	1.69
15	300	408.42	86400.05	30	246.14	10337.08	1.66

provided by the NS-MILO approach for the wing instances in Table 4.15, we can see that the best solution obtained by the full-MILO approach is far from being optimal. Even though we stopped the full-MILO approach after one day, still the NS-MILO approach is significantly faster than the full-MILO approach. Note that the weight of the best solution obtained from the full-MILO approach is 12%-291% higher than that of the NS-MILO approach for the wing instances.

As mentioned earlier, IPOPT is used to provide a local optimal solution for the continuous truss design problem. The weight of the continuous model solution is listed in Table 4.15. We also solved the wing instances with the software package SNOPT [Gill et al., 2005], which converged to the same solutions for the continuous model as those of IPOPT. We may entertain the idea that the solutions are the global optimal solutions of the continuous model, since IPOPT and SNOPT use the interior point method and sequential quadratic programming, respectively. If the solution of the continuous model is the global optimum of the problem, then its weight provides a lower bound for the optimal solution of the discrete truss design problem. The gap between the weight of the continuous model solution and the weight of the solution provided by NS-MILO is, most of the time, less than 12% for the wing instances.

Table 4.15: Weight (kg) and the solution time (s) for the wing trusses using the NS-MILO approach.

# bars	Full-MILO		Cont. model		NS-MILO			t_f/t_n	w_n/w_c	w_f/w_n
	w_f	t_f	w_c	t_c	n_s	w_n	t_n			
81	19,166.90	86,400.02	16,454.83	8.78	42	17,147.00	230.06	375.55	1.04	1.12
99	16,361.55	86,400.02	13,208.30	11.18	41	14,106.27	189.40	456.18	1.07	1.16
117	14,732.28	86,400.02	11,797.77	33.70	59	12,994.64	2,398.43	36.02	1.10	1.13
135	14,791.77	86,400.02	10,845.89	22.99	28	11,941.59	1,794.42	48.15	1.10	1.23
153	16,384.05	86,400.03	10,145.42	274.17	64	11,090.73	7,518.98	11.49	1.09	1.48
171	15,045.04	86,400.01	9,508.32	92.78	49	10,426.03	6,570.13	13.15	1.10	1.44
207	20,527.61	86,400.12	8,656.48	166.32	81	9,657.39	11,249.02	7.68	1.12	2.13
225	21,615.53	86,400.04	8,320.28	332.60	54	9,270.02	9,350.30	9.24	1.11	2.33
243	18,696.90	86,400.13	8,170.89	246.15	23	9,127.25	6,705.11	12.89	1.12	2.05
261	25,324.62	86,400.27	7,923.76	298.48	99	8,708.22	26,799.14	3.22	1.10	2.91
279	16,821.70	86,400.05	7,833.36	874.01	68	8,756.50	25,899.99	3.34	1.12	1.92
297	23,051.34	86,400.03	7,699.78	887.11	67	8,470.54	31,538.54	2.74	1.10	2.72
315	21,016.83	86,400.24	7,590.09	1100.64	62	10,555.56	20,431.61	4.23	1.39	1.99

The objective function improvement of the full-MILO and NS-MILO approach for the wing instance with 315 bars is plotted in Figure 4.8. As we can see, the full-MILO approach improves the objective function slowly, and after 24 hrs, the weight of the best solution found is about double the weight of the solution provided by the NS-MILO approach. On the other hand, the NS-MILO approach took 1.82 hrs to find a feasible solution. The reason is that the first 25 MILO$_2$ subproblems were not able to find a feasible solution in the time limit of the MILO$_2$ subproblems, and the first feasible solution to a MILO$_2$ subproblem was found at 1.82 hrs. The weight of the initial solution obtained by the NS-MILO approach is 210,953.22 kg. The objective function improved quickly and the NS-MILO process stopped at 5.67 hrs

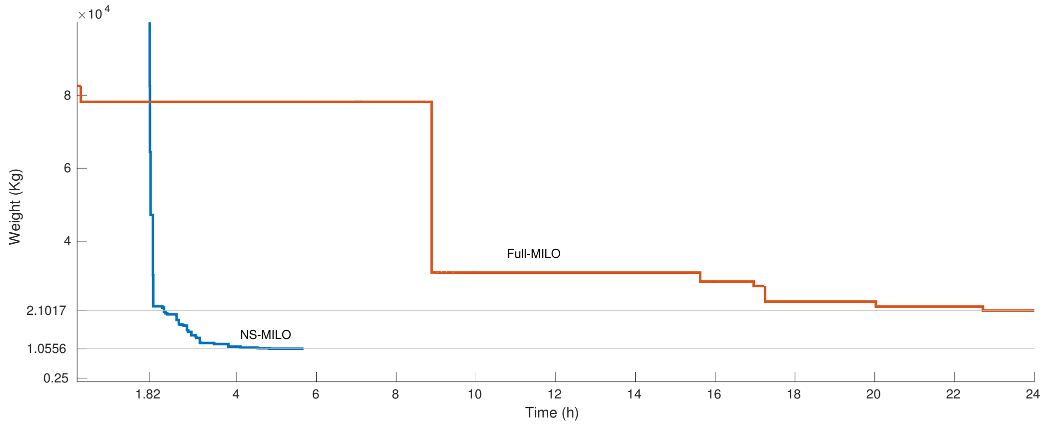
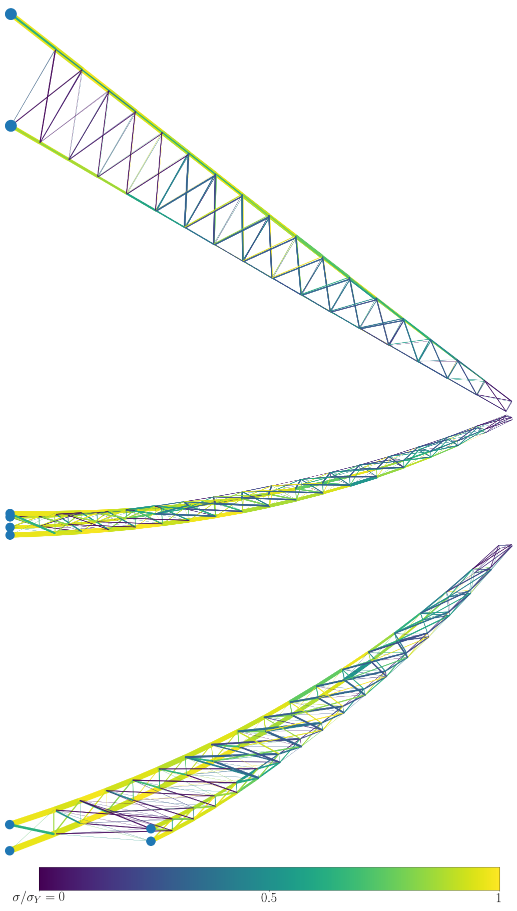


Figure 4.8: Comparison of the convergence of the full-MILO and NS-MILO approaches for the wing problem instance with 315 bars.

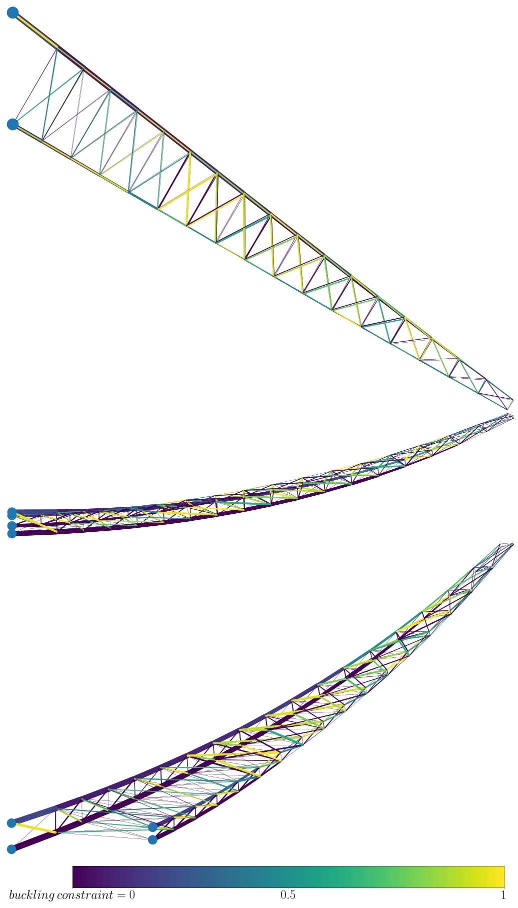
with a solution that has a weight of 10,555.56 kg. The weight of the solution provided by the NS-MILO approach is 50% of the one provided by the full-MILO approach, and is obtained in 23.6% of the time it takes for full-MILO.

To obtain a better physical intuition, in Figure 4.9 we plot the dimensionless stress, $\sigma_i/|\sigma_Y|$, and dimensionless buckling constraints, $\max(-\sigma_i/\gamma_i x_i, 0)$ (stress in bars under tension is substituted with zero for simplicity) for the solution of the wing instance with 315 bars obtained the from NS-MILO approach. The stress constraints are more active for the horizontal bars close to the root of the wing, which is the fixed end of the wing. This is because those bars are mainly responsible for taking the large bending moment around the root. As for the buckling constraints, they are more critical for the upper surface bars because these bars are under compression.

In Figure 4.10, the solution times of the NS-MILO approach for 2D cantilever, 3D cantilever, and wing instances are plotted. As we can see in this figure, the solution time of the 3D cantilever trusses are more than that of the 2D cantilever trusses with more than 160 bars, and the solution time of the wing trusses are significantly more than that of the 3D cantilever trusses. This reflects the increasing complexity



(a) Dimensionless stress distribution.



(b) Dimensionless buckling distribution.

Figure 4.9: Stress and buckling constraint distribution for the wing problem instance with 315 bars.

of the three test sets. Additionally, we can see in Figure 4.10 and Tables 4.13, 4.14, and 4.15 that the number of subproblems and the overall solution time increases only moderately as the problem size increases.

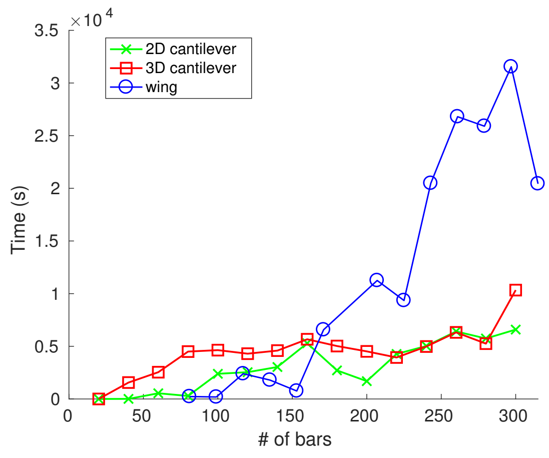


Figure 4.10: Solution time for the NS-MILO approach versus the number of bars in the truss.

4.7.2 Multi-scenario results

In this section we present computational results of solving multi-scenario truss sizing problems by the NS-MILO approach. In Section 4.7.2.1, we present the results on the well-known 72-bar truss and compare the solution obtained by the NS-MILO approach with that of other methods used in the literature to solve the problem. Additionally, to demonstrate how the NS-MILO approach scales as the size of a multi-scenario truss problem grows, we present computational results for the multi-scenario cantilever trusses and wing trusses in Sections 4.7.2.2 and 4.7.2.3, respectively.

In order to use the NS-MILO approach for multi-scenario truss sizing problems, we need to modify it as follows:

- The time limits of the MILO$_2$ subproblems are equal to ℓm seconds, where ℓ and m denote the number of scenarios and the number of bars, respectively. Additionally, the time limits of other MILO$_k$ subproblems are allocated by formula $\alpha dkm(1+\ell^2)$ seconds, where d is the dimension of the truss, and α is the coefficient representing the complexity of the structure. Coefficient α is 1 for the wing trusses and is 0.5 for the 2D and 3D cantilever trusses.

- Let (P_m) be the multi-scenario truss sizing problem, presented in model (3.14), except that we have set $x_i^{\min} = s_{i4}$, i.e., the lower bounds of the cross-sectional areas are set to the 4^{th} value of the discrete set. Let $\bar{x}_0$ be the solution of (P_m) obtained by IPOPT. The first subproblem that we attempt to solve is $\mathrm{MILO}_2(\alpha\bar{x}_0)$, where $\alpha = 1$. If the first MILO_2 subproblem is infeasible or no integer feasible solution is found within the given time limit of the MILO_2 subproblems, then we increase α by 0.1 and attempt to solve $\mathrm{MILO}_2(\alpha\bar{x}_0)$ subproblems until we find an initial integer feasible solution. We have increased $x_i^{\min}$ so as to move away from the boundaries of the feasible set of the continuous problem and help MILO_2 subproblems to find an initial integer feasible solution in the vicinity of $\alpha\bar{x}_0$ faster.

4.7.2.1 72-bar truss

A solution to the 72-bar instance that satisfies Euler buckling constraints has not been reported in the literature. Thus to benchmark the NS-MILO approach for this instance, we show in Table 4.16 the results of the 72-bar instance without Euler buckling constraints. As we can see, the weight of the solution of the continuous model (P_3) matches that of the solution by Haftka and Gürdal [2012] with 0.1% precision. Additionally, the solution of the discrete model obtained by the NS-MILO approach in 0.29 hour is the same as the ones by Kaveh and Ghazaan [2015], Sadollah et al. [2015], and Ho-Huu et al. [2016]. Note that we let the full-MILO approach run for 120 hrs for the 72-bar instance without Euler buckling constraints, and the optimality gap is still 32.61% when the full-MILO approach stops.

Furthermore, the solution provided by the NS-MILO approach for the 72-bar

Table 4.16: Solution (in^2) and the weights (lbm) for the 72-bar truss problem without Euler buckling constraints.

Vars.	Continuous		Discrete							
	Haftka and Gürdal [2012]	Model (P$_3$)	Wu and Chow [1995]	Kaveh and Talatahari [2009]	Sadollah et al. [2012]	Kaveh and Ghazan [2015]	Sadollah et al. [2015]	Ho-Huu et al. [2016]	Full-MILO	NS-MILO
1-4	0.157	0.156	0.196	0.196	1.800	0.196	0.196	0.196	0.196	0.196
5-12	0.536	0.546	0.602	0.563	0.602	0.563	0.563	0.563	0.563	0.563
13-16	0.410	0.410	0.307	0.442	0.111	0.391	0.391	0.391	0.442	0.391
17-18	0.569	0.570	0.766	0.563	0.111	0.563	0.563	0.563	0.602	0.563
19-22	0.507	0.524	0.391	0.563	1.266	0.563	0.563	0.563	0.785	0.563
23-30	0.520	0.517	0.391	0.563	0.563	0.563	0.563	0.563	0.563	0.563
31-34	0.100	0.100	0.141	0.111	0.111	0.111	0.111	0.111	0.111	0.111
35-36	0.100	0.100	0.111	0.250	0.111	0.111	0.111	0.111	0.111	0.111
37-40	1.280	1.268	1.800	1.228	0.442	1.228	1.228	1.228	1.000	1.228
41-48	0.515	0.512	0.602	0.563	0.442	0.442	0.442	0.563	0.563	0.563
49-52	0.100	0.100	0.141	0.111	0.111	0.111	0.111	0.111	0.111	0.111
53-54	0.100	0.100	0.307	0.111	0.111	0.111	0.111	0.111	0.111	0.111
55-58	1.897	1.886	1.563	1.800	0.196	1.990	1.990	1.990	1.990	1.990
59-66	0.516	0.512	0.766	0.442	0.563	0.563	0.563	0.442	0.442	0.442
67-70	0.100	0.100	0.141	0.141	0.442	0.111	0.111	0.111	0.111	0.111
71-72	0.100	0.100	0.111	0.111	0.602	0.111	0.111	0.111	0.111	0.111
W (lbm)	379.66	379.61	427.20	393.38	390.73	389.33	389.33	389.33	392.96	389.33

truss with Euler buckling constraints is presented in Table 4.17. As we can see, the solution provided by the NS-MILO approach is better than that of the full-MILO approach. Note that the NS-MILO approach provided the solution in 27 s, while the full-MILO approach produced the solution in 24 hrs.

4.7.2.2 Multi-scenario 2D and 3D cantilever trusses

Results of the 2D cantilever trusses with two scenarios are presented in Table 4.18. As we can see, the solutions obtained by the NS-MILO approach are equal to the ones obtained by the full-MILO approach for the instances with 20 and 60 bars. In all the trusses with more than 60 bars, the solution provided by the NS-MILO approach is better than the one provided by the full-MILO approach. For instance, the solution provided by the NS-MILO approach for the 2D cantilever truss with 300 bars is %15 lighter than the solution obtained by the full-MILO approach. Furthermore, the NS-MILO approach was able to get the solutions significantly faster than the

Table 4.17: Cross-sectional areas (in^2) and the weights (lbm) for the 72-bar truss problem solutions with Euler buckling constraints.

Vars.	Continuous	Discrete	
		Full-MILO	NS-MILO
1–4	1.470	1.457	1.457
5–12	2.283	2.380	2.380
13–16	1.649	1.990	1.620
17–18	2.774	2.630	2.880
19–22	1.498	1.563	1.563
23–30	1.776	1.800	1.800
31–34	0.100	0.196	0.196
35–36	0.330	0.785	0.442
37–40	1.518	1.563	1.620
41–48	1.933	1.990	1.990
49–52	0.482	0.391	0.442
53–54	0.825	0.602	0.766
55–58	2.084	1.990	1.800
59–66	1.906	1.990	1.990
67–70	0.321	0.391	0.442
71–72	0.100	0.111	0.111
W	1264.750	1316.148	1302.500

full-MILO approach in all the 2D cantilever trusses.

Table 4.18: Weights (kg) and the solution times (s) for the two-scenario 2D cantilever instances using the NS-MILO approach.

n_b	m	ℓ	Full-MILO		NS-MILO			w_f/w_n
			w_f	t_f	n_s	w_n	t_n	
4	20	2	10.31	281.31	9	10.31	3.66	1.00
12	60	2	37.15	86,400.03	9	37.15	1,553.20	1.00
20	100	2	74.20	86,400.07	14	73.95	6,221.25	1.00
28	140	2	119.08	86,400.02	23	119.01	7,309.91	1.00
36	180	2	259.68	86,400.08	10	191.80	7,755.97	1.35
44	220	2	321.42	86,400.11	16	288.22	13,616.79	1.12
52	260	2	518.21	86,400.13	22	397.79	13,684.09	1.30
60	300	2	558.01	86,400.02	19	471.61	23,177.21	1.18

Results of the 3D cantilever trusses with two and three scenarios are presented in Table 4.19. As we can see, the solution provided by the NS-MILO approach for the 3D cantilever truss with 20 bars is the same as th one provided by the full-MILO approach. In all the 3D cantilever trusses with more than 20 bars, the solution obtained by the NS-MILO approach is better than the one provided by the full-MILO approach, and the gap between the weight of the solution provided by the NS-MILO and full-MILO approaches increases rapidly as the size of the problem grows. For instance, the weights of the solutions obtained by the NS-MILO approach for the two-scenario and three-scenario 3D cantilever truss with 300 bars are only %15.8 and %16.3 of the one generated by the full-MILO approach, respectively. In other words, the solution obtained by the full-MILO approach is more than six times heavier than the solution obtained by the NS-MILO approach for the 3D cantilever truss with 300 bars. Except for the three-scenario cantilever with 220 bars, the NS-MILO approach required significantly less time than the 24 hrs allocated to the full-MILO approach.

4.7.2.3 Multi-scenario wing trusses

Results of the 2-scenario and 3-scenario wing trusses are presented in Table 4.20. As we can see, the solutions provided by the NS-MILO approach are significantly better than the ones obtained by the full-MILO approach, and the difference between the weights of the solutions provided by the NS-MILO and full-MILO approaches increases as the size of the wing truss increases. For the 2-scenario and 3-scenario wing truss with 315 bars, solutions provided by the full-MILO approach are 9.61 and 8.91 times heavier than the ones obtained by the NS-MILO approach. The primary reason that full-MILO fails in solving large-scale multi-scenario wing trusses is that solving the continuous relaxation problems at the nodes of the branch and bound tree takes excessive time.

Observe that, within the 24-hour time budget, the number of nodes explored by

Table 4.19: Weights (kg) and the solution times (s) for the multi-scenario 3D cantilever instances using the NS-MILO approach.

n_b	m	ℓ	Full-MILO		NS-MILO			w_f/w_n
			w_f	t_f	n_s	w_n	t_n	
1	20	2	13.47	86400.01	15	13.47	41.33	1.00
		3	14.28	86400.00	13	14.28	1317.91	1.00
3	60	2	41.07	86400.01	13	33.62	5424.24	1.22
		3	39.63	86400.01	8	37.62	7521.81	1.05
5	100	2	73.77	86400.05	13	55.74	5623.09	1.32
		3	105.42	86400.02	18	62.38	17388.69	1.69
7	140	2	199.50	86400.03	23	100.73	16165.49	1.98
		3	202.93	86400.02	24	108.72	37528.54	1.87
9	180	2	345.11	86400.02	16	139.78	19032.60	2.47
		3	375.75	86400.04	20	162.43	47978.13	2.31
11	220	2	549.47	86400.06	11	179.00	20693.89	3.07
		3	701.82	86400.05	20	192.46	121163.95	3.65
13	260	2	714.18	86400.06	9	225.08	17297.92	3.17
		3	978.16	86400.06	11	243.02	48180.25	4.03
15	300	2	1847.84	86400.01	10	293.33	26139.16	6.30
		3	1970.40	86400.08	11	322.91	48251.02	6.10

Gurobi in the B&B algorithm in the full-MILO approach decreases rapidly as the size of the problem grows and as the number of the scenarios increases. For instance, for the 2-scenario wing truss with 315 bars, Gurobi, which is set to use 10 threads, has explored only 288 nodes, implying that on average the continuous relaxation at each node is solved in 3,000 s. For the 3-scenario wing trusses with 279 and 315 bars, Gurobi was not able to solve the continuous relaxation at the root node in 24 hrs. The solutions reported by Gurobi for these two problems are obtained by the heuristics that the solver utilizes to generate initial integer feasible solutions.

Table 4.20: Weights (kg) and the solution times (s) for the multi-scenario wing trusses using the NS-MILO approach.

m	ℓ	Full-MILO			NS-MILO			w_f/w_n
		B&B nodes	w_f	t_f	n_s	w_n	t_n	
81	2	213,902	24,953.10	86,400.01	74	18,730.90	17,120.41	1.33
	3	22,952	45,989.98	86,400.04	70	19,334.45	21,479.01	2.38
117	2	39,808	47,131.95	86,400.05	46	14,246.51	36,734.13	3.30
	3	600	81,981.44	86,400.02	44	14,354.08	50,541.50	5.71
153	2	29,339	43,995.13	86,400.07	19	12,104.99	23,940.06	3.63
	3	20	74,600.89	86,400.32	41	12,306.48	83,625.31	6.06
207	2	6,510	62,157.36	86,400.06	31	10,399.04	47,733.30	5.98
	3	0	67,885.71	86,400.04	39	10,466.81	108,493.33	6.49
243	2	1,009	72,658.38	86,400.06	26	10,228.69	86,057.75	7.10
	3	15	72,658.38	86,400.07	42	10,033.00	180,992.04	7.24
279	2	181	86,170.94	86,400.03	22	9,661.80	67,236.63	8.92
	3	0	86,170.94	86,400.28	26	9,984.50	191,646.64	8.63
315	2	288	91,718.83	86,400.11	16	9,545.63	82,027.25	9.61
	3	0	91,718.83	86,400.17	31	9,742.68	466,123.89	9.41

4.8 Conclusions

We presented novel MILO models for the discrete truss design problems that include both the Euler buckling and the Hooke's law constraints. We also proposed the NS-MILO methodology to provide high-quality solutions in a reasonable time for those problems, where a sequence of MILO subproblems in a moving neighborhood search framework are explored. The new methodology enables us to provide high-quality solutions in a reasonable time for previously unsolvable truss design problems.

Computational experiments with the single-scenario and multi-scenario truss design problems indicate that the NS-MILO approach is significantly faster than the full-MILO approach. Furthermore, the NS-MILO approach obtains significantly better solutions for large scale truss design problems. Specifically, the weight of the solutions provided by the NS-MILO approach for large-scale multi-scenario wing design problems has %15 of the weight of the solution obtained by the full-MILO approach.

Chapter 5

The Inmate Assignment and Scheduling Problem

5.1 Introduction

According to the International Center for Prison Studies, the U.S. incarcerates 698 people for every 100,000 of its population. Despite accounting for approximately 4.5% of the world's population, the U.S. has 21.4 % of the world's incarcerated population [Walmsley, 2017]. In 2010, all levels of government in the U.S. spent more than \$80 billion on corrections [Kyckelhahn and Martin, 2010], implying \$260 tax burden for each U.S. resident. Adjusted to inflation, the expenditures on corrections in 2010 are more than three times of that in 1979 [Schanzenbach et al., 2016].

Due to insufficient capacity of the correctional institutions (CIs), there is a growing problem of *overcrowding* in the CIs. Population management of the inmates is one of the most critical operations within a correctional system, and requires significant monetary and human resources. Efficiently managing the inmate population results in huge savings. Appropriate assignment of the inmates to the CIs is a key element of population management, which can lead to significant savings, as well as enhancing public safety and security of the CIs.

When a court delivers a sentence, the inmate often receives a list of treatment programs based on the various assessments, including the crime committed. Research shows that inmates who complete the programs, offered by the CIs, have lower recidivism rate [Davis et al., 2013]; hence, programs have the capability of saving CI capacity and promoting a safe and healthy society. Inmates usually are given a *minimum sentence length* in "indeterminate sentencing states" like Pennsylvania. Having served the minimum sentence length, they are eligible to be conditionally released, also known as *parole*, if they satisfy all of the parole requirements. *One of the parole requirements is to complete all the required treatment programs.* Overcrowding of CIs adversely affects the way inmates receive their treatment programming and delays scheduling as the resources for the programs are limited. Inmates who receive timely programming have a better chance of becoming eligible for parole and leaving the correctional institution earlier, thereby reducing the population of the CIs.

In 2015, the PADoC had a staggering \$2.15 billion in expenditures to house 50,366 inmates [Mai and Subramanian, 2017]. All inmates, who enter the correctional system, have their own programming needs and special requirements. Often, a CI can offer only certain programs as it has only limited personnel and infrastructure resources and so might not be able to meet the needs of all inmates. We briefly describe the inmate assignment process before this project started. Each new inmate would be assigned to CIs, *manually*, by a staff member of the Office of Population Management (OPM). Numerous factors, i.e., rules and criteria, are considered in assigning inmates to CIs, including but not limited to, security concerns, mental and medical conditions, program needs, separation from other inmates, capacities of the CIs, and home county of the inmates. Having to consider all the factors for the assignment of each inmate, individually, is time-consuming and prone to human errors. Additionally, when inmate assignment is done individually and *sequentially*, inmates assigned later are not considered in the current assignment. This greedy sequential assignment of inmates to CIs makes the process highly inefficient, and results in numerous violations of the factors, or the capacity constraints, or both.

The optimal inmate assignment project in collaboration with the PADoC, spanned five years from idea to successful implementation. The main goal of the project is to develop an Inmate Assignment Decision Support System (IADSS) for the PADoC, which *simultaneously* assigns the inmates to CIs and schedules the treatment programs for the inmates, while all the factors and criteria of the assignment are considered. The IADSS is comprised of a user-friendly web based interface, which is linked to the PADoC databases, and an optimization engine which does the assignment of the inmates to CIs.

The goal of the IAP is to optimize inmate assignments, transfers, and program scheduling, while numerous restrictions and constraints are considered to advance the following objectives:

- reduce the total population of inmates at the CIs,
- minimize inmate movements during prison terms,
- reduce treatment services waiting lists.

5.1.1 Literature Review

The IAP is a novel class of the *assignment problem* [Flood, 1953, Votaw and Orden, 1952] with several side constraints. The classic assignment problem and algorithms to solve it have been extensively studied in the 50s [Dantzig, 1951, Orden, 1951]. Kuhn [1955] suggests the well-known Hungarian method for solving the assignment problem. Assignment models have been used in a large variety of applications of optimization. For instance, crew scheduling is a broadly-used problem class using generalized assignment models. Airline crew scheduling is one of the most important crew scheduling problems that received attention within the optimization community in the 60s [Arabeyre et al., 1969] and it has been extensively studied since then. Furthermore, Caprara et al. [1998] have used the assignment model for crew scheduling in the railway industry. To the best of our knowledge, the only OR paper for the IAP is by Li et al. [2014], who studied the inmate assignment process and developed a decision tree representing all the factors of the inmate assignment to CIs.

5.1.2 Contributions: Novel Modeling and Solution Methodology

The IAP mainly revolves around the assignment of inmates to the CIs and scheduling of programs for the inmates at the CIs. In order to develop a mathematical optimization model, all the processes of the inmate assignment were mapped and formalized, which in fact was a challenging process, because there are no OR experts at the PADoC, nor to the best of our knowledge at DoCs elsewhere today. Due to scarce resources and often conflicting rules, the IAP is inherently an infeasible problem. In order to address the need for simultaneous system-wide optimization of inmate assignments, while considering all the conflicting factors, we developed and fine-tuned a *hierarchically weighted multi-objective MILO model*. In conjunction with model development, data collection and preparation procedures, which interface with the DoC database systems, have been developed. Ultimately, the web-based IADSS was developed which enables the user to make optimal decisions in a fraction of the time needed before. Since September 2016, the integrated IADSS has been in daily use by PADoC. The IADSS makes the assignment process efficient, while significantly improving the quality and consistency of the assignments. These goals are achieved by advanced optimization modeling of system-wide assignment and scheduling needs, and the use of state of the art optimization methodology.

5.1.3 Impact

IADSS enables the PADoC to have high-quality consistent assignments, which also increases security and reduces violence. IADSS has resulted in cost savings by reducing the population of the inmates and the number of transfers between the CIs. It has also enabled the PADoC to reduce the staff needed for making assignments, and it has led to a smaller number of assaults in the CIs. As a result of using the IADSS for the assignment of inmates, the PADoC has saved \$2.9 million in the first year, and it is expected to reduce the cost by \$19.8 million over 5 years.

The broader impact of this project, and the highly successful development of the IADSS is that it can be adapted and used in the correctional systems of other states and countries. Thus, this project, and the developed solution methodology, is opening a new, high impact area for the application of OR and analytics methodologies.

This paper is structured as follows. Section 5.2 presents the IAP and the numerous factors and program scheduling requirements which define the IAP. Modeling and solution methodology details are presented in Section 5.3. The multi-objective MILO model for simultaneously assigning the inmates to the CIs is presented in Section 5.4. Section 5.5 presents the development of the IADSS, and the implementation at the PADoC. We list and quantify the benefits of using IADSS in Section 5.6, and Section 5.7 presents the summary of the paper.

5.2 Preliminaries and Problem Description

In this section, we discuss the preliminary developments at the PADoC and we formally define the IAP and elaborate the rules and criteria used for inmate assignment to the CIs.

5.2.1 Preliminary Development

We now discuss the developmental evolution of the model and the web based IADSS at the PADoC. When the project started, we discussed with PADoC the need for a decision support system to assist the OPM in assigning the inmates to the best possible CI, considering both the needs and limitations of the inmates and the available limited resources of the PADoC. This is a complex problem where ideal assignment of all inmates is not possible. Inmate-specific factors are a combination of several categories such as medical, psychological, educational, custody level, and sentence conditions. On the other hand, CIs have numerous limitations, such as security level, treatment programs availability, and capacity.

Conventionally, the assignment process has been manual and subjective, where a staff member with the provided information of the inmate and the CIs from the PADoC database assigns the inmates one-by-one to the CIs. While the general guidelines for the assignment are known, the large number of factors, the daily changing capacities of the CIs, and the subjective nature of this sequential ad-hoc assignment made the efficiency and quality of the assignment heavily dependent on the experience and judgment of the staff. In order to remove the subjective component of the assignment, initially we developed a decision-tree based decision support system (DTDSS) to reduce bias and variability in assignments, while improving adherence to the guidelines. The DTDSS provided the DoC with a ranked order of the CIs for a particular inmate from which the staff member can choose the assignment. This eliminated much of the tedious work of evaluating various combinations of factors, thus, freeing staff to use their experience to choose from a smaller subset of the most suitable CIs.

Figure 5.1 illustrates the decision tree of the DTDSS. The development and use of the decision tree in the DTDSS was critical in classifying and refining all the relevant factors and their importance level in inmate assignment. After discussing with PADoC personnel the factors which influenced the inmate assignment in detail, we identified and incorporated 60 of the most important factors used in the manual assignment procedure. The DTDSS uses these factors and rules to evaluate and, subsequently, rank the CIs with respect to their suitability for the inmate being assigned. DTDSS assigns weights and penalties for each factor, and the accumulated penalties are used to rank the CIs for the inmates.

This approach could conceivably have been deemed sufficient, while clearly not optimal, if inmates were arriving to the system in a sequence (one by one). The greedy assignment strategy embodied in the sequential application of DTDSS cannot adequately anticipate the bottlenecks in the CIs, several assignments into the future. When a batch of inmates need assignment, there is an opportunity to make resource tradeoffs performing the batch assignment that is not present in the sequential approach. In a sequential assignment, the sequence of the inmates is critical and significantly affects the succeeding assignments. The need for system-wide, simultaneous assignment made clear the need for a multiple-objective optimization model which treats all the inmates needing assignment and considers the current state of all the CIs, simultaneously, from a system's perspective.

5.2.2 Assignment Criteria

In this section, we present the essential elements of the inmate assignment problem. First, we give a brief description of the inmate assignment process. Inmates are evaluated and classified at “intake CIs”. Each period, the accumulated inmates have to be assigned to CIs, while all restrictions and constraints need to be taken into account. This is the basic inmate initial assignment problem. The map of PA with the 25 currently running CIs of the PADoC are shown in Figure 5.2. A crucial feature of the inmate initial assignment problem is that inmates need to go through individually specified programs, which are scheduled according to specific rules and requirements. Furthermore, there are a variety of reasons leading to inmate transfers from their initially assigned CI to another one. The need for this transfer of inmates further complicates the problem. Next we explain the criteria that need to be taken into account at the initial assignment of inmates to the CIs.

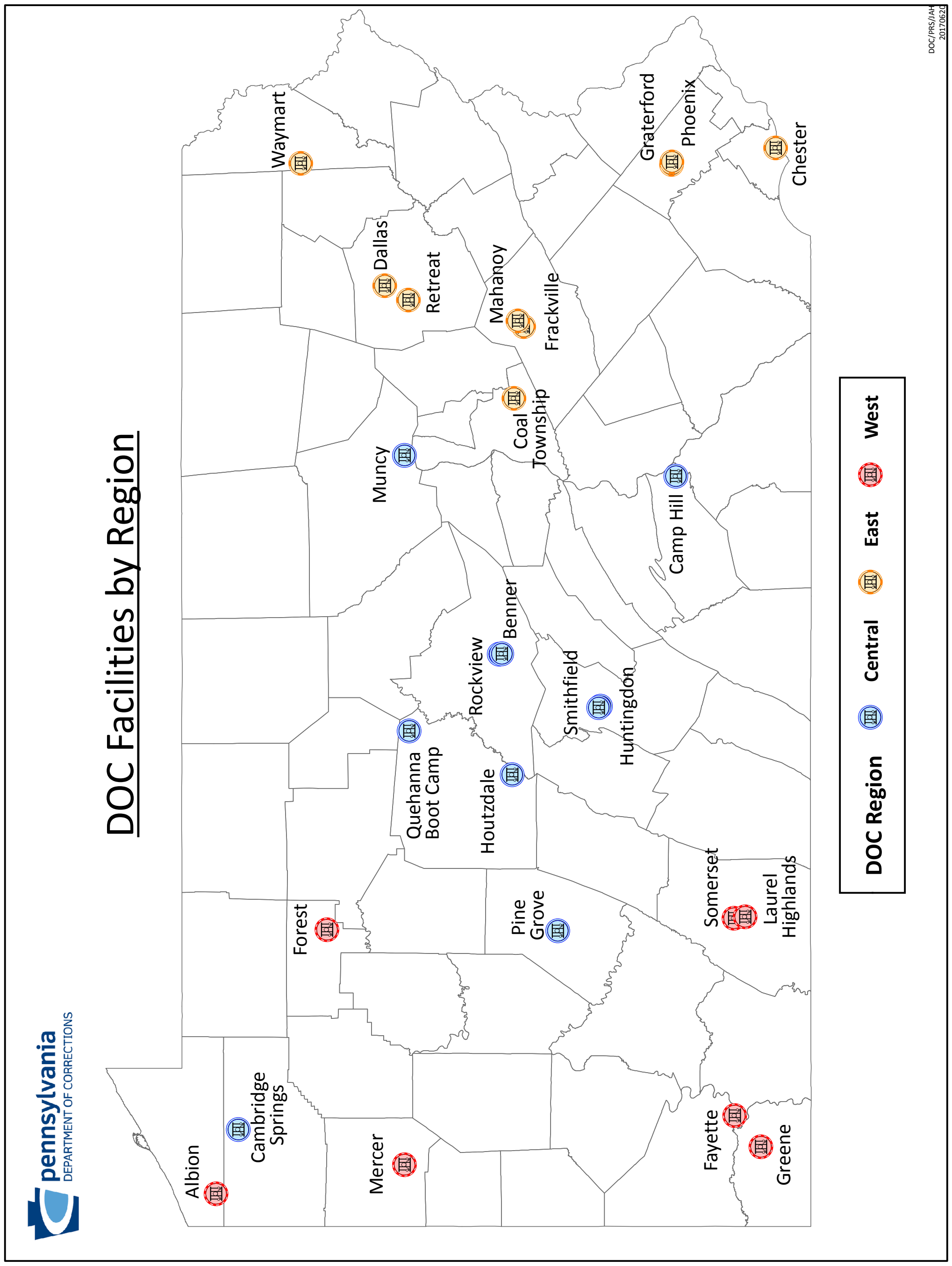


Figure 5.2: The 25 state CIs of the PADoC and their placement in one of the three main regions of the state.

General factors: There are a variety of factors, that have to be satisfied at initial inmate assignment, including but not limited to,

- *High risk inmates* have to be assigned to a predefined set of CIs.
- *Young adult offenders* should be assigned to a predefined set of CIs.
- Inmates with mental health issues should be assigned to specified CIs.
- Inmates serving a *life sentence* have to be assigned to a predefined set of CIs.
- CIs are gender specific; thus, inmates have to be assigned accordingly.

Available beds: The number of available beds for each CI is determined prior to assigning the inmates. At least a minimum number of inmates, which is a function of the available beds, should be assigned to each CI in order to properly and proportionally utilize bed spaces. Additionally, for each CI, the maximum number of inmates, which is again a function of the available beds, is specified to avoid creating long lists of inmates waiting for beds to become available at the CIs. Furthermore, the number of inmates assigned to the CIs should be *proportional to the available beds* when it is in the minimum and maximum range.

Home county: Inmates need to be assigned to a *CI near their home county.*

Separations: Considering previous inmate-inmate and inmate-staff conflicts, some inmates cannot be assigned to certain CIs. Additionally, there might be pairs or groups of inmates, waiting to be assigned, that *cannot be assigned to the same CI.*

5.2.3 Treatment Programs

Inmates usually are given minimum sentence length, i.e., the minimum time they have to stay in CIs, and they have a scheduled parole board interview before their minimum sentence date. To be eligible for parole, they need to satisfy all of the requirements of their sentences. One of the requirements is to complete all of their treatment programs before the parole board interview. Treatment programs are prescribed by the court, or by the correctional system.

Ideally, inmates should be assigned to a CI which can offer their program(s) before their parole board meeting. However, due to limited capacity of the programs at CIs, not all the inmates are able to finish their program(s) before their parole board meeting. This results in creating *inmate waiting lists* for the programs at the CIs, which provides one of the most important criteria in the IAP. Furthermore, inmates can start their programs only within the 24-month window before their minimum sentence date.

Programs can either be *open-enrollment* or *closed-enrollment*. In an open enrollment program, enrollments can happen any time. If an inmate completes an open program, the next inmate can start that program immediately. However, in a closed-enrollment program, a group is identified and they all start and complete the program at the same time.

The number of inmates that start an open-enrollment program at time t is driven by the number of open spots of that program at time t. However, the number of inmates that can start a closed-enrollment program at time t is driven by the number of groups of that program that can start at time t. There is a minimum and maximum for the number of inmates that can be enrolled in a group for each of the closed-enrollment programs.

Another concept which is important in handling the program waiting lists is *clusters*. A cluster is a group of closed-enrollment programs that have *common instructors*, i.e., an instructor can handle all the programs in a cluster and it needs to be determined which program(s) the given instructor runs at a given time. Notice that clusters are only defined for closed-enrollment programs.

One of the main goals of the IAP is to ensure that inmates start their programs as soon as possible. This goal is formalized as minimizing the maximum waiting time of the inmates for starting their required program(s). To reach this goal we schedule the programs for the incoming inmates, while considering the limited available resources of the CIs and the inmates that are already in the CIs.

5.2.4 Transfer Constraints

After the initial assignment, some of the inmates need to be transferred. Some of the reasons for transfers after the initial assignment are as follows

Parole violator: Inmates who are released on parole and have violated their parole terms are brought back to a "parole intake facility" and need to be assigned to a CI afterwards.

Program placement: It may happen that the CI, to which an inmate is initially assigned, does not have all the inmate's required programs. Additionally, treatment programs may be prescribed after the initial assignment and some programs might not be available in the current CI. In these cases, the inmate should be moved to a CI where all the required programs are offered.

Incentive based transfers: Satisfying specific predefined requirements, inmates can request to be moved to other CIs.

Separation: Separation of an inmate from other inmates or from DoC staff can lead to a transfer request.

Constraints and restrictions for transfer placements are the same as the ones explained in Section 5.2.2 for the initial assignment of the inmates. However, the importance of the factors for a transfer placement might differ from those of an initial assignment.

5.3 Modeling and the Solution Methodology

As it was explained in Section 5.2, one of the main goals of the IAP is to assign the inmates to CIs. However, it is not a basic assignment problem, since there are a variety of factors that need to be considered in the assignment of each inmate. General factors, elaborated in Section 5.2.2, need to be satisfied in the assignment of each inmate. We not only need to satisfy the bound constraints on the number of inmates that can be assigned to each CI, but we also need to assign the inmates in proportion to the capacities of the CIs. Another criterion is that inmates should be assigned to CIs that are nearest to their home county. Furthermore, we need to schedule the required programs for the inmates, which brings a scheduling component to the IAP. Due to limited availability of resources and the conflicting rules of the assignment, it is impossible to make an ideal assignment and perfectly satisfy all the factors and program scheduling needs in the assignment of a batch of inmates.

In order to address all the conflicting factors of the assignment, we developed a *hierarchically weighted multi-objective MILO model*. As the problem is inherently infeasible, we allow the violation of the factors, and penalize the violations according to their importance. To do so, we define a weight for each factor of the assignment, which represents the importance of the factor in the assignment process. The violations of the factors are hierarchically weighted according to their importance, and the sum of the hierarchically penalized violations serve as the objective function of the MILO model. The mathematical model is presented in detail in the appendix.

The optimization software package Gurobi (2016) was used to solve the MILO models. Having developed the MILO model, it was extensively tested with various real data sets from PADoC with the goal of specifying and fine-tuning the weights of each of the factors, and ensuring the robustness of the model in recommending appropriate simultaneous assignments and program scheduling.

It is worth mentioning that, any time we solve the MILO model and schedule the programs, not everybody who is going to start the programs in the given time horizon is currently in the system. For instance, inmates with short sentence times who need immediate program enrollment enter the correctional system every week. Thus, there is a lot of freedom in scheduling the programs for periods towards the end of the time horizon. As a result, the MILO model has many equally good solutions. This in turn increases the solution time, since a significant amount of time need to be spent to prove optimality. Knowing that proving optimality requires excessive amount of time, we stop the MILO solver when the absolute optimality gap reaches a predefined threshold.

5.4 Hierarchical Multi-Objective MILO Model

In this section, we present a MILO model for the IAP. We first explain the assignment and the treatment program constraints, and finally the objectives of the problem.

5.4.1 Assignment Criteria Constraints

Let $\mathcal{I}$ be the set of inmates waiting to be assigned and let $\mathcal{J}$ be the set of the available CIs for the assignment. Each inmate should be assigned to one facility, i.e.

$$\sum_{j \in \mathcal{J}} x_{ij} = 1 \quad \forall i \in \mathcal{I},$$

where x_{ij}, for all $i \in \mathcal{I}$ and $j \in \mathcal{J}$ is a binary variable and is equal to 1 if inmate i is assigned to facility j. Let $\mathcal{K}$ be the set of general factors, and let coefficient κ_{ik} for $i \in \mathcal{I}$, $k \in \mathcal{K}$ be equal to 1 if factor k applies to inmate i; and equal to 0 otherwise. Additionally, for all $j \in \mathcal{J}$, $k \in \mathcal{K}$ let ρ_{jk} be equal to 1 if facility j can accommodate inmates with factor k; otherwise, $\rho_{jk} = 0$. The following constraints describe the general-factors violations of inmates.

$$\kappa_{ik}\Big(1 - \sum_{j \in \mathcal{J}} \rho_{jk} x_{ij}\Big) = v_{ik} \quad \forall\ i \in \mathcal{I}, \forall\ k \in \mathcal{K},$$

where v_{ik} indicates the violation of factor k by inmate i and is equal to one if inmate i violates factor k; otherwise, v_{ik} is equal to zero. Furthermore, we have capacity related constraints:

$$\sum_{i \in \mathcal{I}} x_{ij} = s_j \quad \forall j \in \mathcal{J},$$

where s_j, $j \in \mathcal{J}$ denotes the number of the inmates that are assigned to facility j. Let c_j be the capacity of facility j. Ideally, for each pair j_1 and j_2 of CIs, we want to assign inmates proportional to their capacities, i.e., ideally we would have

$$c_{j_1}/s_{j_1} = c_{j_2}/s_{j_2}.$$

Variables $\delta^+_{j_1 j_2}, \delta^-_{j_1 j_2}$ are the decision variables representing the deviation from assigning inmates proportional to the capacities of CIs j_1 and j_2 and are defined as

$$c_{j_2} s_{j_1} - c_{j_1} s_{j_2} = \delta^+_{j_1 j_2} - \delta^-_{j_1 j_2} \quad \forall j_1, j_2 \in \mathcal{J}, j_1 \neq j_2. \tag{5.1}$$

We aim to minimize $\delta^+_{j_1 j_2}, \delta^-_{j_1 j_2}$ by penalizing them in the objective function.

Additionally, we define upper and lower bounds on the number of inmates that can be assigned to each facility. Let $c_j^{\min}$ and $c_j^{\max}$ be, respectively, the minimum required and maximum allowed capacity of facility j, which are functions of the capacity c_j of facility j. For instance, $c_j^{\min} = \zeta_j^- c_j$ and $c_j^{\max} = \zeta_j^+ c_j$ for appropriately chosen constants $\zeta_j^- \leq 1 \leq \zeta_j^+$. Let o_j be the number of inmates assigned over the maximum capacity of facility j and let u_j be the number of inmates needed to reach the minimum capacity of facility j. We have

$$s_j \leq c_j^{\max} + o_j \qquad \forall j \in \mathcal{J},$$
$$s_j \geq c_j^{\min} - u_j \qquad \forall j \in \mathcal{J}.$$

We aim to minimize o_j and u_j by penalizing them in the objective function.

Another important criterion for the inmate assignment is the *separations*. Considering the history of inmates, there might be pairs of inmates that can not be assigned to the same facility. Let $\mathcal{I}^s$ be the set of inmate pairs that should be separated from each other. Additionally, an inmate might have already on his/her file that he/she has to be separated from certain staff or inmates that are already in a facility. Let $\mathcal{J}_i^s$ be the set of CIs that inmate i should be separated from. We have

$$\begin{aligned} &\sum_{j \in \mathcal{J}_i^s} x_{ij} = 0 && \forall i \in \mathcal{I}, \\ &x_{i_1 j} + x_{i_2 j} \leq 1 && \forall (i_1, i_2) \in \mathcal{I}^s. \end{aligned}$$

5.4.2 Treatment Program Constraints

Next, we explain the constraints needed to describe the waiting lists of the programs at the CIs. Let $\mathcal{P}^o, \mathcal{P}^c$ be, respectively, the set of open-enrollment and closed-enrollment programs, and let $\mathcal{C}$ be the set of program clusters. Let $\hat{t}_{ip}$ and $\bar{t}_{ip}$ be, respectively, the latest time and earliest time that inmate i is supposed to start program p, and let $\alpha_{ipt} = 1$ if $t \geq \hat{t}_{ip}$, otherwise $\alpha_{ipt} = 0$, i.e., inmate i should not start program p later than t if $\alpha_{ipt} = 1$. Similarly, $\beta_{ipt} = 1$ if $t \geq \bar{t}_{ip}$, otherwise $\beta_{ipt} = 0$, i.e., inmate i can start program p at time t if $\beta_{ipt} = 1$.

We would like to minimize the number of inmates that can not start their programs earlier than their latest start time $\hat{t}_{ip}$. The decision variable y_{jpt} represents the number of inmates at facility j that are prescribed program p and have to start it by time t but can not do so. We aim to minimize y_{jpt} by penalizing it in the objective function.

Let $\mathcal{T} = \{1, 2, \ldots, t'\}$ be the set of the time periods in our decision horizon. Parameter t' is the last time period in the decision horizon, and let ψ_{jpt}, $\underline{q}_{jpt}$, and $\overline{q}_{jpt}$ be defined as

ψ_{jpt}: The number of inmates starting program p at t in facility j.

$\underline{q}_{jpt}$: The number of inmates, already in facility j, that should start program p at time t or earlier, i.e., the number of inmates with $\hat{t}_{ip} \leq t$

$\overline{q}_{jpt}$: The maximum number of inmates, already in facility j, that can start program p at time t, i.e., the number of inmates with $\bar{t}_{ip} \leq t$.

The following two sets of constraints compute the lower and upper bound on the number of inmates that can start the programs at each time period in the CIs

$$\sum_{i\in\mathcal{I}}\alpha_{ipt}x_{ij} + \underline{q}_{jpt} \leq y_{jpt} + \sum_{\tau=0}^{t}\psi_{jp\tau} \qquad \forall j \in \mathcal{J}, \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$\sum_{i\in\mathcal{I}}\beta_{ipt}x_{ij} + \overline{q}_{jpt} \geq y_{jpt} + \sum_{\tau=0}^{t}\psi_{jp\tau} \qquad \forall j \in \mathcal{J}, \forall p \in \mathcal{P},\ \forall t \in \mathcal{T}.$$

Let R_{jpt} be the number of available spots for open-enrollment program p at time t in facility j. The following constraints assure that the number of inmates starting an open-enrollment program does not exceed the number of spots available for that program at the CIs

$$\sum_{\tau=\max(0,t-d_p)}^{t}\psi_{jp\tau} \leq R_{jpt} \quad \forall j \in \mathcal{J}, \forall p \in \mathcal{P}^o,\ \forall t \in \mathcal{T},$$

where d_p is the duration of program p.

Next, we explain the constraints related to the closed-enrollment programs. As mentioned previously, closed-enrollment programs are categorized in clusters. All the programs in a cluster can be facilitated by one instructor, i.e., programs in a cluster use common instructors. Let R'_{jct} and ψ'_{jpt} be defined as

ψ'_{jpt}: The number of groups of the closed program p that start at time t in facility j.

R'_{jct}: The number of available groups of cluster c that can start at time t in facility j.

Then we have

$$\sum_{p\in\mathcal{P}_c}\sum_{\tau=\max(0,d_p)}^{t}\psi'_{jp\tau} \leq R'_{jct} \qquad \forall j \in \mathcal{J}, \forall c \in \mathcal{C},\ \forall t \in \mathcal{T},$$

where $\mathcal{P}_c$ is the set of the programs of the cluster c. Let $\underline{G}_p$ and $\overline{G}_p$ be, respectively, the minimum and maximum number of inmates that can be enrolled in closed-enrollment program p. The following set of constraints enforce these capacity bounds for the closed-enrollment programs.

$$\underline{G}_p\psi'_{jpt} \leq \psi_{jpt} \leq \overline{G}_p\psi'_{jpt} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P}^c,\ \forall t \in \mathcal{T}.$$

5.4.3 Scheduling of the Programs for the Inmates

One of the main objectives of the IAP is to minimize the maximum waiting time of inmates to start their program(s). In this section, we present the constraints needed to minimize the maximum waiting time of inmates to start their program(s).

Let $\mathcal{T}' = \mathcal{T} \cup \{\infty\}$, and let $\mathcal{P}_i$ be the set of the programs prescribed for inmate i. The new decision variable z_{ijpt}, for $i \in \mathcal{I},\ j \in \mathcal{J}, p \in \mathcal{P}_i, t \in \mathcal{T}'$, is equal to one if inmate i is assigned to facility j, starting program p at time t; otherwise, it is equal to zero. If $z_{ijp\infty} = 1$, it implies that inmate i is not going to start program p in the decision horizon, i.e. later than the last time period of the decision horizon. Following is the set of constraints that define the relationship between z_{ijpt} and x_{ij}

$$\sum_{t\in\mathcal{T}'} z_{ijpt} = x_{ij} \qquad \forall i \in \mathcal{I},\ \forall j \in \mathcal{J},\ \forall p \in \mathcal{P}_i.$$

Let y^a_{jpt} and ψ^a_{jpt}, for $j \in \mathcal{J},\ p \in \mathcal{P},\ t \in \mathcal{T}$, be defined as follows,

y^a_{jpt}: The number of inmates already in facility j, who are prescribed program p and have to start it by time t but can not do so.

ψ^a_{jpt}: The number of inmates already in facility j, starting program p at time t.

We have

$$\underline{q}_{jpt} \leq \sum_{\tau=0}^{t} \psi^a_{jp\tau} + y^a_{jpt} \leq \overline{q}_{jpt} \quad \forall j \in \mathcal{J}, \forall p \in \mathcal{P},\ \forall t \in \mathcal{T}.$$

Additionally, let y^n_{jpt} and ψ^n_{jpt}, for $j \in \mathcal{J},\ p \in \mathcal{P},\ t \in \mathcal{T}$, be defined as follows

y^n_{jpt}: The number of inmates assigned to facility j, who are prescribed program p and have to start it by time t, but can not do so.

ψ^n_{jpt}: The number of inmates assigned to facility j, starting program p at time t.

We have

$$\psi^n_{jpt} = \sum_{i \in \mathcal{I}_p} z_{ijpt} \qquad \forall j \in \mathcal{J}, \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$\sum_{i \in \mathcal{I}} \alpha_{ipt} x_{ij} \leq \sum_{\tau=1}^{t} \psi^n_{jp\tau} + y^n_{jpt} \leq \sum_{i \in \mathcal{I}} \beta_{ipt} x_{ij} \quad \forall j \in \mathcal{J}, \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

where $\mathcal{I}_p$ is the set of the inmates who need program p.

Suppose the number of inmates, already in facility j that should start program p at time t is more than the available spots for program p at time t. Then we have $y^a_{jpt} > 0$. In this case, ψ^n_{jpt} should be equal to zero. In other words, if there are not enough spots of program p for the inmates that are already in facility j, then the number of inmates, assigned to facility j through the model, that are going to start program p at time t should be zero. In order to satisfy this constraint, the indicator variable ϕ_{jpt}, for $j \in \mathcal{J},\ p \in \mathcal{P},\ t \in \mathcal{T}$ is equal to one if $y^a_{jpt} > 0$; otherwise, it is equal to zero. Then we have

$$y^a_{jpt} \leq M\phi_{jpt} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$\psi^n_{jpt} \leq M(1 - \phi_{jpt}) \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

where M is a big number.

Additionally, we have the following set of constraints, which defines the relationship between the decision variables of the problem

$$\psi^a_{jpt} + \psi^n_{jpt} = \psi_{jpt} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$y^a_{jpt} + y^n_{jpt} = y_{jpt} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

Let w_{ip}, for $i \in \mathcal{I},\ p \in \mathcal{P}_i$ be the waiting time of inmate i to start program p after his latest possible start time $\hat{t}_{ip}$. We have

$$w_{ip} = \sum_{j \in \mathcal{J}} \sum_{t \in \mathcal{T}} \max(0, t - \hat{t}_{ip}) z_{ijpt} \qquad \forall i \in \mathcal{I},\ \forall p \in \mathcal{P}_i.$$

Finally, let w'_i be the maximum waiting time of inmate i to start his/her program(s). Then

$$w'_i \geq w_{ip} \qquad \forall i \in \mathcal{I},\ \forall p \in \mathcal{P}_i.$$

5.4.4 Transfer Constraints

The constraints needed to account for the inmate transfers after the initial assignment are the same kind of constraints as the ones for the initial assignment in the current model. However, as the importance of these constraints are frequently different for transfers, the weights of the factors in the objective function differ from an initial assignment.

5.4.5 The Objective Function

The IAP is a multi-objective problem. There are different approaches in the literature to deal with a multi-objective optimization problem. We consider the weighted sum method [Sawaragi et al., 1985] to combine the objectives and have a one-shot optimization in assigning the inmates. The choice of the weighted sum of the objectives is validated by solving real data instances from the Pennsylvania Department of Corrections.

It is worth mentioning that the weights of all the objectives are assumed to be positive. The objectives of the IAP are listed as follows:

- Violation of the general factors should be minimized. The violation is equal to

$$\vartheta = \sum_{i \in \mathcal{I}} \sum_{k \in \mathcal{K}} \lambda^f_{ik} v_{ik},$$

where λ^f_{ik} is the weight of factor k for inmate i.

- Assignment of inmates under the capacity and over the capacity of the CIs should be minimized. The violations of the capacity constraints are defined as

$$o_j = \sum_{i \in \mathcal{I}} x_{ij} - c_j^{\max} \quad \forall j \in \mathcal{J},$$
$$u_j = c_j^{\min} - \sum_{i \in \mathcal{I}} x_{ij} \quad \forall j \in \mathcal{J}.$$

  Then, the overall capacity violation is equal to

$$\eta = \sum_{j \in \mathcal{J}} \lambda_j^o o_j + \lambda_j^u u_j,$$

  where λ_j^o and λ_j^u for $j \in \mathcal{J}$ are, respectively, the weights of over-assignment and under-assignment to the CIs.

- The difference between the capacities of the CIs should be minimized

$$\delta = \lambda^\delta \sum_{j_1 \in \mathcal{J}} \sum_{j_2 \in \mathcal{J} | j_2 \neq j_1} \left(\delta_{j_1 j_2}^+ + \delta_{j_1 j_2}^- \right),$$

  where λ^δ is the weight of the capacity difference, and $\delta_{j_1 j_2}^+$ and $\delta_{j_1 j_2}^-$ are defined in equation (5.1).

- Distance to the home county of the CIs should be minimized.

$$\gamma = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \lambda_i^d d_{ij} x_{ij},$$

  where λ_i^d is the weight of the distance for inmate i.

- The number of inmates that can not start their program on time should be minimized.

$$\omega = \sum_{j \in \mathcal{J}} \sum_{p \in \mathcal{P}} \sum_{t \in \mathcal{T}} \lambda_{jpt}^\omega \, y_{jpt},$$

  where λ_{jpt}^ω is the weight of the wait list of program p at facility j in time t.

- The maximum program waiting time of inmates need to be minimized

$$\theta = \sum_{i \in \mathcal{I}} \lambda_i^{\theta} w_i',$$

where λ_i^t is the penalty weight of waiting time of inmate i.

The weighted sum of the objectives is defined as

$$\lambda_{\vartheta}\vartheta + \lambda_{\eta}\eta + \lambda_{\delta}\delta + \lambda_{\gamma}\gamma + \lambda_{\omega}\omega + \lambda_{\theta}\theta,$$

where the weights of all the objective elements are positive. Objective hierarchies are being enforced through order of magnitude differences in the weight applied. General factors have the highest priority in assigning inmates to CIs. Minimizing the maximum waiting time for each inmate is second in the hierarchy of objectives. Assigning in the range of the minimum and maximum capacity of each facility has the next highest priority. Additionally, in order to reduce the population of the CIs, program waiting lists have a high priority in the objective function. Assigning inmates to a facility near their home county is less important compared to the other objectives of the problem.

5.4.6 The Multi-Objective MILO Model

Now we present the complete optimization model for the inmate assignment and scheduling problem. The lists of parameters and decision variables of IAP are summarized in Tables 5.1 and 5.2, respectively. We utilize the hierarchically weighted sum method to combine the objectives and have a single-objective optimization problem. The MILO model is as follows:

$$\begin{aligned} \min \quad & \lambda_{\vartheta}\vartheta + \lambda_{\eta}\eta + \lambda_{\delta}\delta + \lambda_{\gamma}\gamma + \lambda_{\omega}\omega + \lambda_{\tau}\tau \\ \text{s.t.} \quad & \sum_{j \in \mathcal{J}} x_{ij} = 1 && \forall i \in \mathcal{I}, \end{aligned}$$

$$\sum_{t\in\mathcal{T}'} z_{ijpt} = x_{ij} \qquad \forall i\in\mathcal{I}, \forall j\in\mathcal{J},\ \forall p\in\mathcal{P}_i,$$

$$\psi^n_{jpt} = \sum_{i\in\mathcal{I}_p} z_{ijpt}, \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$\kappa_{ik}\Big(1-\sum_{j\in\mathcal{J}}\rho_{jk}x_{ij}\Big) = v_{ik} \qquad \forall\ i\in\mathcal{I}, \forall\ k\in\mathcal{K},$$

$$\sum_{i\in\mathcal{I}}\alpha_{ipt}x_{ij} + \underline{q}_{jpt} \le y_{jpt} + \sum_{\tau=0}^{t}\psi_{jp\tau} \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$\sum_{i\in\mathcal{I}}\beta_{ipt}x_{ij} + \overline{q}_{jpt} \ge y_{jpt} + \sum_{\tau=0}^{t}\psi_{jp\tau} \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$\sum_{\tau=\max(0,t-\hat{t}_p)}^{t}\psi_{jp\tau} \le R_{jpt} \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P}^o,\ \forall t\in\mathcal{T},$$

$$\underline{G}_p\psi'_{jpt} \le \psi_{jpt} \le \overline{G}_p\psi'_{jpt} \qquad \forall j\in\mathcal{J},\ \forall p\in\mathcal{P}^c,\ \forall t\in\mathcal{T}$$

$$\sum_{p\in\mathcal{P}_c}\sum_{\tau=\max(0,d_p)}^{t}\psi'_{jp\tau} \le R'_{jct} \qquad \forall j\in\mathcal{J}, \forall c\in\mathcal{C},\ \forall t\in\mathcal{T}$$

$$\underline{q}_{jpt} \le \sum_{\tau=1}^{t}(\psi^a_{jp\tau}) + y^a_{jpt} \le \overline{q}_{jpt} \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$\sum_{i\in\mathcal{I}}\alpha_{ipt}x_{ij} \le \sum_{\tau=1}^{t}(\psi^n_{jpt}) + y^n_{jp\tau} \le \sum_{i\in\mathcal{I}}\beta_{ipt}x_{ij}, \qquad \forall j\in\mathcal{J},\ \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$y^a_{jpt} \le M\phi_{jpt} \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P}, \forall t\in\mathcal{T},$$

$$\psi^n_{jpt} \le M(1-\phi_{jpt}) \qquad \forall j\in\mathcal{J}, \forall p\in\mathcal{P}, \forall t\in\mathcal{T},$$

$$\psi^a_{jpt} + \psi^n_{jpt} = \psi_{jpt} \qquad \forall j\in\mathcal{J},\ \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$y^a_{jpt} + y^n_{jpt} = y_{jpt} \qquad \forall j\in\mathcal{J},\ \forall p\in\mathcal{P},\ \forall t\in\mathcal{T},$$

$$w_{ip} = \sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}}\max(0, t-\hat{t}_{ip})z_{ijpt} \qquad \forall i\in\mathcal{I},\ \forall p\in\mathcal{P}_i,$$

$$w'_i \ge w_{ip} \qquad \forall i\in\mathcal{I},\ \forall p\in\mathcal{P}_i,$$

$$\sum_{i\in\mathcal{I}} x_{ij} = s_j \qquad \forall j\in\mathcal{J},$$

$$c_{j_2}s_{j_1} - c_{j_1}s_{j_2} = \delta^+_{j_1j_2} - \delta^-_{j_1j_2} \qquad \forall j_1, j_2\in\mathcal{J},$$

$$s_j \leq c_j^{\max} + o_j \qquad \forall j \in \mathcal{J},$$

$$s_j \geq c_j^{\min} - u_j \qquad \forall j \in \mathcal{J},$$

$$\sum_{j \in \mathcal{J}_i^s} x_{ij} = 0 \qquad \forall i \in \mathcal{I},$$

$$x_{i_1 j} + x_{i_2 j} \leq 1 \qquad \forall (i_1, i_2) \in \mathcal{I}^s,$$

$$z_{ijpt} \in \{0,1\} \qquad \forall i \in \mathcal{I}, \forall j \in \mathcal{J}, \forall p \in \mathcal{P}_i, \forall t \in \mathcal{T}',$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in \mathcal{I},\ \forall j \in \mathcal{J},$$

$$v_{ik} \in \{0,1\} \qquad \forall i \in \mathcal{I},\ \forall k \in \mathcal{K},$$

$$\phi_{jpt} \in \{0,1\} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$y_{jpt}^a, y_{jpt}^n, y_{jpt} \in \mathbb{N} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$\psi_{jpt}^a, \psi_{jpt}^n, \psi_{jpt} \in \mathbb{N} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P},\ \forall t \in \mathcal{T},$$

$$\psi'_{jpt} \in \mathbb{N} \qquad \forall j \in \mathcal{J},\ \forall p \in \mathcal{P}^c,\ \forall t \in \mathcal{T},$$

$$s_j,\ o_j,\ u_j \in \mathbb{N} \qquad \forall j \in \mathcal{J},$$

$$\delta_{j_1 j_2}^+, \delta_{j_1 j_2}^- \in \mathbb{N} \qquad \forall j_1, j_2 \in \mathcal{J},\ j_1 \neq j_2,$$

$$w_{ip} \geq 0 \qquad \forall i \in \mathcal{I},\ \forall p \in \mathcal{P}_i,$$

$$w_i \geq 0 \qquad \forall i \in \mathcal{I}.$$

Table 5.1: The parameters of the IAP

Parameter	Definition
$\mathcal{I}$	The set of inmates that need to be assigned
$\mathcal{J}$	The set of the CIs
$\mathcal{K}$	The set of factors
$\mathcal{P}$	The set of programs
$\mathcal{P}^o$	The set of open-enrollment programs
$\mathcal{P}^c$	The set of closed-enrollment programs
$\mathcal{C}$	The set of program clusters
$\mathcal{P}_i$	The set of the program(s) of inmate i
$\mathcal{P}_c$	The set of closed-enrollment programs of cluster c
$\mathcal{J}_i^s$	Set of CIs that inmate i should be separated from
$\mathcal{I}^s$	Set of inmate pairs that should be separated from each other
$\mathcal{T}$	The set of the time periods in the time horizon
$\mathcal{T}'$	$\mathcal{T} \cup \infty$
κ_{ik}	1 if factor k applies to inmate i 0 otherwise
ρ_{jk}	1 if CI j can accommodate inmates with factor k 0 otherwise
$\hat{t}_ip$	The latest time that inmate i can start program p and finish it before his scheduled board meeting
$\tilde{t}_ip$	The earliest time that inmate i can start program p based on the system regulations
α_{ipt}	1 if inmate i should not start program p later than time t 0 otherwise
β_{ipt}	1 if inmate i can start program p at time t 0 otherwise
$\underline{q}_{jpt}$	The number of inmates, already in facility j, that should have started program p by time t to be able to finish their program before their parole board meeting, i.e., the number of inmates with $\hat{t}_{ip} \leq t$
$\overline{q}_{jpt}$	The maximum number of inmates, already in facility j, that can start program p at time t, i.e., the number of inmates with $\tilde{t}_{ip} \leq t$
R_{jpt}	Number of spots available for open-enrollment program p in CI j at time period t
R'_{jct}	Number of groups available for cluster c in CI j at time period t
$\overline{G}_p$	Maximum number of inmates in a group of program p
$\underline{G}_p$	Minimum number of inmates needed to run program p
d_{ij}	The distance between the home county of inmate i to facility j
c_j	Capacity of facility j
$c_j^{\min}$, $c_j^{\max}$	Minimum and maximum capacity at facility j which are functions of c_j

Table 5.2: The decision variables of the IAP.

Parameter	Definition
x_{ij}	1 if inmate i is assigned to CI j 0 otherwise
z_{ijpt}	1 if inmate i is assigned to CI j, starting program p at time t 0 otherwise
v_{ik}	1 if inmate i violates factor k 0 otherwise
ψ_{jpt}	The number of inmates starting program p at t in facility j
ψ^a_{jpt}	The number of inmates already in facility j, starting program p at time t
ψ^n_{jpt}	The number of inmates assigned to facility j, starting program p at time t
ψ'_{jpt}	The number of groups of the closed program p that start at time t in facility j
y_{jpt}	The number of inmates at facility j that are prescribed program p and have to start it by time t but can not do so
y^a_{jpt}	The number of inmates already in facility j, who are prescribed program p and have to start it by time t but can not do so
y^n_{jpt}	The number of inmates assigned to facility j, who are prescribed program p and have to start it by time t, but can not do so
ϕ_{jpt}	1 if $y^a_{jpt} > 0$ 0 otherwise
w_{ip}	The waiting time of inmate i to start program p from his latest possible start time $\hat{t}_{ip}$
w'_i	Maximum waiting time of inmate i to start his/her program(s)
s_j	Total number of inmates assigned to facility j
o_j	Number of inmates assigned over the maximum capacity of facility j
u_j	Number of inmates assigned under the minimum capacity of facility j
$\delta^+_{j_1 j_2}, \delta^-_{j_1 j_2}$	Variables representing the difference in capacities between the CIs j_1 and j_2

We can strengthen the MILO model formulation by adding a set of constraints for the inmates who have prescribed program(s) as follows

$$\sum_{j\in\mathcal{J}}\sum_{t\in\mathcal{T}'} z_{ijpt} = 1 \qquad \forall i \in \mathcal{I},\ \forall p \in \mathcal{P}_i.$$

While these constraints are redundant, notably if we add them to the model, the solution time decreases significantly. Further, in order to generate a good solution quickly, we set the MILO solver to perform the highest level of preprocessing before starting the branch & bound algorithm, which further reduces the overall solution time.

5.5 Implementation at the PADoC

The project from idea to successful implementation took five years. Before this project started, inmates were assigned to CIs manually by a staff member of the OPM. This manual process had three main drawbacks:

- A variety of factors need to be considered in assigning each inmate to a CI, including security concerns, mental and medical conditions, program needs, separation from other inmates, capacities of the CIs, home county of the inmates, etc. Having all the factors of the assignment and characteristics and capacities of CIs in mind, and considering them for each individual is time-consuming and prone to human errors. As a result there were numerous inappropriate assignment of the inmates.

- If the inmate assignment is done sequentially, then the inmates that are assigned later, are not considered in the earlier assignments. This makes the process inefficient and suboptimal. In fact, if the assignment is done manually, it is hardly possible to consider the following inmate assignments appropriately in the assignment of the current inmate.

- Scheduling of treatment programs was not considered in the manual inmate assignment. This resulted in inmates having longer waiting times to get their programming, thus postponed their eligibility to go on parole, and so increased the population of the CIs.

The DTDSS, which was initially developed, enabled the PADoC to address the first drawback of the manual inmate assignment and consider the rules and criteria of the assignment in assigning each individual inmate to the CIs. However, DTDSS lacks the ability to simultaneously assign a batch of inmates to the CIs, and it does not consider the treatment program scheduling in the assignment. This stressed the need to develop the multi-objective optimization model, which became the heart of the IADSS. The rigorous optimization model enables OPM to consistently account for all the the factors of the assignment. It also enables OPM to simultaneously assign the inmates to CIs, as well as schedule programs optimally to minimize the waiting time of each individual inmate in starting their program(s).

5.5.1 Development of the IADSS

The development of the IADSS took three years. First, a mathematical optimization model was developed as a proof of concept to optimize the simultaneous initial assignment of the inmates to the CIs. It demonstrated to OPM personnel that mathematical optimization provides a powerful tool to optimally assign inmates to the CIs. In conjunction with model development, data had to be harvested from the PADoC databases; thus, data collection and clean up procedures were set up and implemented to link the model to the live databases. The workflow of IADSS is presented in Figure 5.3.

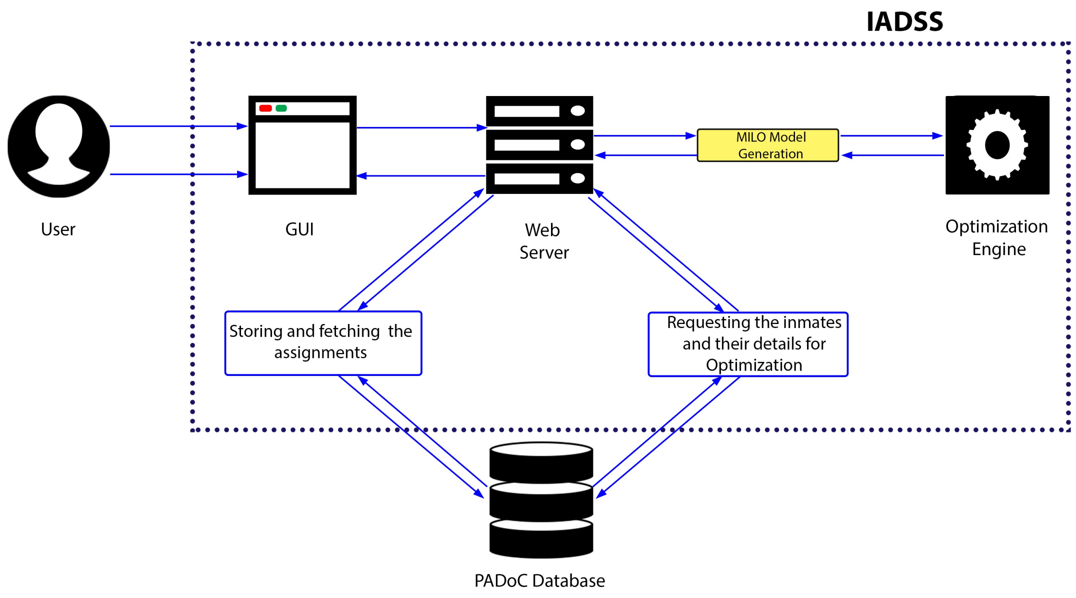

IADSS
User
GUI
Web Server
MILO Model Generation
Optimization Engine
Storing and fetching the assignments
Requesting the inmates and their details for Optimization
PADoC Database

Figure 5.3: Workflow of the IADSS.

The heart of the IADSS is the *optimization module* which generates the mathematical optimization model of the IAP using the data extracted from the PADoC databases, and solves the model. As the inmate assignment to CIs is a multi-objective process, we propose a hierarchical multi-objective optimization model. We consider the weighted sum method [Sawaragi et al., 1985] to combine the objectives. The choice of the weighted sum of the objectives is validated by solving real data instances from the PADoC.

The time sequence of the development phases followed the anticipated increasing mathematical sophistication and complexity of the modules. The violations of the inmate assignment factors were interpreted as the penalty objectives of the assignment and were added one-by-one to the optimization model. As explained in Section 5.2, we need to make two main decisions: assignment of inmates to the CIs and scheduling the start of their program(s). We initially developed a model which only did the assignment of the inmates to the CIs, and tested the model with real data from PADoC to validate the assignment recommendations. Then we extended the model to include the scheduling of the programs for the inmates. Executing the project in this sequence brought meaningful capability online in a judicious manner, while demonstrating to OPM what was possible with an optimization model, and how to utilize a decision support system to optimally execute their most critical task. The model which does the assignment of the inmates and schedules the programs has been used for the daily assignment of the inmates since September 2016.

5.6 Benefits and Impact of the IADSS

The successful development and implementation of IADSS has both significant financial and non-quantifiable human benefits.

5.6.1 High-Quality, Consistent Assignment

- The assignment of the inmates is done simultaneously for all the inmates with a petition for assignment or transfer. Simultaneous assignment ensures system-wide optimum.
- All the factors of the assignment are considered for each individual. As a result, consistently high-quality assignments are made. Current errors are almost exclusively due to data inconsistency, so undesired assignments help OPM to identify data errors.
- The inmate assignment process was previously fragmented in the sense that assignment was done by OPM and the program waiting list was monitored by the Bureau of Treatment Services (BTS) and reported to OPM on a monthly basis. With the implementation of the IADSS, the process is integrated and all the necessary elements of the assignment are considered in one system.

- Program schedules and wait lists at each CI are generated as an integral part of the inmate assignment output. The integrated IADSS minimizes the wait time of the inmates for their required program(s), thus allowing timely release of inmates, and so reducing the inmate population.

- In addition to simultaneous assignment, *individual assignment* can be done for the inmates. Facilities are sorted in the individual assignment for each inmate considering all the factors of the assignment only for that inmate. In case the simultaneous assignment recommendation, for some reason, is not appropriate for an individual, the individual assignment results can be used to evaluate possible assignment to other CIs . The simultaneous assignment recommendation and individual assignment recommendations from the IADSS interface are demonstrated in the first and third panel of Figure 5.4, respectively.

- Three geographical regions (west, central, east) are defined in PA. Counties and CIs are placed in each of these regions. In Figure 5.2, the regions of the CIs are given. Due to the complexity of considering the distance of the home county to the CIs, only assignment of an inmate to his home region was considered before. The IADSS enables DoC to consider the actual distance of the home county to the CIs for each inmate.

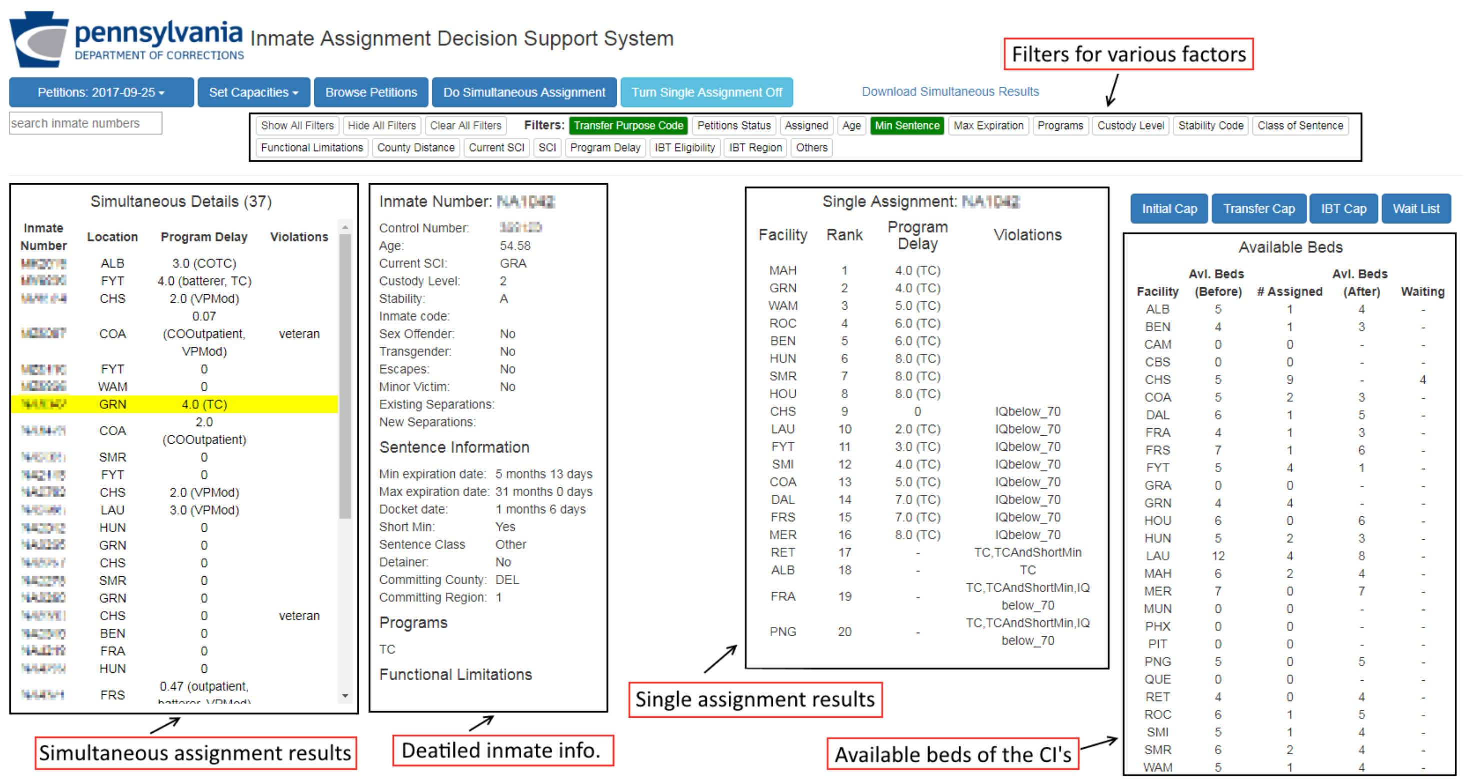
- The rate of acceptance of the simultaneous assignments and individual assignments has been measured to validate the MILO model and ensure that the MILO model captures the hierarchy of the factors of the assignment. In Jan 2017, over 90% of the inmates were assigned to the facility that was suggested by the simultaneous assignment. Among the remaining 10 percent of the inmates, more than 6 percent were assigned to one of the first three CIs recommended by the individual assignment. The remaining 4 percent, that were assigned to other CIs, were either because of data inconsistency, or the special conditions of those inmates. In Table 5.3, results of the IADSS for the first 10 days of the year 2017 are presented.

Table 5.3: Assignment recommendations.

Date	# of inmates	Sim. assignment match	Ind. assignment used and matched	Not ind. nor sim. matched	Sim. assignment match	Ind. or sim. assignment match
3 Jan	15	12	3	0	80%	100 %
4 Jan	54	53	1	0	98.15%	100 %
5 Jan	53	43	5	5	81.13 %	90.57 %
9 Jan	14	12	1	1	85.71%	92.86 %
10 Jan	98	91	5	2	92.86 %	97.96 %
Total	234	211	15	8	90.17%	96.58%

5.6.2 User-Friendly Web Application

- A web-based Graphical User Interface (GUI) is developed to enable interaction with the IADSS. In Figure 5.4, a screenshot of the GUI is demonstrated.



Simultaneous Details (37)

Inmate Number	Location	Program Delay	Violations
[redacted]	ALB	3.0 (COTC)	
[redacted]	FYT	4.0 (batterer, TC)	
[redacted]	CHS	2.0 (VPMod)	
[redacted]	COA	0.07 (COOutpatient, VPMod)	veteran
[redacted]	FYT	0	
[redacted]	WAM	0	
[redacted]	GRN	4.0 (TC)	
[redacted]	COA	2.0 (COOutpatient)	
[redacted]	SMR	0	
[redacted]	FYT	0	
[redacted]	CHS	2.0 (VPMod)	
[redacted]	LAU	3.0 (VPMod)	
[redacted]	HUN	0	
[redacted]	GRN	0	
[redacted]	CHS	0	
[redacted]	SMR	0	
[redacted]	GRN	0	
[redacted]	CHS	0	veteran
[redacted]	BEN	0	
[redacted]	FRA	0	
[redacted]	HUN	0	
[redacted]	FRS	0.47 (outpatient,	

Inmate Number: [redacted]

Control Number:	[redacted]
Age:	54.58
Current SCI:	GRA
Custody Level:	2
Stability:	A
Inmate code:	
Sex Offender:	No
Transgender:	No
Escapes:	No
Minor Victim:	No
Existing Separations:	
New Separations:	

Sentence Information

Min expiration date:	5 months 13 days
Max expiration date:	31 months 0 days
Docket date:	1 months 6 days
Short Min:	Yes
Sentence Class	Other
Detainer:	No
Committing County:	DEL
Committing Region:	1

Programs

TC

Functional Limitations

Single Assignment: [redacted]

Facility	Rank	Program Delay	Violations
MAH	1	4.0 (TC)	
GRN	2	4.0 (TC)	
WAM	3	5.0 (TC)	
ROC	4	6.0 (TC)	
BEN	5	6.0 (TC)	
HUN	6	8.0 (TC)	
SMR	7	8.0 (TC)	
HOU	8	8.0 (TC)	
CHS	9	0	IQbelow_70
LAU	10	2.0 (TC)	IQbelow_70
FYT	11	3.0 (TC)	IQbelow_70
SMI	12	4.0 (TC)	IQbelow_70
COA	13	5.0 (TC)	IQbelow_70
DAL	14	7.0 (TC)	IQbelow_70
FRS	15	7.0 (TC)	IQbelow_70
MER	16	8.0 (TC)	IQbelow_70
RET	17	-	TC,TCAndShortMin
ALB	18	-	TC
FRA	19	-	TC,TCAndShortMin,IQ below_70
PNG	20	-	TC,TCAndShortMin,IQ below_70

Available Beds

Facility	Avl. Beds (Before)	# Assigned	Avl. Beds (After)	Waiting
ALB	5	1	4	-
BEN	4	1	3	-
CAM	0	0	-	-
CBS	0	0	-	-
CHS	5	9	-	4
COA	5	2	3	-
DAL	6	1	5	-
FRA	4	1	3	-
FRS	7	1	6	-
FYT	5	4	1	-
GRA	0	0	-	-
GRN	4	4	-	-
HOU	6	0	6	-
HUN	5	2	3	-
LAU	12	4	8	-
MAH	6	2	4	-
MER	7	0	7	-
MUN	0	0	-	-
PHX	0	0	-	-
PIT	0	0	-	-
PNG	5	0	5	-
QUE	0	0	-	-
RET	4	0	4	-
ROC	6	1	5	-
SMI	5	1	4	-
SMR	6	2	4	-
WAM	5	1	4	-

Figure 5.4: A screen shot of the web-based UI of the IADSS.

- All the personal and sentence information needed for the assignment of an inmate is collected and displayed in the GUI to facilitate the review of the assignment. In the second panel of Figure 5.4, the inmate display page is demonstrated.
- Reporting of the program waiting list alerts BTS for current and future bottlenecks in program schedules and availability.

5.6.3 Security Enhancement

Security enhancement is hard to quantify, however, it was one of the main motivations for initiating this project. The use of IADSS at the PADoC has already resulted in the following identified security enhancements.

- It is stated by the PADoC Secretary, Wetzel, that inmate transportation is one of the riskiest operations at PADoC. By doing proper initial assignment, IADSS has reduced inmate transfers, and so enhancing the security of the CIs and public safety.
- IADSS considers inmates' demographic information and enforces the separations in the assignment, which in turn reduces the number of assaults, thus increasing the security of the CIs.

5.6.4 Quantified Savings

In this section, we present the cost savings resulted from the implementation of IADSS in the first year, and project the benefits to a period of five years. Four areas of significant savings are identified.

- *Reduced waiting time*: IADSS helps to decrease the waiting time for treatment programs, which reduces the length of stay for inmates past their minimum sentence date. We consider the inmates that have less than 9 months to their minimum sentence date at the time of their initial petition, who need at least one treatment program. These inmates must start their programs immediately, since the delay in starting their program(s) directly postpones their parole eligibility. The waiting time of these inmates to start all their programs is calculated with the goal to see how much IADSS has helped to reduce the waiting time for programs. In Figure 5.5, the cumulative distribution functions of the waiting time of these inmates for the first and second quarters of 2016 (2016-1, 2016-2), and the first and second quarters of 2017 (2017-1, 2017-2) are plotted. Notice that the waiting time in the second quarter of 2017 stops at three months, since we did not have the data for longer waiting times at the time of writing the paper. For both quarters 1 and 2, the cumulative distribution function for 2017 is above and to the left of the one for 2016, showing that the use of IADSS has reduced waiting times substantially. Comparing the waiting time of the inmates with initial petition requests in the first quarter of 2016, when IADSS was not yet used, with the first quarter of 2017, we found out that the average waiting time of the inmates in the first quarter of 2016 is 143 days while the average waiting time of the inmates in the first quarter of 2017 is 89 days. Therefore, the average waiting time decreased by 54 days from 2016-1 to 2017-1.

In average, PADoC has 10000 initial petitions annually. 12% of those petitions have less than 9 months to their minimum sentence date and need at least one program. The marginal cost of keeping an inmate in a CI is \$16 per day. As a result, the total annual saving of reducing the inmates' waiting time in starting their program(s) is $10000 \times 0.12 \times 54 \times 16 = \$1,036,800$. As we can see in Figure 5.5, the waiting time is significantly decreasing from 2016 to 2017. Based on already achieved 54 days reduction in the waiting time, 90 days reduction in the waiting time can be projected at the steady state of the system in years 4 and 5. The 90 days reduction in the waiting time of programs enables the PADoC to close a full CI unit. Closing a CI unit allows for more savings than the marginal cost of keeping an inmate in a CI. If a CI unit is closed, the savings per day for each inmate is \$30. Thus, the saving in years 4 and 5 will be $10000 \times 0.12 \times 90 \times 30 = \$3,240,000$.

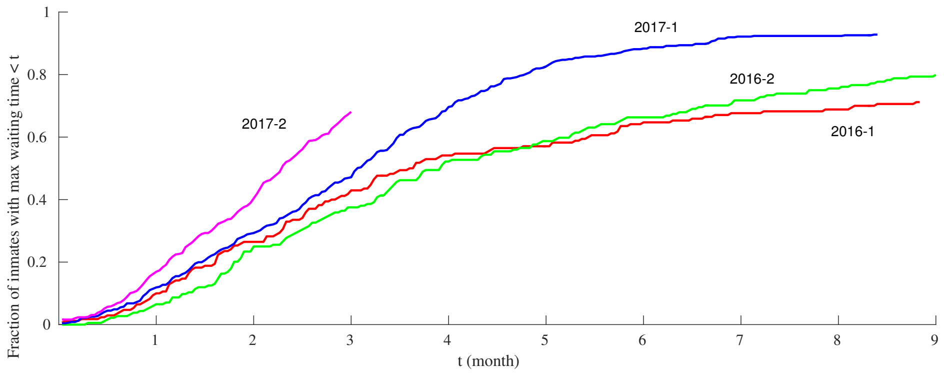


Figure 5.5: Program waiting time for inmates with less than 9 months to their minimum sentence date.

- *Fewer assaults*: There are fewer assaults, due to assigning the right combination of inmates to the most appropriate CIs. We compared the number of assaults in the period January-July of 2017 to the same period in 2016; 95 fewer assaults were reported. If we project this result to the full year, 163 fewer assaults are expected in 2017. The PADoC estimates that approximately 10% to 15% of this reduction is due to the introduction of IADSS, thus IADSS results in 20 fewer assaults in 2017. The criminal justice literature [Cohen, 2005] documents that an assault on average costs \$70,000. Thus IADSS has resulted in $20 \times 70,000 = \$1,400,000$ saving by reducing the number of assaults.

- *Reduced staff*: Fewer staff are required in OPM to oversee inmate assignments and transfers. As a result of using IADSS, one less Captain position is needed to do the inmate assignments at PADoC. The salary and benefits of a captain is $\$134,742$ annually.

- *Fewer transfers*: Due to initially assigning inmates to the correct CI, fewer transfers are later required. By making better assignments with IADSS, $4,672$ fewer transfers were needed in 2017. The cost of each transfer is on average \$82.85 at the PADoC. Hence, the total annual transportation saving is equal to $4,672 \times 82.85 = \$387,075$.

Considering the four main saving points, IADSS has decreased the annual cost at PADoC by $\$2,958,617$, and the projected saving over five years is $\$19,199,485$.

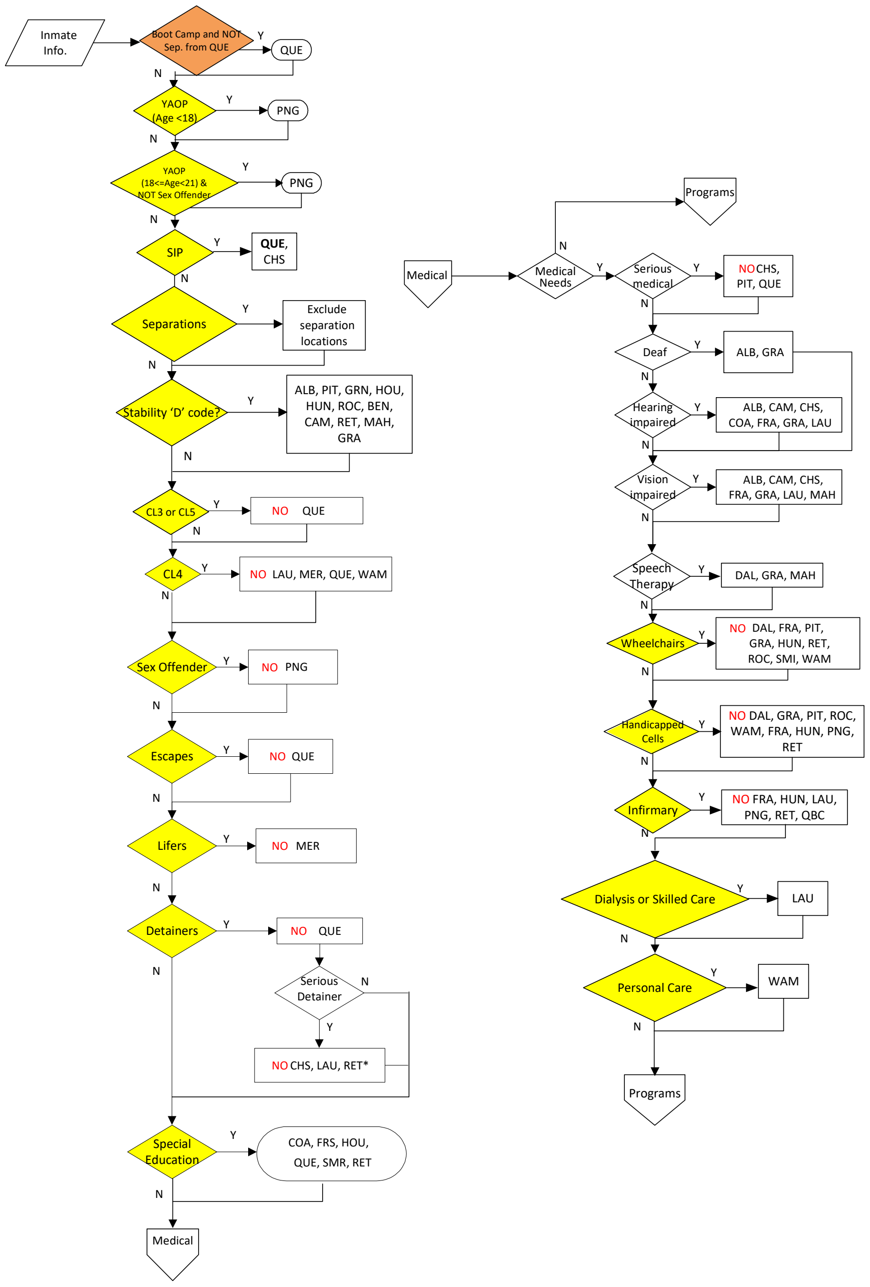
5.7 Summary

Every correctional system faces the inmate assignment problem on a daily basis. Various constraints, including general assignment factors, CI capacity constraints, scheduling of inmate treatment programs, and the assignment of inmates near their home counties, should be satisfied. Making an ideal assignment (i.e., satisfying all the constraints of the assignment) is impossible; thus, the IAP is inherently an infeasible problem. Additionally, the treatment programs must be scheduled at the time of the assignment.

In this chapter, we discuss the development of a novel hierarchical multiobjective MILO model for the IAP. The weighted sum of the violation of the assignment constraints and the treatment-program waiting times serve as the penalty objective of the MILO model. The multiobjective MILO model is the core of the IADSS. The IADSS enables the PADoC to simultaneously and optimally assign inmates to the

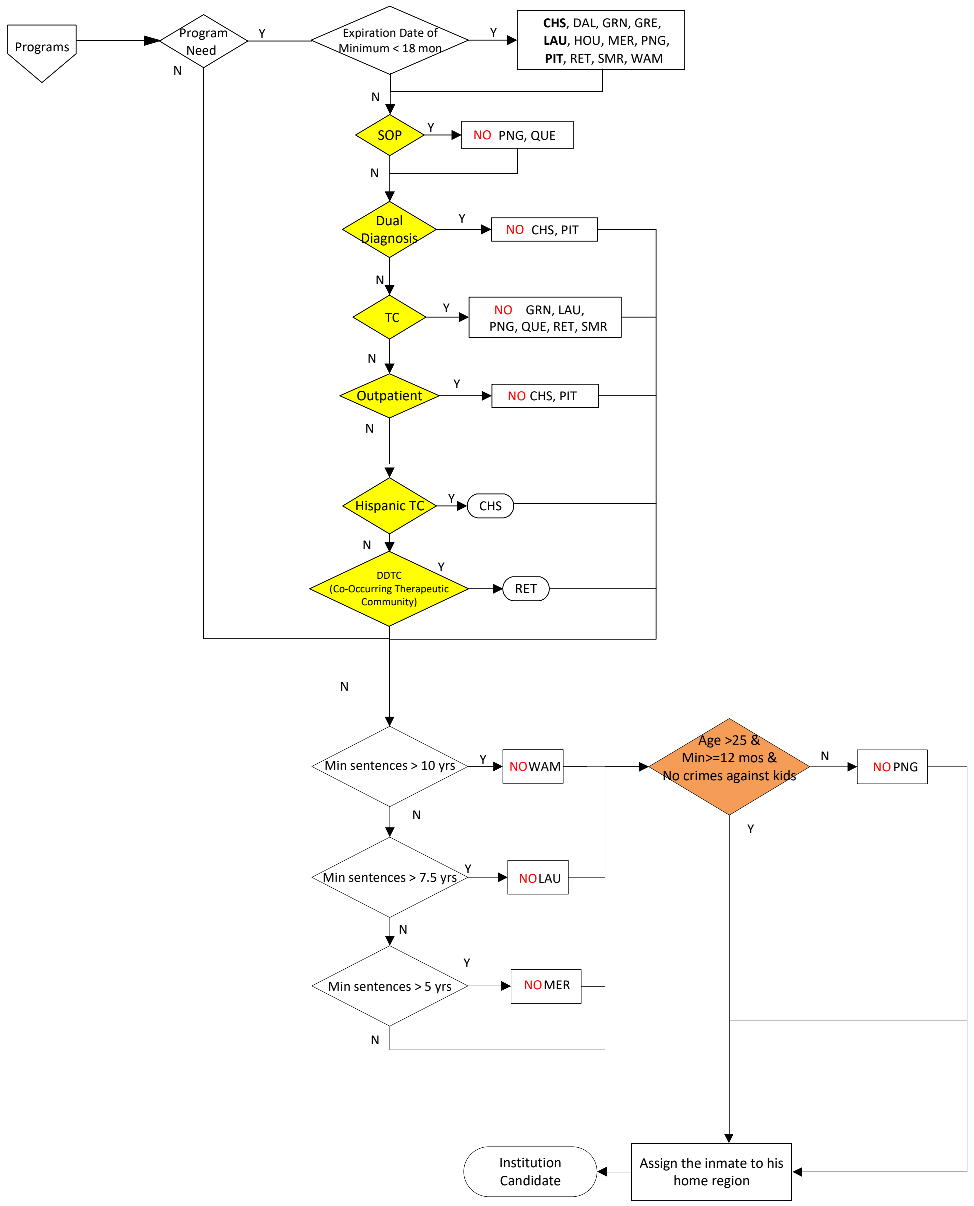
CIs in the PA correctional system and schedules treatment programs for them, while considering all the rules and criteria of the assignment. The IADSS minimizes the waiting time of the inmates for being assigned to their required program(s); hence, it facilitates the timely eligibility of the inmates for parole, which ultimately reduces the inmate population within the correctional system. The PADoC has successfully used the IADSS for the daily assignment of inmates to CIs since September 2016.

To the best of our knowledge, this is the first time that OR methodology has been built directly into the routine business operations of a correctional system. The success of this project opens new avenues to: (1) adapt and introduce the IADSS methodology to optimize the operations of correctional systems of other states and countries, and (2) explore other applications of OR methodology in the complex operations of correctional systems. Correctional systems in the United States and worldwide have numerous problems that cry out for solutions using OR methodologies. This highly successful application of OR in a large correctional system will open a rich application area of OR, just as the first crew-scheduling application did in the airline industry.



(a) Section one

Figure 5.1: The decision tree of the inmate assignment process.


Programs
Program Need
Y
N
Expiration Date of Minimum < 18 mon
Y
CHS, DAL, GRN, GRE, LAU, HOU, MER, PNG, PIT, RET, SMR, WAM
N
SOP
Y
NO PNG, QUE
N
Dual Diagnosis
Y
NO CHS, PIT
N
TC
Y
NO GRN, LAU, PNG, QUE, RET, SMR
N
Outpatient
Y
NO CHS, PIT
N
Hispanic TC
Y
CHS
N
DDTC (Co-Occurring Therapeutic Community)
Y
RET
N
Min sentences > 10 yrs
Y
NO WAM
N
Min sentences > 7.5 yrs
Y
NO LAU
N
Min sentences > 5 yrs
Y
NO MER
N
Age >25 & Min>=12 mos & No crimes against kids
N
NO PNG
Y
Assign the inmate to his home region
Institution Candidate

(b) Section two

Figure 5.1: The decision tree of the inmate assignment process.

Chapter 6

The Inmate Transportation Problem

In Chapter 5, we introduced the inmate assignment problem, and presented our work on developing and solving a mathematical optimization model for the problem. In this Chapter, we present our work on another complex problem that correctional systems face on a daily basis, namely, the *Inmate Transportation Problem (ITP)*.

6.1 Introduction

The ITP is an extension of the Vehicle Routing Problems (VRP) [Cordeau et al., 2007, Toth and Vigo, 2014] with several side constraints. The VRP itself is a generalization of the Traveling Salesman Problem (TSP) [Cook, 2011, Flood, 1956]. As the TSP is an NP-hard problem [Cook, 2011], one can conclude that the VRP and the ITP are NP-hard too. In a TSP, a set of nodes and the distance between each pair of the nodes is given, and the goal is to find the shortest route which visits each node once and returns to the starting node. The classic VRP is a well-known problem in combinatorial optimization. The VRP was first proposed by Dantzig and Ramser [1959], who worked on optimum routing of a fleet of gasoline delivery trucks and developed a MILO model and a solution methodology to find near-optimal solutions for the problem. The VRP is concerned with optimally routing a fleet of vehicles to satisfy the demands of a set of customers from a specified depot, see Cordeau et al. [2007], Toth and Vigo [2014] for more information. The Vehicle Routing Problem with Pickups and Deliveries (VRPPD) extends the VRP by sending the goods/passengers from pickup to delivery points [Desaulniers et al., 2002, Dumas et al., 1991, van der Bruggen et al., 1993]. In a VRPPD, a heterogeneous vehicle fleet located at different nodes satisfy transportation requests. A transportation request is specified with an origin, a destination, and demand of the goods/passengers to be transported. The objective function of the VRPPD is mainly to minimize the costs of transportation. Parragh et al. [2008a,b] did an extensive survey on the VRPPD.

The ITP can be cast as a VRPPD with additional constraints. In a regular VRPPD, each node has one transport request, while in the ITP each node can have multiple transport requests to different nodes. Additionally, a transportation hub is considered in the ITP for the transportation of the inmates, while in the VRPPD transportation is done directly from an origin to a destination. Another important difference is that, unlike the VRPPD, in the ITP a node can be visited more than once in a route, which in turn renders the mathematical optimization models of the VRPPD, developed so far, inapplicable for the ITP.

The remainder of the chapter is organized as follows. We formally define the ITP in Section 6.2, and present a MILO model for the ITP in Section 6.3. Then we present our numerical results in Section 6.4 to demonstrate the effectiveness of the model in reducing the costs of the inmate transportation process. We close the chapter with concluding remarks in Section 6.6.

6.2 Problem Description

The Office of Population Management (OPM) is responsible for the transportation of the inmates at the PADoC. On average, 35,000 inmate transportations are scheduled annually between the 25 CIs at the PADoC, yielding about 650 transportations each week. Conventionally, a staff member of OPM with his/her experience and judgment manually makes the decisions about the transportation of inmates. The decisions are made in two main steps. First, the routes are specified for the vehicles, and then inmates are assigned to the vehicles based on their origin and destination CIs. One of the critical restrictions of the manual assignment is that there is a small set of predefined routes, and the trips are currently scheduled based only on those predefined routes. The limited number of predefined routes in the current policy significantly limits the flexibility of transportation decisions. This manual way of transportation planning is clearly not efficient.

Now, we define the ITP and explain the constraints and objective function of the problem. Given a time horizon, the set of inmates who need to be transported and the origin and the destination for each inmate is predefined. In other words, the decision about the assignment of an inmate to a CI is made prior to deciding on his/her transportation. In the ITP, we decide on the vehicles used at each transportation day, their routes, and the number of inmates that are going to be assigned to the vehicles each day.

Vehicles depart from their home CI, visit a sequence of CIs, and return to their home CI, since the vehicles are maintained by the respective CIs, and the drivers need to return home at the end of the day. Trips should be scheduled in the time window [7 a.m., 7 p.m.]. This means that every route should originate and finish at the same CI, and transport inmates within the given 12 hours time window. Considering the travel time limit, there are a few pairs of CIs which can not be visited in a single trip. In order to be able to transport inmates between any two arbitrary CIs, the PADoC has constructed a transfer hub, which is located at the central region of the state. The hub enables the PADoC to move inmates between any two CIs in one day. Additionally, the hub helps to significantly reduce transportation costs.

The time horizon adds another level of complexity to the problem. Currently, planning for the inmate transportation is done once a week, thus, the time horizon considered for the trips is a week. The actual time horizon depends on the frequency of transportation days and the number of inmates which need to be transported. The MILO model allows to consider a longer time horizon.

6.3 Mathematical model

In this section, we introduce the MILO mathematical model. We define the terms and assumptions we have used to develop the MILO model.

Definition 6.1. *A **route** is a sequence of CIs which starts and ends at the same CI. The starting CI of a route is the **origin** of the route, and two consecutive CIs of the route form a **leg**.*

Definition 6.2. *A **trip** is specified with a vehicle along with its capacity and location at a given CI, a given transportation day, and a route. The given CI is the **origin** and the final destination of the trip.*

Definition 6.3. *A **potential trip** is a trip where the vehicle with its capacity, the origin CI, and the transportation day is specified, but the route is not specified.*

In ITP, we define the set of all potential trips. One of the main decisions to be made is to assign a route – if any – to potential trips and use those trips for inmate transportation.

Due to various policy restrictions and business practices we limited the set of possible routes. Following is the set of assumptions used in generating the set of possible routes:

- We allocate a predefined time duration for getting on and off the vehicle at each CI, except for the route origin.
- The hub may only be visited at most once in a route.
- No consecutive pairs of CIs should be visited more than once.
- Only the legs that are currently used by PADoC are considered in generating the set of the routes. In this case the vehicles will travel only on the paths that are approved by the PADoC.

It is worth mentioning that we do not consider special cases of inmate transportation, such as medical transports, since such requests form a small percentage of the total transportation requests, and are handled by special vehicles. We also do not consider over-night stay for an inmate during the transportation, i.e., all the inmates assigned to a trip will reach their destination at the same day.

We have two main objectives. We aim to minimize the number of the allocated trips and minimize the number of inmates not assigned to a trip. Three main decisions are made in the ITP. We need to allocate trips for transportation, assign routes to the allocated trips, and specify the number of inmates that are going to be transported on each trip.

Let $\mathcal{C}$, $\mathcal{R}$, and $\mathcal{P}$ be the set of the CIs, the set of the possible routes, and the set of the potential trips, respectively. Let the decision variable z_{pr}, for $p \in \mathcal{P}$ and $r \in \mathcal{R}$, be equal to 1 if route r is allocated to potential trip p; otherwise, $z_{pr} = 0$. Constraints (6.1) ensure that at most one route is allocated to a potential trip.

$$\sum_{r\in\mathcal{R}} z_{pr} \leq 1, \quad \forall p \in \mathcal{P}. \tag{6.1}$$

Let $\mathcal{K}_{ri}$ be the set of the stop numbers corresponding to CI i in route r. Inmates can either move directly from their origin to their destination in one trip, or they can go through the hub and get to their destination via two trips. Let y_{ijp}, for all $i, j \in \mathcal{C}$ and $p \in \mathcal{P}$, be the number of inmates moving *directly* from the origin CI i to destination CI j on trip p. Also let ν_r, for all $r \in \mathcal{R}$, be the number of stops in the route r. Let $x_{prk_1k_2}$, for all $p \in \mathcal{P}$, $r \in \mathcal{R}$, and $1 \leq k_1 < k_2 \leq \nu_r$, be the number of inmates *directly* going from the k_1-th CI to the k_2-th CI of route r and trip p. The following constraints assure that the number of inmates directly moving between two CIs in a trip is equal to the sum of all the inmates moving between those two CIs in the route allocated to the trip

$$y_{ijp} = \sum_{r\in\mathcal{R}} \sum_{k_1\in\mathcal{K}_{ri}} \sum_{k_2\in\mathcal{K}_{rj}} x_{prk_1k_2}, \quad \forall i, j \in \mathcal{C}, i \neq j, p \in \mathcal{P}.$$

We need to have a bound on the number of inmates moving between any two CIs in a trip. Let ψ_p, for all $p \in \mathcal{P}$, be the seat capacity of the vehicle used on trip p. Let the parameter ω_{ijr}, for all $i, j \in \mathcal{C}$ and $r \in \mathcal{R}$, be equal to 1 if CI i is before CI j in route r; 0, otherwise. Constraints (6.2) ensure that the number of inmates moving between any two CIs is not more than $S^{\max}$, the maximum capacity of the vehicle.

$$y_{ijp} \leq \psi_p \sum_{r\in\mathcal{R}} \omega_{ijr} z_{pr}, \quad \forall i, j \in \mathcal{C}, i \neq j, p \in \mathcal{P}. \tag{6.2}$$

Let $\mathcal{R}'$ be the set of the routes which go through the hub and let s_{prk}, for all $p \in \mathcal{P}$, $r \in \mathcal{R}$, and $k \leq \nu_r$, be equal to the number of inmates at the k-th stop of route r on trip p. Constraints (6.3) and (6.4) enforce the balance of the inmates on the routes that do not go through the hub. Constraints (6.3) represent the balance equations corresponding to the first stop of a route on a trip without hub.

$$s_{pr1} = \sum_{k=1}^{\nu_r} x_{pr1k}, \quad \forall p \in \mathcal{P}, r \in \mathcal{R} \setminus \mathcal{R}'. \tag{6.3}$$

Constraints (6.4) ensure that the number of inmates at each stop of a route in a trip without hub should be equal to the number of inmates at the previous stop of that route plus the number of inmates *getting on* the trip on that stop minus the number of inmates *getting off* the trip on that stop.

$$s_{prk} = s_{pr,k-1} + \sum_{k_2=k+1}^{\nu_r} x_{prkk_2} - \sum_{k_1=1}^{k-1} x_{prk_1k}, \quad \forall p \in \mathcal{P}, r \in \mathcal{R} \setminus \mathcal{R}', 2 \leq k \leq \nu_r. \tag{6.4}$$

Next, we explain the constraints related to the routes that go through the transportation hub. If an inmate goes through the hub, he/she needs to be assigned to two separate trips to get to his/her final destination. The first trip transports the inmate to the hub, and the second picks him/her up from the hub to the destination.

Let η_r, for all $r \in \mathcal{R}'$, be the stop number of the hub in route r. Constraints (6.5)-(6.8) are equivalent to constraints (6.3) and (6.4) for the transportation through the hub. Constraints (6.5)-(6.8) enforce that the number of inmates *getting on* at each stop is equal to the number of inmates at the previous stop plus the ones that are *getting on* at the stop minus the ones that are *getting off* at that stop.

Let u_{prkj}, for all $p \in \mathcal{P}$, $r \in \mathcal{R}'$, $1 \leq k < \eta_r$, and $j \in \mathcal{C}$, be the number of inmates on trip p moving from the k-th CI of route r to the hub with final destination j. Similarly, let v_{prki} for all $p \in \mathcal{P}$, $r \in \mathcal{R}'$, $\eta_r < k \leq \nu_r$ and $i \in \mathcal{C}$, be the number of inmates on trip p moving from the hub to the k-th CI of route r with origin i. Constraints (6.5) represent the balance equations corresponding to the first stop of a route on a trip.

$$s_{pr1} = \sum_{i \in \mathcal{C}} x_{pr1i} + \sum_{k=1}^{\nu_r} u_{pr1k} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}'. \tag{6.5}$$

Constraints (6.6) represent the balance equations when the k-th stop of the route is before the hub, in which $\sum_{i \in \mathcal{C}} u_{prki}$ is the total number of inmates *getting on* at stop n of route r on trip p and *getting off* at the hub.

$$s_{prk} = s_{pr,k-1} + \sum_{k_2>k} x_{prkk_2} - \sum_{k_1<k} x_{prk_1k} + \sum_{i\in\mathcal{C}} u_{prki} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}', 2 \le k < \eta_r. \tag{6.6}$$

Constraints (6.7) enforce the balance equations at the hub.

$$\begin{aligned} s_{pr\eta_r} = \quad & s_{pr,\eta_r-1} + \sum_{k_2=\eta_r+1}^{\nu_r} x_{pr\eta_rk_2} - \sum_{k_1=1}^{\eta_r-1} x_{prk_1\eta_r} \\ & -\sum_{j\in\mathcal{C}} \sum_{k_1=1}^{\eta_r} u_{prk_1j} + \sum_{i\in\mathcal{C}} \sum_{k_2=\eta_r+1}^{\nu_r} v_{prk_2i}, \qquad \forall p \in \mathcal{P}, r \in \mathcal{R}'. \end{aligned} \tag{6.7}$$

Constraints (6.8) enforce the balance equations for the stops after the hub.

$$s_{prk} = s_{pr,k-1} + \sum_{k_2=k+1}^{\nu_r} x_{prkk_2} - \sum_{k_1=1}^{k-1} x_{prk_1k} - \sum_{i\in\mathcal{C}} v_{prki} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}', k > \eta_r. \tag{6.8}$$

We need to consider the vehicle capacity constraints at each stop. Constraints (6.9) enforce a bound on the variable s_{prk}, making sure that at any given point of time during the transportation, the number of inmates on a trip does not exceed the capacity of the vehicle.

$$s_{prk} \le \psi_p z_{pr} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}, k \le \nu_r. \tag{6.9}$$

Furthermore, constraints (6.10) are the capacity constraints for the transportation of inmates through the hub.

$$\begin{aligned} u_{prk_1j} &\le \psi_p z_{pr} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}', 1 \le k_1 < \eta_r, j \in \mathcal{C}, \\ v_{prk_2i} &\le \psi_p z_{pr} \quad \forall p \in \mathcal{P}, r \in \mathcal{R}', \eta_r < k_2 \le \nu_r, i \in \mathcal{C}. \end{aligned} \tag{6.10}$$

Let $\mathcal{T}$ be the set of the days of the transportation and let $\mathcal{P}_t$ be the set of all the potential trips corresponding to day $t \in \mathcal{T}$. Note that $\mathcal{P} = \bigcup_{t\in\mathcal{T}} \mathcal{P}_t$. Constraints (6.11) enforce that, at each transportation day, the total number of inmates going from CI i to the hub with final destination j is equal to the total number of inmates going from the hub to CI j, with the origin i.

$$\sum_{p\in\mathcal{P}_t} \sum_{r\in\mathcal{R}'} \sum_{k_1\in\mathcal{K}_{ri}} u_{prk_1j} = \sum_{p\in\mathcal{P}_t} \sum_{r\in\mathcal{R}'} \sum_{k_2=\mathcal{K}_{rj}} v_{prk_2i} \quad \forall i, j \in \mathcal{C}, i \ne j, t \in \mathcal{T}. \tag{6.11}$$

6.3.1 Objective Function

The ITP is a multi-objective problem. The PADoC primarily uses two types of vehicles to transport inmates between CIs, buses and vans. We consider two main objectives to reduce the number of inmates not transported in a given transportation time period and to reduce the total number of seats utilized for the inmate transportation. Let d_{ij}, for all $i, j \in \mathcal{C}$, be the number of inmates not assigned to any trip which is defined in equation (6.12)

$$d_{ij} = \xi_{ij} - \sum_{p \in \mathcal{P}} y_{ijp} - \sum_{p \in \mathcal{P}} \sum_{r \in \mathcal{R}'} \sum_{k_1 \in \mathcal{K}_{ri}} v_{prk_1j} \quad \forall i, j \in \mathcal{C}, i \neq j. \tag{6.12}$$

Our aim is to minimize the weighted sum of the two objectives of the MILO model presented in (6.13). Here, α is the weight of the total seats used for transportation.

$$\min \quad \alpha \sum_{p \in \mathcal{P}} \sum_{r \in \mathcal{R}} \psi_p z_{pr} + \sum_{i,j \in \mathcal{C} | i \neq j} d_{ij}. \tag{6.13}$$

6.3.2 MILO Model

In this section, we present the MILO model for the ITP. We utilize the weighted sum method to combine the two objectives. The MILO model is as follows:

$$\begin{aligned}
\min \quad & \alpha \sum_{p \in \mathcal{P}} \sum_{r \in \mathcal{R}} \psi_p z_{pr} + \sum_{i,j \in \mathcal{C} | i \neq j} d_{ij}, \\
\text{s.t.} \quad & \sum_{r \in \mathcal{R}} z_{pr} \leq 1, && \forall p \in \mathcal{P}, \\
& y_{ijp} = \sum_{r \in \mathcal{R}} \sum_{k_1 \in \mathcal{K}_{ri}} \sum_{k_2 \in \mathcal{K}_{rj}} x_{prk_1k_2}, && \forall i, j \in \mathcal{C}, i \neq j, p \in \mathcal{P}, \\
& s_{pr1} = \sum_{k=1}^{\nu_r} x_{pr1k}, && \forall p \in \mathcal{P}, r \in \mathcal{R} \setminus \mathcal{R}', \\
& s_{prk} = s_{pr,k-1} + \sum_{k_2=k+1}^{\nu_r} x_{prkk_2} - \sum_{k_1=1}^{k-1} x_{prk_1k}, && \forall p \in \mathcal{P}, r \in \mathcal{R} \setminus \mathcal{R}', 2 \leq k \leq \nu_r,
\end{aligned}$$

$$s_{pr1} = \sum_{i\in\mathcal{C}} x_{pr1i} + \sum_{k=1}^{\nu_r} u_{pr1k} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}',$$

$$s_{prk} = s_{pr,k-1} + \sum_{k_2>k} x_{prkk_2}$$

$$- \sum_{k_1<k} x_{prk_1k} + \sum_{i\in\mathcal{C}} u_{prni} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}', 2\le k<\eta_r,$$

$$s_{pr\eta_r} = s_{pr,\eta_r-1} + \sum_{k_2=\eta_r+1}^{\nu_r} x_{pr\eta_rk_2} - \sum_{k_1=1}^{\eta_r-1} x_{prk_1\eta_r}$$

$$-\sum_{j\in\mathcal{C}}\sum_{k_1=1}^{\eta_r} u_{prk_1j} + \sum_{i\in\mathcal{C}}\sum_{k_2=\eta_r+1}^{\nu_r} v_{prk_2i}, \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}',$$

$$s_{prk} = s_{pr,k-1} + \sum_{k_2=k+1}^{\nu_r} x_{prkk_2}$$

$$- \sum_{k_1=1}^{k-1} x_{prk_1k} - \sum_{i\in\mathcal{C}} v_{prki} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}', k>\eta_r,$$

$$\sum_{p\in\mathcal{P}_t}\sum_{r\in\mathcal{R}'}\sum_{k_1\in\mathcal{K}_{ri}} u_{prk_1j} = \sum_{p\in\mathcal{P}_t}\sum_{r\in\mathcal{R}'}\sum_{k_2=\mathcal{K}_{rj}} v_{prk_2i} \qquad \forall i,j\in\mathcal{C}, i\ne j, t\in\mathcal{T},$$

$$d_{ij} = \xi_{ij} - \sum_{p\in\mathcal{P}} y_{ijp} - \sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}'}\sum_{k_1\in\mathcal{K}_{ri}} v_{prk_1j} \qquad \forall i,j\in\mathcal{C}, i\ne j$$

$$s_{prk} \le \psi_p z_{pr} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}, k\le\nu_r,$$

$$u_{prk_1j} \le \psi_p z_{pr} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}', 1\le k_1<\eta_r, j\in\mathcal{C},$$

$$v_{prk_2i} \le \psi_p z_{pr} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}', \eta_r<k_2\le\nu_r, i\in\mathcal{C},$$

$$y_{ijp} \le \psi_p \sum_{r\in\mathcal{R}} \omega_{ijr} z_{pr}, \qquad \forall i,j\in\mathcal{C}, i\ne j, p\in\mathcal{P},$$

$$z_{pr} = \{0,1\} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R},$$

$$x_{prk_1k_2} \in \mathbb{N} \qquad \forall p\in\mathcal{P}, r\in\mathcal{R}, 1\le k_1<k_2\le\nu_r,$$

$$y_{ijp} \in \mathbb{N} \qquad \forall i,j\in\mathcal{C}, p\in\mathcal{P}, i\ne j,$$

$$s_{prk} \in \mathbb{N} \qquad \forall p \in \mathcal{P}, r \in \mathcal{R}, 1 \leq k \leq \nu_r,$$

$$u_{prkj} \in \mathbb{N} \qquad \forall p \in \mathcal{P}, r \in \mathcal{R}', 1 \leq k \leq \nu_r, j \in \mathcal{C},$$

$$v_{prki} \in \mathbb{N} \qquad \forall p \in \mathcal{P}, r \in \mathcal{R}', 1 \leq k \leq \nu_r, i \in \mathcal{C},$$

$$d_{ij} \in \mathbb{N} \qquad \forall i, j \in \mathcal{C}, i \neq j.$$

The parameters and sets of the ITP are presented in Table 6.1 and the decision variables of the MILO model are summarized in Table 6.2.

Table 6.1: The parameters of the ITP.

Parameters	**Description**
$\mathcal{C}$	Set of all CIs
$\mathcal{R}$	Set of all possible routes
$\mathcal{R}'$	Set of all possible routes visiting the hub
$\mathcal{T}$	Set of days of the transportation
$\mathcal{P}_t$	Set of the potential trips on day t
$\mathcal{P}$	Set of the all the potential trips ($\mathcal{P} = \bigcup_{t \in \mathcal{T}} \mathcal{P}_t$)
$\mathcal{K}_{ri}$	Set of the stops corresponding to CI i on route r
ω_{ijr}	1, if CI i is before CI j on router; 0, otherwise
ν_r	Number of stops (CIs) on route r
η_r	Stop number of the hub on route r if the route visits the hub; ∞, otherwise
ψ_p	Seat capacity of the vehicle used on trip p
ξ_{ij}	Number of inmates that need to move from CI i to CI j
α	Penalty coefficient of the objective function

Table 6.2: The decision variables of the MILO model for the ITP.

Variables	Description
z_{pr}	1, if route r is assigned to potential trip p; 0, otherwise
y_{ijp}	Number of inmates moving directly (without going to the hub) from CI i to CI j on trip p
$x_{prk_1k_2}$	Number of inmates directly going from the k_1-th CI to the k_2-th CI of route r on trip p
u_{prk_1j}	Number of inmates on trip p going from the k_1-th CI of route r to the hub with final destination j
v_{prk_2i}	Number of inmates on trip p going from the hub to the k_2-th CI of route r with origin i
s_{prk}	Number of inmates on the vehicle at the k-th CI of route r on trip p
d_{ij}	Number of inmates that need to move from CI i to CI j, but not assigned to any trip

The ITP is a multi-objective optimization problem. We had to specify and fine-tune the relative weights of the objectives and ensure robustness of the model in assigning inmates to trips for various datasets.

6.4 Computational Results

In this section, we present the computational results and compare the performance of the MILO model with that of the manual transportation process. We used Google Maps API to calculate the pessimistic travel time between the facilities and create the distance matrix, which is then further used to create routes. In order to test the MILO model, A dataset of 4,682 inmates is used which were transported in an eight-week time period, from April 1st, 2018 to May 26th, 2018. The transportation of inmates is scheduled on a weekly basis. The number of inmates which were transported in each week of the time period are presented in Table 6.3.

Table 6.3: The number of inmates transported in each week between April 1st 2018 to May 26th 2018.

Dates	Week Number	Inmates Transported
04/01/2018 - 04/07/2018	1	550
04/08/2018 - 04/14/2018	2	530
04/15/2018 - 04/21/2018	3	668
04/22/2018 - 04/28/2018	4	657
04/29/2018 - 05/05/2018	5	499
05/06/2018 - 05/12/2018	6	554
05/13/2018 - 05/19/2018	7	581
05/20/2018 - 05/26/2018	8	643
Total inmates transported		4682

For computational experiments a computer with Dual Intel Xeon® CPU E5-2630 @ 2.20 GHz (20 cores) and 64 GB of RAM is used. Gurobi [Gurobi Optimization Inc., 2016] is used to solve the MILO model with its default parameters and it is set to use 10 threads. The solution time limit of Gurobi is set to 43,200 seconds (12 hours) for all datasets.

There are two vehicle types, buses and vans, available at the CIs. Depending on their make and model, the capacities of these buses and vans are different. The capacities of buses are generally larger than those of the vans. Since we minimize the total number of seats used for transportation, the model tends to minimize the number of allocated trips with buses as opposed to vans.

In order to evaluate the MILO model and compare its performance with manually scheduling the transportation, we considered the following indicators:

- Total number of trips allocated.

- Total number of buses and vans used in allocated trips.

- Total number of seats in the vehicles used in allocated trips.

- Total number of inmates transported and number of inmates not transported.

- Percentage of inmates using the hub for transportation.

  If an inmate uses the hub in order to be transported to the destination CI then the transportation is done by two trips.

- Seat utilization ratio, which is the number of inmates transported to the total number of seats available in trips for transportation. The seat utilization ratio can be greater than one, since multiple inmates can occupy the same seat in a trip, as they can get on and get off at different stops. We consider two types of seat utilization ratio. In *without-hub seat utilization ratio*, we count all the inmates once, even though the inmates that go through the hub used two vehicles for transportation. In *with-hub seat utilization ratio*, however, we count the inmates that are transported through the hub twice as they take two seats for transportation.

The results of the manual allocation of the trips and the assignment of the inmates to the trips for the 8 weeks are presented in Table 6.4. Notice that the average number of the seats, buses, and vans used for the transportation and the number of inmates transported are rounded up in the table. The detailed results of each week are presented in Table B.1 of Appendix B.

Table 6.4: Average results of the 8 weeks in manual transportation planning.

Trips	Seats used	Buses	Vans	# Inmates transported	% transported through hub	Seat utilization ratio	
						w/o hub	w/ hub
39	912	21	18	585	57.55	0.64	1.00

The parameter α is the coefficient used in the objective function to penalize the allocation of vehicles for transportation. As α increases, the penalty associated with allocating a vehicle for transportation increases. Thus, the number of the allocated trips and more importantly the number of allocated buses for transportation decreases as α increases. There is a trade-off between the two objectives of the model: minimize the number of the inmates not transported and minimize the number of seats used in the allocated trips. The relative penalty of not assigning inmates to trips decreases as α increases. Thus, the number of inmates that are not assigned to a trip increases as α increases. Additionally, the number of inmates assigned to a trip increases, thus the utilization ratio increases. We tested the MILO model for $\alpha \in \{0.10, 0.25, 0.50, 0.75, 1.00\}$. In Table 6.5 the average results of the MILO model for different values of the penalty coefficient α is presented. Notice that the average number of seats, buses, and vans used for the transportation of the inmates is rounded up in the table. The detailed results of the MILO model for each week is presented in Tables B.2-B.9 of Appendix B.

Table 6.5: Average results of the 8 weeks from the MILO model.

α	Trips	Seats used	Buses	Vans	% not moved	% moved	% moved w/ hub	Seat utilization ratio w/o hub	Seat utilization ratio w/ hub
0.10	28	535	12	16	0.13	99.87	40.25	1.10	2
0.25	26	482	10	16	1.49	98.51	39.63	1.20	1.68
0.50	25	437	9	16	4.21	95.79	38.13	1.29	1.78
0.75	23	385	8	16	8.69	91.31	36.00	1.40	1.90
1.00	19	259	5	15	26.47	73.53	23.25	1.66	2.04

One of the important decisions is to select the appropriate value for α. There is a trade-off between the number of vehicles used for transportation and the number of the inmates moved. Our aim is to choose an α which leads to a small number of inmates not transported while a small number of trips are used to transport the inmates.

From Table 6.5, we can observe that as α increases the percentage of inmates not transported increases. Since for the data that we consider, all the inmates are already transported in those respective weeks, we need to make sure that the number of inmates not transported is small. When $\alpha = 0.25$, on average, %1.49 of the inmates are not transported, which is deemed acceptable, since those few inmates would be transported a week later than their scheduled transportation. Thus, among the values considered for α in Table 6.5, the most appropriate value of α was determined to be 0.25.

When $\alpha = 0.25$, on average, 26 trips are allocated each week for the inmate transportation and less than 1.4% of inmates are not transported. The inmates who were not assigned to any trip can be transported in the following week. As seen in Tables 6.4 and 6.5, on average, weekly transportation of inmates can be done by using less than half of the buses and 3 fewer vans. Furthermore, the without-hub seat utilization for the MILO model with $\alpha = 0.25$, is nearly twice of that for the manual transportation planning, and with-hub seat utilization is increased by %60. Thus, the optimized transportation can significantly improve the seat utilization ratio, while on average less than 1.4 % of inmates are not assigned to trips on a week.

In Table 6.6, the worst case of the manual transportation planning during the 8 weeks is compared with that of utilizing the MILO model with $\alpha = 0.25$ for inmate transportation. In the worst case analysis, the number of the buses is the maximum number of buses that are used in a week for the inmate transportation, and so is the number of the vans in the worst case calculated. In the worst case the MILO model, for $\alpha = 0.25$, allocates 10 less buses and 5 less vans to transport inmates in a week.

Table 6.6: Worst case analysis of manual transportation planning and the MILO model with $\alpha = 0.25$.

	Number of Buses	Number of Vans
Manual	22	21
MILO model	12	16

As we can see in Tables B.2-B.9, none of the problems are solved to global optimality. As the decisions about inmate transportation are made on a weekly basis, we can let the solver Gurobi run for longer time duration than 12 hours to obtain better solutions.

6.5 Benefits and Impact

In this section, we quantify the expected savings that can be achieved by using the MILO model for the inmate transportation process. We have identified two main areas of savings that can be achieved by optimizing the transportation process. In order to compute the savings, we compare the average and the worst case of manual transportation planning with that of the MILO model output.

- **Maintenance and gas**: It was reported by the PADoC in 2013 that the cost maintenance and gas for 21 buses that were used for transportation was \$500,000. On average, the number of the buses used for transportation reduced from 21 to 10. The model reduces the number of buses by 11. Thus, the savings from the maintenance and gas is projected to be \$261,900 annually. In the worst case, the number of buses used for transportation reduced by 10. In the worst case scenario the MILO model results in a saving of \$238,000 annually.

- **Salary**: Each bus and van, used for the transportation, needs three and two correctional officers, respectively. The average annual salary and benefits of a correctional officer is \$135,000. On average, the number of buses and vans used for transportation reduced from 21 and 18 to 10 and 15, respectively. There is a reduction of 11 bus-trips and 3 van-trips weekly. This would translate in a saving of 39 man-day on a weekly basis which is equivalent to 7.8 full-time correctional officer positions. Thus, the saving from the salary would be \$1,053,000, annually. For the worst case, the number of buses and vans used for transportation reduced by 10 and 5, respectively. This could translate in a saving of 40 man-day weekly, which is equivalent to 8 full-time correctional officer positions. Thus, the saving would be \$1,080,000, annually.

The average projected quantified savings in one year and over five years are summarized in Table 6.7. The quantified savings for the comparison between the worst case scenario of the manual transportation process and the worst case scenario of the MILO model output is presented Table 6.8.

Table 6.7: Average quantified savings.

Savings	One Year (\$)	Five years (\$)
Gas & Maintenance	261,900	1,309,500
Salary	1,053,000	5,062,500
Sum	1,314,900	6,574,500

Table 6.8: Worst case quantified savings.

Savings	One Year ($)	Five years ($)
Gas & Maintenance	238,000	1,190,000
Salary	1,080,000	5,400,000
Sum	1,318,000	6,590,000

Besides, the hub is visited on average by %57 of the inmates in the manual transportation planning, while in the solution of the MILO model with $\alpha = 0.1$, the hub is visited by %39 of the inmates, which is more than %30 reduction in hub usage for the transportation of the inmates and can also contribute to significant savings. Another big saving can be achieved by reducing overtime salaries of correctional officers required in transports. Often trips are scheduled for irregular time leading to required extra hours for the correctional officers. Overtime salaries are significantly higher than the normal work hour salaries. Discussions with the PADoC indicate that overtime payments have become a significant monetary burden on the PADoC. To quantify the savings for reduced use of the hub and reduced overtime payment requires the collection and analysis of additional data. The quantification of these savings remains for future analysis.

Conventionally, the PADoC uses a set of about 40 routes to transfer inmates. The model, however, chooses routes from a set of about 1200 possible routes. Regularly changing the routes and letting the model decide the routes make the entire transportation process more efficient and safer, since it becomes harder to identify the pattern of the inmate transportation within the State of PA.

6.6 Conclusion

In this chapter, we presented our work on the ITP. We studied and formalized the inmate transportation process and developed a multi-objective MILO model for the ITP. Numerical results indicate that significant savings can be achieved by using the MILO model in inmate transportation planning. This study was done as a proof of concept for the project of optimizing the inmate transportation process. The MILO model can further be advanced to incorporate other business rules and constraints of the inmate transportation process, and can, additionally, be adapted to other jurisdictions.

Chapter 7

Conclusions and Future Research

In this thesis, we have investigated the theory and applications of Mixed Integer Conic Optimization (MICO). The work is divided into three main areas: Disjunctive Conic Cuts (DCCs) for Mixed Integer Second Order Cone Optimization (MISOCO) problems; developing new models and a novel solution methodology for large-scale discrete truss design problems; and application of mathematical optimization in correctional industries.

7.1 Conclusions

In Chapter 2, we presented two fundamental classes of pathological disjunctions for MICO problems, where the disjunctive cuts do not cut-off any part of the original sets. Then we utilized the pathological disjunctions to identify redundant DCCs and DCyCs for MISOCO problems. We know that if the DCC is redundant, then any other disjunctive cut will be redundant too, since the DCC is the tightest possible disjunctive cut describing the convex hull of the disjunctive set. We also showed how those two cases are the building blocks of more complex instances.

We illustrated that by analyzing some instances in Section 2.4, and showing how those were combinations of the two fundamental pathological disjunctions. Efficient implementation of Branch and Conic Cut (BCC) algorithms for MISOCO requires the identification of pathological disjunctions. In a BCC framework, it is important to keep under control the growth of the problem. For that reason, identifying whether a DCC is redundant, before adding it to the formulation, is essential to obtain an efficient implementation of this methodology.

In Chapters 3 and 4, we presented our work on truss design problems. We proposed several mathematical models for the continuous and discrete truss design problems, and we proposed our novel solution methodology to solve discrete truss sizing problems.

In Chapter 3, we overviewed the continuous truss sizing problem with force balance equations, Hooke's law, yield stress and Euler buckling constraints, and bounds on the nodal displacements and cross-sectional areas of the bars. The resulting model is a non-linear non-convex problem. We presented two MILO models which provide a lower bound for the optimal objective value of the problem. Then we proposed several MILO models for the discrete truss sizing optimization problem and discrete truss topology design and sizing optimization problems, and we extended all the models to account for multi-scenario design problems.

In Chapter 4, we proposed the novel Neighborhood Search Mixed Integer Linear Optimization (NS-MILO) methodology to solve the truss sizing problems, where a sequence of MILO subproblems in a moving neighborhood search framework are solved. It is impossible to get proven optimal solutions for problems of practical relevance. The NS-MILO methodology enables us to provide high-quality solutions for previously unsolvable truss design problems. A variety of different classes of truss problems, including 2D and 3D cantilever trusses, and airplane wing trusses are solved with the NS-MILO approach. Numerical results indicate that the NS-MILO approach is significantly faster than simply using a MILO solver to solve the original truss sizing problems. Additionally, for large scale trusses, the solution of the NS-MILO approach is significantly better than the solution obtained from attempting to directly solve the original problems.

In Chapters 5 and 6, we presented our pioneer work on the application of mathematical optimization in the correctional industry. Specifically, we addressed two important problems that every correctional system faces on a daily basis: the Inmate Assign Problem (IAP) and the Inmate Transportation Problem (ITP).

In Chapter 5, we presented our results on the IAP. The IAP is an assignment problem with various constraints, including general assignment factors, capacity constraints, scheduling of treatment programs for the inmates, etc. Due to limited resources, it is impossible to make an ideal assignment and satisfy all the constraints of the assignment; thus, the IAP is inherently an infeasible problem. We developed a novel hierarchical multi-objective MILO model for the IAP. We penalize the sum of the weighted violation of the constraints of the assignment in the multi-objective MILO model, which is in fact the heart of the Inmate Assignment Decision Support System (IADSS). The IADSS is a decision support system which enables the PADoC to simultaneously and optimally assign the inmates to the correctional institutions, and schedule their treatment programs, while all the rules and criteria of the assignment are considered. The IADSS minimizes the waiting time of the inmates for getting into the required program(s); hence, it facilitates the timely eligibility of the inmates for parole, which ultimately reduces the population of inmates in the correctional system. The IADSS has been used with proven success at the PADoC for the daily assignment of the inmates to CIs since September 2016. The IADSS has decreased the annual cost at PADoC by $\$2,958,617$, and the projected saving over five years is $\$19,199,485$. To our best knowledge, this is the first time that OR methodology is built directly into the routine business operations of a correctional system.

In Chapter 6, we presented our work on the ITP. We studied and formalized the inmate transportation process and developed a multi-objective MILO model for the ITP. Numerical results indicate that significant savings can be achieved by using the MILO model in inmate transportation planning.

7.2 Future research

As a future work to Chapter 2, one can embed the identification of pathological disjunctions and redundant DCCs in the implementation of BCC algorithms with the goal to identify those cases and situations where the addition of DCCs can not tighten the formulation of the problem.

As a future work to Chapter 4, the NS-MILO approach can be extended to solve the discrete truss topology design and sizing optimization problems. Additionally, an other model of airplane wing design includes a discrete ply-angle structure, where the wing is modeled as a multi-layer multi-element plate structure, and the topology and fiber orientation of the structure need to be decided. The discrete ply-angle problem can be modeled as a MISOCO problem, which is challenging to solve when the size of the problem grows. Thus, research need to be done on developing mathematical optimization models and efficient solution methodologies for discrete ply-angle problems.

The success of the inmate assignment project, presented in Chapter 5, opens new avenues to: a) adapt and introduce the IADSS methodology to optimize the operations of correctional systems of other states and countries and, b) explore other applications of OR methodologies in the complex operations of correctional systems. Correctional systems in this nation and elsewhere have numerous problems that cry out for the use of OR methodologies. This highly successful application of OR in a large correctional system opens a rich application area of OR, just as the first crew scheduling application did in the airline industry.

The work on the ITP, presented in Chapter 6, was done as a proof of concept for the project of optimizing the inmate transportation process. The mathematical optimization model developed for the ITP can further be advanced to incorporate other business rules and constraints of the inmate transportation process, and can be adapted to other jurisdictions. Last but not least, having developed the model to solve the inmate transportation problem, it can be integrated with the IADSS to optimize the assignment and transportation of the inmates system-wide.

Bibliography

Achterberg, T., Koch, T., and Martin, A. (2005). Branching rules revisited. *Operations Research Letters*, 33(1):42–54.

Achtziger, W. (1999a). Local stability of trusses in the context of topology optimization part I: Exact modelling. *Structural Optimization*, 17(4):235–246.

Achtziger, W. (1999b). Local stability of trusses in the context of topology optimization part II: A numerical approach. *Structural Optimization*, 17(4):247–258.

Achtziger, W., Bendsøe, M. P., Ben-Tal, A., and Zowe, J. (1992). Equivalent displacement based formulations for maximum strength truss topology design. *IMPACT of Computing in Science and Engineering*, 4(4):315–345.

Achtziger, W. and Stolpe, M. (2006). Truss topology optimization with discrete design variables—guaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization*, 34(1):1–20.

Achtziger, W. and Stolpe, M. (2007a). Global optimization of truss topology with discrete bar areas—part I: theory of relaxed problems. *Computational Optimization and Applications*, 40(2):247–280.

Achtziger, W. and Stolpe, M. (2007b). Global optimization of truss topology with discrete bar areas—part II: Implementation and numerical results. *Computational Optimization and Applications*, 44(2):315–341.

Achtziger, W. and Stolpe, M. (2007c). Truss topology optimization with discrete design variables—guaranteed global optimality and benchmark examples. *Structural and Multidisciplinary Optimization*, 34(1):1–20.

Aktürk, M. S., Atamtürk, A., and Gürel, S. (2014). Aircraft rescheduling with cruise speed control. *Operations Research*, 62(4):829–845.

Al-Khayyal, F. A. and Falk, J. E. (1983). Jointly constrained biconvex programming. *Mathematics of Operations Research*, 8(2):273–286.

Andersen, K. and Jensen, A. N. (2013). Intersection cuts for mixed integer conic quadratic sets. In Goemans, M. and Correa, J., editors, *Integer Programming and Combinatorial Optimization*, volume 7801 of *Lecture Notes in Computer Science*, pages 37–48. Springer.

Arabeyre, J. P., Fearnley, J., Steiger, F. C., and Teather, W. (1969). The airline crew scheduling problem: a survey. *Transportation Science*, 3(2):140–163.

Atamtürk, A. and Narayanan, V. (2010). Conic mixed-integer rounding cuts. *Mathematical Programming*, 122(1):1–20.

Atamtürk, A. and Narayanan, V. (2011). Lifting for conic mixed-integer programming. *Mathematical Programming*, 126(2):351–363.

Atkočiūnas, J., Merkevičiūtė, D., and Venskus, A. (2008). Optimal shakedown design of bar systems: Strength, stiffness and stability constraints. *Computers & Structures*, 86(17):1757–1768.

Balas, E. (1971). Intersection cuts - a new type of cutting planes for integer programming. *Operations Research*, 19(1):19–39.

Balas, E., Ceria, S., and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0–1 programs. *Mathematical Programming*, 58(1):295–324.

Barbosa, H. J., Lemonge, A. C., and Borges, C. C. (2008). A genetic algorithm encoding for cardinality constraints and automatic variable linking in structural optimization. *Engineering Structures*, 30(12):3708–3723.

Belotti, P., Góez, J. C., Pólik, I., Ralphs, T. K., and Terlaky, T. (2013). On families of quadratic surfaces having fixed intersections with two hyperplanes. *Discrete Applied Mathematics*, 161(16-17):2778–2793.

Belotti, P., Góez, J. C., Pólik, I., Ralphs, T. K., and Terlaky, T. (2015). A conic representation of the convex hull of disjunctive sets and conic cuts for integer second order cone optimization. In *Numerical Analysis and Optimization*, volume 134 of *Springer Proceedings in Mathematics & Statistics*, pages 1–35. Springer International Publishing.

Belotti, P., Góez, J. C., Pólik, I., Ralphs, T. K., and Terlaky, T. (2017). A complete characterization of disjunctive conic cuts for mixed integer second order cone optimization. *Discrete Optimization*, 24:3–31.

Ben-Tal, A. and Nemirovski, A. (2001). *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*, volume 2. SIAM.

Bendsøe, M. P. and Ben-Tal, A. (1993). Truss topology optimization by a displacements based optimality criterion approach. In Rozvany, G., editor, *Optimization of Large Structural Systems*, volume 231 of *NATO ASI Series*, pages 139–155. Springer.

Bendsøe, M. P., Ben-Tal, A., and Zowe, J. (1994). Optimization methods for truss geometry and topology design. *Structural Optimization*, 7(3):141–159.

Bergamini, M. L., Aguirre, P., and Grossmann, I. (2005). Logic-based outer approximation for globally optimal synthesis of process networks. *Computers & Chemical Engineering*, 29(9):1914 – 1933.

Bertsekas, D. P. (2009). *Convex Optimization Theory.* Athena Scientific Belmont.

Bertsimas, D. and Shioda, R. (2009). Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications*, 43(1):1–22.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization*, volume 6. Athena Scientific Belmont, MA.

Bienstock, D. and Munoz, G. (2014). On linear relaxations of OPF problems. *Preprint arXiv:1411.1120.*

Bland, J. A. (2001). Optimal structural design by ant colony optimization. *Engineering Optimization*, 33(4):425–443.

Brooks, T. R., Kenway, G. K. W., and Martins, J. R. R. A. (2018). uCRM: An aerostructural model for the study of flexible transonic aircraft wings. *AIAA Journal.* (In press).

Cai, J. and Thierauf, G. (1993). Discrete optimization of structures using an improved penalty function method. *Engineering Optimization*, 21(4):293–306.

Camp, C. and Farshchin, M. (2014). Design of space trusses using modified teachingâĂŞlearning based optimization. *Engineering Structures*, 62-63:87–97.

Camp, C. V. (2007). Design of space trusses using big bang&#x2013;big crunch optimization. *Journal of Structural Engineering*, 133(7):999–1008.

Camp, C. V. and Bichon, B. J. (2004a). Design of space trusses using ant colony optimization. *Journal of Structural Engineering*, 130(5):741–751.

Camp, C. V. and Bichon, B. J. (2004b). Design of space trusses using ant colony optimization. *Journal of Structural Engineering*, 130(5):741–751.

Caprara, A., Toth, P., Vigo, D., and Fischetti, M. (1998). Modeling and solving the crew rostering problem. *Operations Research*, 46(6):820–830.

Cerveira, A., Agra, A., Bastos, F., and Gromicho, J. (2009). New branch and bound approaches for truss topology design with discrete areas. In Long, C., Sohrab, S. H., Bognar, G., and Perlovsky, L., editors, *Proceedings of the American Conference on Applied Mathematics. Recent Advances in Applied Mathematics*, pages 228–233.

Çezik, M. T. and Iyengar, G. (2005). Cuts for mixed 0-1 conic programming. *Mathematical Programming*, 104(1):179–202.

Cohen, M. A. (2005). *The Costs of Crime and Justice*. Routledge.

Cook, W. (2012). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press.

Cook, W. J. (2011). *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D. (2007). Chapter 6 vehicle routing. In Barnhart, C. and Laporte, G., editors, *Transportation*, volume 14 of *Handbooks in Operations Research and Management Science*, pages 367–428. Elsevier.

Crowder, H., Johnson, E. L., and Padberg, M. (1983). Solving large-scale zero-one linear programming problems. *Operations Research*, 31(5):803–834.

Dadush, D., Dey, S. S., and Vielma, J. P. (2011). The split closure of a strictly convex body. *Operations Research Letters*, 39(2):121–126.

Dakin, R. J. (1965). A tree-search algorithm for mixed integer programming problems. *The Computer Journal*, 8(3):250–255.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Operations Research*, 2(4):393–410.

Dantzig, G. B. (1951). Application of the simplex method to a transportation problem. In Koopmans, T. C., editor, *Activity Analysis of Production and Allocation*, volume 13, pages 359–373. John Wiley and Sons.

Dantzig, G. B. (1957). Discrete-variable extremum problems. *Operations Research*, 5(2):266–288.

Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1):80–91.

Davis, L., Bozick, R., and Steele, J. (2013). *Evaluating the Effectiveness of Correctional Education: A Meta-Analysis of Programs That Provide Education to Incarcerated Adults*. RAND Corporation.

De Klerk, E., Roos, C., and Terlaky, T. (1995). *Semi-definite problems in truss topology optimization*. Delft University of Technology, Faculty of Technical Mathematics and Informatics, Report 95-128.

Desaulniers, G., Desrosiers, J., Erdmann, A., Solomon, M. M., and Soumis, F. (2002). VRP with pickup and delivery. In Toth, P. and Vigo, D., editors, *The Vehicle Routing Problem*, chapter 9, pages 225–242. SIAM.

Dorn, W. S., Gomory, R. E., and Greenberg, H. J. (1964). Automatic design of optimal structures. *Journal de Mecanique*, 3:25–52.

Drewes, S. (2009). *Mixed Integer Second Order Cone Programming*. PhD thesis, Technische Universität, Darmstadt, Germany.

Dumas, Y., Desrosiers, J., and Soumis, F. (1991). The pickup and delivery problem with time windows. *European Journal of Operational Research*, 54(1):7–22.

Flood, M. M. (1953). On the Hitchcock distribution problem. *Pacific Journal of Mathematics*, 3(2):369–386.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, 4(1):61–75.

Giambanco, F. and Palizzolo, L. (1995). Optimality conditions for shakedown design of trusses. *Computational Mechanics*, 16(6):369–378.

Gill, P. E., Murray, W., and Saunders, M. A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131.

Glover, F. (1975). Improved linear integer programming formulations of nonlinear integer problems. *Management Science*, 22(4):455–460.

Glover, F. (1984). An improved MIP formulation for products of discrete and continuous variables. *Journal of Information and Optimization Sciences*, 5(1):69–71.

Góez, J. C. (2013). *Mixed Integer Second Order Cone Optimization Disjunctive Conic Cuts: Theory and Experiments*. PhD thesis, Lehigh University.

Góez, J. C. and Anjos, M. F. (2018). Second order conic optimization formulations for service system design problems with congestion. In Pinter, J. D. and Terlaky, T., editors, *Modeling and Optimization: Theory and Applications - 2017 MOPTA Conference , Selected Contributions*. Springer.

Gomory, R. (1960a). An algorithm for the mixed integer problem. Technical report, Santa Monica, Calif.: RAND Corporation, RM-2597-PR.

Gomory, R. E. (1958). Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278.

Gomory, R. E. (1960b). Solving linear programming problems in integers. *Combinatorial Analysis*, 10:211–215.

Gomory, R. E. (1963). An algorithm for integer solutions to linear programs. In Graves, R. L. and Wolfe, P., editors, *Recent Advances in Mathematical Programming*, volume 64, pages 260–302. New York.

Gounaris, C. E., Misener, R., and Floudas, C. A. (2009). Computational comparison of piecewiseâĹŠlinear relaxations for pooling problems. *Industrial & Engineering Chemistry Research*, 48(12):5742–5766.

Gurobi Optimization Inc. (2016). Gurobi optimizer reference manual.

Haftka, R. T. and Gürdal, Z. (2012). *Elements of Structural Optimization*. Springer Science & Business Media.

Hajela, P. and Lee, E. (1995). Genetic algorithms in truss topological optimization. *International Journal of Solids and Structures*, 32(22):3341–3357.

Hanafi, S. (2016). New variable neighbourhood search based 0-1 MIP heuristics. *Yugoslav Journal of Operations Research*, 25(3).

Hansen, P. and Mladenović, N. (2003). Variable neighborhood search. In Glover, F. and Kochenberger, G. A., editors, *Handbook of Metaheuristics*, pages 145–184, Boston, MA. Springer US.

Ho-Huu, V., Nguyen-Thoi, T., Vo-Duy, T., and Nguyen-Trang, T. (2016). An adaptive elitist differential evolution for optimization of truss structures with discrete design variables. *Computers and Structures*, 165:59–75.

IBM Knowledge Center (2017). IBM ILOG CPLEX Optimization Studio V12.8.0 documentation.

Jabr, R. A., Singh, R., and Pal, B. C. (2012). Minimum loss network reconfiguration using mixed-integer convex programming. *IEEE Transactions on Power Systems*, 27(2):1106–1115.

Kaliszky, S. and Lógó, J. (2002). Plastic behaviour and stability constraints in the shakedown analysis and optimal design of trusses. *Structural and Multidisciplinary Optimization*, 24(2):118–124.

Karuppiah, R. and Grossmann, I. E. (2006). Global optimization for the synthesis of integrated water systems in chemical processes. *Computers & Chemical Engineering*, 30(4):650 – 673.

Kaveh, A., Azar, B. F., and Talatahari, S. (2008). Ant colony optimization for design of space trusses. *International Journal of Space Structures*, 23(3):167–181.

Kaveh, A. and Ghazaan, M. I. (2015). A comparative study of CBO and ECBO for optimal design of skeletal structures. *Computers & Structures*, 153:137–147.

Kaveh, A. and Kalatjari, V. (2004). Size/geometry optimization of trusses by the force method and genetic algorithm. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 84(5):347–357.

Kaveh, A. and Mahdavi, V. (2014). Colliding bodies optimization method for optimum discrete design of truss structures. *Computers & Structures*, 139:43–53.

Kaveh, A. and Talatahari, S. (2009). A particle swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research*, 65(8):1558–1568.

Kılınç-Karzan, F. and Yıldız, S. (2014). Two-term disjunctions on the second-order cone. In Lee, J. and Vygen, J., editors, *Integer Programming and Combinatorial Optimization*, volume 8494 of *Lecture Notes in Computer Science*, pages 345–356. Springer International Publishing.

Kripka, M. (2004). Discrete optimization of trusses by simulated annealing. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 26:170–173.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.

Kyckelhahn, T. and Martin, T. (2010). Justice expenditure and employment extracts, 2010 - preliminary. http://www.bjs.gov/index.cfm?ty=pbdetail&iid=4679. Accessed: 2017-08-15.

Land, A. H. and Doig, A. G. (1960). An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520.

Lazić, J. (2010). *New variants of variable neighbourhood search for 0-1 mixed integer programming and clustering*. PhD thesis, Brunel University, School of Information Systems, Computing and Mathematics.

Li, D., Plebani, L., Terlaky, T., Wilson, G. R., and Bucklen, K. B. (2014). Inmate classification: decision support tool gives help to pennsylvania department of corrections. *Industrial Engineer*, 46:44–48.

Li, L., Huang, Z., and Liu, F. (2009). A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures*, 87(7):435–443.

Mahfouz, S. Y. (1999). *Design optimization of structural steelwork*. PhD thesis, University of Bradford, United Kingdom.

Mai, C. and Subramanian, R. (2017). The price of prisons: examining state spending trends, 2010-2015. `https://storage.googleapis.com/vera-web-assets/downloads/Publications/price-of-prisons-2015-state-spending-trends/legacy_downloads/the-price-of-prisons-2015-state-spending-trends.pdf`. Accessed: 2017-08-15.

McCormick, G. P. (1976). Computability of global solutions to factorable nonconvex programs: Part 1 convex underestimating problems. *Mathematical Programming*, 10(1):147–175.

Mela, K. (2014). Resolving issues with member buckling in truss topology optimization using a mixed variable approach. *Structural and Multidisciplinary Optimization*, 50(6):1037–1049.

Mellaert, R. V., Mela, K., Tiainen, T., Heinisuo, M., Lombaert, G., and Schevenels, M. (2017). Mixed-integer linear programming approach for global discrete sizing optimization of frame structures. *Structural and Multidisciplinary Optimization*, 57(2):579–593.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100.

Modaresi, S., Kılınç, M. R., and Vielma, J. P. (2015). Split cuts and extended formulations for mixed integer conic quadratic programming. *Operations Research Letters*, 43(1):10–15.

MOSEK (2017). *The MOSEK Optimization Suite. Version 8.1 (Revision 34).*

Nemhauser, G. L. and Wolsey, L. A. (1990). A recursive procedure to generate all cuts for 0-1 mixed integer programs. *Mathematical Programming*, 46(1-3):379–390.

Orden, A. (1951). A procedure for handling degeneracy in the transportation problem. *DCS/Comptroller, Headquarters US Air Force, Washington, DC.*

Parragh, S., Doerner, K., and Hartl, R. (2008a). A survey on pickup and delivery problems, part ii: transportation between pickup and delivery locations. j für betriebswirtschaft 58 (2): 81–117. *doi. org/10.1007/s1130*, pages 1–008.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008b). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51.

Petersen, C. C. (1971). A note on transforming the product of variables to linear form in linear programs. *Working Paper, Purdue University,.*

Petrovic, N., Kostic, N., and Marjanovic, N. (2017). comparison of approaches to 10 bar truss structural optimization with included buckling constraints. *Applied Engineering Letters*, 2:98–103.

Rahami, H., Kaveh, A., and Gholipour, Y. (2008). Sizing, geometry and topology optimization of trusses via force method and genetic algorithm. *Engineering Structures*, 30(9):2360–2369.

Rajeev, S. and Krishnamoorthy, C. S. (1992). Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5):1233–1250.

Rajeev, S. and Krishnamoorthy, C. S. (1997). Genetic algorithms-based methodologies for design optimization of trusses. *Journal of Structural Engineering*, 123(3):350–358.

Rasmussen, M. and Stolpe, M. (2008). Global optimization of discrete truss topology design problems using a parallel cut-and-branch method. *Computers & Structures*, 86(13):1527–1538.

Rockafellar, R. T. (1997). *Convex Analysis*. Princeton University Press.

Sadollah, A., Bahreininejad, A., Eskandar, H., and Hamdi, M. (2012). Mine blast algorithm for optimization of truss structures with discrete variables. *Computers & Structures*, 102-103:49–63.

Sadollah, A., Eskandar, H., Bahreininejad, A., and Kim, J. H. (2015). Water cycle, mine blast and improved mine blast algorithms for discrete sizing optimization of truss structures. *Computers & Structures*, 149:1–16.

Sawaragi, Y., Nakayama, H., and Tanino, T. (1985). *Theory of Multiobjective Optimization*, volume 176 of *Mathematics in Science and Engineering*. Elsevier.

Schanzenbach, D. W., Nunn, R., Bauer, L., Breitwieser, A., Mumford, M., and Nantz, G. (2016). *Twelve Facts about Incarceration and Prisoner Reentry*. The Brookings Institution, Washington, DC.

SeokLee, K. and Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9–10):781–798.

Sonmez, M. (2011). Discrete optimum design of truss structures using artificial bee colony algorithm. *Structural and Multidisciplinary Optimization*, 43(1):85–97.

Stolpe, M. (2004). Global optimization of minimum weight truss topology problems with stress, displacement, and local buckling constraints using branch-and-bound. *International Journal for Numerical Methods in Engineering*, 61(8):1270–1309.

Stolpe, M. (2007). On the reformulation of topology optimization problems as linear or convex quadratic mixed 0–1 programs. *Optimization and Engineering*, 8(2):163–192.

Stolpe, M. (2011). To bee or not to bee—comments on "discrete optimum design of truss structures using artificial bee colony algorithm". *Structural and Multidisciplinary Optimization*, 44(5):707–711.

Stolpe, M. (2016). Truss optimization with discrete design variables: a critical review. *Structural and Multidisciplinary Optimization*, 53(2):349–374.

Stubbs, A. R. and Mehrotra, S. (1999). A branch-and-cut method for 0-1 mixed convex programming. *Mathematical Programming*, 86(3):515–532.

Svanberg, K. and Werme, M. (2005). A hierarchical neighbourhood search method for topology optimization. *Structural and Multidisciplinary Optimization*, 29(5):325–340.

Svanberg, K. and Werme, M. (2007). Sequential integer programming methods for stress constrained topology optimization. *Structural and Multidisciplinary Optimization*, 34(4):277–299.

Tawarmalani, M. and Sahinidis, N. V. (2004). Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99(3):563–591.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. SIAM.

van der Bruggen, L. J. J., Lenstra, J. K., and Schuur, P. C. (1993). Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science*, 27(3):298–311.

Van Roy, T. J. and Wolsey, L. A. (1987). Solving mixed integer programming problems using automatic reformulation. *Operations Research*, 35(1):45–57.

Vinel, A. and Krokhmal, P. A. (2014). Polyhedral approximations in p-order cone programming. *Optimization Methods and Software*, 29(6):1210–1237.

Votaw, D. F. and Orden, A. (1952). The personnel assignment problem. *Symposium on Linear Inequalities and Programming, SCOOP 10, USAF*, pages 155–163.

Wächter, A. and Biegler, L. T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57.

Walmsley, R. (2017). World prison population list. http://www.prisonstudies.org/sites/default/files/resources/downloads/world_prison_population_list_11th_edition_0.pdf. Accessed: 2017-08-15.

Wu, S.-J. and Chow, P.-T. (1995). Steady-state genetic algorithms for discrete optimization of trusses. *Computers and Structures*, 56(6):979–991.

Zeng, S. and Li, L. a. (2012). Particle swarm-group search algorithm and its application to spatial structural design with discrete variables. *International Journal of Optimization in Civil Engineering*, 2(4):443–458.

Appendix A

The discrete set of the cross sectional areas

The discrete set of the cross-sectional areas for the various problems are listed below.

10-bar truss

$$\begin{aligned} S = \; & \{1.62,\ 1.8,\ 1.99,\ 2.13,\ 2.38,\ 2.62,\ 2.63,\ 2.88,\ 2.93, \\ & 3.09,\ 3.13,\ 3.38,\ 3.47,\ 3.55,\ 3.63,\ 3.84,\ 3.87,\ 3.88, \\ & 4.18,\ 4.22,\ 4.49,\ 4.59,\ 4.80,\ 4.97,\ 5.12,\ 5.74,\ 7.22, \\ & 7.97,\ 11.50, 13.50,\ 13.90, 14.20,\ 15.50,\ 16.00, 16.90, \\ & 18.80,\ 19.90,\ 22.00, 22.90,\ 26.50,\ 30.00,\ 33.5,\ 35, \\ & 37.5,\ 40,\ 42.5,\ 45,\ 47.5,\ 50,\ 52.5,\ 55,\ 57.5,\ 60, \\ & 62.5,\ 65,\ 67.5,\ 70,\ 72.5,\ 75,\ 77.5,\ 80,\ 82.5,\ 85, \\ & 87.5,\ 90\ \}(\mathrm{in}^2). \end{aligned}$$

72-bar truss

$$\begin{aligned} S = \{ \;& 0.111,\ 0.141,\ 0.196,\ 0.250,\ 0.307,\ 0.391,\ 0.442,\\ & 0.563,\ 0.602,\ 0.766, .785,\ .994,\ 1,\ 1.228,\ 1.266,\\ & 1.457,\ 1.563,\ 1.62,\ 1.8,\ 1.99,\ 2.13,\ 2.38,\ 2.62,\\ & 2.63,\ 2.88,\ 2.93,\ 3.09,\ 3.13,\ 3.38,\ 3.47,\ 3.55,\\ & 3.63,\ 3.84,\ 3.87,\ 3.88,\ 4.18,\ 4.22,\ 4.49,\ 4.59,\\ & 4.8,\ 4.97,\ 5.12,\ 5.74,\ 7.22,\ 7.97,\ 8.53,\ 9.3,\\ & 10.85,\ 11.5,\ 13.5,\ 13.9,\ 14.2,\ 15.5,\ 16,\ 16.9,\\ & 18.8, 19.9,\ 22,\ 22.9, 24.5,\ 26.5, 28,\ 30, 33.5\ \}(\mathrm{in}^2). \end{aligned}$$

2D cantilever truss problem

$$\begin{aligned} S = \{ \;& 0.25,\ 0.5,\ 0.75,\ 1,\ 2,\ 3,\ 4,\ 6,\ 7,\ 8,\ 9,\ 10,\ 12,\ 14,\\ & 16,\ 18,\ 20,\ 22,\ 24,\ 26,\ 28,\ 30,\ 32,\ 34,\ 36,\ 38,\\ & 40,\ 42.5,\ 45,\ 47.5,\ 50,\ 52.5,\ 55,\ 57.5,\ 60,\ 62.5,\\ & 65,\ 70,\ 75, 80,\ 85\}(\mathrm{cm}^2). \end{aligned}$$

3D cantilever truss problem

$$\begin{aligned} S = \{ \;& .25,\ .5,\ .75,\ 1,\ 2,\ 3,\ 4,\ 6,\ 7,\ 8,\ 9,\ 10,\ 12,\ 14,\\ & 16,\ 18,\ 20,\ 22,\ 24,\ 26,\ 28,\ 30,\ 32,\ 34,\ 36,\ 38,\\ & 40,\ 42.5,\ 45,\ 47.5,\ 50,\ 52.5,\ 55,\ 57.5,\ 60,\ 62.5,\\ & 65,\ 70,\ 75, 80,\ 85\}(\mathrm{cm}^2). \end{aligned}$$

Wing truss problem

$$\begin{aligned} S = \{ \; & .25,\ 1,\ 5,\ 10,\ 15,\ 20,\ 25,\ 30,\ 35,\ 40,\ 45,\ 50, \\ & 55,\ 60,\ 65,\ 70,\ 75,\ 80,\ 85,\ 90,\ 95,\ 100,\ 150, \\ & 200,\ 250,\ 300,\ 350,\ 400,\ 450,\ 500,\ 550,\ 600, \\ & 650,\ 700,\ 750,\ 800,\ 850,\ 900,\ 950,\ 1000,\ 1050, \\ & 1100,\ 1150,\ 1200\}(\mathrm{cm}^2). \end{aligned}$$

Appendix B

The detailed output of the ITP problem

Table B.1: Manual transportation planning results.

Week	Trips	Seats used	Buses	Vans	Inmates moved	% moved w/ hub	Utilization ratio w/o hub	Utilization ratio w/ hub
1	42	948	21	21	550	58	0.58	0.93
2	40	943	21	19	530	53	0.56	0.85
3	37	931	22	16	668	52	0.72	1.09
4	39	862	19	20	657	55	0.76	1.16
5	38	823	18	20	499	62	0.61	0.96
6	40	925	20	20	554	70	0.60	1.01
7	36	912	22	14	581	58	0.64	0.98
8	38	955	21	17	643	52	0.67	0.99

Table B.2: Week 1 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	25	444	9	16	1	549	44	1.24	1.78	13.60
0.25	23	430	9	14	4	546	40	1.27	1.78	12.10
0.50	23	430	9	14	1	549	41	1.28	1.80	9.11
0.75	22	404	8	14	14	536	39	1.33	1.85	7.49
1.00	19	265	4	15	129	421	19	1.59	1.89	3.92

Table B.3: Week 2 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	23	430	9	14	0	530	30	1.23	1.60	10.00
0.25	22	437	9	13	1	529	35	1.21	1.63	13.90
0.50	24	418	8	16	10	520	30	1.24	1.61	15.00
0.75	21	312	5	16	65	465	30	1.49	1.94	8.29
1.00	17	251	4	13	116	414	11	1.65	1.82	6.11

Table B.4: Week 3 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	28	550	12	16	0	668	34	1.12	1.63	14.10
0.25	26	517	11	15	5	663	34	1.28	1.72	12.10
0.50	27	510	11	16	19	649	29	1.27	1.65	13.80
0.75	23	437	9	14	60	608	30	1.39	1.80	9.91
1.00	16	230	4	12	240	428	8	1.86	2.01	5.21

Table B.5: Week 4 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	30	644	14	16	1	656	41	1.02	1.44	33.30
0.25	27	531	11	16	13	644	39	1.21	1.69	25.10
0.50	25	458	9	16	41	616	39	1.34	1.86	19.30
0.75	25	444	9	16	54	603	33	1.36	1.80	15.90
1.00	21	284	5	16	197	460	30	1.62	2.11	12.50

Table B.6: Week 5 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	26	484	10	16	1	498	43	1.03	1.48	24.70
0.25	22	371	7	15	24	475	41	1.28	1.81	22.30
0.50	22	338	6	16	41	458	30	1.36	1.77	18.20
0.75	21	298	5	16	56	443	30	1.49	1.93	10.10
1.00	19	284	5	14	61	438	29	1.54	2.00	6.00

Table B.7: Week 6 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	30	644	14	16	0	554	49	0.86	1.28	36.60
0.25	26	505	10	16	13	541	52	1.07	1.62	26.70
0.50	24	404	8	16	38	516	51	1.28	1.93	15.10
0.75	22	338	6	16	87	467	46	1.38	2.01	11.70
1.00	19	232	3	16	171	383	40	1.65	2.31	4.86

Table B.8: Week 7 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	27	510	11	16	2	579	39	1.14	1.58	22.60
0.25	27	510	11	16	2	579	40	1.14	1.59	20.70
0.50	25	444	9	16	27	554	45	1.25	1.82	17.50
0.75	22	371	7	15	48	533	40	1.44	2.01	6.73
1.00	19	251	4	15	159	422	21	1.68	2.03	6.22

Table B.9: Week 8 results.

α	Trips	Seats used	Buses	Vans	Inmates not moved	Inmates moved	% moved w/ hub	Seat util. ratio w/o hub	Seat util. ratio w/ hub	Opt. gap %
0.10	28	571	12	16	1	642	42	1.12	1.59	23.50
0.25	28	550	12	16	5	638	36	1.16	1.58	22.10
0.50	25	491	10	15	18	625	40	1.27	1.79	16.10
0.75	26	470	10	16	15	628	40	1.34	1.87	10.00
1.00	20	272	4	16	193	450	28	1.65	2.12	8.19

Vita

Mohammad Shahabsafa was born in Yazd, Iran in 1987 to Asghar Shahabsafa and Soraya Ghadirpanah. He received his Bachelor's and Master's degrees in Industrial Engineering from Sharif University of Technology, Tehran, Iran. He entered the Ph.D. program at Lehigh University in 2013. His research mainly revolves around discrete optimization and its applications. Mohammad was honored to receive the Informs Daniel H. Wagner Prize in 2017. He was additionally recognized on the floor of Pennsylvania House and Senate for the contributions of the inmate assignment project. Mohammad was awarded the ISE PhD student of the year in 2017. He is a co-founder of Optamo LLC and will be joining Optamo after graduation.