Draping Optimisation Software for Dry Composite Preform Manufacturing

by

Fabian Basaldua-Robledo

Thesis submitted to the University of Ottawa

in partial fulfilment of the requirements

for the Ph.D. degree in

Mechanical Engineering

Department of Mechanical Engineering

Faculty of Engineering

University of Ottawa

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. François Robitaille, for his exceptional guidance and support throughout this journey. To describe him merely as a supervisor would be an understatement; he has been a mentor, a trusted advisor, and a friend whose impact has extended far beyond the academic sphere. During the most challenging periods of this PhD, when the path forward seemed uncertain, he was always present with invaluable advice and encouragement. His dedication, high standards, and generosity have shaped not only this research but also my professional and personal development. Furthermore, I am deeply grateful for the financial support he provided, which was vital to the completion of my studies. My family and I will remain forever thankful for his unwavering trust and unparalleled mentorship.

I would also like to sincerely thank Dr. Juan F. Luna and Dr. Nasser Mohammed Noriega from the Faculty of Mechanical Engineering at FIME – Universidad Autónoma de Nuevo León, who believed in me and supported my application for the scholarship that made this possible.

I wish to express my gratitude to the institutions that made this doctoral research possible. I am grateful to the University of Ottawa, the Faculty of Engineering and the Department of Mechanical Engineering, for providing the academic environment and resources necessary for this work. To the Hutchinson team members for supporting this project and providing access to facilities and direct links to industrial practices. I also extend my deepest thanks to the Universidad Autónoma de Nuevo León (UANL) and the Facultad de Ingeniería Mecánica y Eléctrica (FIME) for their institutional support and for nominating me as a candidate for the scholarship granted by the Instituto de Innovación y Transferencia de Tecnología (I2T2) of Nuevo León and the Consejo

Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT). Their investment and trust were fundamental to the successful completion of this PhD.

To the leadership team at STELIA North America for their encouragement during the final two years of this work, and especially to Tim Dellapinna for his flexibility and support, which made it possible to pursue and complete this thesis alongside my professional responsibilities.

I am grateful to my fellow graduate students—Jacob Hoffer, Philippe Kanz, Zhengyang Zhang—for the insightful discussions, collaboration and camaraderie shared throughout this journey. On a personal level, I extend my heartfelt thanks to my friends at the University of Ottawa—Payam, Nazanin, Pouya, Gilmar, and Chuzhen—who became an extension of my family in Canada and were always there when I needed them most.

Finally, I dedicate this work to my family. To my father and mother, thank you for your unwavering belief in my path and for always cheering me on from afar. To my wife, Elizabeth, who has stood by me through all these years: this achievement belongs to you as much as it does to me. This work would not have been possible without your support, understanding, patience, and love.

To my daughters, Maria and Carlotta—this work is a legacy I leave for you. Every page carries a piece of my effort, my perseverance, and my dreams. I hope that one day, when you read this, you will see that dedication, curiosity, and resilience can take you further than you ever imagined.

Abstract

Hand lay-up draping of textile reinforcements is widely used for manufacturing polymer-matrix composite parts by liquid moulding, as it enables the production of good-quality complex parts with the limited resources typically available for small-volume projects. Implementation of hand lay-up draping presents challenges in ensuring cost-effective and consistent manufacturing. Procedures for the design of composite parts and their manufacturing processes require concurrent analysis of part geometry and manufacturing operations. The capability to define in advance the reinforcement, textile patterns, and the sequence of manual draping operations is critical to reducing costs and maximising reproducibility in production. Whilst draping is recognized as a complex sequential task, little attention has been given to identifying a superior draping route by generating and evaluating alternative draping sequences. Instead, trial-and-error remains the industrial norm.

In this work, hand lay-up draping is modelled as a sequence of decisions, where each draping step consists in placing a single Base Surface BaS and requires a specific amount of strain energy imparted to the fabric reinforcement stack. The total strain energy is defined as the sum of contributions from different deformation modes, with an additional term accounting for statistical variability in material behaviour. Based on this formulation, different approaches for optimising draping cost and quality are implemented, and the effects of sequence parameters are quantified using data extracted from pre-existing databases.

As a final product, a software tool generates draping sequences, a 3D visual guide, wrinkle warnings, and darting recommendations for the fabric pattern.

Table of Contents

List of Figures

List of Tables

List of Algorithms

List of Symbols

b_f	Width of fabric testing coupon
cp_i	Centre point for BaS #i
d_i	i-th draping decision
e_n	Edge number n
k	Adjustable weight coefficient for strain energy variability
k_q	Adjustable weight coefficient for the Quality component of $f(C, Q)$
l	Length - lab testing
l_{en}	Length of edge
m	Mass
m_c	Predominant mould curvature type, concave or convex
mep	Middle edge point
n_{BaS}	Total number of base surfaces in a mould
$\tilde{n}$	Neighbour
tds	Total number of draping steps
B	Bending stiffness
C_i	Cost of draping decisions previously taken
C_R	Connecting radius
D_i	Decision space
$\boldsymbol{E}$	Set of graph-edges representing neighbouring relationships
$\boldsymbol{F_{CH}}$	Set of 3D facets from a mould
$\boldsymbol{G}$	Undirected graph

G_{Peirce}	G – Peirce's criterion
$\boldsymbol{H^*}$	Optimal solution of function H
L	Distance between ellipses
M	Bending moment
M_m	Set of points from a mould
P_m	Set of middle edge points from a mould
P_s	Starting position
R	Fabric radius
R_1	Upper arc radius - Conical Frustum
R_2	Lower arc radius - Conical Frustum
R_f	Radius after spring back
R_m	Single curvature mould radius
S	State
$\boldsymbol{S}$	Draping sequence as a set of ordered list of nodes
S^*	Optimal solution of state S
$\hat{\boldsymbol{S}}$	Set of all feasible sequences under proposed constraints
S_{BaS}	BaS area
S_{PF}	Picture frame surface
U	Strain energy
U_B	Bending strain energy
$U_{BSB\text{-}CV}$	Bending spring back strain energy - Convex curvature
$U_{BSB\text{-}CC}$	Bending spring back strain energy - Concave curvature
U_{Di}	Energy required for a draping step

U_S	In-plane shear strain energy
U_{SSB}	In-plane shear strain energy spring back
v_{nm}	Vertex number *nm*
$\boldsymbol{V}$	Set of BaS
V_m	Set of vertices from a mould
$\boldsymbol{x}$	Decision variables
$\boldsymbol{X}$	Constraints on decision variables
α	Fabric opening angle (rad)
α_f	Final opening angle (rad)
α_m	Mould angle (rad)
β	Spring back angle (rad)
$\beta_{1,2}$	Frustum opening angles (rad)
θ_{Di}	Shear angle calculated for each draping step (rad)
Π_{Ni}	Plane defined for every BaS
$\boldsymbol{\mathcal{R}}$	Set of restrictions for HLU process model
$\boldsymbol{\mathcal{S}}$	Set of possible states
$\hat{\boldsymbol{S}}$	Set of draping sequences

Acronyms

BaS	Base Surface
BF	Brute Force
BFS	Breadth-First Search
C&Q	Cost and Quality
CAD	Computer-Aided Design
CFRP	Carbon Fibre Reinforced Polymer
CH	Convex Hull
CM	Counter-Mould
CSV	Comma-Separated Values
DFS	Depth-First Search
DDP	Differential Dynamic Programming
DC	Double curvature BaS
DP	Dynamic Programming
GPU	Graphics Processing Unit
GRST	Guided Random Search Techniques
HLU	Hand Lay-up
LCM	Liquid Composite Moulding
LRTM	Light Resin Transfer Moulding
MOO	Multi Objective Optimisation
MSVB	Microsoft Visual Basic
NB	Natural Branching

PMC	Polymer Matrix Composite
R&D	Research and Development
RAM	Random Access Memory
RFQ	Request For Quotation
RS	Reinforcement Stack
RTM	Resin Transfer Moulding
TS	Tabu Search

Chapter 1. Introduction

1.1 Background

Usage of polymer-matrix composites (PMC) in aerospace applications is continuously on the rise, driven by economics and regulations promoting reduced carbon emissions [1]. For example, carbon fibre reinforced PMC parts (CFRP) make up 53% of Airbus A350 vehicle mass [2], Figure 1.1. Approximately 85% of PMC parts in aircraft are categorized as secondary structures, with hand lay-up (HLU) of prepreg semi-products being the most widely used manufacturing technique for such components [3].

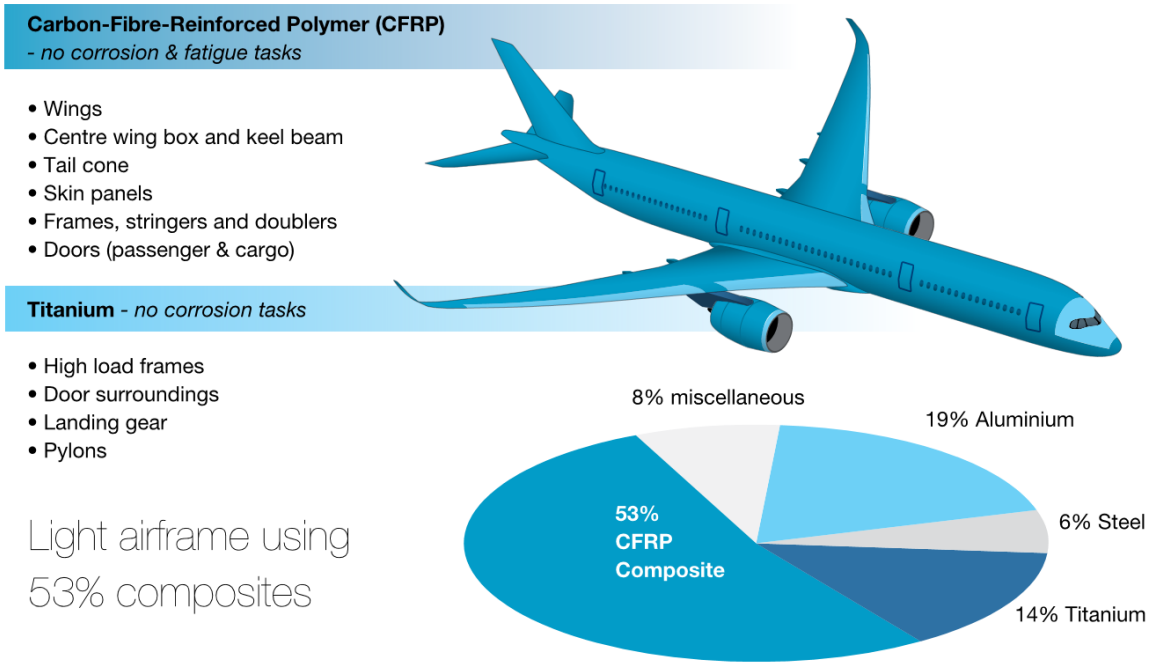


Figure 1.1 Materials usage by weight in A350 XWB aircraft [11]

Manufacturing processes for PMC parts differ fundamentally from processes used for making metal parts as PMC material and part are created simultaneously. Consequently, the design of PMC parts and manufacturing operations requires an iterative approach involving material selection, part geometry and manufacturing operations. Compromises on one aspect are often made to accommodate another aspect.

For example, changes intended to improve manufacturability may affect part quality, while improvements in quality may increase manufacturing effort and cost. Expert technicians also know, through experience, which regions of a mould should generally be draped first and which should be draped later to facilitate subsequent draping operations. However, these decisions are largely based on experience and not supported by a systematic optimisation methodology.

Manufacturing textile-reinforced PMC parts using vacuum-assisted resin transfer moulding (VARTM) presents specific challenges. Dry fibre preforms have been successfully adopted in aerospace applications, including the resin-infused composite wing of the Airbus A220 aircraft. This thesis focuses on dry fibre preform manufacturing, specifically on the sequencing of hand lay-up draping operations leading to easier manufacturing of preforms that demonstrate higher consistency. For each specific PMC part geometry, a well designed HLU strategy will make the best use of the capability of stacked reinforcement fabric layers to cover the mould whilst reducing in-plane shear in fabric layers, in the most reproducible manner. Complementary to HLU sequencing, the preforming strategy will include determining the location and extent of darts. A well designed preforming strategy will reduce the numbers of darts and possible wrinkles, whilst also reducing in-plane shear in fabric layers and variability of the preforms produced. Effective consideration of all these aspects of preforming will reduce the time required by technicians for building preforms on the shop floor, and it will lead to PMC parts of higher consistency.

Software tools that predict surface draping are available [4]–[6]. Starting from user-specified initial conditions, existing software tools can simulate draping over complete mould surfaces. However, they do not minimize shear or maximize consistency, and darts are not considered: existing software tools simply cover a mould surface from given initial conditions. Therefore, industrial simulations proceed as lengthy trial and error processes where several preforming simulations are conducted arbitrarily and a manufacturing scenario that is workable rather optimal is eventually identified.

The foundational approach to the work reported in this thesis is intrinsically different. First, the geometry of the PMC part is decomposed into multiple adjacent parametrized geometries labelled BaS [7]. Two databases are queried repeatedly towards identifying preferred draping strategies: database A features yarn orientations and in-plane shear results for a selection of parametrized BaS, and database B features extensive results of characterization tests conducted on stacked layers of industrial textile reinforcements. Databases A and B are queried by an optimisation engine that is the central focus of this thesis. This optimization engine, which features different algorithms, was developed to probe a large number of sequences along which the different BaS defining a part may be draped, leading to the identification of a preferred sequence enabling easier and more reproducible preforming. The optimisation engine constitutes the main contribution of this thesis. The work reported here is the centerpiece of a wider project that involved a team of researchers over several years.

Individual BaS are classified as flat, single curvature or double curvature surfaces. They are defined using the conical Frustum, a geometric modeling tool developed by Hoffer [8] as part of the project. The Conical Frustum uses 12 parameters in a more complex version or 5 parameters in a simplified version that can model most local mould features faithfully. Database A lists warp

and weft yarn orientations and in-plane shear angles resulting from BaS draping. It was built using Drape-it™ and ANSYS™ software packages, resulting in a set of linear equations that describe yarn orientations as a function of Frustum parameters. Database A was populated and structured using Taguchi experimental plans [9]. The BaS Frustum and Database A, both developed by Hoffer [8], are discussed in Sections 1.2.1 and 1.2.2 of this thesis.

Experimental characterization of the mechanical behaviour of single layer and multiple layer stacks of dry textile reinforcements used for manufacturing industrial parts was conducted by Kanz [9]. Some properties were reported for other reinforcement fabrics in the literature [10]–[12] and others properties are novel to this project. Database B lists properties measured on single and multiple layer stacks of industrial reinforcement fabrics for bending stiffness, bending spring back, in-plane shear, in-plane shear spring back and friction between plies. It was also populated and structured using similar Taguchi experimental plans [9], resulting in similar linear equations. Property data and Database B are revisited in Section 1.2.3 and Chapter 4 of this thesis.

1.2 Previous Project Work

1.2.1 Industrial Geometry File Formats and Base Surfaces

Industrial partner Hutchinson Aéronautique et Industrie (Montréal) makes PMC parts for major aircraft manufacturers. A contract starts with the reception of a CAD file for a PMC part. Clients use different software and formats; SolidWorks™, Inventor™, AutoCAD™, CATIA™ and Pro/ENGINEER™ are the most common [13]. In this project, files were transferred under

standardized STEP format based on ISO standard 10303-21:2016 and managed in SolidWorks™ [14].

The algorithm developed in this project starts by extracting geometric information from a STEP file through MS Visual Basic (MSVB) macro points_creation_and_export.swp programmed by Salles, a process engineer employed by Hutchinson for the project. The MSVB macro interacts with SolidWorks™, gathering relevant geometric data in a coherent and structured format, Table 1.1. Headers from each column indicate the type of information extracted from the CAD file. Finally, extracted data are saved as a Excel™ file.

Table 1.1 Structure of geometric data extracted by macro points_creation_and_export.swp

Base surface #	Edge #	Edge length	Point #	x	y	z
BaS#i	e_1	l_{e1}	v_{1a}	x_{1a}	y_{1a}	z_{1a}
			mep_{1b}	x_{1b}	y_{1b}	z_{1b}
			v_{1c}	x_{1c}	y_{1c}	z_{1c}
	e_2	l_{e2}	...	...	...	...
			...	...	...	...
	e_3	l_{e3}	...	...	...	...
			...	...	...	...
	e_4	l_{e4}	...	...	...	...

Industrial geometries are processed as provided by the CAD model, with the requirement that every sub-geometry be delimited by either 3 or 4 edges. When a sub-geometry exceeds this limit, it must be subdivided into multiple sub-geometries until compliance is achieved. Figure 1.2 shows the case of demonstrator 01_Francois, where the original top BaS #3 delimited by 7 edges was manually subdivided into three BaS: BaS #3, BaS #4 and BaS #5. Each edge is subsequently approximated as a constant-radius curve connecting two vertices through a middle edge point.

Macro points_creation_and_export.swp determines the length of every edge and the coordinates of its middle edge point.

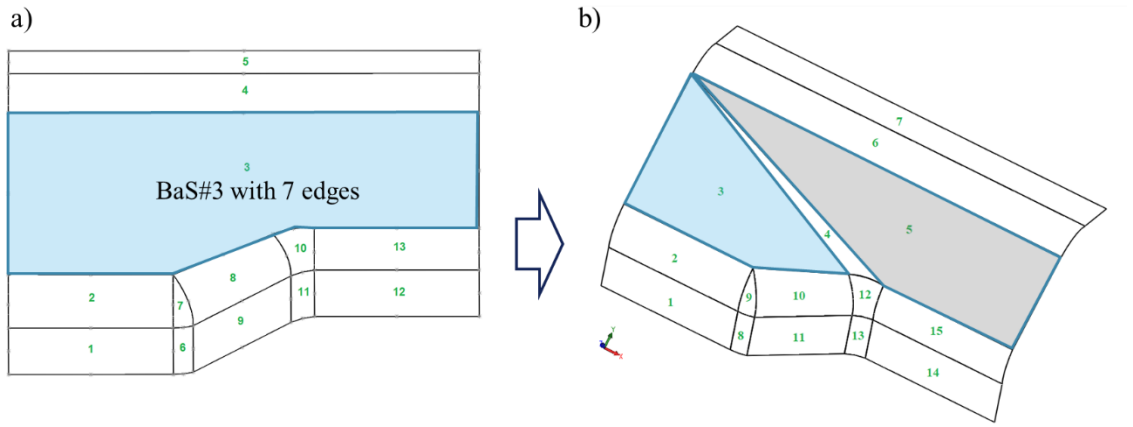


Figure 1.2 Demonstrator 01_Francois: a) Original configuration; b) Only 3 or 4 edges BaS

Figure 1.3 shows the decomposition of a mould surface into a number of flat, single curvature and double curvature BaS delimited by 3 or 4 edges, with their vertices and middle edge points.

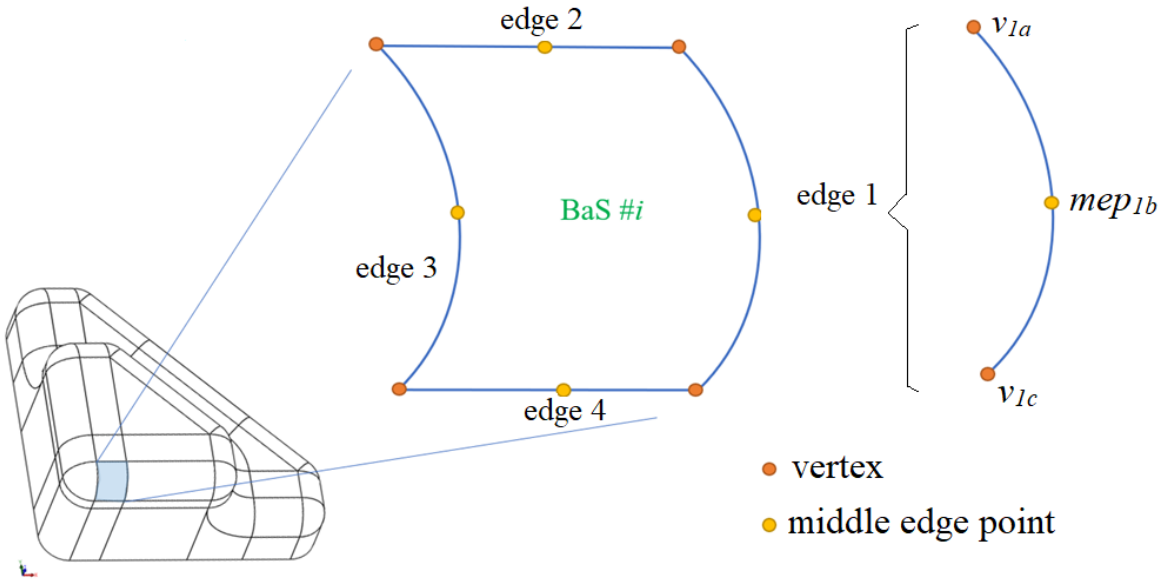


Figure 1.3 Demonstrator 02_Jacob_solid divided into its geometric elements

Extraction of geometric data is demonstrated in Appendix A for demonstrator 01_Francois. The need for all BaS to have a maximum of 4 edges stems from the nature of the conical Frustum used for parametrizing them, as discussed in the following section.

1.2.2 Conical Frustum and Database A

Figure 1.4 shows the conical Frustum used as a multi-parameter base surface modelling and parametrization tool. Hoffer [8] initially developed a Frustum featuring 12 parameters, Figure 1.4 a). Figure 1.4 b) shows a simplified version of the Frustum featuring 5 parameters, developed based on a hierarchy of original Frustum parameters most significant in modelling BaS for industrial geometries.

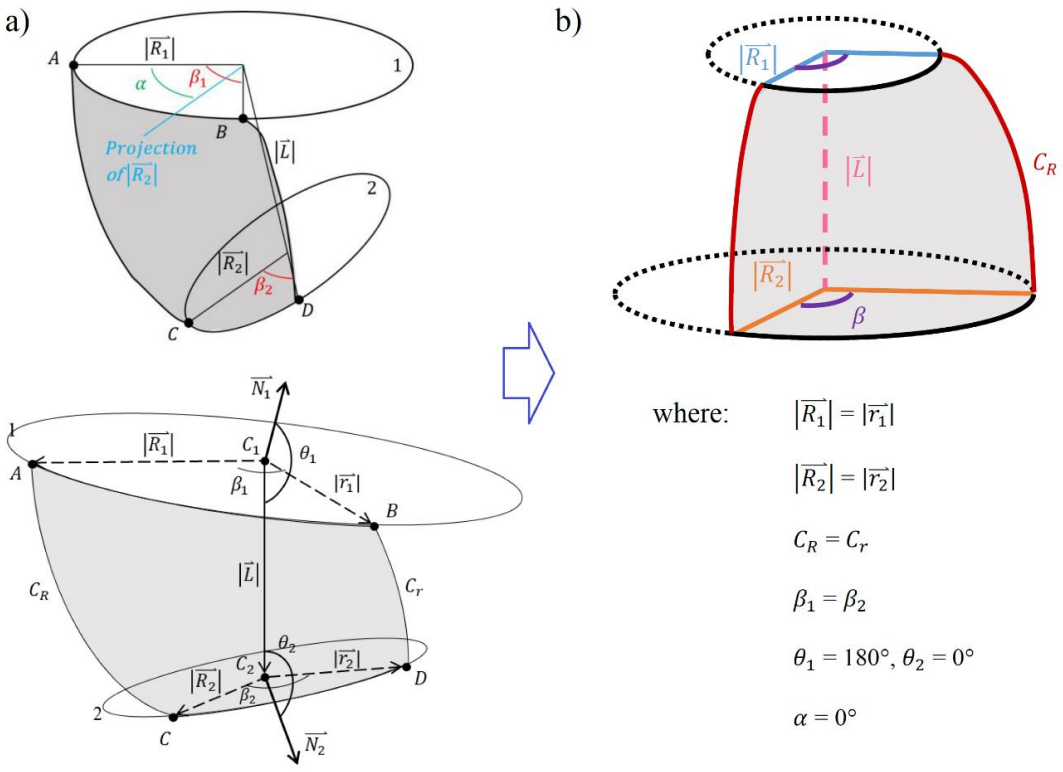


Figure 1.4 Conical Frustum: a) 12 parameters; b) 5 parameters

Using 12 parameters would have required that Hoffer performed a very large number of BaS models and simulations. Hoffer [8] determined the Frustum parameter hierarchy and populated yarn orientations and in-plane shear angles in Database A using 16 BaS models of 3- and 4-edges. ANSYS™ was used for BaS geometry creation and meshing, while subsequent draping simulations were performed using DrapeIt™ kinematic draping software [8] .

Frustum parameters were used as input variables to a Taguchi plan used for structuring and populating Database A with yarn orientation and in-plane shear angle data obtained from simulations conducted on specific BaS. The Taguchi plan and database structure enable interpolation for any BaS identified on a given industrial geometry that does not exist as such in the database. Database A input features two groups of parameters: the 5 conical Frustum geometric parameters and 3 draping parameters. The first group features R_1, R_2, L, C_R, and β_i, shown in Figure 1.4 b). The second group consists of draping starting position Ps in the BaS, initial shear angle, and initial yarn orientations. Database A outputs yarn orientations and in-plane shear angles calculated at both vertices and middle edge point of every BaS exit edge. Results were used for creating linear equations interpolating any BaS and draping parameters from results available in Database A.

1.2.3 Draping Strain Energy, Variability and Database B

Kanz [9] evaluated the draping performance of reinforcement fabrics through comprehensive testing, quantifying bending stiffness, bending spring back, in-plane shear, in-plane shear spring back, and friction. Linear functions were created, interpolating the strain energy required for draping as a function of the type of fabric, yarn orientations and number of layers in the fabric for

each of the above deformation modes. Results were included in Database B, along with their associated variability.

Individual terms of strain energy associated with draping each parametrized BaS representing the mould surface are calculated by the software tool developed in this thesis, using mechanical behaviour data from Database B, based on yarn orientations and in-plane shear angles interpolated from Database A. Variability is similarly calculated by the software tool developed in this thesis for the same strain energy terms, using dispersion results also available in Database B and yarn orientation data interpolated from Database A.

Two foundational premises of the work are that the ease or difficulty of draping a textile preform is commensurate with the sum of strain energy terms imparted to the textile during draping, and that the variability of the result is commensurate with the combined variability of the same energy terms. These premises are discussed at length in Chapter 4, where the optimisation Objective Function is introduced. In this work, cost (C) represents the cumulative strain energy associated with a draping sequence, while quality (Q) represents the variability associated with that strain energy. The formulation of both terms and their experimental basis are detailed in Chapter 4.

1.3 Objectives

The main objective of this thesis is to develop a software tool capable of optimising the cost (C) and quality (Q), collectively referred to as C&Q, for dry PMC preform manufacturing. C&Q are defined in this thesis according to the requirements expressed by the industrial partner.

To achieve this main objective, the following specific objectives are pursued:

- Develop a modelling framework for HLU draping. This objective consists of geometric parameterization and the development of a HLU progression model. The HLU progression model includes defining initial conditions and a set of logic decisions summarizing the most common draping progression actions made by a human operator.
- Develop a global optimisation approach for HLU cost and quality. This objective consists in developing a combinatorial algorithm capable of generating a wide range of possible draping sequences through topological connections between BaS. The algorithm shall be adaptable to different sequencing conditions. After generating the complete space of HLU sequences, C&Q are calculated for every sequence. The software tool then identifies the sequence(s) offering the best balance between cost and quality, as defined by the Objective Function $f(C,Q)$.
- Develop an optimisation approach based on solving multiple local optima. This objective consists in developing a software tool capable of creating partial sequences or sub-spaces simulating draping progression in HLU. For each sub-space, the algorithm evaluates every draping step and solves the local optimum. Memory from previously evaluated draping steps is consulted to allow draping along multiple branches. A near-optimal solution is obtained through the conjunction of all solved local optima.

1.4 Scope and Limitations

The present work focuses on the modelling and optimisation of the HLU step of the moulding stage, which occurs prior to resin infusion in VARTM. The industrial context considered is HLU of stacks of dry textile reinforcements for non-structural aerospace PMC parts, particularly aircraft interior panels used in civilian aircraft.

Patterns considered in this work feature 1 to 4 layers of stacked textile reinforcements oriented in the same direction within each pattern, as dictated by the industrial context. Consequently, maintaining prescribed fibre orientations throughout the complete mould surface is not treated as a requirement, in contrast with structural composite applications where fibre orientation directly affects structural performance.

The optimisation framework relies on geometric parameterization, discretization, interpolation, extrapolation, and experimentally derived databases to estimate draping behaviour. While these approximations are intended to provide meaningful agreement with actual shop-floor draping operations, the objective is not to predict precise mechanical quantities or part performance. Rather, the software tool is intended to support manufacturing decisions through structured and consistent comparisons between alternative draping strategies. This approach enables the rapid evaluation of very large design spaces, allowing up to millions of possible draping sequences to be assessed using practical computational resources.

The software tool is intended to support quotation and manufacturing planning activities, where response time is critical for preparing customer bids and responding to requests for quotation (RFQ). Consequently, the work does not aim to perform high-fidelity finite element

simulations or formal statistical analyses. Instead, computationally efficient approximations are adopted to enable the evaluation of a large number of possible draping sequences within practical timeframes.

The software tool developed in this thesis is intended to operate on computational resources typically available to manufacturing and engineering personnel. Software development and testing were performed using an Intel Core i9-13950HX processor featuring 8 performance cores and 16 efficient cores with a maximum frequency of 5.5 GHz.

1.5 Contributions

The approach to HLU manufacturing optimisation presented in this work is unique in the way that it addresses the process. The proposed software tool uses two databases to determine yarn orientations, material behaviour, and the cost and quality associated with draping each BaS along a given preform manufacturing sequence. This enables automated optimisation of draping sequences without additional user interaction during software execution. The work presented in this thesis led to the following original contributions to the field of preform engineering:

- This work is the first to generate a very large number of possible draping sequences, and to evaluate and compare them systematically and automatically based on engineering parameters previously characterized and instantly available through database querying.
- This work is the first to simulate HLU preforming as a sequence of discrete steps, to evaluate the physical properties of the fabric and the energy needed to execute the

operation, to identify possible surfaces to drape next, to estimate C&Q for all BaS, and to identify the best draping strategy for a given part and mould.

- This works presents a novel and unique modelling approach for fast analysis of HLU draping process that can be replicated and used for further evaluation of optimisation algorithms. Additionally, the algorithms developed can be adapted for different applications where complex trajectory optimisation is required.
- This work is the first to propose the optimisation of draping through the rapid evaluation of multiple draping scenarios and sequences, using methods that are sufficiently fast in an industrial context where rapid identification of a good manufacturing process design and draping strategy is required for industrial costumers.

1.6 Thesis Outline

This thesis is organized into seven chapters, structured as follows.

Chapter 1 introduces the background and motivation for the work, presents relevant previous project developments, and defines the objectives and main contributions of the thesis. It also outlines the overall structure of the document.

Chapter 2 reviews the state of the art related to composite manufacturing processes, with a focus on VARTM and HLU. It further examines existing approaches for HLU process optimisation, including progression algorithms and optimisation strategies such as brute force (BF) and dynamic programming (DP).

Chapter 3 presents the modelling framework for the HLU process. It describes the definition of the initial state, the evolution of yarn orientations across BaS, and the generation of draping sequences using search algorithms. Additional functionalities, including wrinkle detection and edge darting recommendations, are also introduced.

Chapter 4 defines the C&Q Objective Function used for evaluating draping sequences. It details the material characterisation methods, strain energy components, their variability, and their integration into the optimisation framework through Database B.

Chapter 5 presents the BF optimisation approach applied to HLU draping. Demonstrators with up to 10 base surfaces are analysed, enabling exhaustive generation and evaluation of all feasible draping sequences. The results are used for validating the modelling framework and the Objective Function.

Chapter 6 introduces the DP optimisation framework for HLU draping. Two variants are developed and evaluated: a 1-next state approach and a 2-next states approach. Their performance, computational efficiency, and suitability for complex geometries with large numbers of BaS are analysed.

Chapter 7 discusses the results obtained from both optimisation approaches. It evaluates the modelling assumptions and algorithm performance in terms of computational cost and solution quality. The Chapter concludes with a summary of the main contributions, limitations of the work, and recommendations for future research.

Chapter 2. Literature Review

2.1 Vacuum-Assisted Resin Transfer Moulding

Composite materials combine two or more constituent materials that remain as distinct phases [15]–[19], often leading to properties superior to those of these individual constituent materials [20],[21]. Several PMC manufacturing processes were developed including liquid composite moulding (LCM) processes such as resin transfer moulding (RTM), VARTM and others, which are commonly used for manufacturing industrial and aerospace PMC parts [22]–[24].

VARTM is widely used for making PMC parts. As shown in Figure 2.1, VARTM proceeds under atmospheric pressure at room temperature, emits limited amounts of volatile compounds, processes multiple reinforcement layers simultaneously, enables uniform resin distribution with relatively high fibre volume fraction and shows potential for reproducibility [25],[26]. Crucially, short processing times make VARTM cost-effective [27].

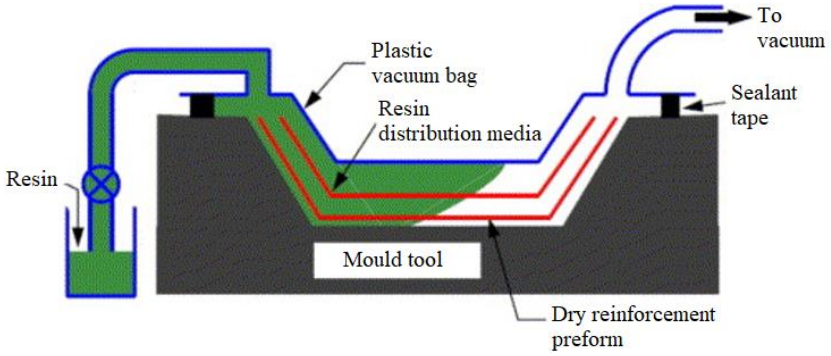


Figure 2.1 Diagram of the VARTM process [28]

Advantages of VARTM include:

- Flexibility in mould material selection and tooling design [29];
- Capability to manufacture large parts offering reasonable dimensional accuracy with tensile and compression strengths [30] comparable to those of prepreg composites [31];
- Ease of implementation with cure times shorter than resin film infusion [32];
- Possibility to store resin and hardener separately and to mix them immediately prior to infusion;
- Capability to identify dry spots through translucent vacuum bag and to remediate during infusion [33].

Conversely, VARTM is marred by some limitations [34]:

- Requires skilled workers and a clear operations plan prepared prior to execution;
- Potential for vacuum leaks requires continuous monitoring during infusion;
- Results are dependent on worker experience and skills;
- Surface finish on bagging side is poor compared to mould side;
- Ancillary materials such as peel ply, sealant tape and resin tubes must be prepared each time and are not reusable.

The VARTM process features three main stages:

- Moulding, where the composite material is made from separate constituent materials. The moulding stage can be divided into three distinct steps:

    - Hand lay-up (HLU), where dry reinforcements are tailored, draped and positioned onto the mould;
    - Resin infusion, where resin flow ancillary disposable parts are positioned and a liquid resin is infused in the dry reinforcements under vacuum;
    - Resin cure, where the liquid resin undergoes a chemical reaction that turns it into a solid.
- De-moulding, where the cured PMC part is removed from the mould;
- Trimming, where the manufactured PMC part is deburred to its final shape.

2.2 Hand Lay-up

HLU consists in manually laying and draping stacks of reinforcement fabric layers with a determined outline shape known as patterns over the surface of a mould, following a specific sequence of manual operations [35]. After HLU, draped patterns are sealed under a flexible membrane along with resin flow ancillaries. An uncured thermosetting polymer, colloquially known as the resin, is then infused by a pressure difference resulting from vacuum applied at an outlet. Resin cure follows [36]. Thickness of the resulting PMC part is largely dependent on the number of reinforcement layers present in the patterns.

HLU is the oldest preforming technique for textile-reinforced PMC [37] and it remains widely used. Prominent studies [35],[38]–[41] reveal a scarcity of information regarding manual operations in HLU, summarizing the lay-up procedure in a single sentence. This lack of scientific, structured analysis of HLU despite widespread industrial use justifies the present work.

HLU consists in covering the surface of a mould with one or more layers of fabric reinforcement, ideally as one single, tailored, continuous fabric layer or stack of similar layers labelled pattern in either case, covering the entire surface of the mould. More realistically, a mould will typically be covered using a minimum number of patterns that is larger than one, hence a typical preform will be made from a number of adjacent patterns. In either case, each multilayer pattern may be draped layer after layer or all layers at once; the latter is more productive and therefore common practice in industry.

Figure 2.2 shows the relationship between in-plane shear and wrinkling in textile reinforcements. In-plane shear is the main deformation mode occurring during draping. The need for multiple patterns results from physical limits to the amount of in-plane shear deformation that can be induced without wrinkling. Once a reinforcement fabric reaches its locking angle, further in-plane shear ultimately leads to wrinkling [42].

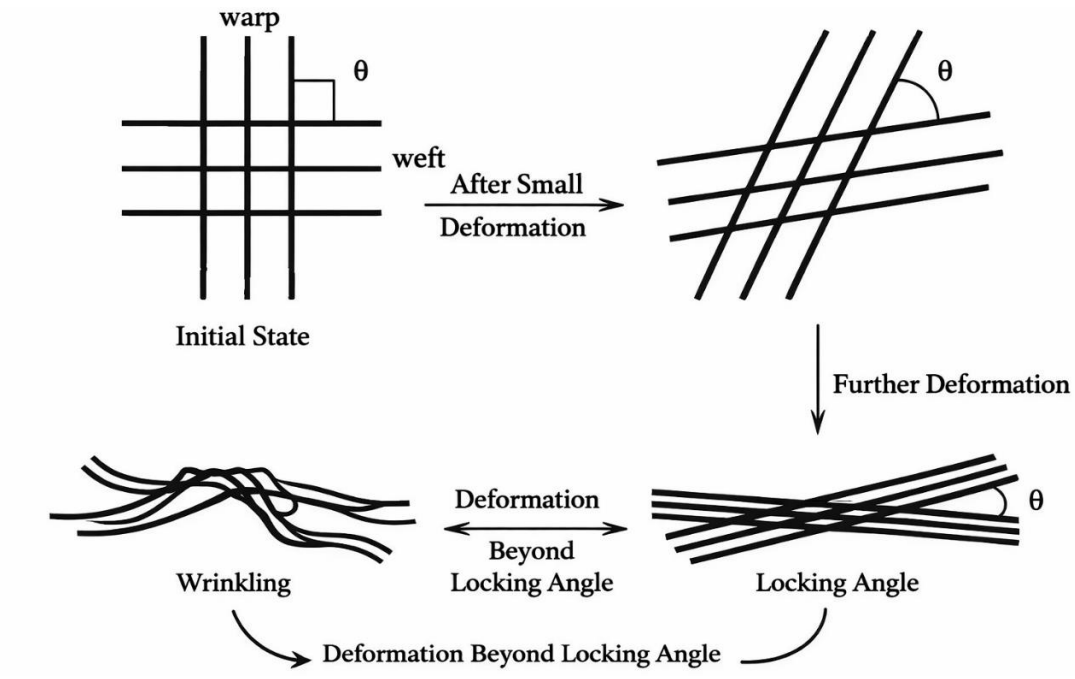


Figure 2.2 Wrinkling due to in-plane shearing beyond locking angle [42]

The locking angle defines the critical shear limit for woven reinforcement fabrics, occurring when warp and weft yarns come into lateral contact and can no longer rotate freely. Before this threshold, shear is accommodated by yarn rotation. Beyond it, shear stiffness rises sharply leading to wrinkling that compromises laminate quality. The locking angle depends on fabric architecture, tow size, compaction and inter-yarn friction [43]. Its determination, often via picture frame tests, is essential for accurate draping simulations and defect prediction in textile reinforcement draping.

Figure 2.3 shows the draping of a tetrahedral tool, where high in-plane shear develops in the corner region of the reinforcement [60]. This example illustrates how complex geometries can generate localized shear deformations that may approach the locking angle and increase the likelihood of wrinkling.

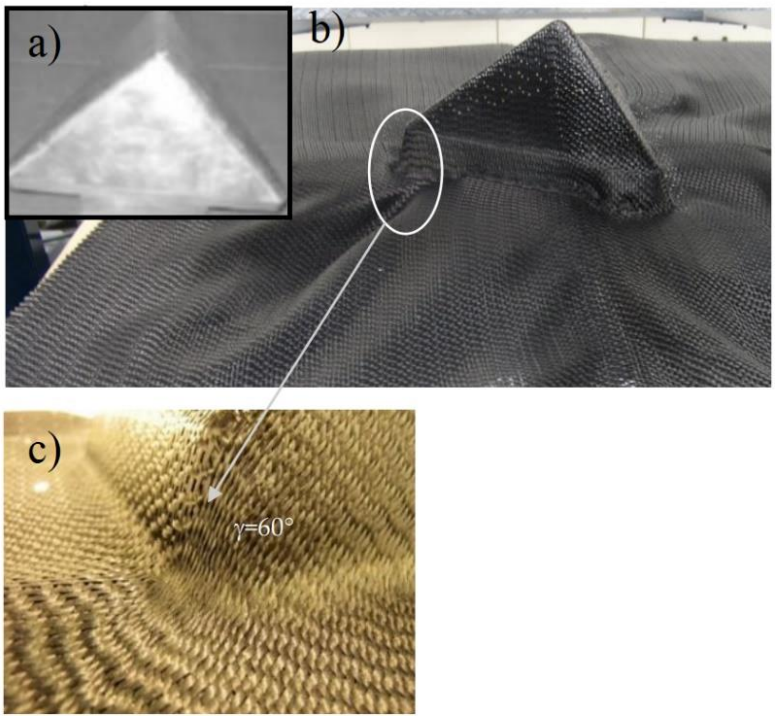


Figure 2.3 Draping of a tetrahedron: a) Tetrahedral tool; b) Draping experiment; c) High in-plane shear in corner [60]

In-plane shear is the most studied deformation mode for textile reinforcements, as reviewed by Boisse [44]. Experimental characterization began in the 1960s with the work of Lindberg [45], Grosberg [46], [47] and Spivak [48]. Experimental data constitutes important input for draping simulations [49]–[53] and predictive models used for estimating the onset of locking and wrinkling [54]–[56]. Different sequences of manual operations used when draping patterns on a mould can increase or reduce in-plane shear deformation and its variability [57], [58]. Consequently, some draping sequences may bring the reinforcement fabric closer to its locking angle than others, increasing the likelihood of wrinkling and leading to different levels of success in draping moulds, as reported by Eitzinger [59].

Figure 2.4 shows different in-plane shear results from different initial pattern orientations, obtained from simulations conducted using software Nottingham DrapeIt™.

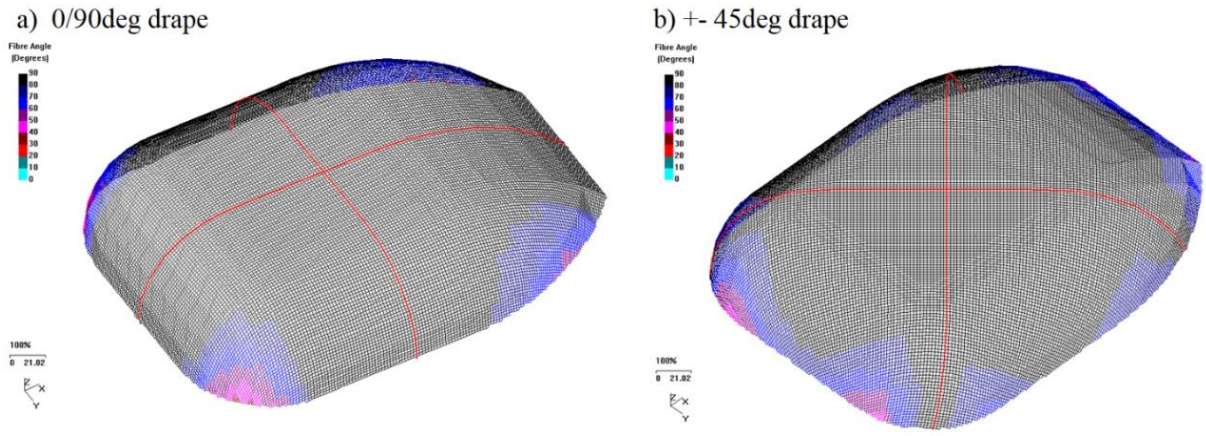


Figure 2.4 Simulation results for different initial orientations: a) 0°/90°; b) +/-45° [66]

Figure 2.5 shows an example of a plybook used during hand lay-up draping. These documents provide instructions for positioning separate patterns and progressively covering the surface of a mould according to a prescribed draping sequence. Plybooks are typically created by highly experienced R&D engineers whose specific skills are sparsely available.

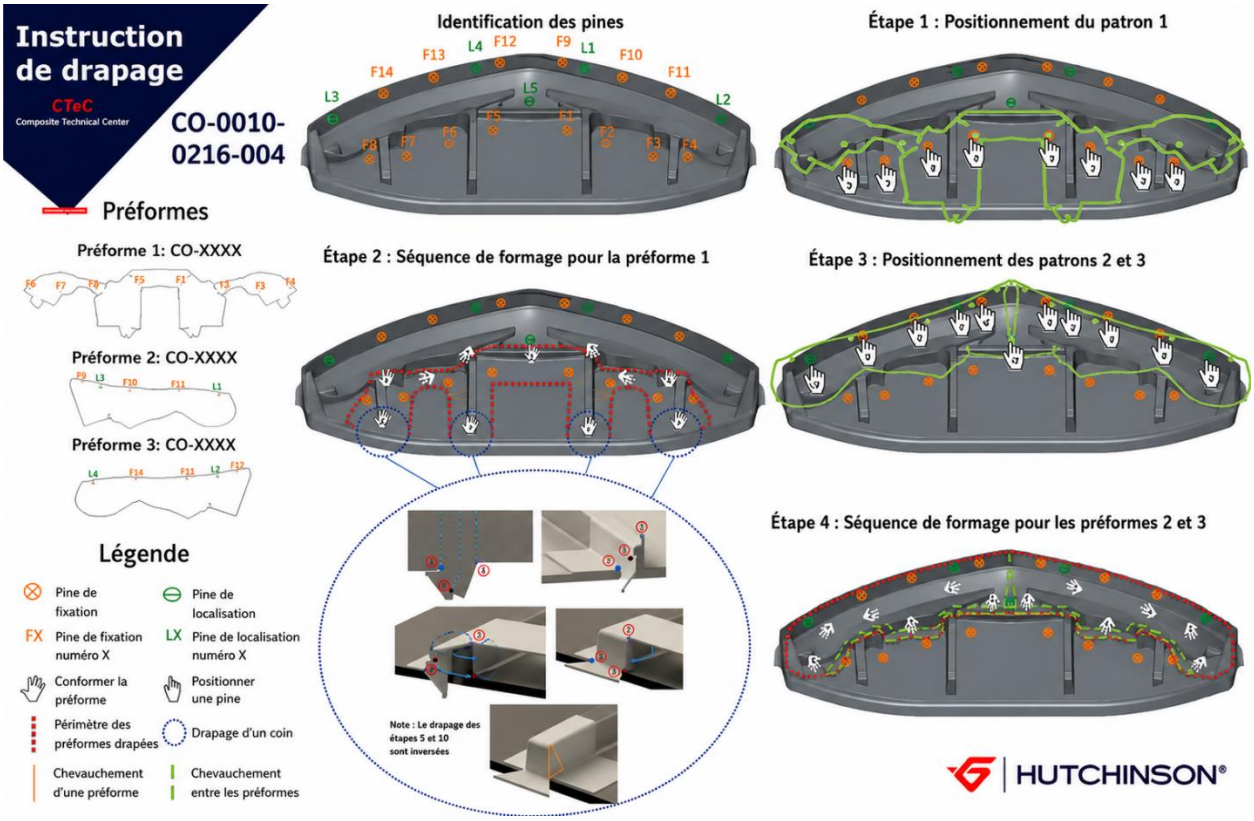


Figure 2.5 Extract from draping plybook from industrial partner Hutchinson Aéronautique [7]

HLU is labour-intensive and manual operations can result in inconsistent outcomes [64]. Industrial competitiveness dictates that the sequence of preforming operations in HLU be optimised towards minimizing manufacturing time and cost, and towards minimizing variability and maximizing preform quality. Labour costs associated with preforming represent around 60%

of PMC manufacturing costs for non-structural PMC parts as opposed to 30% to 40% for more automated methods [65].

HLU analysis focuses on the sequential operations involved in the manual positioning of individual patterns towards building up the reinforcement in the shape of the final part. Draping analysis focuses more specifically on resulting yarn orientations and in-plane shear distribution in the draped reinforcement. Ultimately, the two methods complement one another: draping modelling generates the essential yarn orientation and shear data that inputs into the HLU process simulation. [67].

Draping can be simulated using two main approaches: purely geometric methods and methods based on first-principles solid mechanics. Geometric methods use geometric information only, primarily the part geometry, using variations of the fisherman's net algorithm [68]. Figure 2.6 shows the generation of crossover points during draping simulation. Two geodesics are generated from a selected starting position and yarn orientations, then divided into regular intervals *d*. Further yarn crossover points are located by intersecting two spheres of radius d and the surface to be draped until the surface is covered. This fast method generates results in the form of yarn orientations, in-plane shear angles and flat patterns. However, reinforcement behaviour and any sequence for draping operations are not considered [69]. Geometric methods are used mainly for identifying areas of large in-plane shear and for determining whether a reinforcement may be draped over a given geometry knowing its locking angle [70].

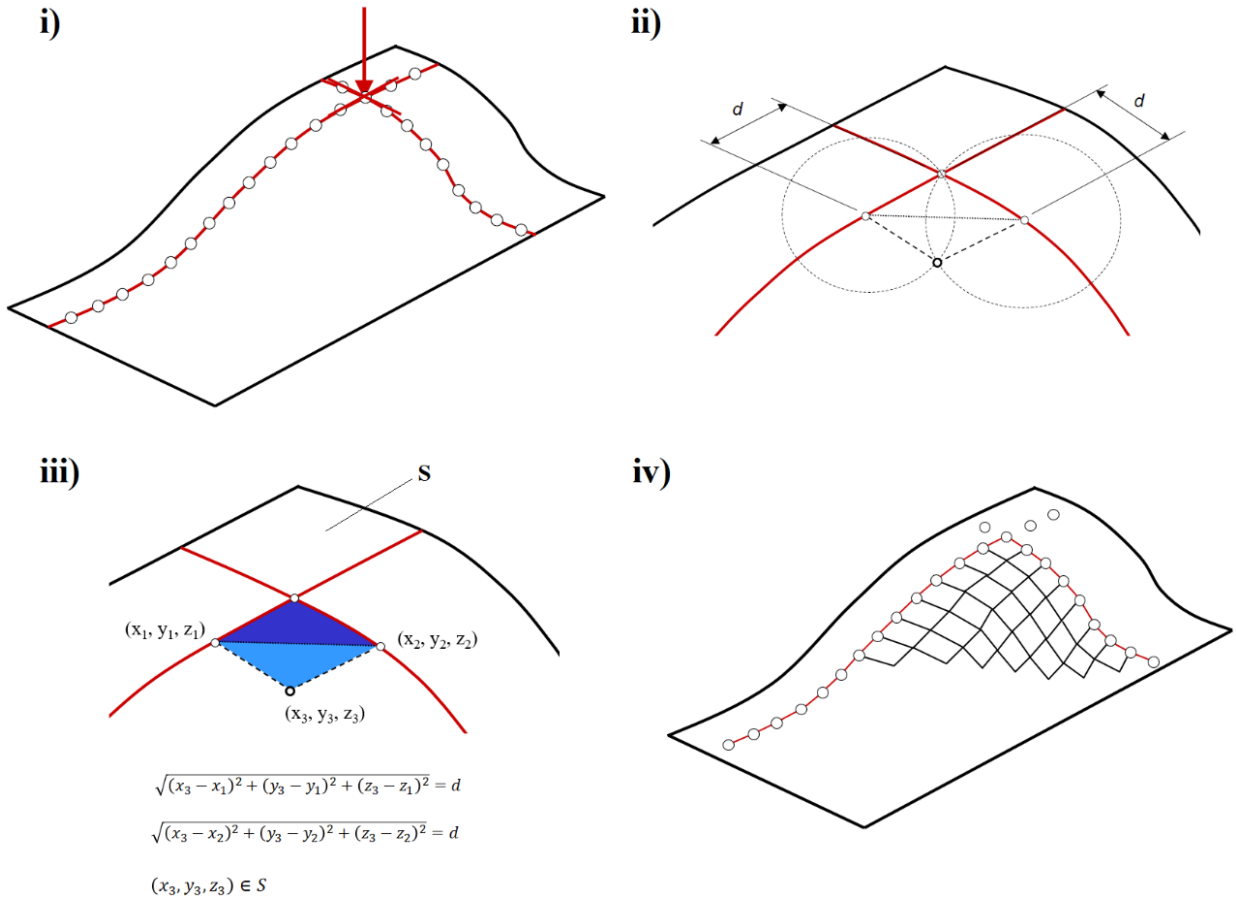


Figure 2.6 Fisherman's net algorithm [66].

Conversely, the solid mechanics approach implements the constitutive behaviour of the reinforcement through numerical methods [71]. The approach must handle non-linearities encountered in draping including large displacements, rotations and strains as well as non-linear and non-conservative constitutive behaviour of reinforcement fabrics and extensive contact non-linearities [72]. Over the last decade, research on draping has focused primarily on solid mechanics rather than geometric approaches with an emphasis on reducing processing times rather than increasing accuracy [73]–[75]. The ability of solid mechanics approaches to predict the formation and spread of defects with greater precision makes them valuable for examining the impact of

design decisions on semi-automated processes [76]. However, despite significant advances, solid mechanics approaches will never be as fast as geometric approaches.

The reviewed literature provides the geometric modelling methods, textile characterization approaches, and optimisation concepts that form the basis of the methodology presented in the following chapters.

2.3 Hand Lay-up Process Optimisation

The analysis, modelling and optimisation of HLU manual operations has received limited attention. Potter [35] studied the approach and methods used by technical staff in performing HLU. In the first study, four technical staff performed HLU on 15 different moulds. Their actions were categorized into eight distinct techniques through video analysis, showing clear correlations between mould features and techniques used, hence indicating a systematic approach to HLU. This finding lead to a manufacturing knowledge base for HLU that can be used for enhancing the process, train technical staff and support the design of future automated solutions.

Potter and Hancock [67] combined the above findings with geometric drape modelling towards generating detailed, unambiguous HLU instructions. Figures 2.7 and 2.8 show the proposed methodology and resulting manufacturing instructions. The aim was to enhance the information provided by kinematic simulations and produce clear and effective instructions for technical staff. The strategy combined experimentally established HLU procedures with yarn alignment patterns. Preforms were divided into regions of similar in-plane curvature, and specific draping operation sequences were determined for these regions. Out-of-plane yarn curvature was

analyzed to pinpoint areas that may induce bridging issues, which would necessitate local or global fabric manipulation.

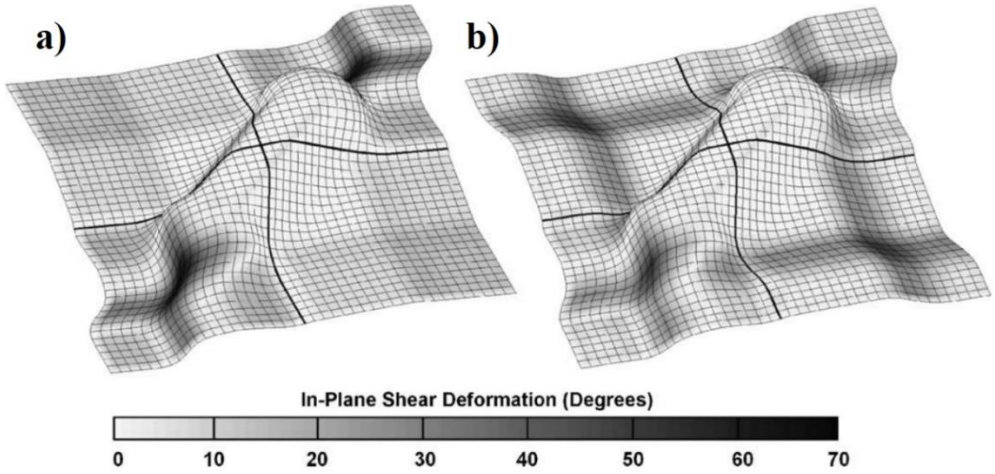


Figure 2.7 a) Straight line drape; b) Optimised drape [67]

The work showed that for complex geometries, use of simulation-based draping can significantly improve the fit of reinforcement fabrics to mould surfaces. Even then, HLU strategies were still determined by humans; no optimisation algorithms were implemented, only single patterns were considered, and no darting assessment was included in the authors' work.

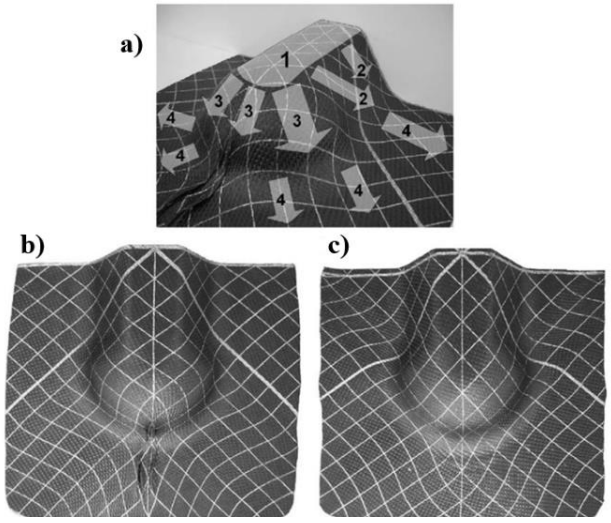


Figure 2.8 Kinematic draping simulation: a) Draping sequence; b) Draping enforcing warp orientations from Figure 2.7-a; c) Draping enforcing warp orientations from Figure 2.7-b [67]

Subsequent developments to modelling were proposed for robotized draping. Malhan [64] combined kinematic drape modelling, Potter's techniques [69], usage of end-effectors for handling plies [77] and studies on automated handling for pick and place operations [78]. Figure 2.9 shows the proposed methodology. Mould surfaces were divided based on inputs from skilled technical staff, after which HLU sequences were determined from kinematic draping simulations. Parts were then inspected to assess lay-up quality through conformity, fibre alignment, resin distribution, and defects including foreign object deposition, air pockets, wrinkles, and fibre damage. While previous robotic motions were optimised primarily to reduce robotic arm displacement, the literature reveals opportunities for optimisation approaches that directly consider manufacturing cost and quality.

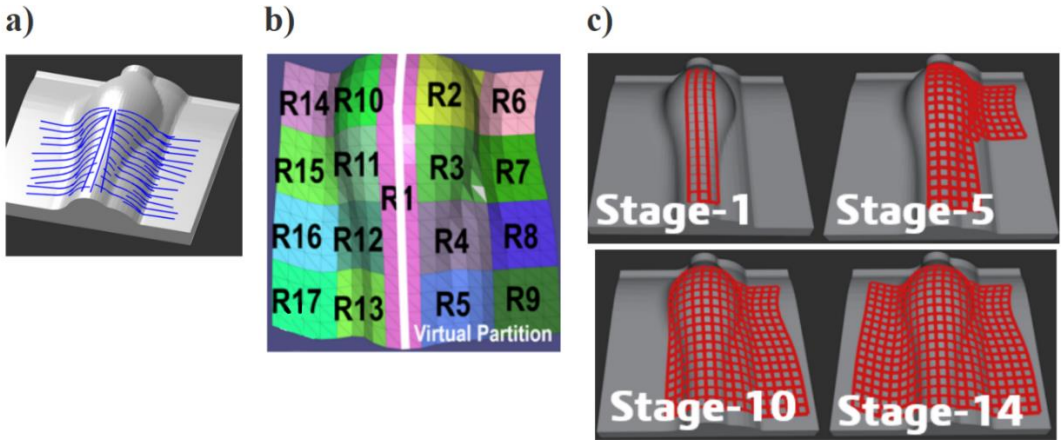


Figure 2.9 a) Mould surface; b) Partition by regions; c) Draping stages [64]

2.3.1 Progression Algorithms

Draping of a mould composed of multiple BaS can be modeled as a progression from one BaS to an adjacent BaS through shared edges. A draping sequence is an ordered list of such steps,

executed chronologically. For example, draping sequence [1–2, 2–4, 2–3] starts by draping BaS #1, then moves into BaS #2 across their common edge, followed by draping BaS #4 from BaS #2. The final step drapes BaS #3 from BaS #2; note that draping of BaS #2 itself is not repeated.

Such discrete progression compares with the traveling salesman problem [79] represented by graphs featuring nodes and edges. The objective of the travelling salesman problem is to find a route, known as a Hamiltonian circuit, that visits every node in the graph exactly once without repetition and minimizes an objective value such as the total distance, risk, or time [80]. The travelling salesman problem is solvable using search algorithms [81], including direct search algorithms [82] such as depth-first search (DFS) and breadth-first search (BFS) [83], extracting information from data structures or computing within a problem's domain search space [84].

Numerous problems can be addressed by graph algorithms, utilising appropriate search algorithms [85]. Graph traversal algorithms [86] including DFS and BFS form the most basic approach: DFS visits all reachable nodes from a starting point while BFS examines nodes by ascending distance [87]. More complex searches involve finding the shortest paths in weighted graphs [88], with Dijkstra's algorithm [89] suitable for this thesis' goals given its aim of visiting all nodes in the graph in the most cost-effective manner without cycles.

DFS, also known as backtracking and originating from Trémaux's work [90], is widely used in combinatorial theory and artificial intelligence. Applications include VARTM resin impregnation optimisation [91], materials optimisation, water management, health sciences, control theory, and others [92]–[97]. DFS starts from a node and repeatedly follows an unexplored edge until no further progression is possible, at which point the search backtracks to the most recently visited node with unexplored edges [98]. In the HLU model considered, a BaS is

represented by a node, while an edge represents a draping step from one BaS to a neighbouring BaS.

Graph theory states that graph G is a pair $G = (V, E)$ of sets such that $E \subseteq [V]^2$; thus, elements of *E,* the edges, are 2-element subsets of *V*, vertices or nodes [99]. Therefore, path $p: v \overset{*}{\Rightarrow} w$ in G is a sequence of nodes and edges leading from node *v* to node *w.* A path is simple if all its nodes are distinct. A directed rooted tree *T* has a starting node called the root, and every node receives only one incoming edge. A directed graph *G* has a spanning tree *T* if *T* is a subgraph of *G* that encompasses all nodes of *G* [100]. Figure 2.10 depicts an in-progress tree-like graph with the node at the top being the root and representing the starting point, hence identified by number 1. Node numbers indicate DFS algorithm visitation order. Gray shaded nodes are the next possible nodes to be visited. The DFS process ends when all nodes have been visited.

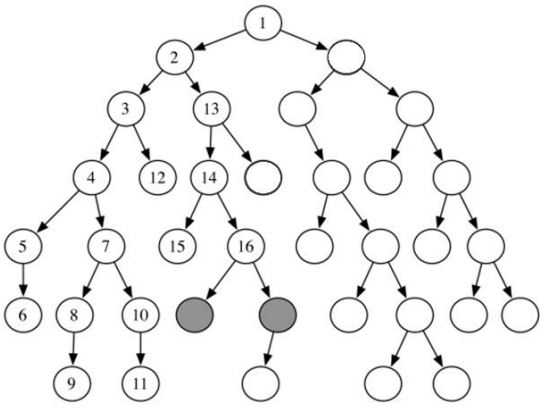


Figure 2.10 Depth-first search (DFS) path [101]

In applying DFS to HLU of a mould, several similar tree-like graphs can be elaborated with each graph representing a different draping sequence. In a HLU DFS graph representation, each node represents a BaS. Arrows in the tree diagram, known as edges in DFS terminology but

distinct from BaS edges in geometric modelling, represent draping steps where directionality is important; hence the graphs are known as directed graphs. Care must be taken to distinguish DFS terminology from geometric elements constituting a BaS.

A path's length is the number of edges that it contains. DFS naturally generates the longest possible paths because it always continues from the most recently visited node, only branching when no further progression is possible. A path is a non-empty graph $P = (V,E)$ of the form $V = \{x_0, x_1, \ldots, x_k\}$, $E = \{x_0x_1, x_1x_2, x_{k-1}x_k\}$, where x_i are all distinct. Nodes x_0 and x_k are linked by P and called its ends; nodes $x_1, \ldots, x_{k-1}$ are the inner nodes of P.

Based on the spanning tree concept [99], Figure 2.11 shows 3 different spanning trees of graph G. The spanning tree shown in Figure 2.11c) cannot be generated by DFS. While fast, limited exploration of spanning trees by DFS constrains its effectiveness.

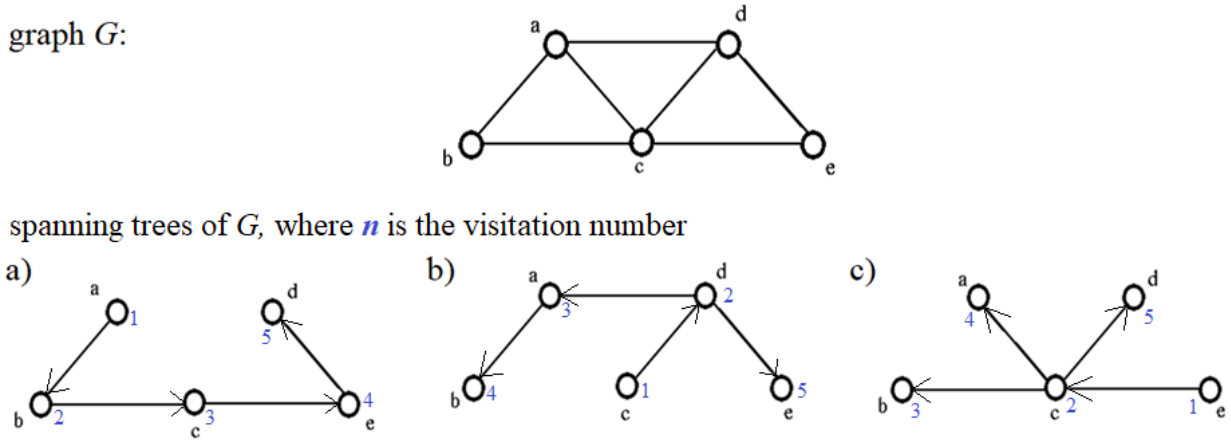


Figure 2.11 Spanning trees of G: a) , b) Spanning trees generated by DFS; c) Spanning tree impossible to generate using DFS [102]

BFS is more complex than DFS. BFS examines graph nodes sequentially by increasing distance from the initial node, facilitating distance computation that can be used for determining

paths and node visitation sequences [103]. Advancing level by level, BFS continues until all nodes at a given level are explored. BFS has been used for optimising the design of PMC plates [104], automated fibre placement [105], and other PMC-related applications [106]–[109]. Figure 2.12 shows a BFS traversal initiating at node 1, visiting nodes 2 and 3 (distance level 1), then nodes 4, 5, 6 and 7 (distance level 2), and eventually nodes 8 to 14 (distance level 3). Gray nodes represent those pending visit at distance level 4.

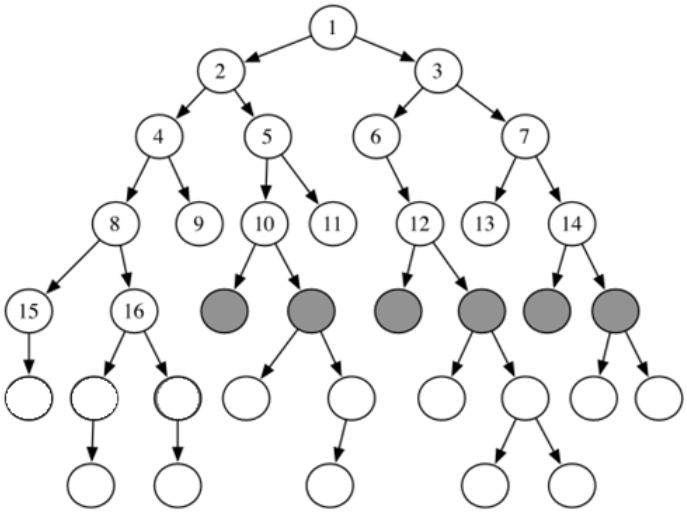


Figure 2.12 Breadth-first search steps [101]

BFS is more challenging to implement than DFS because it explores multiple parts of a graph simultaneously. While BFS is advantageous for smaller problems, its memory requirements increase rapidly with problem size because large numbers of partial solutions must be stored during the search. Consequently, BFS may become impractical for large, dynamically generated graph problems.

DFS and BFS operate on unweighted graphs and therefore do not identify least-cost paths directly, as edge costs are not considered during graph traversal. Although BFS can identify paths with fewer edges, these paths do not necessarily correspond to lower-cost solutions. For this reason, DFS and BFS are commonly used for graph exploration and solution-space generation, while cost-based evaluation is performed separately.

Unlike DFS and BFS, lowest cost path algorithms are used for analysing weighted graphs featuring edge cost data [110]. Dijkstra's algorithm determines the lowest cost path from the starting node to all graph nodes [111]. The algorithm was used for analysing carbon nanotube PMC electrical conductivity [112], [113] and PMC fatigue life prediction [114]. It offers greater efficiency than the oft-used Bellman-Ford algorithm [115] and enables large graph processing, but forbids edges carrying negative cost.

Dijkstra's algorithm maintains a set of visited nodes and a set of unvisited nodes. Initiating at the source node, it selects sequentially the next unvisited node with the smallest tentative cost from the source. Then, it examines neighbours of this chosen node and updates their tentative cost if a path with lower cost is detected [116]. This process continues until the destination node is reached, or all reachable nodes have been visited. Efficiency is achieved by processing each graph edge only once, given that no negative edges are present [89].

Figure 2.13 shows the progression of Dijkstra's algorithm on a weighted graph. Initial costs to all nodes except the starting node are infinite in step 1. In step 2, the options from node 1 are evaluated and node 5 is selected for its lower cost. In step 3, the algorithm compares visiting node 4 from node 5 (cost = 3) with visiting node 2 from node 1 (cost = 5), selecting node 4. In step 4, node 2 is selected over node 3 because of its lower tentative cost. Finally, node 3 can be reached either from node 4 (cost = 6) or from node 2 (cost = 2); the latter option is selected.

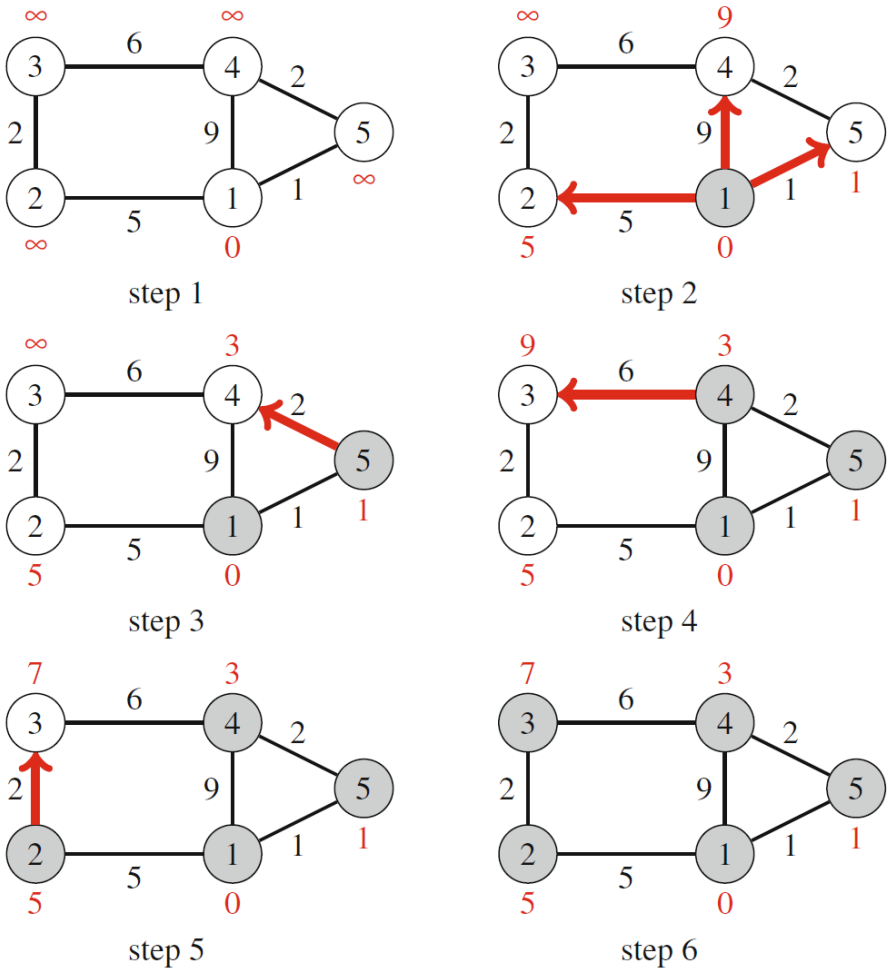


Figure 2.13 Dijkstra's algorithm in action [103]

Dijkstra's primary constraint is its greedy nature, potentially hindering the pursuit of optimal paths. As shown in Figure 2.14, the lowest cost path from node 1 to node 4 is $1 \rightarrow 3 \rightarrow 4$ with a cost of 4. Nonetheless, Dijkstra's algorithm erroneously identifies path $1 \rightarrow 2 \rightarrow 4$ by greedily pursuing minimum weight edges at each step.

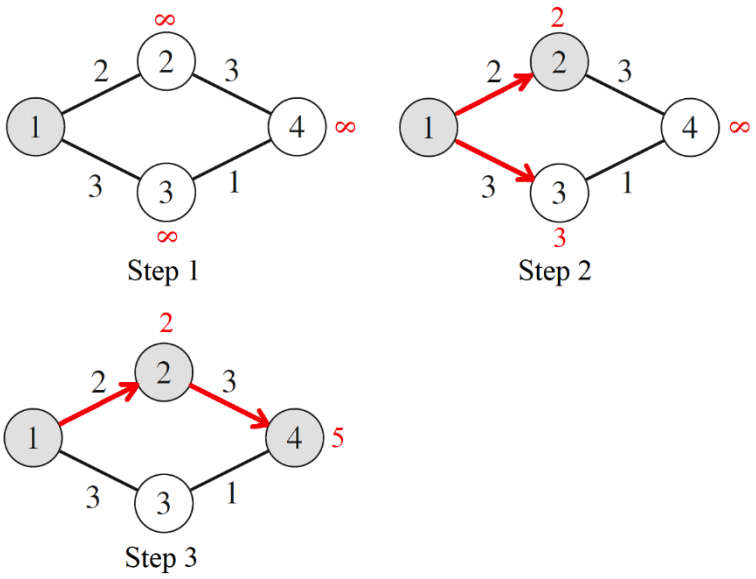


Figure 2.14 Dijkstra's algorithm failing greedily [103]

Summarizing the three search methods presented above, DFS and BFS are used for unweighted graph search problems. Dijkstra is a short path algorithm where path weights are considered. In the next section, HLU optimisation is discussed.

2.3.2 Classification of Optimisation Algorithms

Optimisation seeks to identify values of input variables that maximise or minimise an Objective Function within set bounds [117]. The Objective Function quantifies system performance from specific independent input variables. Establishing an accurate yet efficient model is crucial as overly simplistic or complex models can lead to inaccurate or laborious solutions respectively [118]. Upon model formulation, various optimisation algorithms designed for specific system types are considered and a suitable choice is made to determine the solution accurately and efficiently. Finally, optimisation algorithm performance must be evaluated [119]–

[121]. The solution must also be evaluated in terms of the actual application, possibly leading to model refinement and improvement. Each time the model is modified, the optimisation problem is solved again, in an iterative process.

A mathematical optimisation problem can be described as follows [122]:

$$\begin{aligned} &\text{Minimize} \quad f_0(\boldsymbol{x}) \\ &\text{subject to} \quad f_i(\boldsymbol{x}) \leq \boldsymbol{b}_i \quad i = 1, \ldots, m \end{aligned} \tag{2.1}$$

Vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ is the optimisation variable of the problem, where individual components are input variables. Function $f_0 : \mathbf{R}^n \rightarrow \mathbf{R}$ is the Objective Function, a scalar function of $\boldsymbol{x}$ to maximize or minimize. Functions $f_i : \mathbf{R}^n \rightarrow \mathbf{R}$, $i = 1, \ldots, m$ are inequality constraint functions with constants $b_1, b_i, \ldots, b_m$ as bounds. Vector $\boldsymbol{x}^*$ is called optimal or a solution to the problem, Equation 2.1, if it has the minimum or maximum objective value among all vectors that satisfy constraints f_i . Therefore, for any $\boldsymbol{z}$ with $f_1(\boldsymbol{z}) \leq b_1, \ldots, f_m(\boldsymbol{z}) \leq b_m$, inequality $f_0(\boldsymbol{z}) \geq f_0(\boldsymbol{x}^*)$ is always respected. Figure 2.15 shows a graphical representation of a general optimisation problem.

Understanding the different types of optimisation algorithms and how they can be grouped is the first step towards algorithm selection. Optimisation algorithms can be classified in multiple ways [117], [123]–[126].

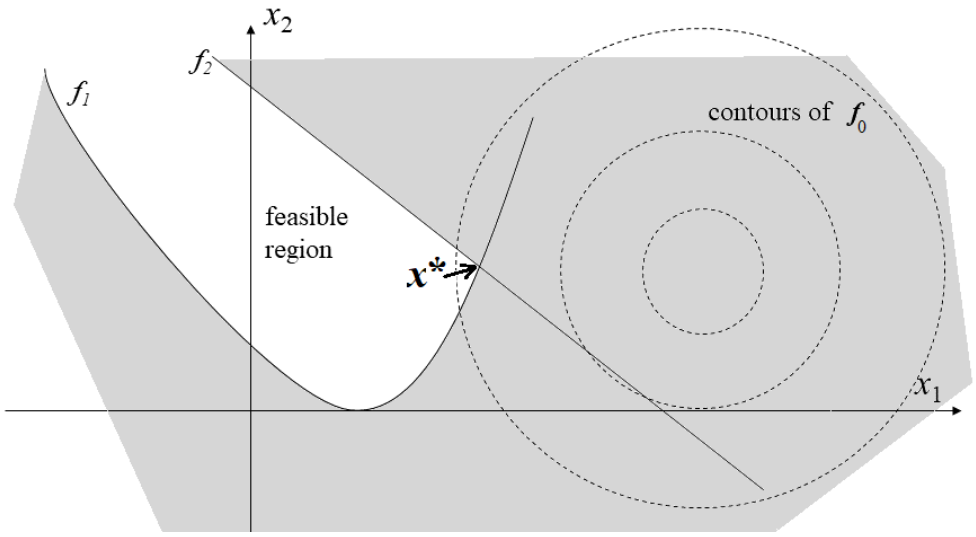


Figure 2.15 Graphic representation of an optimisation problem [117]

Optimisation problems can be discrete or continuous [127]. Continuous models are solved using numerical or calculus-based optimisation [128]. These solution methods further subdivide into direct and indirect search methods. Direct search methods rely on local gradients for moving towards a local optimum [129]. Indirect methods are local in scope and assume the existence of derivatives to the model function. These specificities narrow the applicability of numerical methods to a limited number of optimisation problems [130].

Numerical optimisation methods are generally not well suited to problems of a discrete nature, where decisions are selected from a finite set of alternatives. In such cases, the system is represented by discrete states and transitions rather than continuous variables. HLU draping sequence optimisation belongs to this class of problems, as draping is represented as a sequence of discrete operations performed over interconnected regions of the mould surface.

Optimisation problems can be constrained or unconstrained [131]. Unconstrained optimisation problems are usually set as such for simplifying the algorithm; constraints can be

substituted by penalization terms that discourage their violation. Conversely, constrained optimisation problems feature boundaries that are essential to fidelity of the model to the real-world system; examples include limits on displacement, stress, strain, or in-plane shear angle.

Optimisation problems can be classified as linear, nonlinear or convex according to the nature of the Objective Function and constraints [132]. An optimisation problem is linear when the Objective Function and all constraints are linear functions of $\boldsymbol{x}$ [133]. Linear constraints and objectives are frequently favoured in problem formulation due to their simplicity and ease of definition [134]. Even if an Objective Function is not inherently linear, defining it as such is often more straightforward than a complex alternative. Nonlinear models are often better suited to problems arising from physical sciences and engineering but they require complex formulation and more computational resources. Therefore, linearization of convex problems is often attempted, even when valid only within particular regions of the domain. Linear models are frequently used in engineering applications to represent complex material behaviour while maintaining computational efficiency.

Optimisation can be classified as local or global. Identifying a single global optimisation solution is often difficult for nonlinear problems. Local optima constitute an attractive option for identifying an acceptable solution. For convex problems and more particularly for linear problems, local solutions are also global solutions [135]. The identification of a global solution to a nonlinear problem typically requires the solution of many local optimisation problems [117]. Consequently, optimisation methods that combine local decision-making with global search strategies are frequently employed to address large and complex optimisation problems.

Optimisation problems can be categorised as single objective or multi-objective [136]. Multi-objective optimisation (MOO) deals with the simultaneous optimisation of two or more

conflicting objectives, whilst respecting all constraint functions [137]. If the objectives are linearly related, the model shall not be treated as a MOO problem [138]. For HLU optimisation, cost and quality can be combined into a single Objective Function and optimised using a single-objective approach.

Figure 2.16 shows a classification of optimisation methods proposed by Bandyopadhyay et al. [139], featuring notably Brute Force optimisation and Dynamic Programming, which are discussed in the following sections.

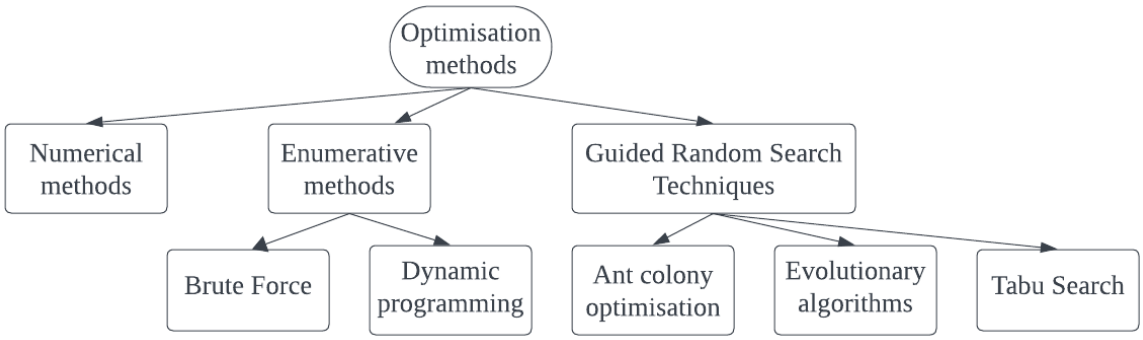


Figure 2.16 Classification of optimisation methods [139]

2.3.3 Initialization for Enumerative Optimisation Methods

Initialization plays a fundamental role in optimization algorithms, as the quality of the initial solution often influences both the convergence rate and the likelihood of reaching a global optimum. Different strategies exist, ranging from purely random initialization to heuristics that exploit prior knowledge of the problem structure [79],[80]. In many combinatorial or engineering optimization problems, heuristic initialization can reduce the search space substantially and guide the algorithm toward promising regions of feasible solutions from the outset [81],[82]. The choice

of initialization method should therefore reflect both the nature of the problem and the characteristics of the data or geometry being optimised.

Hasanzadeh and Keynia [143] demonstrated that Pareto-based initialization can improve convergence speed and solution quality of metaheuristic algorithms significantly, compared to purely random initialization. In their study, several population-based algorithms such as genetic algorithms, particle swarm optimization, and differential evolution were tested under both random and Pareto-based initial conditions. Results consistently showed that assigning higher initial weight to the top 20% of influential elements, while maintaining diversity through the remaining 80%, produced faster convergence and better global optima.

These findings suggest that initialization can influence both optimisation efficiency and solution quality. Consequently, heuristic initialization strategies are commonly used to define representative starting points and guide the search process toward promising regions of the solution space.

2.3.4 Brute Force Algorithms

Increases in computational power have led researchers to investigate the feasibility of solving mathematical and industrial problems through bold approaches. Brute force (BF) methods, where every possibility is probed, have been avoided historically due to exponential growth in complexity and data management needs, even in cases where creating and solving a mathematical model was complex. As stated by Heule [144], it may never be possible to produce a proof for some truth statements. Recent BF algorithms applications include Konev's partial solution [145]

of the combinatorial problem put forth by Erdos [146]; the size of that solution is 13 GB. Similarly, Lamb's solution [147] for the Boolean Pythagorean triples problem extends 200 TB [148].

BF methods solve optimisation problems through exhaustive evaluation of all feasible solutions. Although computationally demanding, they guarantee identification of the global optimum provided that the complete solution space can be explored. BF optimisation is applied in the following form:

$$\text{Minimize } \{f(\boldsymbol{x}) \mid \boldsymbol{x} \in \boldsymbol{X}\} \tag{2.2}$$

where set $\boldsymbol{X}$ summarizes constraints on decision variables $\boldsymbol{x}$, and

$$\boldsymbol{X} \subset \ddot{X} \mid \ddot{X} = nPr$$

$$n = r = \#BaS$$

where $\#BaS$ is the total number of BaS in the geometry.

For sequence-planning problems, BF can be used to generate and evaluate all feasible decision sequences, making it a useful benchmark for assessing the performance of more computationally efficient optimisation methods.

2.3.5 Dynamic Programming Algorithms

Dynamic programming (DP) is a method for solving complex problems by breaking them into simpler overlapping subproblems and solving each subproblem only once, storing the solutions to subproblems in a table to avoid redundant computations. This technique is particularly useful for optimization problems where the goal is to find the best solution among a set of possible solutions [149].

DP is a suitable alternative to BF for combinatorial problems featuring large decision spaces. DP is a efficient enumerative search technique that identifies a near optimal solution through the recursive solution of multiple local optima [150]. For combinatorial models, solving local optima within feasible subspaces can significantly reduce computational time and resource requirements [151]. DP algorithms have proven their efficacy for many optimisation applications, taking advantage of quick solutions to small subproblems to achieve a near optimal solution [152]. DP is also well suited to trajectory optimisation problems [153], and other problems that can be represented as a sequence of interconnected decisions, such as draping sequence planning.

DP problem specification is commonly done in the form of a nonlinear recursive equation, labelled dynamic programming functional equation (DPFE). The key task is to formulate the problem in terms of a DPFE, such that the solution of the recursive DPFE is the solution of the optimisation problem. This multistage decision process optimisation is based on the principle of optimality, which states that whatever the initial state and initial decision of an optimal policy are, remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision [154]. Equation (2.1) describing BF can be modified and solved using DP as shown in Equation (2.3):

$$\boldsymbol{H}^{*} = \text{opt}\ (d_1, d_2, \ldots, d_n) \in \Delta\{H(d_1, d_2, \ldots, d_n)\} =$$

$$= \text{opt}_{d1 \in D1}\{\ \text{opt}_{d2 \in D2(d1)}\{\ldots\{\ \text{opt}_{dn \in Dn(d1,\ldots,dn-1)}\{\ H\ (d_1, \ldots, d_n)\}\}\ldots\}\}. \quad (2.3)$$

where $\boldsymbol{H^*}$ is the optimal solution of function H obtained through recursive solving of the DPFE for each decision d_i taken from an element D_i in a set of $\{D\}$.

For a DP problem, conditions must exist where the recursive DPFE must stop being solved. Let function $f\ (d_1, \ldots, d_{i-1})$ be defined as the optimal solution of the sequential decisions. Decisions $d_1, \ldots, d_{i-1}$ are already made, and decisions $d_i, \ldots, d_n$ are yet to be made. Then, $f\ (d_1, \ldots, d_{i-1})$ can be rewritten as Equation (2.4):

$$f\ (d_1, \ldots, d_{i-1}) = \text{opt}_{di \in Di(d1,\ldots,di-1)}\{C_i(d_i|d_1, \ldots, d_{i-1}) + f(d_1, \ldots, d_i)\}. \quad (2.4)$$

where the optimal solution $f\ (d_1, \ldots, d_{i-1})$ is equal to the optimum decision d_i found inside decision space D_i, which evaluates the cost of decision d_i from a state defined by previous decisions $d_1, \ldots, d_{i-1}$, plus the cost of all previous decisions, C_i. Equation (2.4) can be rewritten as follows:

$$\boldsymbol{S^*} = \text{opt}_{di \in Di(S)}\{C_i(d_i|S) + f(S')\}. \quad (2.5)$$

where $\boldsymbol{S^*}$ is the optimal solution of the system state, S is a state within a set $Ş$ of possible states, and $S' = (d_1, \ldots, d_i)$ is a next state. Depending on the problem formulation, the cost function $C(d|S)$ may account only for the immediate effect of decision d, or it may incorporate information from future states through look-ahead strategies. Such approximations are commonly used to reduce

computational complexity whilst retaining part of the future information relevant to the optimisation process, as observed in Equation (2.6) for 1-next state and (2.7) for 2-next states:

$$\boldsymbol{S}^{*} = \mathrm{opt}_{d \in S}\{C(d|S) + f(S')\}. \tag{2.6}$$

$$\boldsymbol{S}^{*} = \mathrm{opt}_{d \in S}\{C(d|S) + f(S') + f(S'')\}. \tag{2.7}$$

where every element being summed in (2.5) to (2.7) belong to the set of real numbers $\mathbb{R}$.

Decision space D_i is the feasible subspace in which local optima are sought by a DP algorithm. It contains all decisions that may be selected from a given system state. The definition of D_i is problem-dependent and directly influences both the quality of the solution and the computational effort required. The decision space adopted for HLU optimisation is presented in Chapter 6.

DP facilitates the identification of local and global optimal solutions [155]. DP algorithms tend to be compact as they use recursive functions. Both linear and non-linear problems can be addressed using DP [156]. Conversely, recursion demands more memory, potentially causing stack overflow during runtime.

Memoization is a top-down dynamic programming technique that speeds up recursive algorithms by storing the results of subproblems the first time they are solved. When the same subproblem reappears, the algorithm reuses the stored value instead of computing again. This eliminates repeated work and can improve runtime dramatically, at the cost of using additional

memory [158]. One disadvantage is that storing solutions for each sub-problem consumes memory with no guarantee that stored values will be utilised latter in execution [157].

One of the main disadvantages of DP is that it can get trapped in local optima [159] due to its dependency on the assumption that a complex problem can be broken down into smaller subproblems, the optimal solutions of which can be combined into a near optimal solution. If this condition known as the optimal substructure property is absent, the method may propagate solutions that are only optimal within their restricted scope. Consequently, the final result may represent a local rather than a global optimum, limiting the validity of the approach.

Cool et al. [160] reviewed various connectivity constraints in topology optimization and emphasized that enforcing connectivity strongly affects design outcomes. Similar principles can be applied to sequence-planning problems, where connectivity constraints help prevent infeasible or impractical intermediate states. In the context of draping, connectivity preservation can be used to avoid situations where undraped BaS become surrounded by previously draped regions, leading to more realistic draping sequences.

While the DP formulation presented in (2.4) - (2.7) describes the exact recursive solution of the optimisation problem, its direct application becomes computationally prohibitive due to the factorial growth of the decision space. Therefore, practical implementations often rely on approximate dynamic programming strategies, where the value function is estimated using limited lookahead or heuristic evaluations. This trade-off between optimality and computational tractability is particularly relevant for high-dimensional combinatorial problems such as draping sequence optimisation.

2.3.6 Optimisation Summary

Good optimisation algorithms possess the following properties [117]:

- Robustness: the algorithm performs satisfactorily for a wide variety of problems of similar nature, and for a wide range of initial values;
- Efficiency: the solution is obtained within a reasonable amount of time, without need for sizeable computational resources;
- Accuracy: the algorithm identifies the solution with precision whilst avoiding over-sensitivity to error in input data.

The HLU optimisation problem presents the following characteristics: discrete nodes, multiple interconnected decisions, and a finite search space. Consequently, it can be classified as a discrete, non-convex optimisation problem where both the Objective Function and feasible subspace may be arbitrary. Solving such problems presents a significant challenge due to the potentially large number of feasible solutions [161]. The literature reviewed in this chapter identifies BF and DP as suitable optimisation approaches for addressing these characteristics.

Before these optimisation algorithms can be applied, a mathematical representation of the HLU process is required. Such a model must describe how draping progresses across a mould surface, how yarn orientations evolve between neighbouring BaS, and how alternative draping sequences can be represented. The development of this modelling framework is presented in Chapter 3 and forms the foundation upon which the optimisation methods introduced in subsequent chapters are built.

Chapter 3. Modelling the Hand Lay-up Process

This chapter presents the mathematical and algorithmic framework developed to model the HLU process. Following the geometric parameter extraction and processing stage described previously, two distinct sets of geometric parameters are used within the overall methodology. The first set, introduced in Section 1.2.2, consists of the Frustum parameters that are used for interacting with Database A. The second set of geometric parameters is introduced in this chapter and is specifically defined to support the HLU modelling and optimisation algorithms. Figure 3.1 provides a high-level overview of the methodology developed in this thesis and illustrates how these geometric parameters interact with the main components of the framework.

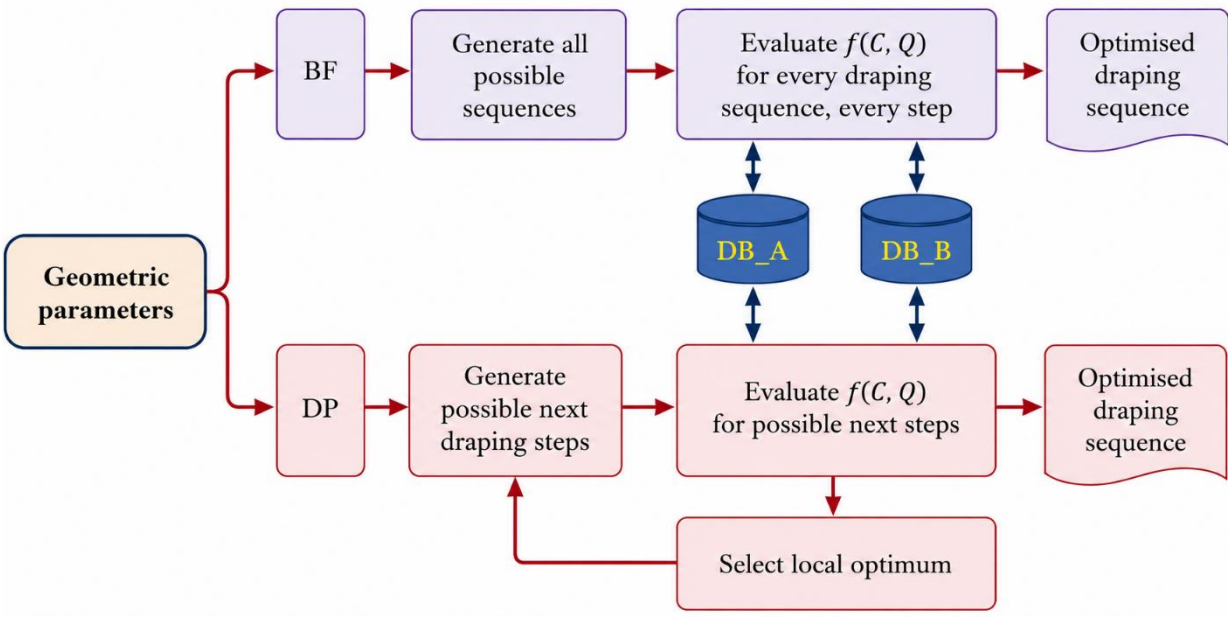


Figure 3.1 High-level workflow of the proposed optimisation framework

This chapter focuses on simulating the progression of fabric draping across complex geometries by determining the evolution of yarn orientations as they transition between adjacent BaS. It begins with a detailed explanation of the rotation and interpolation methods used for transferring warp and weft vectors across flat, single-curvature and double-curvature surfaces. It then introduces the logic for handling multiple draped neighbours and the integration of search algorithms to generate viable draping sequences. By establishing this geometric foundation, the model provides the data necessary for evaluating the Objective Function and predicting manufacturing defects such as wrinkling.

3.1 Yarn Orientations

Figure 3.2 shows the normalized vector reference frame used for defining warp and weft yarn orientations on each BaS. This section details how warp and weft yarn orientations are determined upon draping successive BaS. Initial warp and weft orientation vectors on the first BaS are set by the user and saved as *orientation* using a normalized vector reference frame for each BaS, where the first reference frame normalized vector *v_red* is parallel to the first edge of the BaS and the second reference frame normalized vector *v_black* completes the set of coplanar orthogonal normalized vectors associated with the BaS.

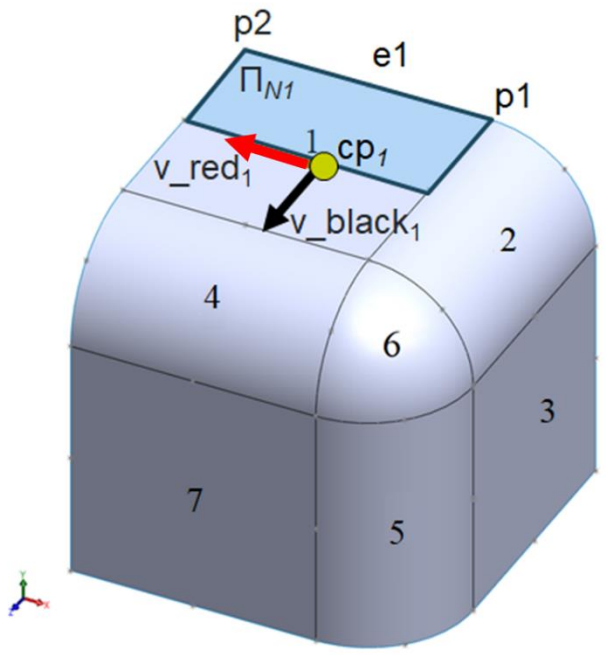


Figure 3.2 Geometric parameters for calculating reference frame vectors *v_red* and *v_black*

Each BaS #*i* is assigned a plane Π_{Ni} defined by its v_red_i and its centre point cp_i. Algorithm 3.1 for determining *v_red* and *v_black* vectors is as follows.

Algorithm 3.1 Generation of reference frame vectors Red and Black

```
For every BaS_i:
     Define centre_point (BaS_i):
          centre_point = mean(middle_edge_points (BaS_i))
          return centre_point
     Define v_red (first_edge(BaS_i):
          extract p1 and p2 from first_edge(BaS_i)
        v_red = normalized vector from p1 to p2
        return v_red
     Define v_black(v_red, centre_point, first_edge(BaS_i)):
          extract p1 and p2 from first_edge(BaS_i)
```

```
    normal_plane = (p2 - p1) × (centre_point - p1)
    v_black = (normal_plane × v_red)
    return v_black
```

Functions for calculating yarn orientations when draping subsequent flat, single or double curvature surfaces beyond the first BaS are presented below. For flat or single curvature BaS, warp and weft vectors are transferred to the next BaS using rotations. For double curvature BaS, in-plane shear induced upon draping the surface is added to yarn orientations in a three-step method. In all cases, information from all previously draped neighbours is used.

To support these calculations, yarn orientations are managed through the Orientations Matrix, a data structure that stores the warp and weft orientation vectors together with the corresponding in-plane shear angle for every BaS. The matrix is updated after each draping decision and queried whenever a subsequent draping step is evaluated, ensuring consistent propagation of yarn orientations throughout the draping sequence.

When analyzing a draping step of the type [*BaS_a* → *BaS_b*], the draping step can be a starting draping step or not. Starting draping steps have the particularity that *BaS_a* does not have any draped neighbour(s) hence yarn orientations for *BaS_a* must be calculated through Algorithm 3.2 as shown below.

Algorithm 3.2 Calculation of warp and weft vectors for starting BaS

```
Define oriented_warp_weft(v_red, v_black, orientation):
    normal_vector = (v_red × v_black)
    rotation_matrix(normal_vector, orientation)
```

```
        oriented_vector_warp = (rotation_matrix · v_red)
    Calculate orthogonal vector weft in the plane:
        projection = (oriented_vector_warp · v_black) /
    (oriented_vector_warp · oriented_vector_warp)
        oriented_vector_weft = v_black - projection *
        oriented_vector_warp
    Normalize oriented vectors warp and weft
return oriented_vector_warp, oriented_vector_weft
```

For a randomly selected draping sequence, for example [1–2, 2–3, 2–6, 3–5, 5–7, 7–4], Figure 3.3 shows the reference frame vectors *v_red* and *v_black* in red and black, along with warp and weft orientation vectors *oriented_vector_warp* and *oriented_vector_weft* in green and yellow. The local initial orientation is defined as a 45° warp orientation with an initial 0° shear angle between warp and weft.

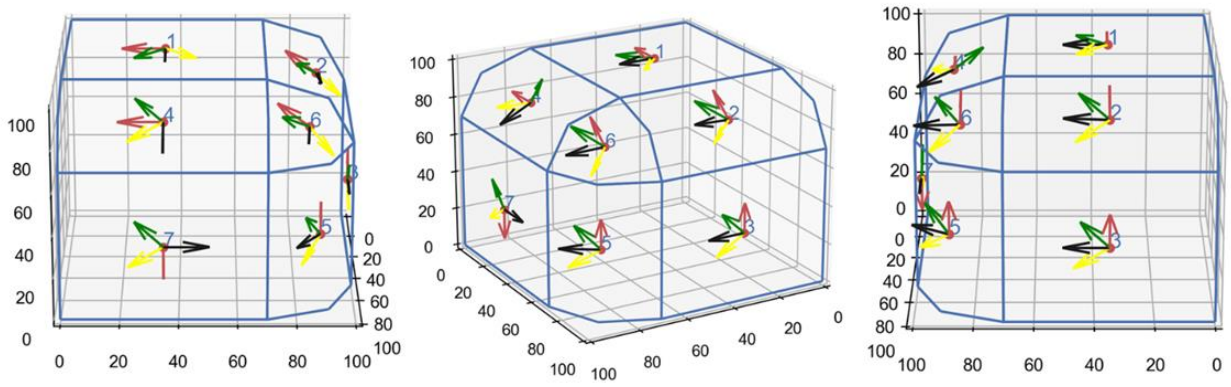


Figure 3.3 Reference vectors and warp & weft orientation vectors over neighbouring BaS

3.1.1 Yarn Orientations for Flat and Single Curvature Base Surfaces

Figure 3.4 shows successive draping steps for demonstrator 27_Hutchinson, highlighting the evolution of warp and weft yarn orientations as draping progresses across adjacent BaS. Draping starts from BaS #3 with an initial yarn orientation of 0°/90° and follows the sequence [3-4, 4-1, 4-5, 5-6, 3-2]. The figure shows how yarn orientations are propagated and updated after each draping step as the reinforcement advances across the mould surface.

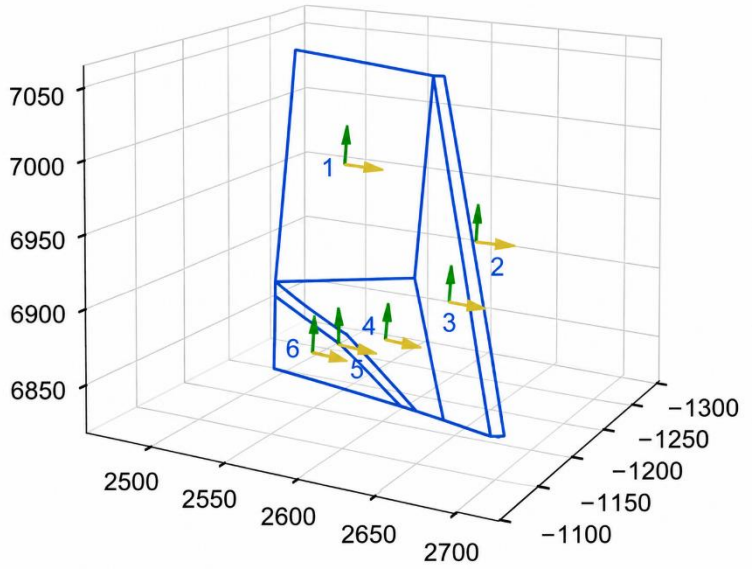


Figure 3.4 Warp and weft orientations passed from BaS to BaS, case with low in-plane shear

If starting *BaS_a* is flat or single curvature, then warp and weft vectors are determined, saved and transferred to the next draped *BaS_b* using function **rotate_vectors_in_plane** as shown in Algorithm 3.3 below, where index 1 is used for *BaS_a* data and index 2 is used for *BaS_b* data.

Algorithm 3.3 Warp and weft vectors rotation for BaS with draped neighbours

Define rotate_vectors_in_plane (*v_warp1, v_weft1, v_red1, v_black1, v_red2, v_black2*):

Calculate perpendicular vectors *perp1* and *perp2*

perp1 = (*v_red1* × *v_black1*)

perp2 = (*v_red2* × *v_black2*)

Move *perp1* and *perp2* to origin <0, 0, 0>

perp1_origin =

*perp1 – (perp1 · v_red1) * v_red1 – (perp1 · v_black1) * v_black1*

perp2_origin =

*perp2 – (perp2 · v_red2) * v_red2 – (perp2 · v_black2) * v_black2*

Create *Rotation Matrix*

ortho = (perp1_origin × perp2_origin)

Create *initial_plane_matrix*

initial_plane_matrix = [$perp1\_origin^T$, $ortho^T$, $(perp1\_origin \times ortho)^T$]

Create final plane matrix

final_plane_matrix = [$perp2\_origin^T$, $ortho^T$, $(perp2\_origin \times ortho)^T$]

Calculate rotation matrix

rotation_matrix = (final_plane_matrix · $initial\_plane\_matrix^{-1}$)

Rotate vectors using rotation matrix

v_warp2 = (rotation_matrix · v_warp1)

v_weft2 = (rotation_matrix · v_weft1)

Normalize rotated vectors

return v_warp2, v_weft2

v_warp2 and v_weft2 are in the same plane than *v_red2* and *v_black2*

3.1.2 Yarn Orientations for Double Curvature Base Surfaces

If *BaS_a* has double curvature, initial shear angle on every edge of *BaS_a* is interpolated from linear equations and *BaS_a* draping data contained in Database A. Then, a single value of the shear angle is set for *BaS_a*: shear angle over each individual BaS takes a single value in the model. Two different settings were developed for determining these single values of the shear angle. The first setting consists in selecting the maximum shear value calculated at every edge, Equation (3.1); the second setting consists in calculating a weighted average shear angle value, Equation (3.2). Under the first setting, in-plane shear angle for *BaS_a* is the maximum value,

$$\theta_{Max\_BaS\_a} = \max(\boldsymbol{\theta_a}) \tag{3.1}$$

where $\boldsymbol{\theta_a}$ is the set of shear angle values calculated at the midpoint of every edge of *BaS_a.* Under the second setting, in-plane shear angle for *BaS_a* is a weighted average,

$$\theta_{Avg\_BaS\_a} = \frac{\sum_1^n \theta_a l_{ei}}{\sum_1^n l_{ei}} \tag{3.2}$$

where n is the total number of edges for BaS and l_{ei} is the length of edge i.

Figure 3.5 shows the three steps used to update warp and weft yarn orientations for BaS #6. The blue arrows indicate the sequential draping steps corresponding to draping sequence [6-2, 2-1, 2-3, 1-4, 3-5, 5-7]. After a single value of in-plane shear is calculated for the double-curvature

BaS, this value is added to the initial warp and weft vectors through the three steps shown in the figure.

Firstly, warp and weft vectors are rotated to an *x-y* plane using Algorithm 3.3, where *v_red2* = (1,0,0) and *v_black2* = (0,1,0) . Secondly, shear is added to the warp and weft vectors using Algorithm 3.4 In-plane shear angle addition in plane *x-y*. Finally, 2D sheared warp and weft vectors are returned to their original 3D plane using Algorithm 3.3 .

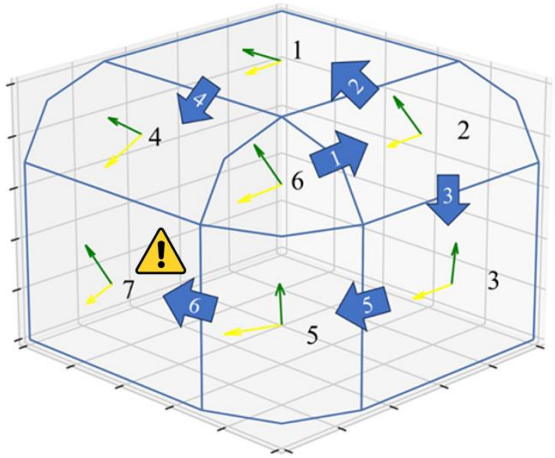


Figure 3.5 Yarn orientations for demonstrator 30_Corner.

Algorithm 3.4 In-plane shear angle addition in plane *x-y*

```
Define add_shear_2D(warp, weft, shear):
    Normalize warp and weft vectors
    Calculate mean_vector = warp + weft
    Calculate rotation_angle = angle between mean_vector and
    positive x-axis, 0° to 360°
    Rotate (warp, weft, mean_vector) by rotation_angle
    Calculate angle_warp between positive x-axis and
    rotated_warp
    Add shear:
        If i element of rotated_warp(i,j)>= 0,
```

```
            theta = angle_warp + shear/2
        else:
            theta = angle_warp - shear/2
            sheared_warp = Rotate (rotated_warp) by theta
    Normalize sheared_warp
    Calculate sheared_weft = sheared_warp(i, -j)
    Rotate vectors to their original region in 3D plane:
    Rotate (sheared_warp, sheared_weft) by -1*rotation_angle
    return sheared_warp, sheared_weft
```

3.1.3 Yarn Orientations for a Base Surface with Draped Neighbours

If draping step [*BaS_a* → *BaS_b*] is not the first draping step, *BaS_a* is identified as draped and the software proceeds to analyzing the draping of *BaS_b*. All draping geometric information from the draped neighbours of *BaS_b* is transferred to *BaS_b*. First, all draped neighbours from *BaS_b* are identified and listed, along with yarn orientations and length of the edge that every draped neighbour shares with *BaS_b*. Then the algorithm proceeds to calculating a weighted average to determine the yarn orientations for *BaS_b*. This operation may generate in-plane shear.

Algorithm 3.5 Affectation of draped neighbours' warp and weft orientations

```
For all draped_neighbours of BaS_b:
   Rotate warp and weft vectors into BaS_b plane
   Save rotated_vectors in neighbours_warp_weft_rotated matrix
   If amount of draped_neighbours = 1
         Do nothing.
   Else, if amount of draped_neighbours > 1:
      For all rotated_vectors in neighbours_warp_weft_rotated:
         # Permutation list for memoization purposes #
```

```
Create permutation_list of all rotated_vectors
Calculate weighted_avg_warp
Calculate weighted_avg_weft
```

If the generated in-plane shear is larger than the experimentally defined threshold observed in step 6 of Figure 3.2, then the software alerts the user of possible wrinkling and provides an edge darting recommendation when applicable, as described in Section 3.4.

Finally, if *BaS_b* is either flat or has single curvature, the computed warp and weft vectors accounting for the influence of neighbouring BaS are stored for use in subsequent draping step analyses. Otherwise, if *BaS_b* has double curvature, in-plane shear is calculated and added to the previously calculated warp and weft orientations for *BaS_b*.

3.2 Hand Lay-up Progression and Draping Sequences

HLU drape modelling aims at describing the progressive, sequential covering of a mould by a pattern. In this thesis, a discrete description of draping is proposed where the following conditions are met:

- Draping of a pattern always starts from only one BaS;
- Draping progresses from a previously draped BaS;
- Draping progresses one BaS at a time;
- Draping ends when all BaS of a mould are draped.

Different approaches to HLU draping optimisation are explored in this thesis. Each optimisation algorithm searches draping possibilities, evaluates them, and compares them. Generating draping sequences first without evaluating the cost or quality associated with these sequences at the same time is labelled an unweighted search task. In this work, DFS and BFS algorithms were used in unweighted searches, generating as many draping sequences as possible for a given mould.

3.2.1 Generating Draping Sequences Using Unweighted Depth-First Search

A DFS algorithm was developed to explore systematically all valid draping sequences across a mould. This approach identifies every potential draping sequence by going through the mould's base surfaces as a search graph, ensuring physical continuity.

At each step, the sequence is extended by selecting an undraped neighbor of the most recently draped BaS. This enforces a strictly local propagation of the draping process. However, when no such neighbour exists (i.e., the sequence reaches a dead-end), a recovery mechanism is activated: the stuck function. In this case, the next BaS may be selected from the set of undraped neighbours of any previously draped BaS, thereby enabling branching and preventing premature termination of the sequence.

The generation of feasible draping sequences is formulated as a constrained traversal over a graph. Let

$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}) \tag{3.3}$$

be an undirected graph, where $\boldsymbol{V}$ is the set of BaS, $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ is the set of graph-edges representing neighbouring relationships, and $\boldsymbol{V} \times \boldsymbol{V} = \{(u, v) \mid u \in \boldsymbol{V}, v \in \boldsymbol{V}, u \neq v\}$ generates all possible ordered pair of graph-nodes, i.e., all the possible pairs of draping steps.

A draping sequence is defined as an ordered list of nodes:

$$\boldsymbol{S} = [v_1, v_2, \dots, v_k], \quad v_i \in \boldsymbol{V} \tag{3.4}$$

where each element represents the order in which BaS are draped.

A draping sequence $\boldsymbol{S}$ is considered valid if it satisfies the following conditions:

i) Uniqueness constraint – Each BaS is draped at most once, preventing revisiting previously draped BaS.

$$v_i \neq v_j \quad \forall\, i \neq j \tag{3.5}$$

ii) Conditional adjacency and expansion rule – At each step, the sequence is extended by selecting a new BaS according to a priority rule. First, define the set of forward candidates from the last node:

$$Next_{forward}(\boldsymbol{S}) = \{u \in \boldsymbol{N}(v_k) \mid u \notin \boldsymbol{S}\} \tag{3.6}$$

where $\boldsymbol{N}(v_k)$ denotes the set of neighbours of the last node v_k

iii) Local propagation – Always continue draping from the last draped BaS, if forward candidates exist:

$$Next_{forward}(\boldsymbol{S}) \neq \emptyset \tag{3.7}$$

then the next node must satisfy:

$$v_{k+1} \in Next_{forward}(\boldsymbol{S}) \tag{3.8}$$

iv) Stuck condition – dead-end detection. A draping sequence is considered stuck when no forward candidates are available:

$$Next_{forward}(\boldsymbol{S}) = \emptyset \tag{3.9}$$

v) Branching – When condition iv) is valid, a branching mechanism is activated. The next node is selected from the set of undraped neighbours of any previously draped BaS:

$$Next_{branch}(\boldsymbol{S}) = \bigcup_{v_j \in \boldsymbol{S}} \{u \in \boldsymbol{N}(v_j) \mid u \notin \boldsymbol{S}\} \tag{3.10}$$

and

$$v_{k+1} \in Next_{branch}(\boldsymbol{S}) \tag{3.11}$$

This allows the algorithm to branch from earlier nodes in the sequence, avoiding premature termination.

The sequence extension can be compactly expressed as:

$$v_{k+1} \in \begin{cases} \{u \in \boldsymbol{N}(v_k) \mid u \notin \boldsymbol{S}\}, & if\ \exists u \in \boldsymbol{N}(v_k) \backslash \boldsymbol{S} \\ \bigcup_{v_j \in \boldsymbol{S}} \{u \in \boldsymbol{N}(v_j) \mid u \notin \boldsymbol{S}\}, & otherwise \end{cases} \tag{3.12}$$

where the objective of the algorithm is to construct the set:

$$\hat{\boldsymbol{S}} = \{\boldsymbol{S} \mid \boldsymbol{S}\ satisfies\ all\ constraints\ above\} \tag{3.13}$$

Algorithm 3.6 outlines the iterative logic and backtracking routines used for generating these sequences.

Algorithm 3.6 DFS draping sequences generator

```
# Inputs:                                                    #
# G = (V, E)    i.e. MinMax matrix and 3D neighbouring matrix  #
# N(v)          Function returning neighbours of BaS v         #
# Output:                                                    #
# S_all         Set of all valid draping sequences             #
```

```
# List to store all valid sequences
Initialize S_all = []
# Loop over all possible starting BaS (v_start)
For each node v_start in V:
   Initialize queue stack Q
   Push [v_start] into Q    #start sequence from v_start#
   #Explore all sequences starting from v_start#
   While Q is not empty:
      S = Pop(Q)            #Current sequence#
      v_k = last element of S   #Most recently draped BaS#
      #Step 1: Forward expansion – priority rule#
      #Try to continue from the last node#
      Next_forward = {u in N(v_k) such that u not in S}
      If Next_forward is not empty:
         #Local growth#
         For each node u in Next_forward:
            S_new = S + [u]     #Extend sequence#
            Push S_new into Q   #Continue exploration#
      Else:
         #Step 2: Stuck condition → Branching enabled#
         Next_branch = empty set
         #Generate all possible next steps from all previously#
         #draped BaS#
         For each node v_j in S:
            Neighbours_j = N(v_j)
            For each node u in Neighbours_j:
                If u not in S:
                    Add u to Next_branch
         #Branching Expansion#
         For each node u in Next_branch:
```

```
            S_new = S + [u]      #Extend sequence#
            Push S_new into Q
        #Break if full sequence reached#
        If length(S)==|V|:
          Add S to S_all
#Output all valid sequences#
return S_all
```

Algorithm 3.6 was validated on industrial demonstrators. Figures 3.6–3.8 illustrate the geometry of demonstrator 27_Hutchinson, the search graphs of two randomly selected draping sequences (DFS), and their representation on the mould geometry.

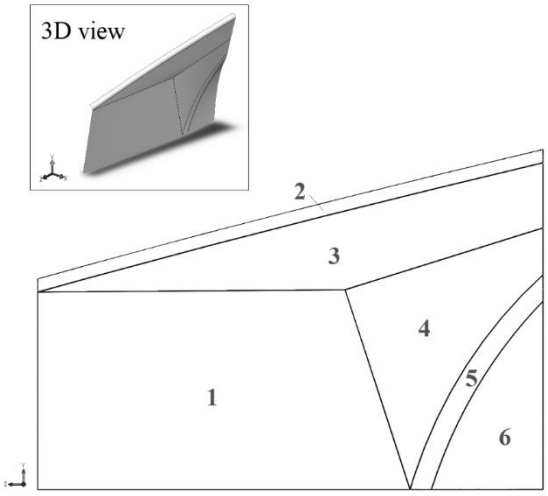


Figure 3.6 Demonstrator 27_Hutchinson; Labelled BaS

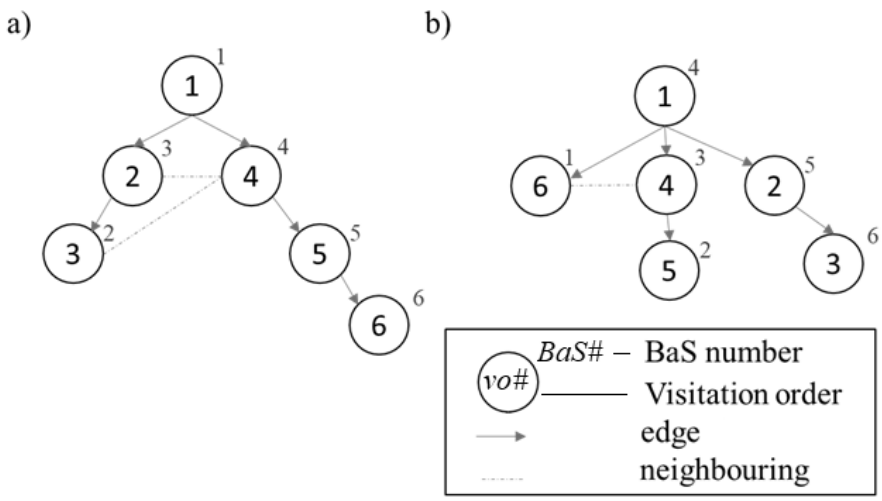


Figure 3.7 DFS for 27_Hutchinson: a) Starting from BaS #1; b) Starting from BaS #4

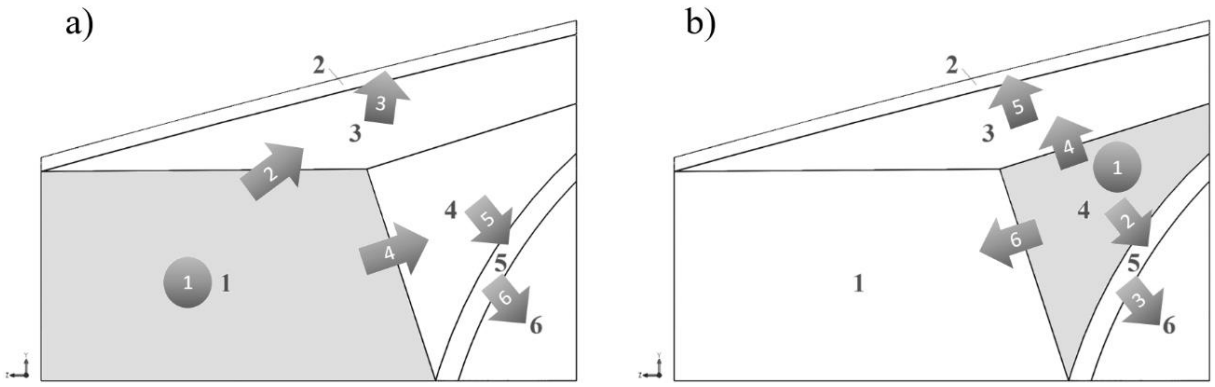


Figure 3.8 DFS sequences: a) Starting from BaS #1; b) Starting from BaS #4

Individual DFS generated draping sequences shown in Figures 3.8 a) and 3.8 b) are saved as individual rows in arrays; all sequences starting at the same BaS are saved in one distinct array under the format shown in Equations (3.14) and (3.15) . For example, all sequences starting at *BaS_1* are saved in *seqs_array_1* , and all sequences starting at *BaS_4* are saved in *seqs_array_4* , etc.

$$seqs\_array\_1 = \begin{bmatrix} [sequence\ 1\ starting\ at\ BaS\_1] \\ \vdots \\ [1-3,\ 3-2,\ 1-4,\ 4-5,\ 5-6] \\ \vdots \\ [sequence\ n\ starting\ at\ BaS\_1] \end{bmatrix} \quad (3.14)$$

$$seqs\_array\_4 = \begin{bmatrix} [sequence\ 1\ starting\ at\ BaS\_4] \\ \vdots \\ [4-5,\ 5-6,\ 4-3,\ 3-2,\ 4-1] \\ \vdots \\ [sequence\ n\ starting\ at\ BaS\_4] \end{bmatrix} \quad (3.15)$$

3.2.2 Generating Draping Sequences Using Unweighted Breadth-First Search

The BFS approach explores sequences level by level, unlike DFS which goes deep first. In the context of draping sequences, this ensures that all sequences of length *k* are explored before moving to length *k*+1. BFS graph formulation is as follows. Let Equation (3.3)

$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$$

be an undirected graph, where $\boldsymbol{V}$ is the set of BaS, $\boldsymbol{E} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ is the set of graph-edges representing neighbouring relationships between BaS, and $\boldsymbol{V} \times \boldsymbol{V} = \{(u, v) \mid u \in \boldsymbol{V}, v \in \boldsymbol{V}\}$ generates all possible ordered pair of graph-nodes, i.e., all the possible pairs of draping steps.

A draping sequence is defined by Equation (3.4) as an ordered list of nodes where each element represents a BaS in the order it is draped:

$$\boldsymbol{S} = [v_1, v_2, \dots, v_k], \quad v_i \in \boldsymbol{V}$$

The proposed Breadth-first search algorithm is subjected to the following conditions:

i) Forward expansion from the last draped node, Equation (3.6) – Attempt to extend the draping sequence from v_k

$$Next_{forward}(\boldsymbol{S}) = \{u \in \boldsymbol{N}(v_k) \mid u \notin \boldsymbol{S}\}$$

ii) Expansion from all previously draped nodes – To ensure complete exploration, BFS also considers neighbours of all nodes in the current sequence

$$Next_{all}(\boldsymbol{S}) = \bigcup_{v_j \in \boldsymbol{S}} \{u \in \boldsymbol{N}(v_j) \mid u \notin \boldsymbol{S}\} \tag{3.16}$$

iii) Sequence generation – for every $u \in Next_{all}(S)$, create a new sequence $S_{new} = S + [u]$ and enqueue it in the BFS queue. This ensures that all possible draping sequences are explored breadth-first while giving priority to forward expansion from the last draped BaS.

Algorithm 3.7 presents the BFS draping sequences generation.

Algorithm 3.7 BFS draping sequences generator

```
# Inputs:
# G = (V, E)     # i.e. MinMax matrix and 3D neighbouring matrix
# N(v)           # Function returning neighbours of BaS v
# Output:
# S_all          # Set of all valid draping sequences
# List to store all valid sequences
Initialize S_all = []
# Loop over all possible starting BaS (v_start)
For each node v_start in V:
    Initialize queue stack Q
    Enqueue [v_start] into Q    # Start sequence from v_start

    # Explore all sequences starting from v_start
    While Q is not empty:
        S = Dequeue(Q)              # Current sequence
        v_k = last element of S     # Most recently draped BaS

        # Step 1: Forward expansion - priority rule
        # Try to continue from the last node
        Next_nodes = set()
        # Forward-first: last node expansion
        For each u in N(v_k):
            If u not in S:
                Add u to Next_nodes

        # Step 2: Full neighbour expansion from all previously
        #draped BaS
        For each v_j in S:
            For each u in N(v_j):
```

```
                If u not in S:
                    Add u to Next_nodes

        # Step 3: Enqueue all new sequences
        For each u in Next_nodes:
            S_new = S + [u]       # Extend sequence
            Enqueue S_new into Q  # Continue exploration

        # Store sequence if full length reached
        If length(S) == |V|:
            Add S to S_all

# Output all valid sequences
return S_all
```

Figure 3.9 shows a search graph of a draping sequence generated by BFS for demonstrator 27_Hutchinson, starting from BaS #1.

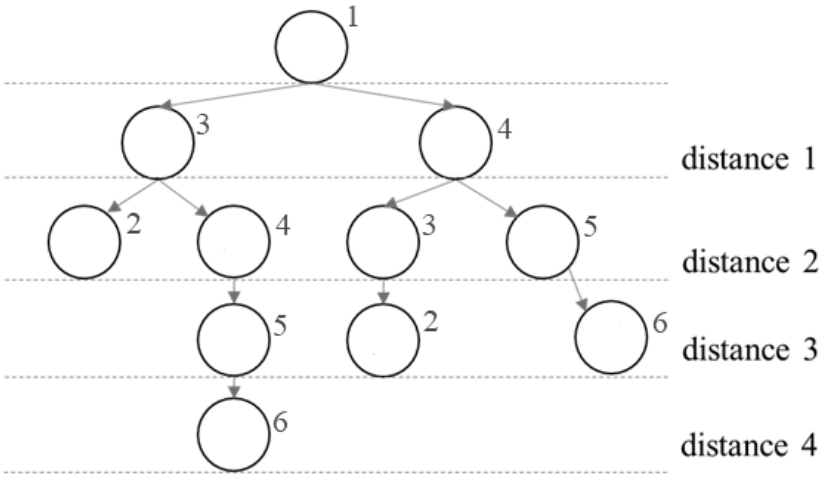


Figure 3.9 BFS graph for demonstrator 27_Hutchinson

Every node in Figure 3.9 represents a BaS. The arrows represent the possible draping steps at every distance stage. When draping cannot continue from a given BaS but other undraped BaS remain, the stuck function searches all other options to keep draping, starting from another draped BaS, until all BaS are draped. For example, sequence [1-3, 3-2, …] cannot continue from BaS#2 beyond distance 2 as BaS#2 has no undraped neighbours. Other options exist, such as [1-3, 3-2, 3-4, …] and [1-3, 3-2, 1-4, …]. Both options are written in array *seqs_array_1*.

For every distance, BFS generates the possible draping steps to be saved in the array corresponding to the initial BaS draped. The amount of possible draping sequences generated by BFS grows exponentially, compromising computational capacity.

3.3 Wrinkle Warning and Edge Darting Recommendation Functionalities

Yarn orientations and fabric in-plane shear angles are determined for each BaS upon each draping step. These quantities are used for quantifying strain energy imparted to the textile reinforcement stack upon draping and its variability. Energy calculations are performed as part of the Objective Function algorithm presented in Chapter 4.

For the same mould, different draping sequences will result in different yarn orientations and in-plane shear angles over each BaS. It may be that the locking angle of the reinforcement fabric is reached over some BaS when following some specific draping sequences, and not when following other sequences. If fabric in-plane shear induced in a given draping step exceeds the fabric locking angle, wrinkling occurs. Wrinkling is a highly unfavourable outcome that must be

avoided [162]. Wrinkle occurrence can be alleviated by darting or patterning. The optimal draping sequence is expected to generate the lowest number of wrinkles, requiring minimal darting.

Figure 3.10 shows an example of a mould corner where a specific draping step results in different yarn orientations being imposed on neighbouring BaS. Such orientation incompatibilities can lead to wrinkling when neighbouring regions attempt to accommodate conflicting deformation states. If the attempted warp and/or weft yarn orientations in neighbouring BaS differ beyond a predefined threshold, a darting recommendation is generated for peripheral BaS, as described in Algorithm 3.8 below.

Algorithm 3.8 Wrinkle warning

```
For all draped_neighbours of BaS_b:
   Rotate warp and weft vectors into BaS_b plane
   Save rotated_vectors in neighbours_warp_weft_rotated matrix
   If amount of draped_neighbours = 1
     Do nothing.
   Else, if amount of draped_neighbours > 1:
     For all rotated_vectors in neighbours_warp_weft_rotated:
        Create permutation_list of all rotated_vectors
          # Wrinkle warning #
        For all permutations in permutation_list:
           Calculate angle between vectors warp_m vs. warp_n
           If
           angle between vectors warp_m and warp_n > locking_angle:
             Print "Possible wrinkle for BaS_i"
           Calculate angle between vectors weft_m vs. weft_n
           If
```

```
        angle between vectors weft_m and weft_n > locking_angle:
          Print "Possible wrinkle for BaS_i"

    For all rotated_vectors in neighbours_warp_weft_rotated:
      Calculate weighted_avg_warp
      Calculate weighted_avg_weft
```

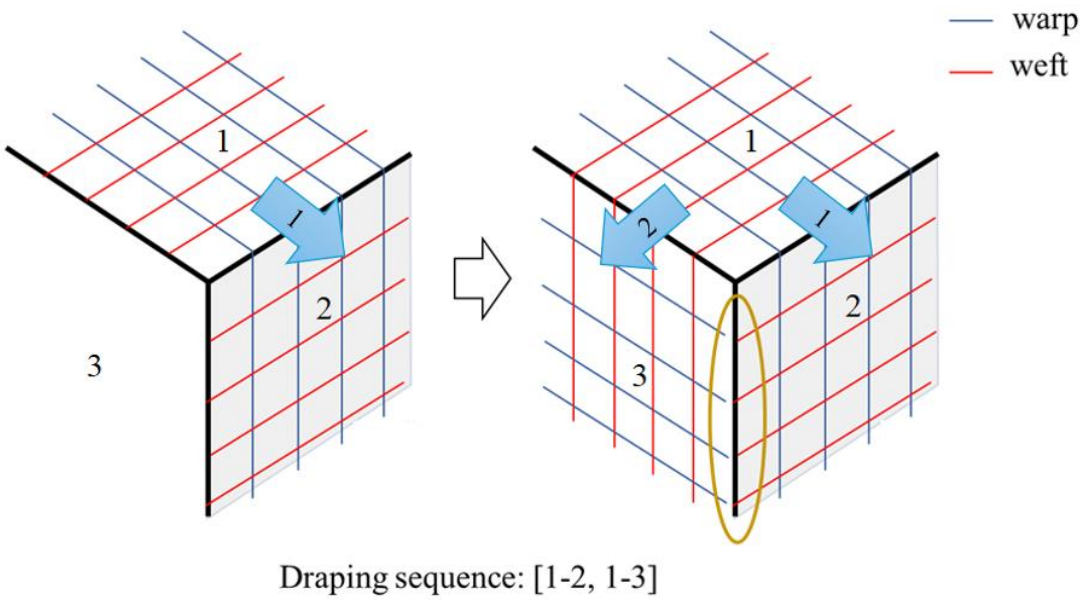


Figure 3.10 Warp and weft yarn misalignment at a corner

Wrinkle avoidance through darting is implemented in two different ways in this thesis, because two draping sequences optimisation approaches are explored. The first approach, BF, evaluates entire draping sequences generated by DFS or BFS algorithms. The second approach, DP, evaluates partial draping sequences as they are generated from a local pool of possibilities for the next few draping steps.

For BF, potential wrinkle events are recorded and flagged at the end, upon sequence evaluation. A draping sequence with no or few potential wrinkle events is preferred to one showing

more potential wrinkle events. For DP, wrinkles are inherently avoided by decision making. Wrinkles generated from reaching the locking angle over one BaS or from yarn orientation mismatch between neighbouring BaS are reported in the results summary. Figure 3.11 shows a square box convex mould [163] where different initial orientations are tested, demonstrating its effect on wrinkle generation.

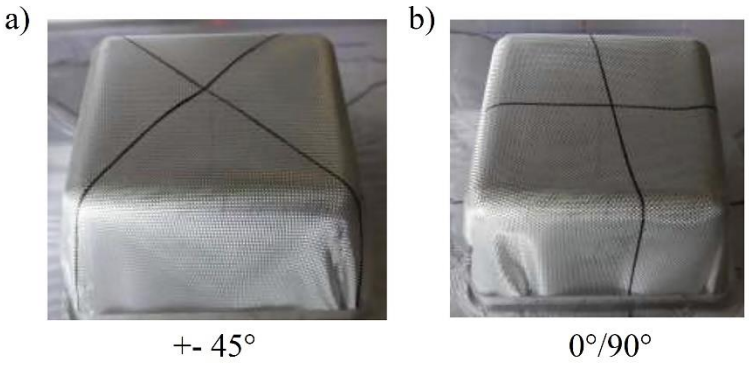


Figure 3.11 Square box draped from initial orientations: a) +/- 45°; b) 0°/90° [163]

Algorithm 3.9 identifies peripheral edges where a darting operation is recommended as a means of mitigating wrinkles. A darting recommendation is generated when in-plane shear exceeds the locking angle and a peripheral vertex is detected for BaS_b, indicating that excess deformation may be relieved through a peripheral cut.

Algorithm 3.9 Darting recommendations

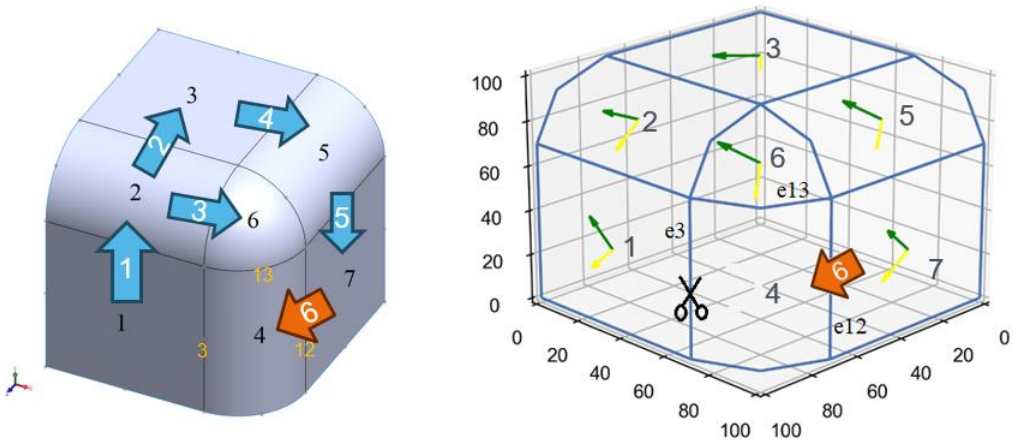
```
If shear_angle(BaS_b) >= locking_angle or
entering_yarn_orientations_conflict == True:
   If BaS_b in peripheral_BaS:
      Print "Recommended darting operation for BaS_b"
      Identify neighbours with orientations conflict
      Identify shared edges
```

```
      From shared edges – [List] the ones with a peripheral vertex
      Print "Recommended darting in edges [list]"
# Internal Functions definition #
Define entering_yarn_orientations_conflict (BaS_a → BaS_b):
   Extract list of draped neighbours of BaS_b
   Extract yarn orientations of every draped neighbour of BaS_b
   Calculate yarn_orientations_difference between all
   draped_neighbours(BaS_b)
   If yarn_orientations_difference ≥ set_limit:
     Print (conflict between neighbour_m and neighbour_n)
     Print (Entering BaS_b through edge_ei and edge_ej)
   Return (True)

Define dart (BaS_b):
   If dart triggered by reaching locking_angle:
   Extract peripheral_edges(BaS_b)
   Recommend darting at edge which is peripheral but not from BaS_a
   If dart triggered by conflict in yarn orientations:
     Extract from entering_yarn_orientations_conflict:
       List of conflictive neighbouring couples
       For every conflictive neighbouring couple:
          Recommend darting at edge which is peripheral
```

Figure 3.12 shows an example of darting being generated as a recommendation during draping. In draping step 6, two issues are flagged. Firstly, incompatible yarn orientations from previously draped neighbouring BaS #7 and BaS #1 are detected when attempting to drape BaS #4. In this case, both common edges e12 and e3 are peripheral and could therefore be darted. As draping proceeds from BaS #7 in the sequence shown, only common edge e3 remains available for darting.

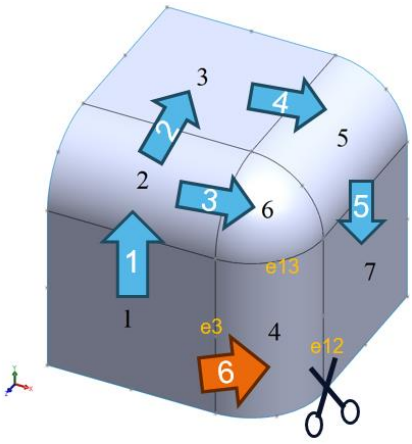
Secondly, incompatible yarn orientations from previously draped neighbouring BaS #7 and #6 are detected when attempting to drape BaS #4. In this case, only common edge e12 is peripheral hence could be darted. As draping proceeds from BaS #7 in this sequence, edge e12 cannot be darted hence no feasible recommended solution is available.




```
step  6  - from  7  to  4
ALERT --> High difference in yarn orientations entering BaS# 4  from BaS# 1  and BaS# 7
Possible wrinkle trying to drape peripheral BaS# 4 through edges  12 3
recommended to dart through edge  3
ALERT --> High difference in yarn orientations entering BaS# 4  from BaS# 7  and BaS# 6
Possible wrinkle trying to drape peripheral BaS# 4 through edges  12 13
```

Figure 3.12 Yarn orientations conflicts at step #6.

Figure 3.13 also shows that wrinkling warnings and darting recommendations are specific to each draping sequence. The comments discussed above apply only to the sequence progressing from BaS #7 to BaS #4. Alternative draping sequences may lead to different yarn orientation compatibility conditions and therefore different recommendations. One such example is the sequence [1-2, 2-3, 2-6, 3-5, 5-7, 1-4], shown in Figure 3.13, where wrinkling is avoided by darting common edge e12 only. In this case, yarn orientations over previously draped neighbouring BaS #6 and #1 remain compatible and the combination is therefore not flagged.




```
step  6  - from  1  to  4
ALERT --> High difference in yarn orientations entering BaS# 4  from BaS# 1  and BaS# 7
Possible wrinkle trying to drape peripheral BaS# 4 through edges  12 3
recommended to dart through edge  12
ALERT --> High difference in yarn orientations entering BaS# 4  from BaS# 7  and BaS# 6
Possible wrinkle trying to drape peripheral BaS# 4 through edges  12 13
recommended to dart through edge  12
```

Figure 3.13 Orientations conflicts and darting recommendations

If darting were enforced by the software as a draping action, yarn orientations associated with the darted edge would no longer be considered when determining yarn orientations over BaS_b and subsequent draping steps. Such an implementation would affect the propagation of yarn orientations throughout the draping sequence. In the present work, however, darting is not executed by the software and remains a recommendation to the user. When a darting condition is detected, a penalization is added to the Objective Function

The modelling framework presented in this chapter enables the calculation of yarn orientations and in-plane shear distributions for any draping sequence. These quantities provide the inputs required for evaluating the optimisation criteria developed in this thesis. The following chapter describes how yarn orientation and in-plane shear data are combined with experimental results from Database B to calculate the C&Q terms used in the Objective Function.

Chapter 4. Cost and Quality Objective Function

The previous chapter presented the modelling framework used to calculate yarn orientations and in-plane shear distributions for any given draping sequence. The present chapter builds upon those results by defining the cost and quality terms used to evaluate and compare alternative draping sequences. The developed Objective Function features terms proportional to the strain energy required for draping patterns over a mould, collectively identified as the cost, and terms proportional to the variability of the strain energy, collectively identified as the quality. Quality is considered inversely proportional to variability. Cost and quality are terms favoured by the industry.

4.1 Introduction to Strain Energy Terms and Objective Function

An Objective Function quantifies the performance of a system, using any quantity or combination of quantities that can be represented by a number. The following sections describe the strain energy sources considered in the Objective Function and the experimental methodology used to characterize them.

The work developed in this thesis is based on two fundamental hypotheses. Firstly, patterns requiring more strain energy for reaching a draped configuration will require more operator time and attention, hence demand higher cost. Secondly, patterns for which this strain energy is more consistent show more consistent constitutive behaviour, leading to results of higher

reproducibility, or quality. The terms proportional and inversely proportional are used in a general sense; mathematical details appear below.

Cost C is a function of strain energy terms calculated upon draping each BaS, based on BaS geometry, draping parameters and average reinforcement stack behaviour as characterized by Kanz [9]. Individual strain energy terms associated with different deformation modes are discussed in this Chapter. Likewise, quality Q is a function of the variability of individual strain energy terms calculated upon draping each BaS, based on BaS geometry, draping parameters and statistical distributions of the same material parameters. C and Q functions enable quantitative comparisons of notional cost and notional quality for different draping sequences.

During HLU, patterns progressively drape flat, single curvature and double curvature BaS parametrized using the conical Frustum proposed by Hoffer [2], Section 1.2.2 . Draping flat or single curvature BaS does not induce additional in-plane shear. On the other hand, patterns draped on double curvature surfaces undergo additional in-plane shear, requiring additional strain energy associated with this specific deformation mode. Strain energy terms used in calculating C and Q for flat, single curvature and double curvature BaS arise from the deformation modes listed in Table 4.1; all terms are expressed in N·m . It must be highlighted that different yarn orientations coming into a BaS from different neighbouring BaS can also lead to additional in-plane shear over said BaS.

Table 4.1 Deformation modes included in strain energy calculations for different BaS types

Flat	Single curvature	Double curvature
In-plane shear (U_S)	U_S	U_S
In-plane shear spring back (U_{SSB})	U_{SSB}	U_{SSB}
	Bending (U_B)	
	Bending spring back (U_{BSB})	

As observed in Table 4.1, bending energy terms are excluded from the analysis of double-curvature surfaces. This exclusion is justified by the relative magnitude of the strain energy terms. While bending is present on these surfaces, its contribution is much smaller than that of in-plane shear. More importantly, shear strain generated on a given BaS is propagated through subsequent draping steps, whereas bending primarily affects individual surfaces. Consequently, strain energy from shear is consistently much greater than that from bending for double-curvature surfaces, making the inclusion of bending energy terms unnecessary. This simplification maintains sufficient modelling accuracy while reducing computational requirements during optimisation.

All reinforcement fabrics and stacks characterized for this work require different amounts of strain energy for bending, bending spring back, in-plane shear, and in-plane shear spring back. Section 4.2 summarizes the materials and characterization methods used [9]. Section 4.3 details the calculation of strain energy terms and their variability from experimental data and geometric parameters of the BaS being draped. Finally, Section 4.4 details the cost and quality Objective Function.

4.2 Materials and Characterization Methods

The industrial partner manufactures parts using certified reinforcement fabrics. Figure 4.1 presents the three fabrics considered in this work. Fabric #1 is a 319 g/m^2 glass plain weave known internally as 7500GF. Experimental results are available for single layer and four-layer stacks, the latter being used in production and the former enabling interpolation. Fabric #2 is a 299 g/m^2 glass 8-harness satin known internally as 7781GF, tested as single layer and four-layer stacks. Fabric #3 is a 196 g/m^2 carbon 4-harness satin known internally as 94932CF, tested as single layer and three-layer stacks. These three different fabrics were tested as a single layer SL or multilayer ML. Kanz [9] performed all experiments towards calculating strain energy terms for bending, bending spring back, in-plane shear and in-plane shear spring back.

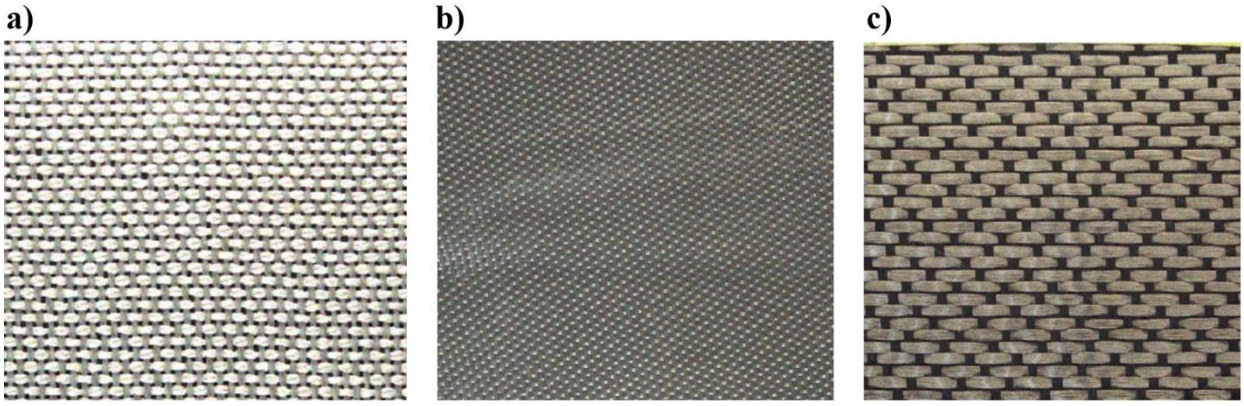


Figure 4.1 a) Fabric #1; b) Fabric #2; c) Fabric #3

Bending tests were performed under standard ASTM-F3260 [164]. Bending stiffness was quantified in compliance with the standard. Experimental details appear in [9]. Standard ASTM-F3260 requires four readings of overhang length l (m) and bending rigidity B_{Peirce} (μN·m^2/m) for

each specimen, as well as averages and standard deviations for each surface orientation (face/back) and each direction (warp/weft).

Convex and concave bending spring back test methods were developed by Kanz; experimental details appear in [9]. Figure 4.2 shows the two test configurations. Convex tests bend the fabric around a convex 90° angle, Figure 4.2 a), while concave tests bend the fabric into a concave 90° angle, Figure 4.2 b). Both tests were performed using five mould radii ranging from 1.59 mm to 12.70 mm, in addition to a sharp edge.

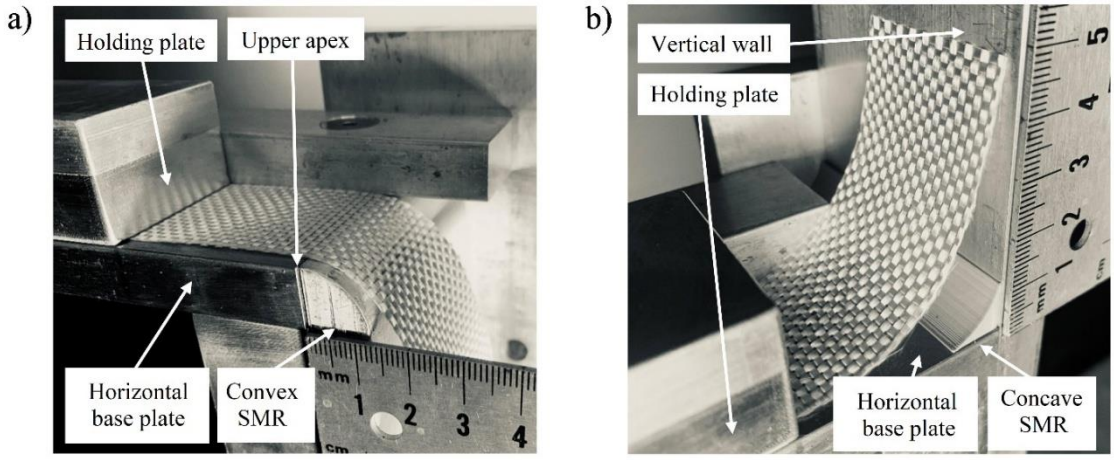


Figure 4.2 Bending testing device: a) Convex configuration; b) Concave configuration [9]

In-plane shear tests were performed using the articulated picture-frame rig shown in Figure 4.3 for all selected fabrics and fabric stacks. Experimental details appear in [9]. The normalized shear load was measured along with the shear angle in the fabrics. It was also used in characterizing the shear angle at which shear force returned to zero on the return phase of the cycle, referred to as the return angle, quantifying in-plane shear spring back.

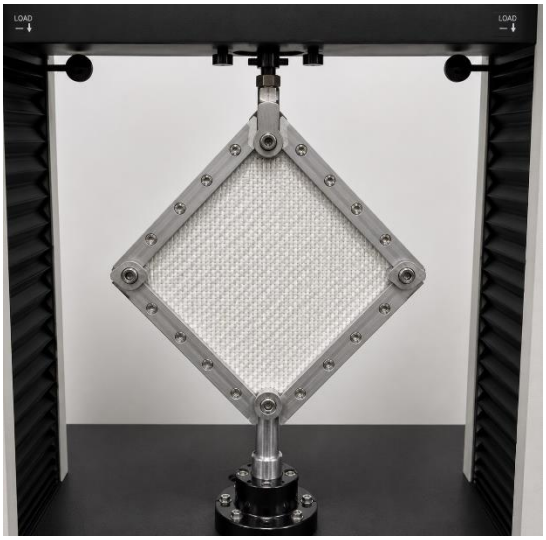


Figure 4.3 Picture-frame testing rig [9]

4.3 Strain Energy Terms and Their Variability

4.3.1 Bending

Bending strain energy U_B is calculated for radii up to 1.0 m. Single curvature surfaces with radii above 1.0 m are considered flat, yet in this case a bending energy term is calculated based on this 1.0 m radius to account for general handling of reinforcement fabrics. As with all other strain energy terms, bending strain energy is proportional to the area of the BaS being draped. In addition to bending, radii below .0125 m led to an additional term of strain energy U_{BSB} associated with bending spring back discussed in Section 4.3.3 .

Calculation of U_B assumes that bending response is linear elastic, single curvature BaS feature constant radii, fabric thickness is small relative to the radius, and bending moment is constant along the length of the fabric which undergoes pure bending [165]:

$$U_B = \int_0^l \frac{M}{2B}^2 dx \tag{4.1}$$

where M, B and l are the bending moment (N·m), bending stiffness (N·m^2) and length (m). The relationship between bending moment and bending radius in pure bending is [166]:

$$M = \frac{B}{R} \tag{4.2}$$

where R represents the fabric radius (m), which is assumed to be the same as the mould radius. This assumption can be contextualised by considering the non-structural nature of parts manufactured by the industrial partner, making the difference between the fabric radius and the mould radius negligible. Bending moment M is calculated from cantilever bending stiffness normalized by unit width as characterized by Kanz [9]; U_B is obtained by substituting Equation (4.2) in the solution of Equation (4.1):

$$U_B = \frac{\left(\frac{B}{R}\right)^2 l}{2B} = \frac{M\alpha}{2} \tag{4.3}$$

where α is the fabric opening angle defined between both ends of the fabric (rad). Equation (4.5) expresses U_B from experimental bending stiffness G_{Peirce} (μN·m^2/m), with Equation (4.4) extracted from standard ASTM-F3260:

$$G_{Peirce} = 9.81 \times 10^{-6} \times m \times \left(\frac{1}{2}\right)^3 \tag{4.4}$$

$$U_B = \frac{G_{Peirce} b_f}{2} \frac{\alpha}{R} \times 10^{-6} \tag{4.5}$$

where b_f is fabric width (m) and m is the mass per unit area (g/m^2).

4.3.2 Bending Variability

Variability of bending energy is characterized using the dispersion of bending stiffness. All terms in Equation (4.4) being constant except for G_{Peirce}, bending energy variability is calculated as:

$$\Delta U_B = \frac{b_f}{2} \frac{\alpha}{R_m} \cdot \Delta G_{Peirce} \tag{4.6}$$

where ΔG_{Peirce} is the dispersion of bending stiffness. Standard deviations of fabric characteristics are correlated to the same independent variables as the linear functions for interpolating averages.

4.3.3 Bending Spring-back

Experimental results show that BaS curvature affects bending behaviour below certain radii. Bending spring back was observed by Kanz for $R < 0.0125$ m [9]. The behaviour is different for concave and convex curvatures. As mentioned in Chapter 2, the user identifies whether a mould is predominantly concave or convex and the software proceeds accordingly.

Fabrics and fabric stacks bent to concave or convex radii below 0.0125 m spring back, requiring additional energy to conform to their intended draped configuration [167]. Equation (4.2) states that a fabric bent to a larger radius requires less strain energy than the same fabric bent to a smaller radius, for the same opening angle. Calculation of U_{BSB} for convex BaS proceeds as follows:

$$U_{BSB-CV} = \frac{G_{Peirce} b_f}{2} \left(\frac{\alpha_m}{R_m} - \frac{\alpha_f}{R_f} \right) \tag{4.7}$$

where α_m is the angle between both ends of the fabric (*rad*) referred to as the fabric opening angle. Once the fabric is draped onto the mould, α_m is the same for the fabric and the mould. R_m is the mould radius (m) and α_f and R_f are respectively the final opening angle and radius after spring back. U_{BSB} is expressed in N·m .

Bending spring back characterisation was conducted for a 90° bending angle and various mould radii. Figure 4.4 a) shows convex spring back testing configuration, where fabrics were securely positioned on the horizontal plane. Spring back β measurements (R_f) were done horizontally relative to the simulated mould radius origin. Circles representing R_m and final fabric

form R_f were tangent at the point where the fabric leaves the horizontal surface at point A. Consequently, radius centres align with point A and the mould radius origin.

A different equation was derived for concave BaS, reflecting the different test method needed for evaluating α_f and R_f, Figure 4.4 b); circles representing the draped fabric stack and the mould are both tangent to both horizontal and vertical walls of the mould corner. The calculation of R_f from β measurement is further explained by Kanz [9]. Equation (4.7) simplifies to:

$$U_{BSB-CC} = \frac{G_{Peirce} b_f \alpha}{2} \left(\frac{1}{R_m} - \frac{1}{R_f} \right) \tag{4.8}$$

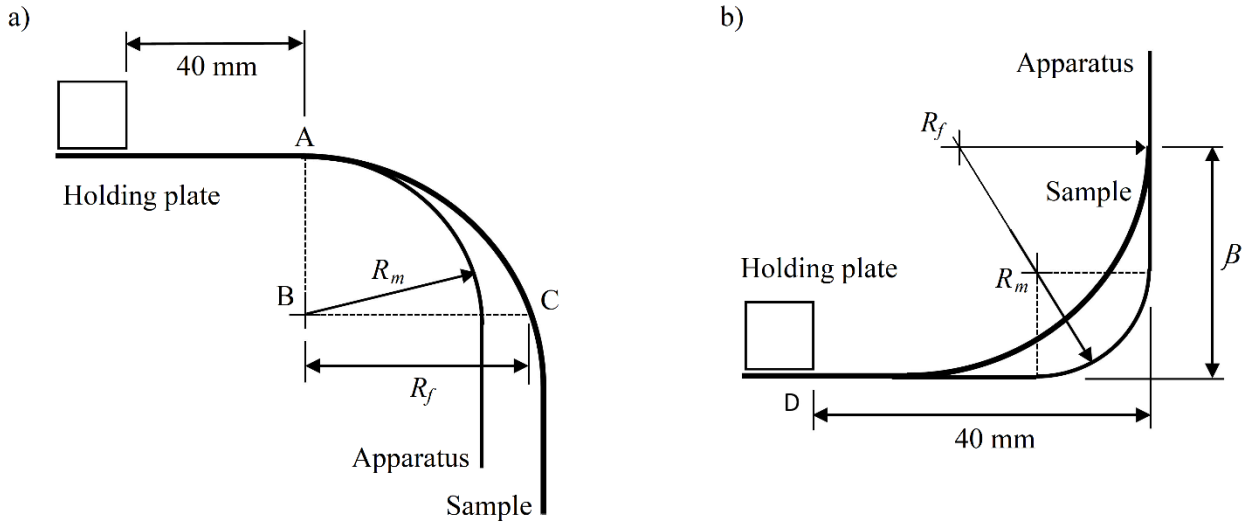


Figure 4.4 Bending spring back variables: a) Convex; b) Concave

4.3.4 Bending Spring-back Variability

Variability for U_{BSB} is expressed as two different equations. For convex radii, the final opening angle of the material differs from the opening angle of the mould. As a result, Equation (4.9) is used for determining the variation in convex spring back energy:

$$\Delta U_{BSB-CV} = \frac{b_f}{2}\left(G_{Peirce} \cdot \frac{\alpha_m}{R} + \frac{G_{Peirce}\alpha_f}{R_f} \cdot \left(\frac{\Delta\alpha_f}{\alpha_f} + \frac{\Delta R_f}{R_f} + \frac{\Delta G_{Peirce}}{G_{Peirce}}\right)\right) \tag{4.9}$$

where ΔR_f represents the final fabric radius and $\Delta\alpha_f$ denotes the final fabric opening angle:

$$\Delta R_f = \frac{\beta}{R}\Delta\beta \tag{4.10}$$

$$\Delta\alpha_f = \frac{\beta}{R_f}\Delta\beta \tag{4.11}$$

where $\Delta\beta$ is the dispersion of raw spring-back data (m/m).

For concave radii, the final opening angle of the fabric is equal to the opening angle of the mould, and the final fabric radius is equal to the raw spring back measurement. Equation (4.12) is used for calculating the variability of bending spring back energy (N·m):

$$\Delta U_{BSB-CC} = \frac{\alpha b_f}{2}\left(\frac{\Delta G_{Peirce}}{R} + \frac{G_{Peirce}}{R_f} \cdot \left(\frac{\Delta G_{Peirce}}{G_{Peirce}} + \frac{\Delta R_f}{R_f}\right)\right) \tag{4.12}$$

For concave radii, spring back measurements correspond to the final fabric radius hence final fabric radius dispersion defines the dispersion of spring back measurements.

4.3.5 In-plane Shear

In-plane shear arises upon draping double-curvature BaS, driven by surface geometry. It also occurs when yarns with different orientations from multiple neighbouring draped surfaces meet on a given BaS. Lastly, in-plane shear can be transferred from one draped BaS to another. In all three cases, the strain energy U_s (N·m) required for elastic in-plane shear is expressed by:

$$U_S = \iiint \frac{\tau^2}{2G} dxdydz \tag{4.13}$$

where τ and G are the shear stress (N/m^2) and shear stiffness (N/m^2). In this work, in-plane shear strain energy is calculated by numerical integration of experimental data. Experiments were performed by Kanz [9] using a picture frame rig, depicted in Figure 4.5 , to evaluate shear stress for Fabric #1, Fabric #2 and Fabric #3 in SL and ML configurations. For this calculation the shear angle (rad) is converted to displacement of the picture frame during in-plane shear tests, defined as x_{frame} (m) [9].

The rig consisted of a square articulated frame with a hinge-to-hinge length of 175 mm. Fabric specimens were mounted as cruciform samples and sheared up to 53°, corresponding to an extension of 85 mm. The effective sheared area of the test was 19 600 mm², providing the experimental data used for determining locking angles, shear energy, and variability as shown in Figure 4.6 graph.

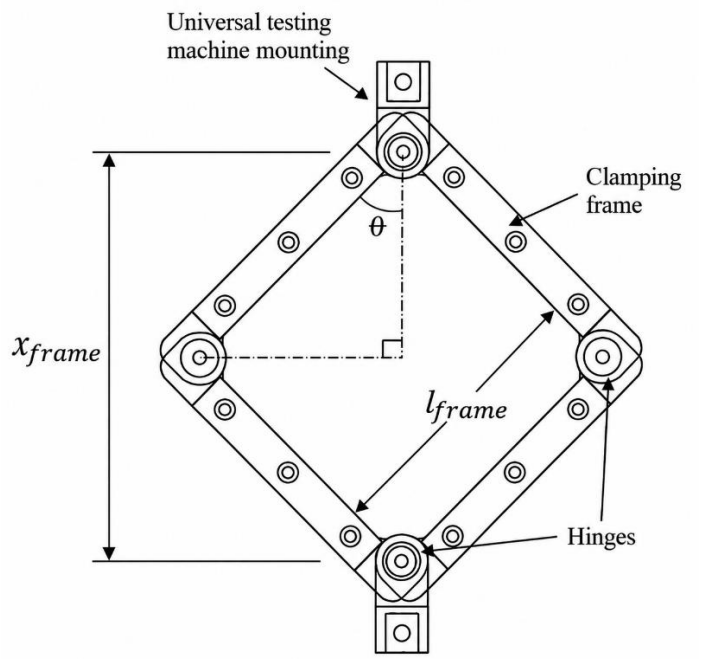


Figure 4.5 Picture frame rig for evaluating in-plane shear stress [9]

Picture frame extension x_{frame} (m) is calculated as:

$$x_{frame} = 2l_{frame}\cos\left(\frac{90-\theta}{2}\right) - \sqrt{2}l_{frame} \tag{4.14}$$

where l_{frame} (m) is the side length from the centres of the hinges and θ is the shear angle.

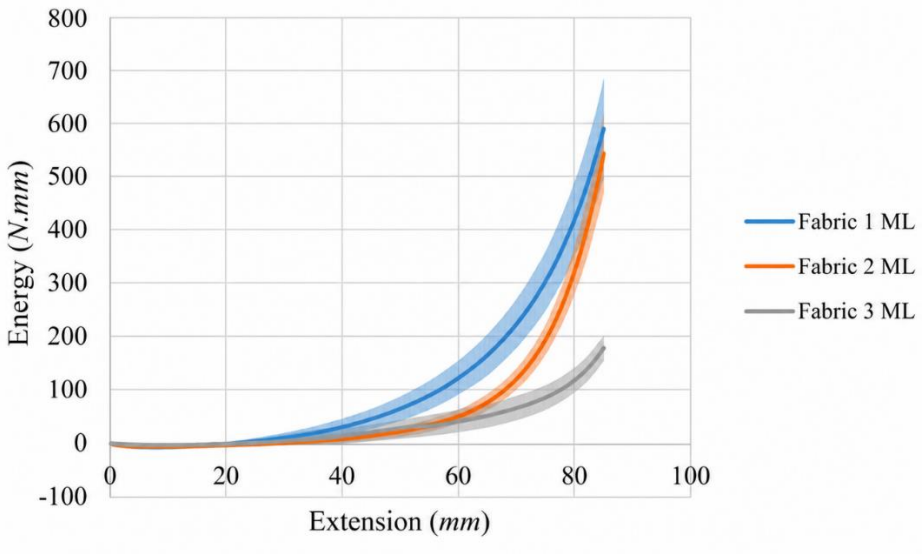


Figure 4.6 Average and standard deviation, picture frame test results - multilayer fabrics [9]

The strain energy required for shearing a fabric in its plane to a specific angle is then calculated using a table look-up function in Database B [9]. Linear interpolation is used for multilayer fabrics and a scaling factor adjusts for surface area. Equation (4.15) applies for Fabrics #1 and #2 where 1 and 4 layers were used in experiments by Kanz [1], and Equation (4.16) applies for Fabric #3 where 1 and 3 layers were used:

$$U_S(n,\theta) = \frac{S_{BaS}}{S_{PF}}\left(\left(\frac{n-1}{3}\right)\cdot\left(U_S(4,\theta) - U_S(1,\theta)\right) + U_S(1,\theta)\right) \tag{4.15}$$

$$U_S(n,\theta) = \frac{S_{BaS}}{S_{PF}}\left(\left(\frac{n-1}{2}\right)\cdot\left(U_S(3,\theta) - U_S(1,\theta)\right) + U_S(1,\theta)\right) \tag{4.16}$$

where $U_S(n, \theta)$ is the strain energy (N·m) required for shearing n layers in-plane to shear angle θ (rad), S_{BaS} is the area (m^2) of the BaS and S_{PF} is the surface area (m) of the fabric sample sheared

in the articulated picture frame, equal to 0.0196 m^2 at test start. The shear angle θ reached upon draping a double curvature BaS is calculated by interpolating from Database A [8].

4.3.6 In-plane Shear Variability

Figure 4.7 shows the experimental standard deviations obtained from the in-plane shear tests. Calculation of variability for in-plane shear energy is performed through a lookup function applied to these experimental data, quantifying the standard deviation associated with in-plane shear deformation [9]. Equations (4.17) and (4.18) are then used for interpolating standard deviation for number of layers and shear angle. Equation (4.17) is used for Fabrics #1 and #2; Equation (4.18) is used for Fabric #3.

$$\Delta U_S(n,\theta) = \frac{S_{BaS}}{S_{PF}}\left(\left(\frac{n-1}{3}\right)\cdot\left(\Delta U_S(4,\theta) - \Delta U_S(1,\theta)\right) + \Delta U_S(1,\theta)\right) \quad (4.17)$$

$$\Delta U_S(n,\theta) = \frac{S_{BaS}}{S_{PF}}\left(\left(\frac{n-1}{2}\right)\cdot\left(\Delta U_S(3,\theta) - \Delta U_S(1,\theta)\right) + \Delta U_S(1,\theta)\right) \quad (4.18)$$

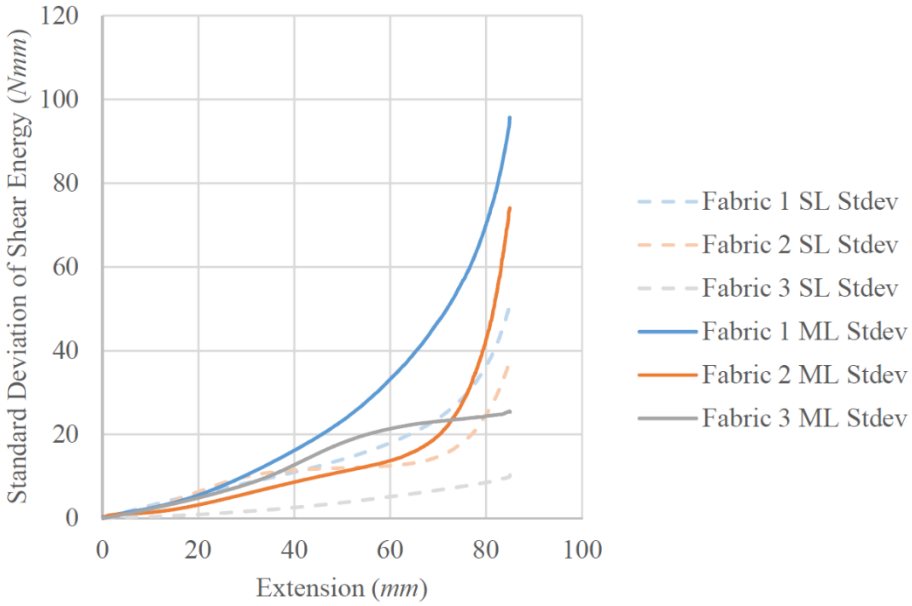


Figure 4.7 Standard deviations of in-plane shear strain energy [9]

where $\Delta U_S(n, \theta)$ is the variability of in-plane shear energy (N·m/m), n is the number of layers, S_{BaS} is BaS area (m^2) and S_{PF} = 0.0196 m^2.

4.3.7 In-plane Shear Spring-back

In-plane shear spring back was quantified through the return angle of SL and ML reinforcement fabric stacks tested in shear, defined as the angle where shear force reduces to zero in the return phase [9]. Return angles were characterized for Fabrics 1, 2 and 3. ML samples for Fabrics 1 and 2 featured four layers while Fabric 3 featured three layers. Five SL and five ML samples were tested, each undergoing five cycles from 0° to 53° shear angles.

Figures 4.8 and 4.9 show the shear energy and spring-back energy calculations obtained from picture-frame testing. Numerical integration of the raw picture-frame test results was used to

calculate shear energy [9]. In-plane shear spring-back energy corresponds to the area beneath the curves during the return phase.

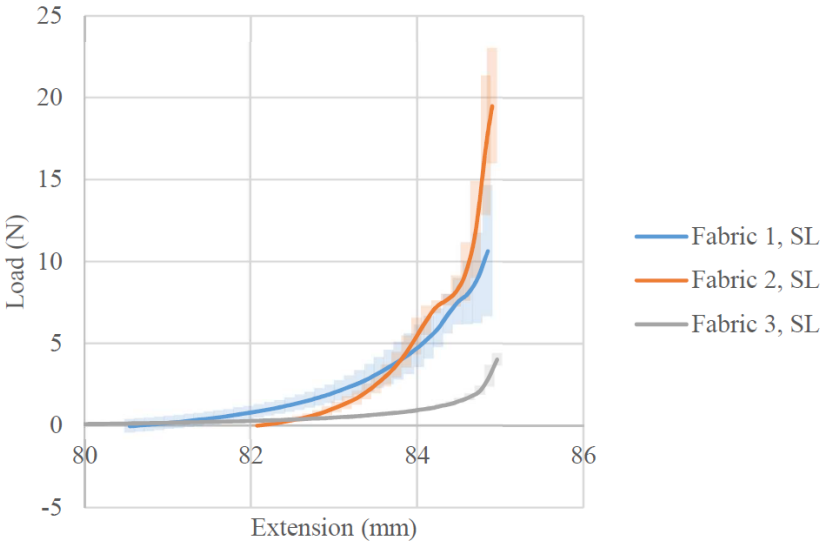


Figure 4.8 Average and standard deviation of return phase of single layer fabrics [9]

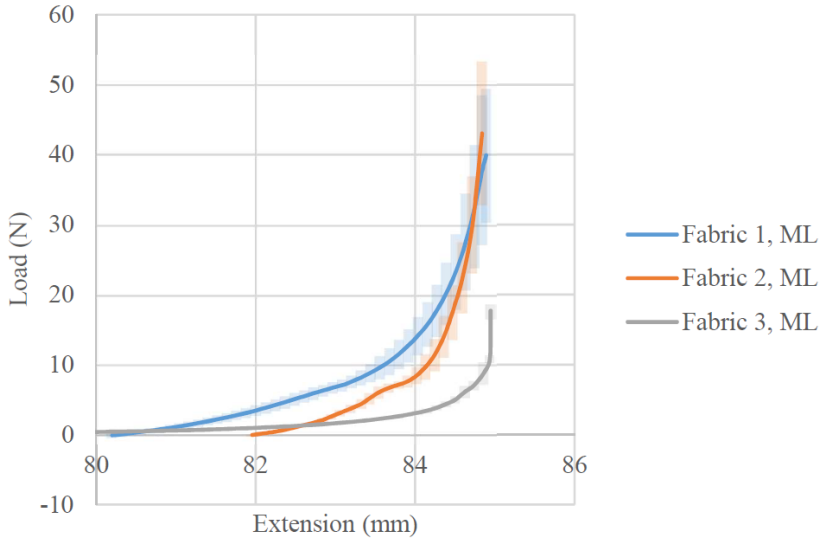


Figure 4.9 Average and standard deviation of return phase of multilayer fabrics [9]

In-plane shear spring back energy (N·m) was calculated for a 53° shear angle corresponding to 85 mm extension on the picture-frame rig. In-plane shear spring back energy ratios were calculated for each fabric in single and multiple layer configurations [1]; average ratio for all fabrics is 5.68×10^{-2} [9]. If the dispersion of the return phase energy ΔU_{return} (N·m) is considered negligible [9], the standard deviation of the spring back ratios ΔSBR_{shear} (N·m/N·m) can be estimated using:

$$\Delta U_{SSB} = \pm \frac{U_{return}}{U_{53°}} \left(\frac{\Delta U_{53°}}{U_{53°}} \right) \tag{4.19}$$

Assuming constant in-plane shear spring back ratio and shear angle for all fabrics tested, U_{SSB} (N·m) can be derived by multiplying the energy needed to shear the fabric to a given shear angle by the average in-plane shear spring back ratio across all fabrics, 5.68×10^{-2} [9]:

$$U_{SSB} = 5.68 \times 10^{-2} \cdot U_s(n, \theta) \tag{4.20}$$

and in-plane shear spring back energy variability is calculated as:

$$\Delta U_{SSB} = 5.68 \times 10^{-2} \cdot \Delta U_s(n, \theta) \tag{4.21}$$

where $\Delta U_s(n, \theta)$ is the variability of in-plane shear energy (N·m/ N·m) required for shearing n layers of fabric to angle θ.

4.3.8 Database B Calculator – Strain Energy and its Variability

All strain energy sources and their variability terms described above were obtained through standardized laboratory tests. The resulting experimental data were subjected to rigorous statistical analysis to characterize both the mean response and the associated variability. Results are contained in Database B along with the linear equations describing the behaviour. Database B is queried by the software tool developed in this thesis, accelerating the evaluation process and producing results in a fraction of the time required for direct calculations.

4.4 Objective Function

The HLU process is defined by sequential decisions labelled draping steps. Each possible complete set of draping steps is labelled a draping sequence. Each draping step consists of draping a single BaS, requiring a specific amount of strain energy imparted to the fabric reinforcement stack as the sum of individual strain energy terms associated to different deformation modes. Furthermore, a term of statistical variability exists for each term of strain energy associated with each deformation mode and each BaS, as a result of variability in the constitutive behaviour of fabric reinforcement stacks.

The total amount of strain energy imparted to the reinforcement when draping all BaS featured in a mould along a specific draping sequence is the sum of energy terms required at each draping step:

$$U_{total} = \sum_{i=1}^{tds} U_{D_i} \tag{4.22}$$

where *tds* is the total number of draping steps required for draping all BaS over a mould, equal to the number of BaS minus one. The energy U_{Di} required for draping step i is calculated as:

$$U_{D_i} = U_B + U_{BSB} + U_s + U_{SSB} \tag{4.23}$$

From Section 4.3, U_{Di} is a function of the following parameters when draping a single curvature BaS:

$$U_{D_i}(n, \alpha_m, R_m, \alpha_f, R_f, m_c) \tag{4.24}$$

where m_c is the predominant curvature type of the mould, concave or convex; all other terms were introduced previously. When draping a double curvature BaS, U_{Di} is a function of: (n, θ, S_{BaS}).

Shear angle θ_{Di} (rad) is calculated for each draping step based on BaS geometry and taken as input parameter for calculating terms of energy required for draping the BaS. The input shear angle can be affected by in-plane shear originating from previously draped BaS:

$$\theta_{D_i}\left(\theta_{D_i-1}\left(\theta_{D_{i-2}}\left(\ldots\left(\theta_{D_i-i}\right)\right)\right)\right) \tag{4.25}$$

Therefore, energy terms associated to a draping step and BaS are also function of previous draping steps:

$$U_{D_i}\left(U_{D_i-1}\left(U_{D_{i-2}}\left(\ldots\left(U_{D_i-i}\right)\right)\right)\right) \tag{4.26}$$

It is important to mention that the first draping step in a sequence, $D_{i\text{-}i} = D_0$, consists in draping a BaS with initial conditions set by the user: shear angle θ_0 and yarn orientation γ_0.

The total variation of energy generated for draping all BaS of a mould following a specific draping sequence consisting of *m* draping steps is the sum of variation of energy for every draping step:

$$\Delta U_{total} = \sum_{i=1}^{m} \Delta U_{D_i} \tag{4.27}$$

$$\Delta U_{D_i} = \Delta U_{bending} + \Delta U_{BSB} + \Delta U_s + \Delta U_{SSB} \quad (4.28)$$

From equations (4.27) and (4.28), energy variability for draping step *i* is also affected by previous draping steps:

$$\Delta U_{D_i}\left(\Delta U_{D_i-1}\left(\Delta U_{D_{i-2}}\left(\ldots\left(\Delta U_{D_i-i}\right)\right)\right)\right) \quad (4.29)$$

Based on validation tests performed at the industrial partner R&D premises on industrial moulds, an adjustable coefficient k_q was proposed for tuning the contribution of variability calculated from Database B to the overall Objective Function: $\Delta U_{D_i} \rightarrow k_q \Delta U_{D_i}$. The thesis optimization problem is stated as:

$$Optimize: \qquad U_{total} + \Delta U_{total} \quad (4.30)$$

Subject to: ***R***

where ***R*** is a set of restrictions arising from the nature of the HLU process and the mechanics of fabric deformation as explained in this thesis.

Optimisation algorithms presented in this thesis evaluate the HLU draping process through an interaction between Python routines and supporting Excel® Databases A and B. The

optimisation algorithms require inputs such as Frustum parameters, initial orientations, fabric type, number of plies, and in-plane shear angle from Database B.

4.5 Conclusion

This chapter detailed a systematic methodology for quantifying sources of strain energy and their variability, with a focus on the industrial application of the software tool for efficiently generating optimal draping sequences. Future enhancements, including fine-tuning weighting constants and expanding the database with additional test data, are recommended to further augment the tool's capabilities. The foundational role of the Objective Function is emphasized, as it quantifies system performance through strain energy and variability, which are essential metrics for comparative evaluations.

Chapter 5. Brute Force Optimisation

As described in Section 2.3.4, increasingly accessible computational resources have transformed BF optimisation into a practical choice for tackling optimisation problems with small populations. In this thesis, BF optimisation is used for evaluating the Objective Function for all BaS in all draping sequences generated by combinatorial algorithms DFS and BFS.

These combinatorial algorithms are constrained with the aim that simulated draping steps proceed as real ones would when performed by a technician during hand lay-up draping. As described in Section 3.3, draping starts from one BaS and progresses from draped BaS to contiguous non-draped BaS; progressing in more than one single direction and from any previously draped BaS is possible, this latter possibility being labelled colloquially as branching.

Figure 5.1 shows the workflow implemented for BF draping optimisation. The algorithm integrates different software functionalities including geometric processing through mould parameterization, draping sequence generation using DFS or BFS, the objective Function evaluation of individual sequences, and selection of the best draping sequence as the final objective. For clarity, the algorithm number corresponding to each software functionality is indicated beside the relevant blocks in the figure. For example, A 5.1 refers to Algorithm 5.1.

Algorithm 5.1 Brute force optimisation

```
Define Brute_Force_Optimisation (CAD_file):

# Mould_parameterization: #

   Run Algorithm 3.1 to generate reference frame vectors for
   all BaS.

# Sequence_Generation: #

   Run Algorithm 3.6 (DFS) or Algorithm 3.7 (BFS) to generate
   all valid permutations.

Save lists of draping sequences (saved_lists)

# Full Draping seqs. Objective Function (C&Q) Evaluation #

For every sequence in saved_lists:

   For every BaS in the saved_lists:

      Retrieve geometric data from Database_A.

      If BaS is the starting_surface:

        Run Algorithm 3.2 for initial Warp and Weft
        orientations.

      Else:

        Run Algorithm 3.3 for vector rotation and

        Run Algorithm 3.5 to account for draped neighbors.

      If BaS has double curvature (DC):

        Run Algorithm 3.4 to calculate in-plane shear angle
        addition.

# Perform Quality Checks: #

      Run Algorithm 3.8 for wrinkle warnings.

      Run Algorithm 3.9 for darting recommendations.

Retrieve material variability data from Database_B.

Calculate Cost and Quality (C&Q) based on Objective Function.
Register Wrinkle warnings and Darting recommendations.

Save results (C&Q_value, sequence #).

Best_Result = Select the sequences with the optimal value for
the Objective Function and less amount of Wrinkle warnings and
Darting recommendations. (Manually)
```

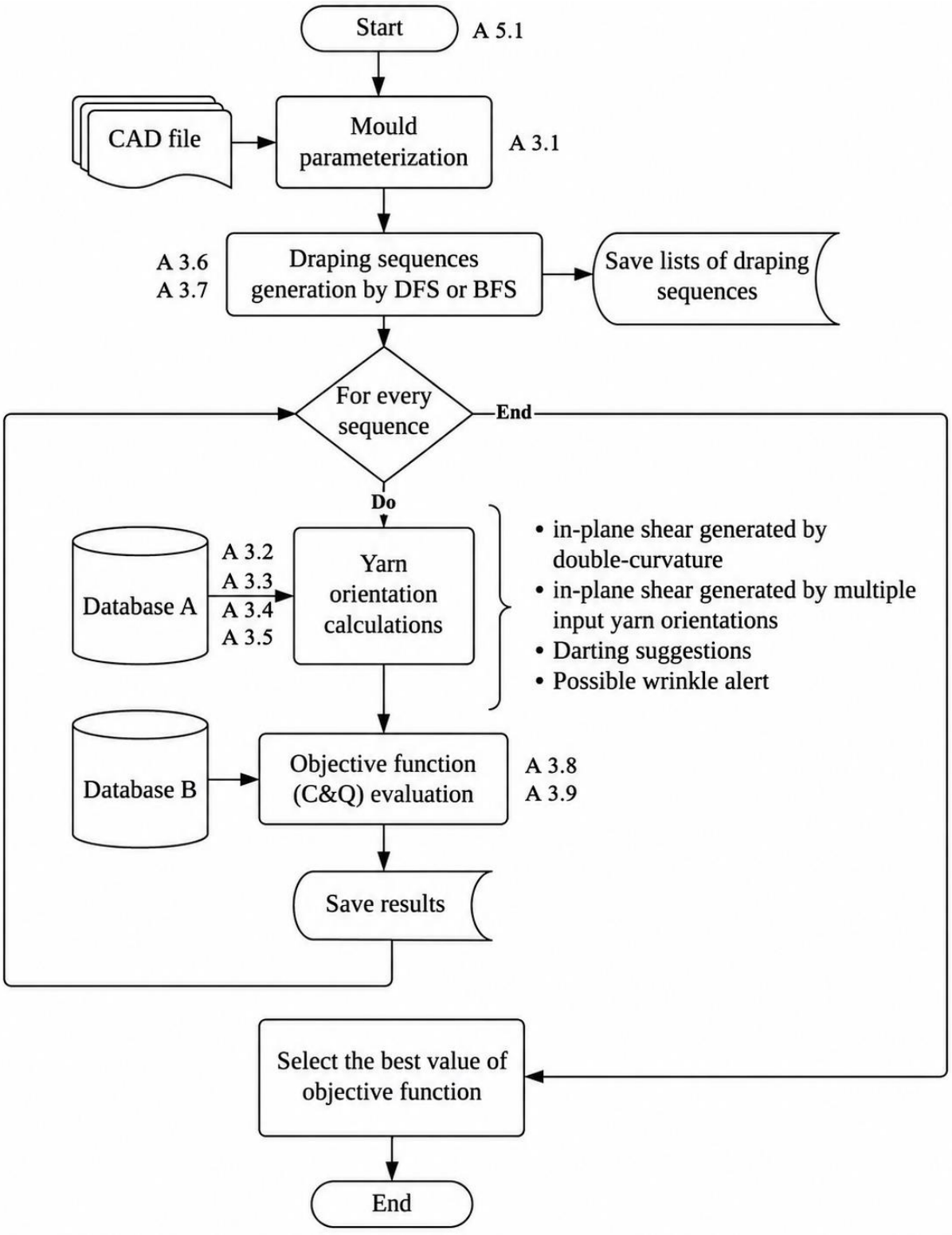


Figure 5.1 Brute Force algorithm flowchart

5.1 Demonstrators for Brute Force Evaluation

Figures 5.2 and 5.3 show the demonstrator parts used for validating BF: the industrial case 27_Hutchinson and the academic case 30_Corner. Selection was limited to parts with 10 BaS or fewer due to computational constraints. The chosen demonstrators, featuring 6 and 7 BaS respectively, confirmed that BF consistently identifies the global optimum through exhaustive evaluation of the complete feasible solution space.

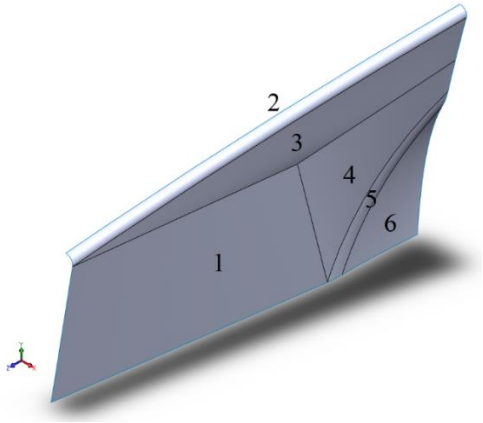


Figure 5.2 Demonstrator 27_Hutchinson with labelled BaS

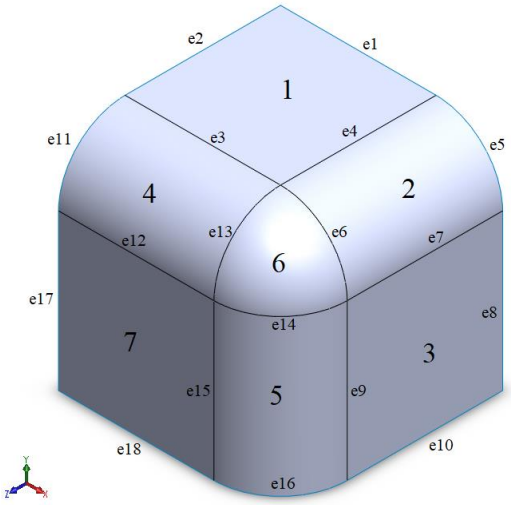


Figure 5.3 Demonstrator 30_Corner with labelled BaS and edges

5.2 Generation of Draping Sequences

Draping sequences were generated using adaptations of the two combinatorial approaches DFS and BFS detailed in Sections 3.3.1 and 3.3.2. Brute force optimization was applied to the two demonstrators. Performance metrics of the combinatorial algorithms for sequence generation are presented in Table 5.1, while evaluation results are provided in Table 5.3. The data reveal clear differences in the numbers of sequences generated by each approach: BFS produces substantially more sequences than DFS, i.e. DFS constitutes a subset of BFS. For the present demonstrators, differences in computational costs are negligible; however, for geometries of greater complexity with larger numbers of BaS, computational costs increase significantly and scale exponentially with problem size.

The numbers of sequences obtained for DFS and BFS combinatorial algorithms are different because BFS enables branching events: at each *i*th draping step, BFS enumerates all possible continuations from every currently draped BaS to all their non-draped neighbours. In contrast, DFS extends the sequence only from the most recently draped BaS; if no non-draped neighbours remain, it backtracks and explores new branches from previously draped BaS, until all BaS have been visited.

For each combinatorial approach, the numbers of sequences generated for demonstrator 30_Corner are much larger than for demonstrator 27_Hutchinson. These differences result in part from the larger number of BaS making the academic demonstrator, but also from more extensive BaS connectivity in demonstrator 30_Corner.

Table 5.1 DFS and BFS performance comparison

Demonstrator	Combinatorial algorithm	Sequences generated	Time (s)
27_Hutchinson	DFS	47	0.023
	BFS	164	0.07
30_Corner	DFS	156	0.047
	BFS	7536	1.17

5.3 Brute Force Performance Evaluation

After the generation of draping sequences, the software calculates values of the Objective Function for all sequences, optimises it by selecting an extrema, and delivers the corresponding preferred draping sequence amongst all those generated initially using either DFS or BFS. The software also provides wrinkling warnings and suggestions for darting, as described below.

For a given demonstrator, the optimal sequence(s) may be sought among those generated by BFS, which systematically explores the full set of feasible draping sequences. DFS, in contrast, explores individual sequences along single paths without guaranteeing full coverage of the solution space. While both approaches can generate identical non-branching sequences, BFS is exhaustive, whereas DFS is path-dependent and does not ensure that all possible sequences are evaluated.

Shear angles used for yarn orientation calculations may be selected by the user as either the maximum or average value from all edges for every BaS, Section 3.2.2. The user may wish to identify the fabric reinforcement leading to the best result hence investigate and optimise for fabrics #1, #2 and #3 characterized in Database B. The target number of layers is typically set in

advance by the client; however, different initial fabric orientations may be investigated, typically using the extreme cases 0°/90° and +/-45°.

Table 5.2 summarizes the factors and levels considered for evaluating BF optimisation performance with each demonstrator, assuming a fixed number of fabric layers in each case.

Table 5.2 Scenarios for evaluating BF optimisation performance

Demonstrator	Combinatorial algorithm	Shear angle used	Fabric	Initial orientation
27_Hutchinson	DFS	Max shear	Fabric #1	0°/90°
30_Corner	BFS	Avg shear	Fabric #3 Fabric #3	+/-45°

Table 5.3 presents optimisation results and execution times for the BF algorithm, based on representative combinations of parameters selected from the factors and levels listed in Table 5.2. Values labelled best *f(C&Q)* correspond to the optimised output of the Objective Function, with computing time expressed in minutes.

Table 5.3 reports the best Objective Function values obtained after evaluating draping sequences generated by both DFS and BFS combinatorial algorithms, considering average and maximum shear angle options. The following section discusses these outcomes and presents the sequences leading to the best Objective Function value.

Table 5.3 Objective Function optimisation results for representative cases.

		Fabric #1, 1 layer 0°/90° initial orientation				Fabric #3, 3 layers +/- 45° initial orientation			
		DFS		BFS		DFS		BFS	
		Avg shear	Max shear	Avg shear	Max shear	Avg shear	Max shear	Avg shear	Max shear
27_Hutchinson	best *f(C&Q)*	19.07	26.29	19.07	26.29	16.97	27.96	16.97	27.96
	time (min)	0.45	0.44	1.32	1.32	0.47	0.45	1.32	1.46
30_Corner	best *f(C&Q)*	2.82	2.92	2.80	2.92	8.3	9.17	6.55	7.42
	time (min)	1.02	1.15	48.8	51.6	1.03	1.02	52.2	50.3

5.4 Analysis of Results

DFS and BFS are combinatorial algorithms that explore the solution space by generating extensive sets of possible draping sequences, analogous to the traveling salesman problem. For Demonstrator 30_Corner, each algorithm produced a distinct list of sequences. The Objective Function was then evaluated for every sequence in both lists.

Upon comparison of results it was observed that both algorithms converged on the same set of sequences as optimal, despite differences in the algorithms creating the explored solution space. For the first test configuration with 1 layer of Fabric #1, the best sequence generated by both DFS and BFS and evaluated using the maximum shear angle condition was [6-5, 5-4, 4-1, 1-3, 3-2], Figure 5.4. When evaluating using the average shear angle, two sequences – generated by both DFS and BFS - rated best: [6-5, 5-4, 4-3, 3-2, 3-1] and [6-5, 5-4, 4-3, 3-2, 4-1], Figure 5.5 a) and Figure 5.5 b) respectively.

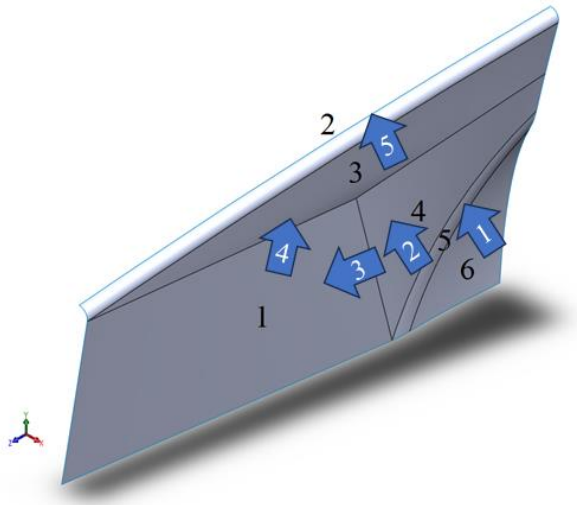


Figure 5.4 BF optimised sequence, maximum shear, demonstrator 27_Hutchinson

For the case of 3 layers of Fabric #3, the best sequences were the same as identified above for Fabric #1, in all cases, but larger values of total strain energy were required for draping the mould, which was expected from the higher bending stiffness of the carbon fibre fabric.

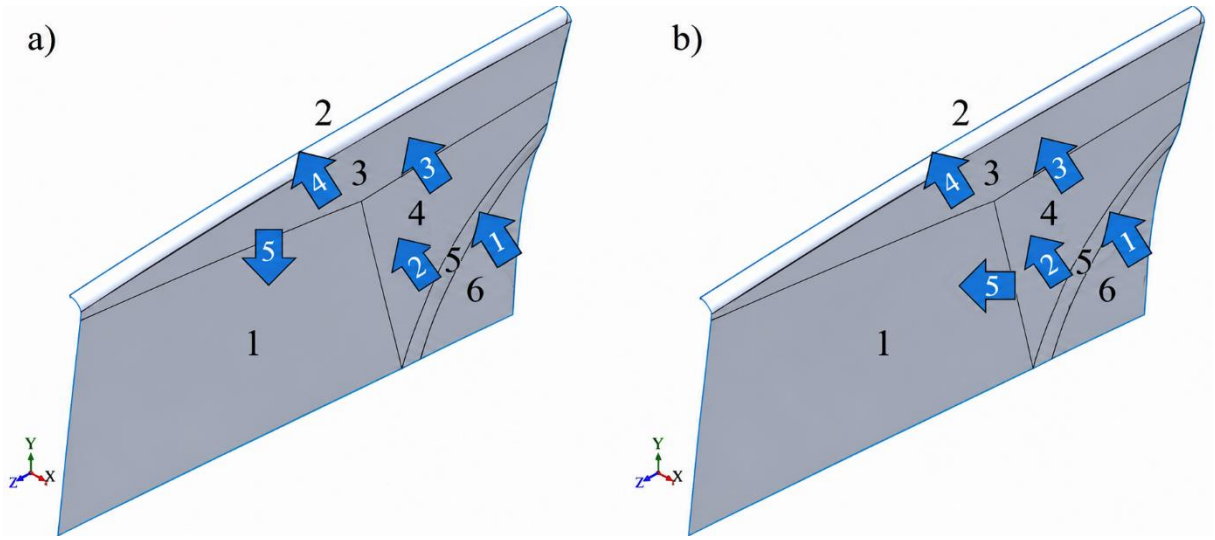


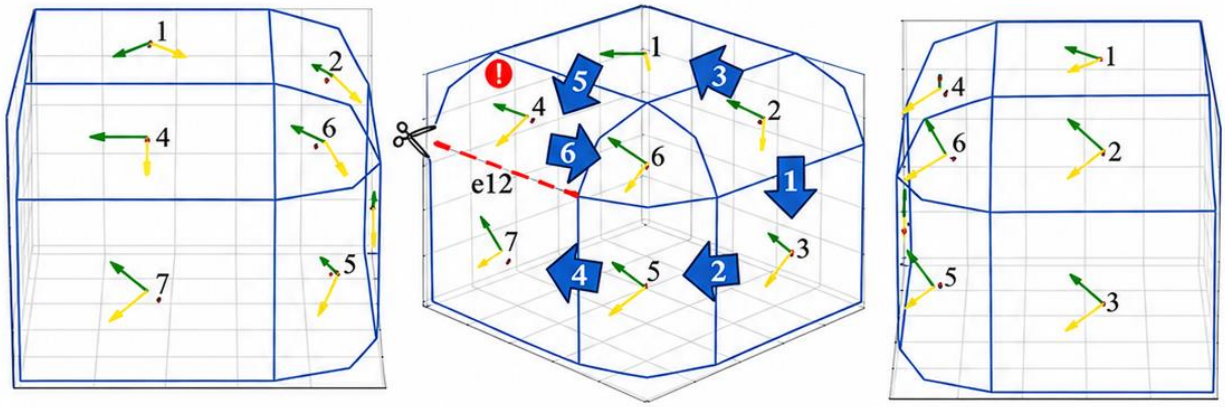
Figure 5.5 BF optimised sequences, average shear, demonstrator 27_Hutchinson: a) Sequence 1; b) Sequence 2

For demonstrator 30_Corner, BFS produced a search space more than 48 times larger than that of DFS, thereby enabling the exploration of a wider range of options. When evaluating the Objective Function for a single layer of Fabric #1 using maximum shear angle for yarn orientation, DFS yielded one sequence attaining the best objective value of 2.92, whereas BFS identified 66 distinct sequences that achieved the same value. When the average shear angle was considered instead, BFS produced 240 sequences reaching the best objective value of 2.80, which is slightly better than the best DFS result of 2.82 . The marked difference lies in the redundancy of optimal solutions: BFS generated hundreds of sequences with identical best values, while DFS produced only one. This multiplicity of equivalent solutions arises from the geometrical symmetry of the 30_Corner demonstrator.

For the case of three layers of Fabric #3 evaluated with average shear angle, BFS identified 96 optimal draping sequences with an Objective Function value of 7.42, whereas DFS failed to generate any sequence approaching this performance. A similar outcome was obtained when the maximum shear angle was used.

Demonstrator 30_Corner provided a good test case for the wrinkle warning algorithm because of its corner geometry. When in-plane shear of a BaS exceeded a set threshold, the algorithm warned the user of potential wrinkling. If the wrinkle occurred in a peripheral BaS, darting was recommended, as illustrated in the draping sequences of Figure 5.6 analyzed using the maximum shear angle.

a)



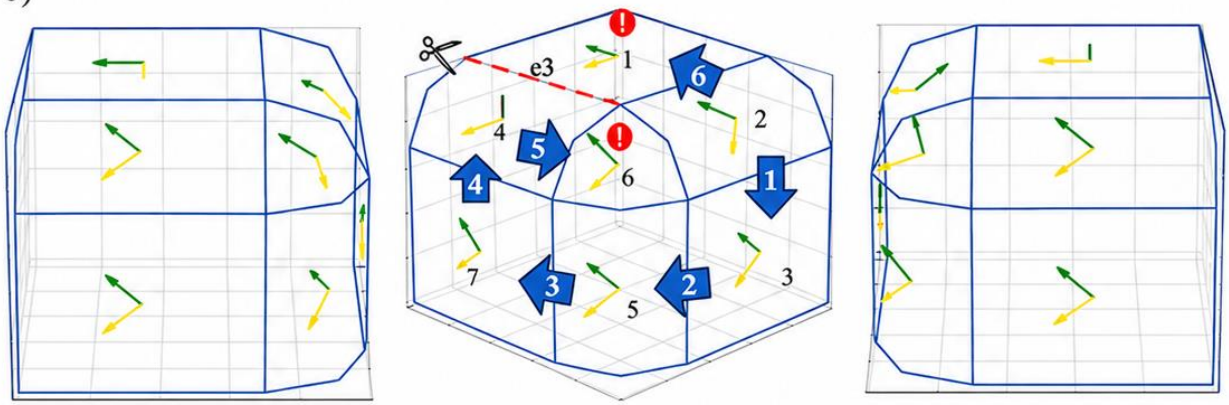

```
# Printed Screen from Command-Prompt window #

ALERT --> High difference in yarn orientations entering
Bas#4 from Bas#1 and Bas#7

Wrinkle trying to drape peripheral Bas#4 through edges 12
and 3

Recommended to dart through edge 12
```

b)




```
# Printed Screen from Command-Prompt window #

ALERT --> High difference in yarn orientations entering
Bas#6 from Bas#2 and Bas# 4

ALERT --> High difference in yarn orientations entering
Bas#6 from Bas#4 and Bas#5

ALERT --> High difference in yarn orientations entering
Bas#1 from Bas#4 and Bas#2

Wrinkle trying to drape peripheral Bas# 1 through edges 4
and 3

Recommended to dart through edge 3
```

Figure 5.6 Draping sequences for 30_Corner demonstrator with yarn orientations displayed

Sequence [2-3, 3-5, 2-1, 5-7, 1-4, 4-6] presented in Figure 5.6 a), BaS #6 does not trigger a high-shear warning, even though a wrinkle was previously detected in BaS #4. This outcome is explained by the mechanism of orientation averaging: in draping step #5, BaS #4 is draped from BaS #1 but its yarn orientations conflict with those of BaS #7. The algorithm resolves this conflict by enforcing average warp and weft orientations, which are then propagated forward. As a result, the orientations reaching BaS #6 are not in conflict with draped neighbouring BaS #5 and BaS #2, preventing the algorithm from issuing a wrinkle warning. The results in Figure 5.6 a) are reflected in a real case scenario, Figure 3.13 a), where the corner does not exhibit a wrinkle, whereas the single-curvature BaS does.

Sequence [2-3, 3-5, 5-7, 7-4, 4-6, 2-1] illustrated in Figure 5.5 b) exhibits yarn orientation conflicts in BaS #6 and BaS #1. However, only BaS #1 can be darted since BaS #6 is not located at the periphery of the mould. Consequently, the software recommends darting edge 3 of BaS #1. As expected, this sequence requires more energy to be draped and is ranked second, after sequence shown in Figure 5.5 a) .

5.5 Conclusions

Brute force optimisation proved effective for moulds featuring a limited number of BaS, consistently identifying the global optimum through exhaustive evaluation of all feasible draping sequences. However, the exponential growth of the solution space with increasing numbers of BaS makes BF optimisation impractical for larger moulds.BFS combinatorial algorithm demonstrated

its superiority in generating better populated search spaces, which enabled the identification of better draping sequences. Even though both DFS and BFS may identify the same best draping sequence for simpler demonstrators, this is less likely to happen with more complex moulds. Therefore, BFS is recommended over DFS as the preferred BF algorithm.

The wrinkle warning functionality and darting recommendation tool showed their efficacy in tests. Both functionalities are of informative nature in the BF algorithm, without any action taken by the software that affects neither sequences nor calculated values of the Objective Function.

Chapter 6. Dynamic Programing

6.1 Introduction

Dynamic programming solves complex problems by decomposing them into overlapping subproblems that are solved once and stored for reuse, as discussed in Section 2.3.5. It is well suited for optimization over large solution spaces, particularly when the exact recursive formulation becomes intractable due to factorial growth of the decision space. To address this, the work presented here adopts a limited-horizon approach, approximating the value function using 1-next state and 2-next states lookahead with memoization to ensure computational feasibility.

This approach fundamentally integrates sequence generation into the solution process, evaluating each step as the algorithm progresses. Consequently, DP achieves efficiency through an enumerative search that optimizes global outcomes by solving interdependent local subproblems. This provides an effective framework for optimizing draping sequences for complex geometries, featuring more than 10 BaS.

As introduced in Section 2.3.5, the DP algorithms proposed in this chapter require a predefined search space: a set of draping steps derived from the initial or previously draped BaS. Algorithms 3.6 DFS and 3.7 BFS form the core of this search space generator. Notably, Algorithm 3.7 enforces branching at every step, identifying possible draping operations from all previously draped BaS. This contrasts with Algorithm 3.6, which prioritizes forward expansion, continuing only from the last draped BaS unless a dead-end triggers the stuck functionality to allow branching.

Sections 6.2 through 6.6 detail the functionalities incorporated into the proposed DP algorithms. Section 6.7 presents the demonstrators used for evaluating performance, while Section 6.8 outlines the testing plan. Finally, Sections 6.9 and 6.10 present the results for the 1-next state and 2-next states DP algorithms, respectively.

6.2 Next-State Generation and Algorithmic Variants

To evaluate the impact of search depth on optimization performance, two variants of the sequence generation DP algorithm were developed. These variants differ in how they explore the available search space and utilize the underlying generation algorithms.

The first variant, labelled 1-next state, evaluates all immediate neighbors that can be draped from the existing set of draped BaS as defined by Equation (2.6). This version utilizes Algorithm 3.7 for BFS to identify and evaluate every possible transition within the immediate neighborhood.

The second variant, labelled 2-next states, extends this search horizon to incorporate a second level of look-ahead. As presented in Equation (2.7), the algorithm first identifies all 1-next state options using Algorithm 3.7. For each of these candidates, it then explores the subsequent search space using Algorithm 3.6 for DFS. This hybrid approach allows the algorithm to evaluate the next states based on the specific yarn orientations resulting from the previous state configuration. The resulting decision-making breadth for both the 1-next state and 2-next states versions is illustrated in Figure 6.1.

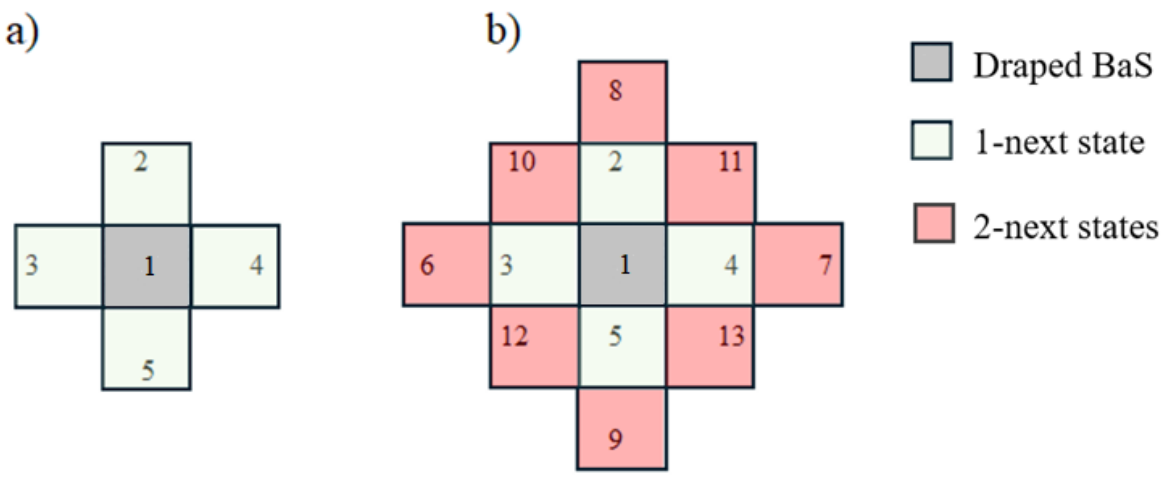


Figure 6.1 Decision space: a) 1-next state; b) 2-next states

Figure 6.1 shows one example of a neighbouring structure, though these structures vary from case to case. In certain scenarios such as the one represented in Figure 6.2, 1-next state BaS such as BaS #2 and BaS #3, BaS #2 and BaS #4, or BaS #4 and double curvature (DC) BaS #5 share a common edge. Consequently, the draping of these surfaces directly influences neighbouring surfaces during the evaluation of 2-next states.

This interaction necessitated a new approach for evaluating 2-next states within the DP algorithm, specifically the implementation of a new data structure to assess multiple draping scenarios before selecting the optimum. To facilitate this, 1-next state draping steps are temporarily treated as draped. This enables the algorithm to explore additional possibilities starting from these temporary positions, as shown in Figure 6.2. These temporary scenarios are defined as Hypothetical States, and they are described in detail in Section 6.3.

6.3 Hypothetical States For 2-Next States Evaluation

The 2-next states DP algorithm relies on parallel Hypothetical States to temporarily store data associated with every 1-next state BaS candidate. After all 1-next state options are generated using BFS, the software evaluates the corresponding 2-next states using DFS. This evaluation is performed without branching, considering only the immediate neighbours of each 1-next state BaS candidate, as shown on the right side of Figure 6.2. Consequently, 2-next states that could be reached through additional branching via BFS, shown in the bottom left of Figure 6.2, are excluded from this variant as they constitute sub-cases of the initial 1-next state.

After the list of 2-next states draping steps is created, the algorithm evaluates each search space to identify local optima. To prevent confusion between parallel Hypothetical States, the algorithm performs full evaluations within loops defined by 1-next state options. The main Orientations Matrix memory is cleared after finishing the evaluation of each parallel Hypothetical State. Orientations Matrices were introduced previously in Section 3.1 .

Following the evaluation of the Objective Function for all 2-next states draping steps, the software selects the optimal pair of steps. It then extracts warp and weft orientations from the corresponding Hypothetical State and stores them in the appropriate rows of the Orientations Matrix.

Figure 6.2 illustrates the BFS draping steps excluded from the 2-next states search space generation. Notably, BaS #5 is a DC type; this geometry generates in-plane shear that is transmitted to its immediate neighbours during the 2-next states evaluation.

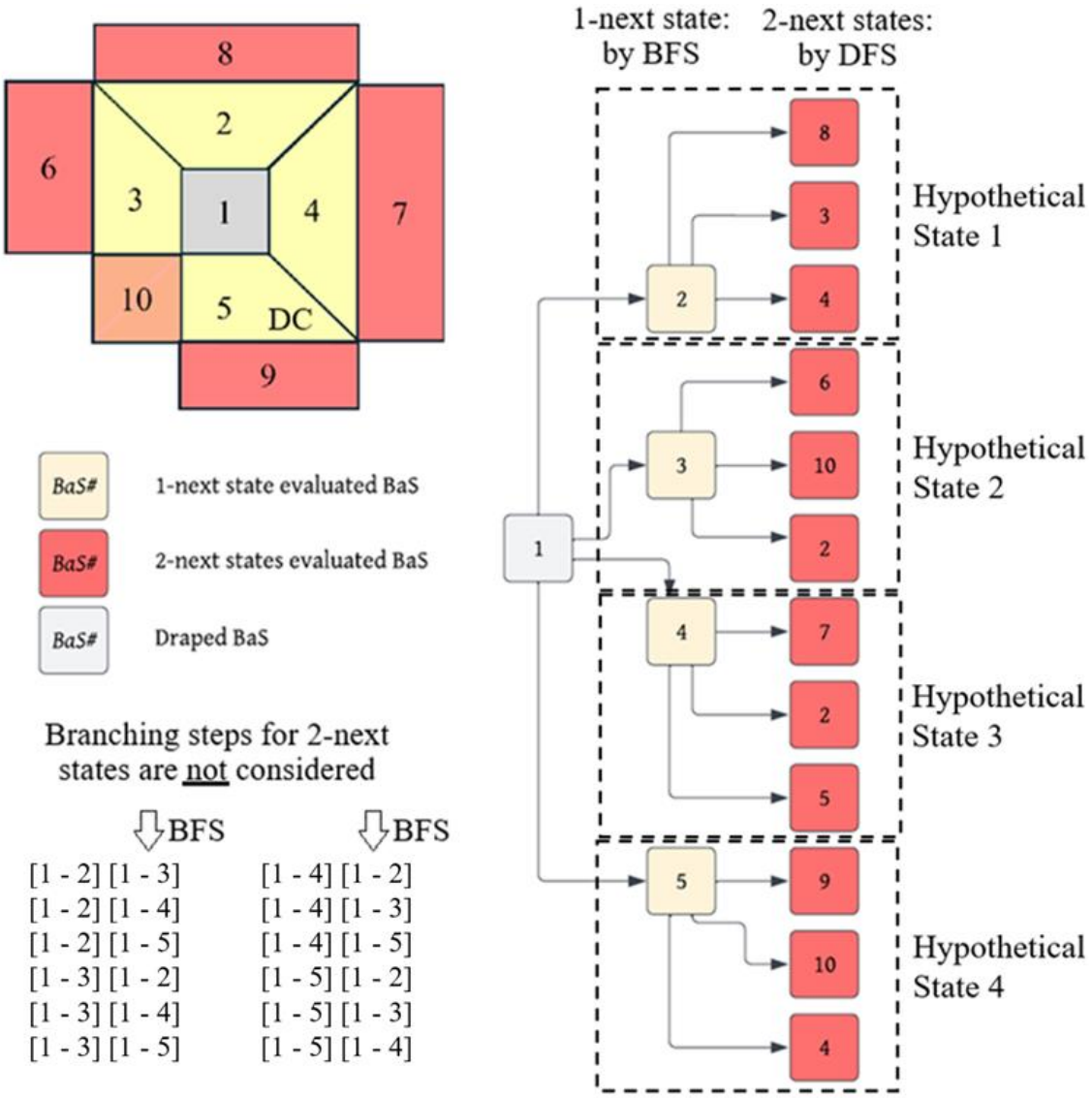


Figure 6.2 Draping steps for 2-next states DP algorithm

The implementation of Hypothetical States and the temporary memory required for their management significantly increase memory requirements. Preliminary tests showed that incorporating memoization into the 2-next states algorithm consistently exceeded the available 32 GB RAM, preventing execution of the demonstrators considered in this work. Consequently, memoization was not implemented for the 2-next states approach.

6.4 Memoization for Dynamic Programming

Figure 6.3 shows the structure of the Memoization Matrix used in this work. As discussed in Section 2.3.5, memoization reduces redundant computations, lowering execution time from exponential to polynomial complexity at the expense of increased memory usage [158], [168], [169]. Each stored draping step [*BaS_a – BaS_b*] includes the shear angles of neighbouring BaS at the time of evaluation. These values are retrieved whenever the same draping state is encountered.

Figure 6.3 compares the same 2-next states pair of draping steps, [1-4, 4-7], demonstrating how they result in distinct memory datasets based on the draping status of their neighbours. This distinction is critical because the Objective Function evaluation requires the warp and weft orientations from both *BaS_a* and *BaS_b*; furthermore, the calculation of *BaS_b* orientations is directly affected by the state of its neighbours.

Memoization was implemented exclusively in the 1-next state variant of the Dynamic Programming algorithm. The 2-next states version generates a reduced Hypothetical State matrix on-the-fly for each evaluated pair, but these matrices are erased after every iteration to manage total memory usage.

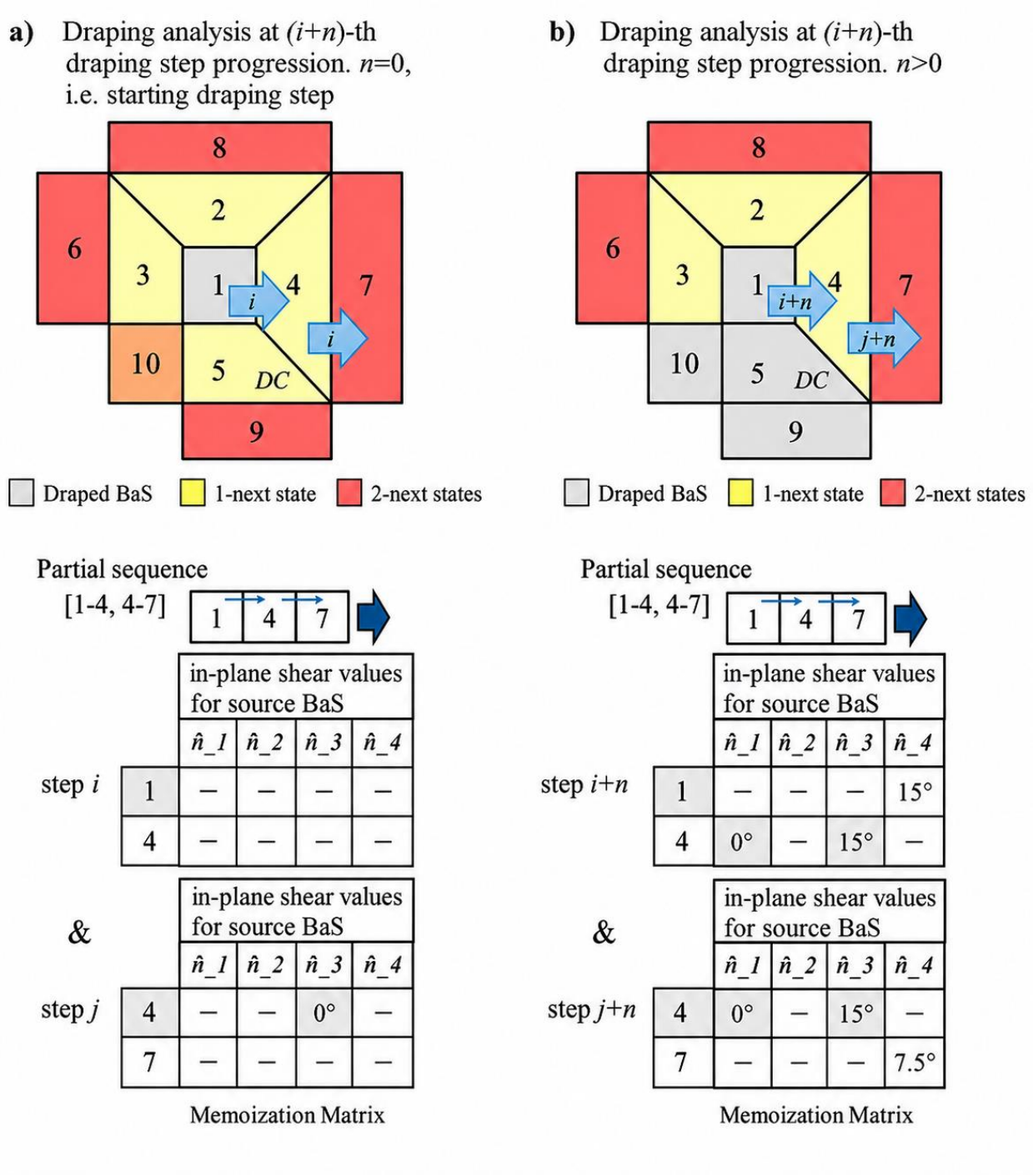


Figure 6.3 Memoization examples

6.5 Wrinkle Warnings and Darting Recommendations

The wrinkle warning and darting recommendation algorithms in DP are adapted from those used in the BF approach. The primary challenge lies in their integration into the 2-next states DP algorithm, where darting a BaS without propagating yarn orientations across the darted edge modifies the Hypothetical States in a non-local manner. This directly impacts the evaluation of the Objective Function.

Algorithm 3.8 for wrinkle warnings and Algorithm 3.9 for darting recommendations form the basis of this integration. Assessing the effect of darting through peripheral edges requires experimental validation to incorporate this behaviour into Database B. Such validation would enable darting to be used as an active operation within the draping process rather than a post-analysis recommendation.

Consequently, darting recommendation is currently implemented only within the 1-next state DP formulation, where state transition remains defined explicitly after each draping decision. In the 2-next states DP algorithm, darting is excluded because the resulting state evolution cannot be formulated within the current transition model, due to the absence of yarn propagation across darted edges. Therefore, darting is treated as a post-processing recommendation rather than a decision variable within the DP framework.

6.6 Enclosure Avoidance Constraint

Upon testing the 1-next-state DP algorithm it was observed that its greedy, local-optimisation nature led to decisions that do not mimic decision-making of a highly experienced technician. Specifically, the algorithm visited sequences that produced local enclosure situations, Figure 6.4 a). In practical terms, the algorithm could drape fabric around an undraped BaS whilst leaving it surrounded by previously draped regions, which is not a realistic way to drape a mould. A functionality was implemented to forbid such local enclosure scenarios, ensuring that no draping step leads to one enclosed undraped BaS. This constraint ensures that no single draping step leads to a fully enclosed, undraped BaS, though it does not yet apply to groups of BaS unless they are enclosed by peripheral elements.

It was also observed that the algorithm prioritized the draping of peripheral BaS before addressing internal BaS, Figure 6.4 b). This latter case, labelled peripheral enclosure, is more likely to occur when working with mould models rather than part models due to flat peripheral surfaces typically added at mould periphery to facilitate vacuum bagging. By constraining both situations, the algorithm achieves more realistic and efficient representation of the draping process. The process for achieving these constraints is detailed below.

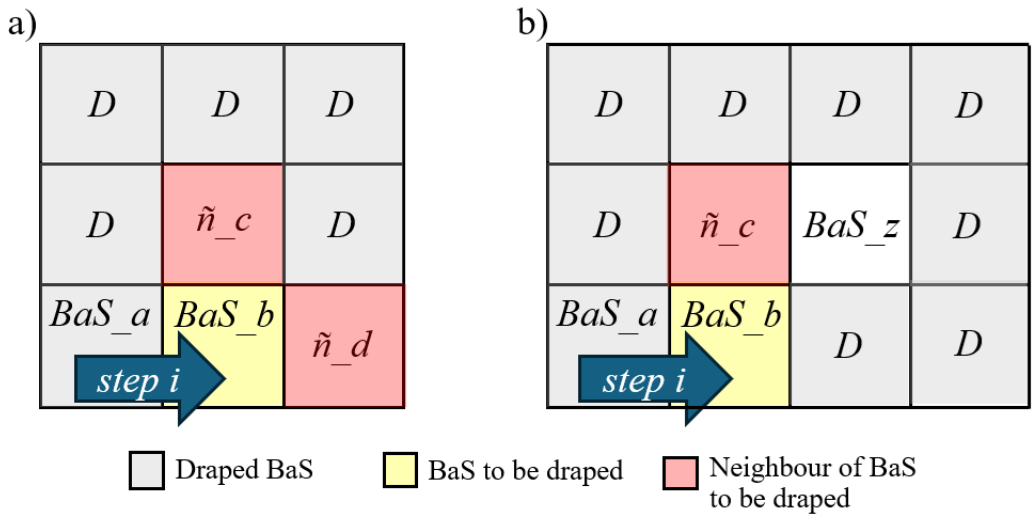


Figure 6.4 a) Enclosure of one non-draped BaS; b) Enclosure by draping peripheral BaS

While the No_Enclosure Algorithm 6.1 is categorised as a functionality, its importance to the feasibility of the search space warrants the presentation of its logic below.

Algorithm 6.1 No_Enclosure functionality

```
Define No_Enclosure (active_seq, candidates, peripheral_list):
  # Initialization: #
  Identify all current non-draped Base Surfaces (BaS).
  Create a filtered_list for feasible next steps.
  For each candidate_BaS in candidates:
    # 1. Local Enclosure Check: #
    # Evaluate if selecting the candidate_BaS isolates any
    # neighboring BaS.
    If a neighbor would become completely surrounded by draped
    regions:
      Mark candidate_BaS as "Rejected" (Enclosure risk).
```

```
    # 2. Peripheral Enclosure Check: #
    # Evaluate if candidate_BaS is a peripheral surface.
    If candidate_BaS is last remaining peripheral BaS while
    internal BaS are undraped:
      Mark candidate_BaS as "Rejected".

    # 3. Filtering rejected BaS: #
    If candidate_BaS is not rejected:
      Add candidate_BaS to the filtered_list.

# 4. Safety Override – avoid premature end: #
If filtered_list is empty:
  return original candidates
Else:
  return filtered_list
```

Algorithm 6.2 addresses both enclosure cases above by filtering the list of possible next draping steps. The function is called every time a draping step is to be optimised, and it receives the following inputs: current partial draping sequence, list of possible draping steps, list of peripheral BaS, and the sequential number of the draping step being optimised. Then, Algorithm 6.2 performs the following operations:

- Local enclosure: for each candidate BaS the function inspects its neighbours. Any neighbour that is still undraped is checked through its own neighbours. If this second-level neighbourhood is completely draped, the neighbour would become fully enclosed if the candidate was selected. Since such an enclosed BaS would be

unreachable later by a continuous draping front, the candidate step is removed. The function iteratively rejects all steps that would cause enclosure or surrounding. If every candidate is rejected, leaving no feasible step, the function returns the original list of possible draping steps to avoid a situation where the algorithm ends without draping all BaS.

- Peripheral enclosure: the algorithm identifies which peripheral BaS remain undraped. If the candidate step would drape the last remaining peripheral BaS while non peripheral BaS are still undraped, the step is removed from the list of possible draping steps. This prevents forming a fully enclosed undraped region.

Algorithm 6.2 is overridden whenever its constraints eliminate all possible next draping steps, allowing the algorithm to proceed.

The individual functionalities discussed in the previous sections, from memoization to enclosure avoidance, are consolidated into a single optimization framework. This integration ensures that search space is effectively pruned to exclude physically impractical sequences while remaining computationally manageable. The resulting DP model and algorithm, incorporating these combined constraints, are presented in the next two sections.

6.7 Dynamic Programming Model

To represent the sequence optimization process, the draping problem is formulated as a discrete-state DP model. The model is defined by a state space representing the configuration of draped BaS and a transition logic that governs the progression of the draping front. Due to exponential growth of the decision space, the model employs a limited-horizon policy to evaluate the Objective Function over 1-next state and 2-next states lookahead trajectories.

Previously presented Equation (3.3) is shown again below:

$$\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E})$$

where $\boldsymbol{G}$ is an undirected graph, $\boldsymbol{V}$ is the set of BaS and $\boldsymbol{E}$ is the set of graph-edges representing neighbouring relationships between BaS.

A draping sequence is defined as an ordered list of nodes, as per Equation (3.4), shared again below:

$$\boldsymbol{S} = [v_1, v_2, \ldots, v_k], \quad v_i \in \boldsymbol{V}$$

where each element represents the order in which BaS are draped.

DP generated draping sequence $\boldsymbol{S}$ must satisfy the uniqueness constraint, where each BaS is draped at most once, preventing revisiting previously draped BaS as per Equation (3.5) displayed again here:

$$v_i \neq v_j \quad \forall\, i \neq j$$

DP feasible decision space is generated by DFS and BFS Algorithms 3.6 and 3.7 respectively. Let 1-next state decision space be defined as

$$D_1(S) = Next_{forward}(\boldsymbol{S}) \cup Next_{branch}(\boldsymbol{S}) \tag{6.1}$$

and 2-next states decision space

$$D_2(S) = \begin{cases} Next_{forward}(\boldsymbol{S}), & if\ Next_{forward}(\boldsymbol{S}) = \emptyset \\ Next_{branch}(\boldsymbol{S}), & otherwise \end{cases} \tag{6.2}$$

Let DP Objective Function be

$$C(v\,|S) = f(C, Q) \tag{6.3}$$

where cost depends on previously draped sequence *S* and the cost of new draping step *v.* Let the DP Objective Function be

$$f(S) = min\{f(C,Q)\}\, from\ state\ S\ to\ completion \qquad (6.4)$$

DP 1-next state practical approximation is defined by Equation (6.7) . Let DP functional equation be

$$f(S) = \min_{v \in D(s)} \{C(v|S) + f(S')\} \qquad (6.5)$$

$$S' = S \cup \{v\} \qquad (6.6)$$

therefore, the 1-next state DP approximation can be expressed as:

$$f(S) \approx \min_{v \in D(s)} \{C(v|S)\} \qquad (6.7)$$

To incorporate limited future information, i.e. 2-next states formulation, let

$$f(S) \approx \min_{v \in D(S)} \left\{ C(v|S) + \min_{w \in D(S')} C(w|S') \right\} \tag{6.7}$$

where

$$S' = S \cup \{v\},\ S'' = S' \cup \{w\} \tag{6.8}$$

DP global objective can be then represented as

$$S^* = \arg\min_{S \in \hat{\boldsymbol{S}}} \sum_{i=1}^{|V|} C(v_i|S_{i-1}) \tag{6.9}$$

where $\hat{\boldsymbol{S}}$ is the set of all feasible sequences generated under proposed constrains, and S_{i-1} is the partial sequence before decision v_i .

6.8 Dynamic Programming Algorithm

Figure 6.5 shows the workflow implemented for DP optimisation through Algorithm 6.2. The workflow integrates the main software functionalities including geometric processing through mould parameterization, initialization, decision-space generation, two-level decision evaluation, wrinkle warnings and recommendations, as well as memoization and parallel Hypothetical States.

Algorithm 6.2 Dynamic programming optimisation

```
Define DP_Optimisation Input(Geometric_Data_File):
  # Mould_parameterization: #
  Run Algorithm 3.1 to generate reference frame vectors for
  all BaS.

  # Initialization - Optional #
  Run 80_20_Function to generate [List of starting BaS]

  # Iterative Draping Progression and Memoization: #
  For every starting BaS:
    While not all BaS are draped in the current sequence:
      # Find Next Steps: #
      Run Algorithm 3.7 BFS to list potential 1-next steps
      If 2-next states:
        Run Algorithm 3.6 DFS to list potential 2-next steps
        for every 1-next state

      For all possible draping steps identified:
        # Memoization Check: #
        If state [sequence + BaS] has been evaluated
        previously:
          Retrieve partial results from memory.
        Else:
          # Yarn Orientation, wrinkle warnings and darting: #
          Retrieve geometric data from Database_A.
          Run Algorithms 3.2, 3.3, 3.4, and 3.5 to calculate
          orientations and shear.
          Run Algorithm 3.8 for wrinkle warnings and
          Run Algorithm 3.9 for darting recommendations.
```

```
        # Cost Evaluation: #
        Retrieve mechanical performance data (Database_B)
        Calculate partial Cost and Quality (C&Q)
        Save partial results + wrinkle + darting to memory
        (Memoization).

    # Local Optima Selection: #
    Select local optima: Identify the best draping step(s)
    based on lowest C&Q and least wrinkle warnings + darting
    recommendations for current search space.
    Update active draping sequence with selected local
    optima.

Save final sequence for current starting BaS.

Best_Sequence = Select sequence with optimal global value
from generated list of optimal draping sequences.
```

For increased clarity in understanding Figure 6.5 flowchart, the algorithm number corresponding to each software functionality is indicated beside the relevant blocks in the figure. For example, the notation A 3.1 denotes Algorithm 3.1.

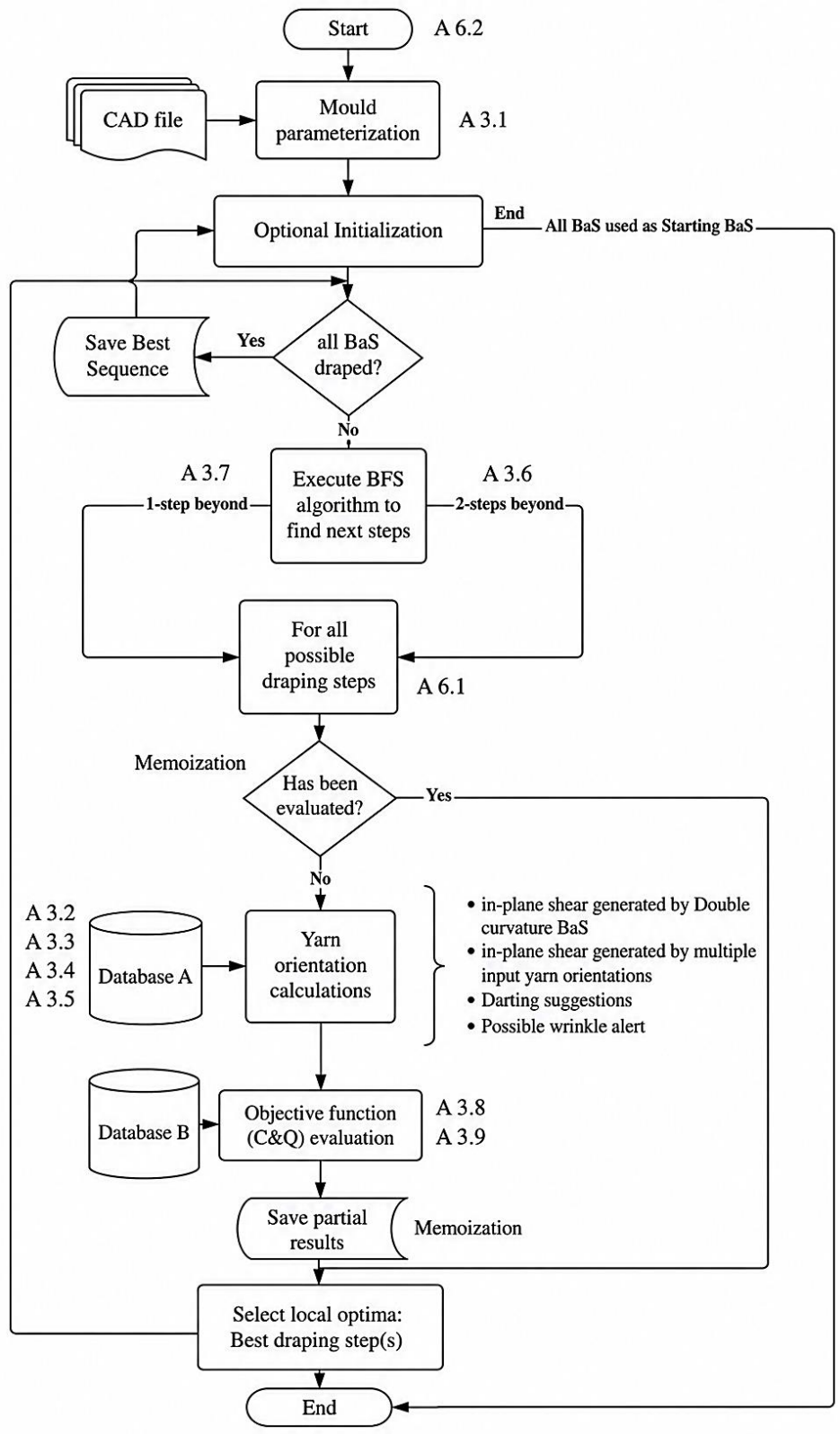


Figure 6.5 Dynamic programing algorithm flowchart

6.9 Dynamic Programming Evaluation Demonstrators

Four demonstrator parts were used in evaluating performance of the DP approach. The first is academic model demonstrator 31_Benchmark derived from the mould originally introduced by Hancock [67] described in Section 2.3 and shown in Figure 6.6. The second, academic demonstrator 32_Variation, is a variation of demonstrator 31_Benchmark shown in Figure 6.7. The third is academic demonstrator 01_Francois shown in Figure 6.8, and the fourth is industrial demonstrator 24_Hutchinson shown in Figure 6.9 .

Compared to cases analysed in Chapter 5, selection of these test cases was motivated by their geometry characterized by the presence of sharp internal corners, multiple regions of double curvature and/or by a larger number of BaS: 10 BaS for demonstrators 31_Benchmark and 32_Variation, 15 BaS for 01_Francois and 66 BaS for demonstrator 24_Hutchinson.

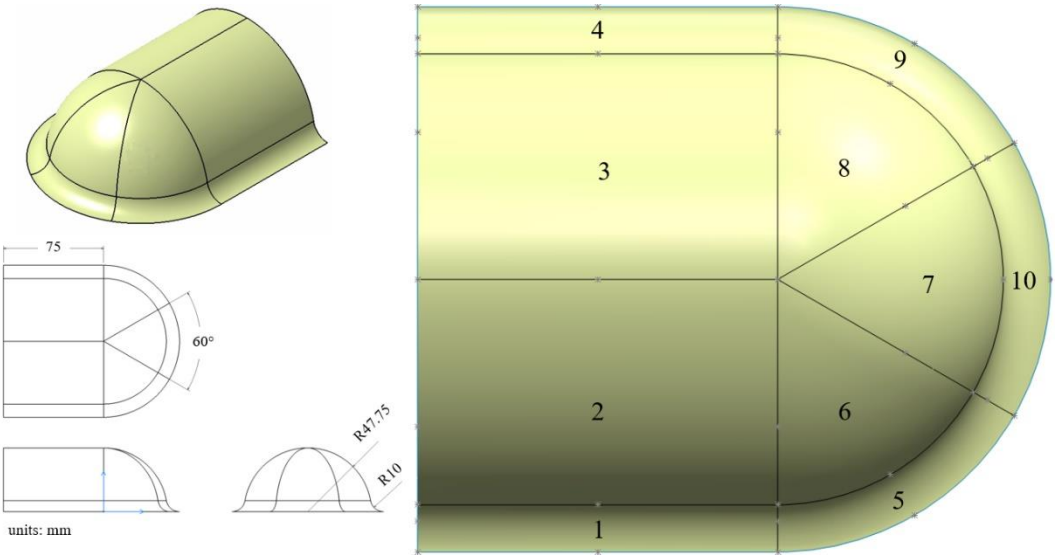


Figure 6.6 Demonstrator 31_Benchmark; Surface area: 20 258 mm^2

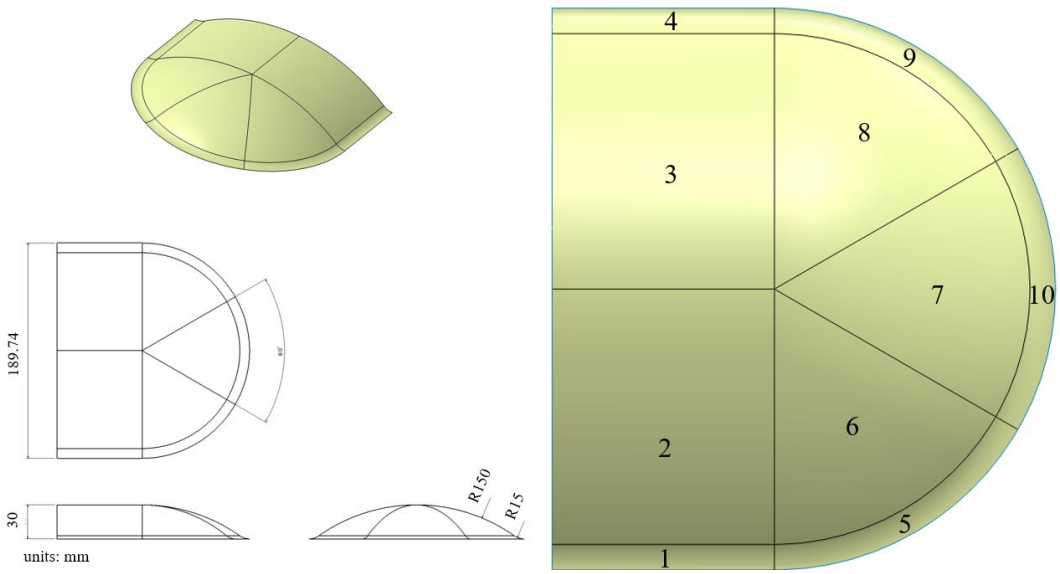


Figure 6.7 Demonstrator 32_Variation; Surface area: 30 645 mm^2

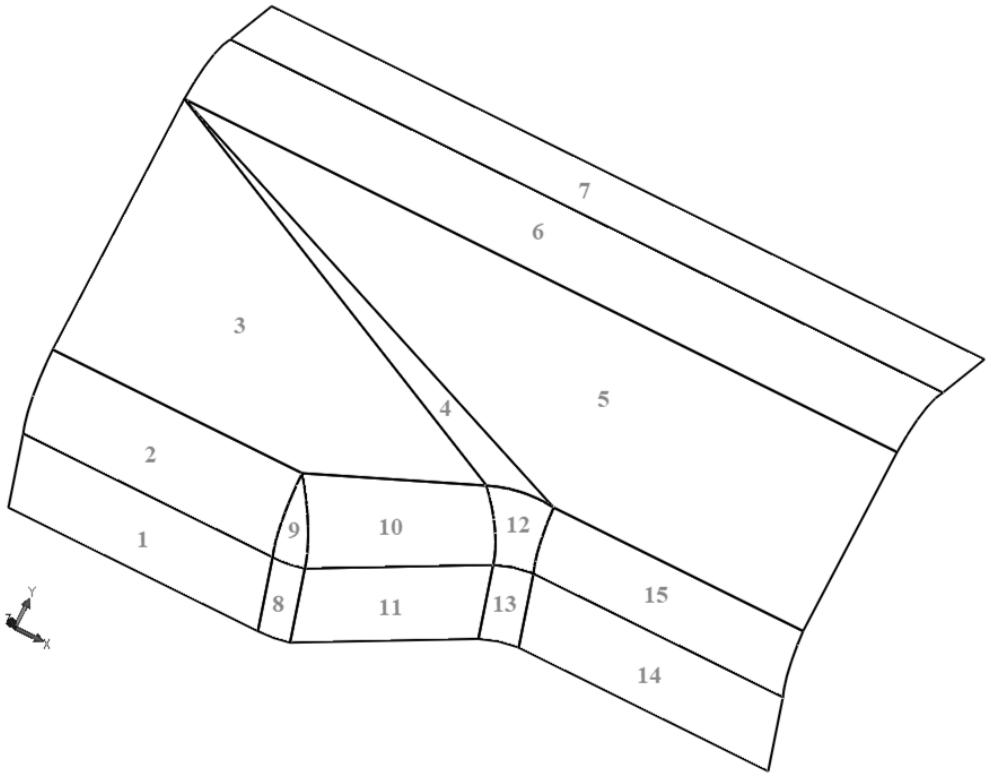


Figure 6.8 Demonstrator 01_Francois; Surface area : 33 871 mm^2

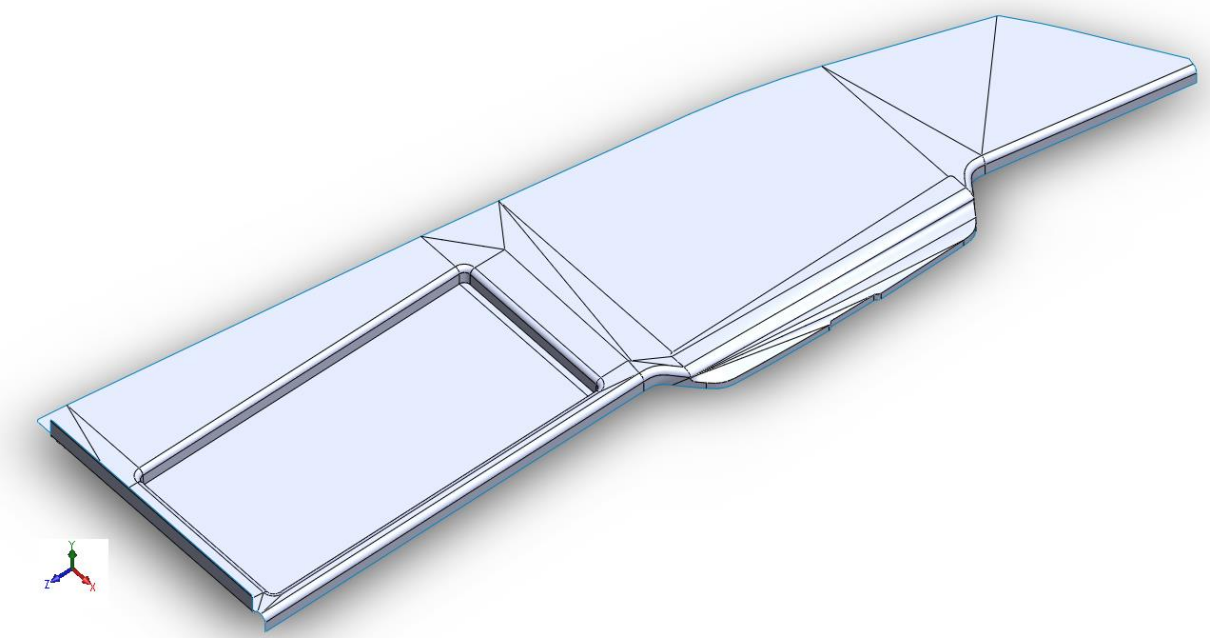
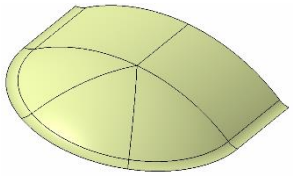


Figure 6.9 Industrial demonstrator 24_Hutchinson; Surface area: 354 838 mm^2

6.10 Design of Dynamic Programming Evaluation Programme

The first testing of the DP algorithm was comparative in nature: BF and DP algorithms were run on demonstrator 32_Variant. The number of possible sequences is shown in Table 6.1.

Table 6.1 Brute Force analysis for demonstrator 32_Variant

Geometry	BF generated sequences - computing time	BF Objective Function evaluation – estimated computing time
	3 550 144 sequences	4.5 seconds/sequence (avg)
	594 seconds	986 hours = 41 days

A computing time of 41 days for analysing a 10-BaS mould is incompatible with industrial performance requirements for the software tool, which is intended to operate on an average personal computer and propose answers to client requirements within a few days. For more complex geometries featuring 10 or more BaS, DP is preferred.

Table 6.2 presents a set of 8 experimental testing runs selected for evaluating 1-next state and 2-next states variants. Both academic and industrial demonstrators were represented. The 8 testing runs cover a representative range of geometries, materials and initial configurations, enabling testing of the DP algorithm under realistic conditions. Both DP variants were tested on equivalent scenarios. Section 6.11 present the results obtained from DP algorithms testing.

Table 6.2 Dynamic Programming testing runs, 1-next state and 2-next states

Demonstrator	Shear angle used	Fabric and layers	Initial orientation	Run #
31_Benchmark	Maximum	Fabric #1, 1 ply	0°/ 90°	1
			+/- 45°	2
32_Variation	Maximum	Fabric #1, 1 ply	0°/ 90°	3
			+/- 45°	4
01_Francois	Maximum	Fabric #2, 2 plies	0°/ 90°	5
			+/- 45°	6
24_Hutchinson	Maximum	Fabric #3, 3 plies	0°/ 90°	7
			+/- 45°	8

6.11 Dynamic Programming Results and Computational Performance Analysis

Next Section 6.11.1 presents a brief comparison between the BF and DP 1-next state algorithms for demonstrators 30_Corner and 27_Hutchinson. This benchmarking exercise was conducted using the same fabrics and initial yarn orientations for both optimisation approaches.

The remainder of this Chapter presents the results obtained from the eight DP testing runs defined in Section 6.10, for each of the 2 DP algorithms developed in this thesis. Representative results are presented for the 1-next state and 2-next states algorithms in Sections 6.11.3 to 6.11.5 , while complete results including optimised draping sequences, yarn orientations, wrinkle warnings, darting recommendations, validation photographs and individual discussions for each run are provided in Appendix B.

6.11.1 Benchmarking of Dynamic Programming Against Brute Force Optimisation

Brute Force optimisation guarantees identification of the global optimum by evaluating all feasible draping sequences in an exhaustive manner. For this reason, BF results obtained in Chapter 5 were used as a benchmark for assessing the performance of the DP 1-next state algorithm. Demonstrators 30_Corner and 27_Hutchinson were selected because exhaustive evaluation was computationally feasible for these geometries. The comparison was conducted using the same fabrics, initial yarn orientations and Objective Function formulation. Table 6.3 summarizes the results in terms of optimised Objective Function values and computing times.

Table 6.3 Benchmarking of DP against BF Optimisation

	Brute Force		**Dynamic Programming**	
	best *f(C, Q)*	Computing time	best *f(C, Q)*	Computing time
27_Hutchinson	26.29	1 m 19 s	26.29	36 s
30_Corner	2.92	51 min 36 s	2.92	43 s

Two important conclusions can be drawn from Table 6.4. First, the DP 1-next state algorithm was able to identify the same global optimum Objective Function value as BF in a consistent manner. Although only the best Objective Function values are presented in the table, optimisation results obtained from every starting BaS were compared for each demonstrator. In all cases, the DP algorithm reached the same Objective Function value as the global optimum identified through exhaustive BF evaluation.

The second important observation is the substantial reduction in computing time. This effect was particularly evident for demonstrator 30_Corner, featuring 7 BaS, where computing time was reduced by 98.6%.

Based on these findings, it can be concluded conservatively that the proposed DP algorithm is capable of generating at least near-optimal solutions while requiring only a fraction of the computational resources and execution time associated with BF optimisation.

6.11.2 Summary of Dynamic Programming Results

This section summarizes the results for both the 1-next state and 2-next states algorithms, allowing direct comparison of optimisation performance across the different demonstrators, fabrics and initial yarn orientations for every run as introduced in the previous Section.

Optimised Objective Function values and total computing time are presented for every run in Table 6.4. This computing time includes main algorithm processing as well as generation of six rotated 2D views of the final optimised draping sequence obtained for each starting BaS, including the resulting warp and weft vectors. Both processes were executed on an Intel Core i9-13950HX processor with 8 performance cores and 16 efficient cores with maximum frequency up to 5.5 GHz.

Table 6.4 Summary of Dynamic Programming optimisation results

Demonstrator	Run #	DP 1-next state		DP 2-next states	
		f(C, Q)*	**Computing time**	***f(C, Q)****	**Computing time**
31_Benchmark	Run 1	11.27	3 min 6 s	14.44	2 min 25 s
	Run 2	11.27	3 min 4 s	24.48	2 min 54 s
32_Variation	Run 3	13.93	3 min 29 s	17.82	2 min 30 s
	Run 4	13.93	3 min 11 s	19.2	3 min 0 s
01_Francois	Run 5	4.22	8 min 31 s	3.58	4 min 48 s
	Run 6	5.03	5 min 7 s	5.01	4 min 12 s
24_Hutchinson	Run 7	84.43	115 min 24 s	87.24	103 min 48 s
	Run 8	83.98	91 min 36 s	86.1	93 min

It is important to consider that the computing times presented include optimisation and visual guides generation for implementing the optimised draping sequence. For example, for Run 1 the total computing time with visualization was approximately 2.7 times the optimisation time alone.

As a rule of thumb, roughly 40% of total computing time is dedicated to optimisation, while about 60% corresponds to visualization.

General observations from the summarized results presented in Table 6.5 are listed below:

- For the simpler academic demonstrators, the larger decision horizon of the DP 2-next states algorithm was not as effective as expected. For these demonstrators, the branching nature of the BFS-based search space generation used by the DP 1-next state algorithm led to a more diverse exploration of candidate solutions and, consequently, lower Objective Function values.
- On average, DP 2-next states reduced computing time by 16%. For the two simpler demonstrators, the best solutions obtained were approximately 50% inferior to those obtained with DP 1-next state. Conversely, for the two more complex demonstrators, DP 2-next states produced Objective Function values that were approximately 2% lower on average.
- Industrial scalability of the proposed DP algorithms was successfully demonstrated. The draping process for demonstrator 24_Hutchinson was optimised in approximately 1 hour and 30 minutes using a commercially available laptop computer.

Representative testing results for the DP 1-next state and DP 2-next states algorithms are presented in the following three sections. Complete results for all testing runs are provided in Appendix B, along with further case by case discussion of results.

6.11.3 DP 1-Next State - Run 2

A total of 378 partial draping sequences were evaluated and optimised for DP 1-next state Run 2 with initial orientation at +/- 45°. All BaS were tested as starting BaS. Total computing time was 3 min 4 s . Results are summarized in Table 6.5 .

Figures 6.10 and 6.11 show the optimal draping sequence for an initial orientation of +/- 45°, and the resulting warp and weft vector orientations, respectively. The optimised sequence clearly shows the greedy nature of DP, with a noticeable tendency to delay the draping of double-curvature BaS.

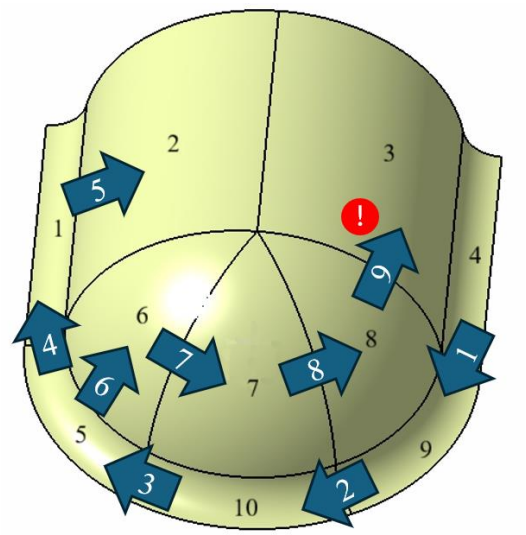


Figure 6.10 DP 1-next state - draping sequence for optimised Run 2

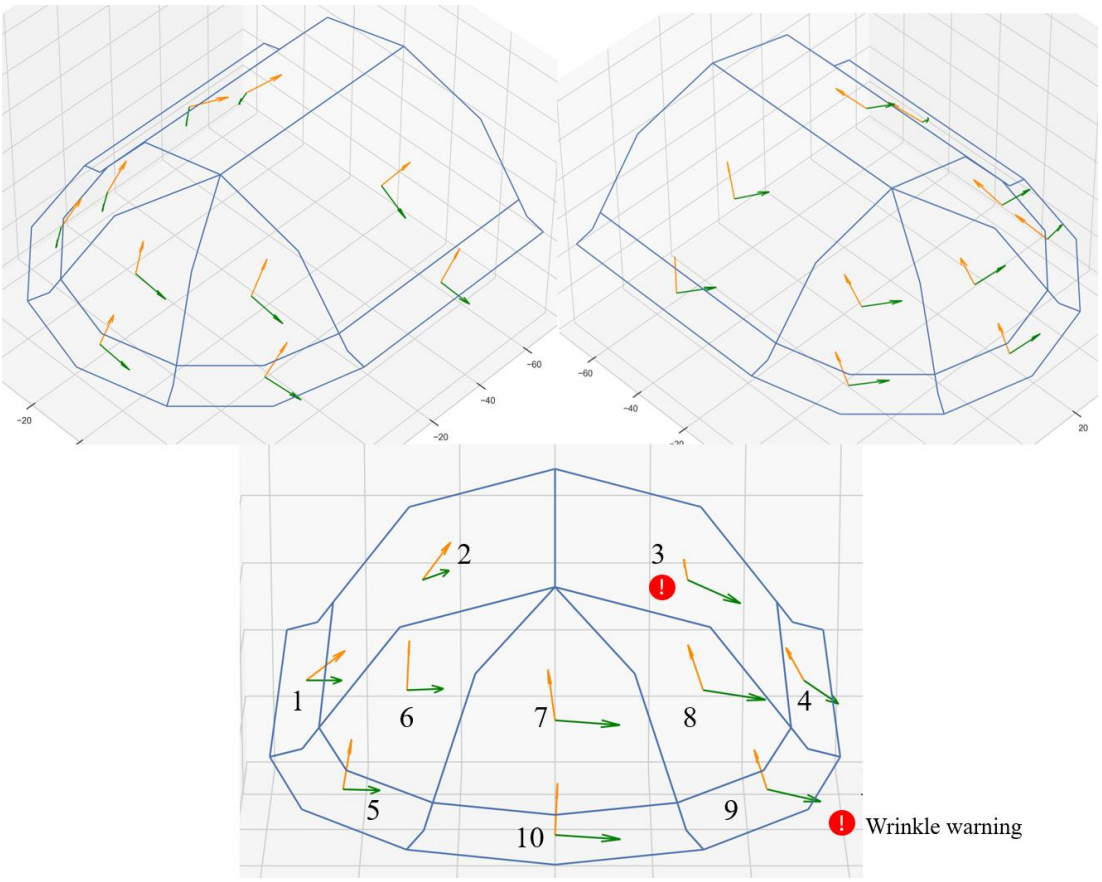


Figure 6.11 DP 1-next state resulting orientations for optimised Run 2

Table 6.5 DP 1-next state - Run 2 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Wrinkle warnings	Optimised draping sequence:
31_Benchmark 1-next state Run2	1	11.27	1	[1-5, 5-10, 10-9, 9-4, 1-2, 5-6, 6-7, 7-8, 8-3(W, D)]
	4	11.27	1	[4-9, 9-10, 10-5, 5-1, 1-2, 5-6, 6-7, 7-8, 8-3(W, D)]

The optimised draping sequences for Run 2 exhibit realistic progression in which high in-plane deformation was deferred to the latter stages of the draping process. Starting from lateral single-curvature regions led the algorithm to favour peripheral draping steps initially; however,

the no-enclosure condition prevented the algorithm from enclosing undraped BaS. As a result, the progression model produced a feasible draping sequence with a wrinkle warning issued for BaS #3 at a late stage of the sequence. As observed, the optimal draping sequences for BaS #1 and BaS #4 are nearly symmetrical; they differ only in their initial stages before progressing through identical steps.

Figure 6.12 shows the resulting draping sequence. During validation, considerable operator intervention was required as the fabric was progressively draped toward the opposite half of the mould. This behaviour is consistent with that observed in Run 1, the details of which are provided in Appendix B.

Figure 6.12 DP 1-next state - validation for optimised Run 2

Figure 6.12 validates the formation of a wrinkle during final draping step [BaS #8 – BaS #3] and demonstrates the suitability of darting to manage this issue. Crucially, the optimised sequence for Run 2 displaces the wrinkle to the periphery of BaS #3, facilitating the recommended darting operation.

6.11.4 DP 1-Next State - Run 5

Optimisation performance and results for this run are presented in Appendix B.5 the optimal draping sequence is illustrated in Figure 6.13 and its laboratory validation is shown in Figure 6.14, where warp and weft orientations are presented in blue and red colors.

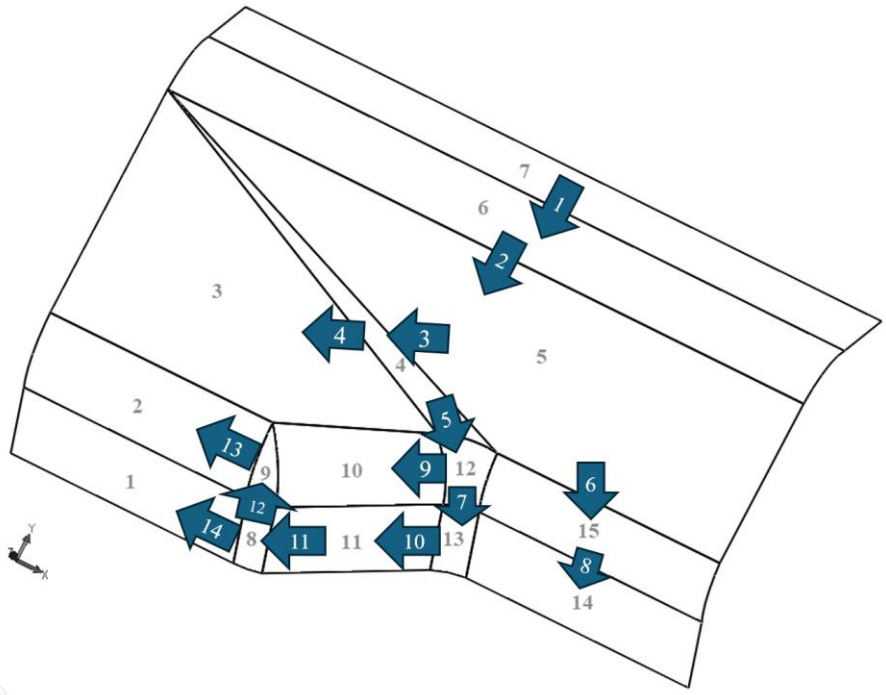


Figure 6.13 DP 1-next state - draping sequence for optimised Run 5

Figure 6.14 DP 1-next state - validation for optimised Run 5

Figure 6.15 presents the case of a non-optimised draping sequence [1-8, 1-2, 2-9, 9-10, 8-11, 11-13, 10-12, 12-15, 13-14, 12-4, 2-3, 15-5, 5-6, 6-7] which was generated and tested manually for comparison purposes; it was not produced by the optimisation algorithm. Its inclusion highlights an undesirable outcome associated with poor shear management and reinforces the benefit of the optimised sequencing strategy.

Figure 6.15 Non-optimised draping sequence illustrating shear dispersion effects

6.11.5 DP 2-Next States - Run 7

Given the substantially larger search space explored by the 2-next states DP algorithm, memory usage becomes a critical consideration. For this reason, wrinkle warning and darting recommendation parameters were not stored for hypothetical states, preventing excessive memory usage while preserving the core optimisation functionality.

This section presents results obtained using the DP 2-next states optimisation algorithm for Run 7 on the 66-BaS demonstrator 24_Hutchinson, with the initial yarn orientation set at 0°/90°. Table 6.6 summarizes optimisation results from the starting BaS selected by the initialization algorithm, while Table 6.7 lists the optimised draping sequence obtained from the selected starting BaS #12. Figures 6.16 to 6.18 show the corresponding draping sequence. A total of 11 276 partial draping sequences were evaluated and optimised in 103 min and 48 s.

Table 6.6 DP 2-next states - Optimisation results for all selected starting BaS for Run 7

Starting BaS#	Cost & Quality	Starting BaS#	Cost & Quality
2	279.06	30	110.19
5	237.47	31	280.25
10	278.17	44	110.98
11	280.66	57	237.50
12	87.24	58	187.73
20	232.52	59	185.83
29	112.43	61	113.93

Table 6.7 DP 2-next states - Best draping sequence for Run 7

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Optimised draping sequence:
24_Hutchinson 2-next states Run 7	12	87.24	[12-11, 11-41, 41-40, 40-38, 11-10, 10-47, 10-30, 30-29, 12-28, 28-9, 9-2, 2-3, 38-44, 44-39, 3-22, 22-32, 3-15, 15-18, 9-13, 13-16, 38-36, 36-34, 39-37, 37-35, 32-33, 33-23, 3-1, 1-4, 18-21, 21-25, 40-45, 45-51, 16-19, 19-24, 30-49, 49-54, 54-55, 55-63, 51-56, 56-62, 62-65, 65-64, 23-5, 5-6, 29-31, 31-48, 49-50, 50-57, 21-20, 20-17, 44-46, 46-52, 63-66, 66-59, 9-8, 12-8, 6-7, 48-53, 25-27, 24-26, 17-14, 2-14, 63-58, 57-58, 20-42]

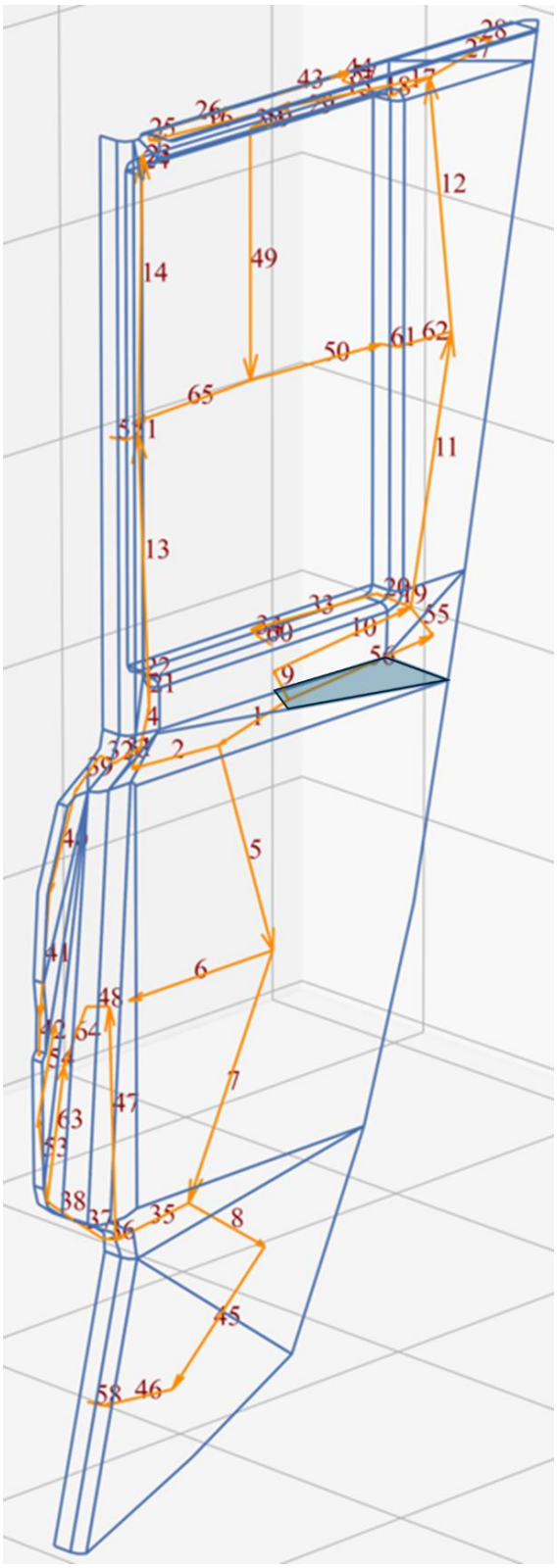


Figure 6.16 DP 2-next states - draping sequence for optimised Run 7

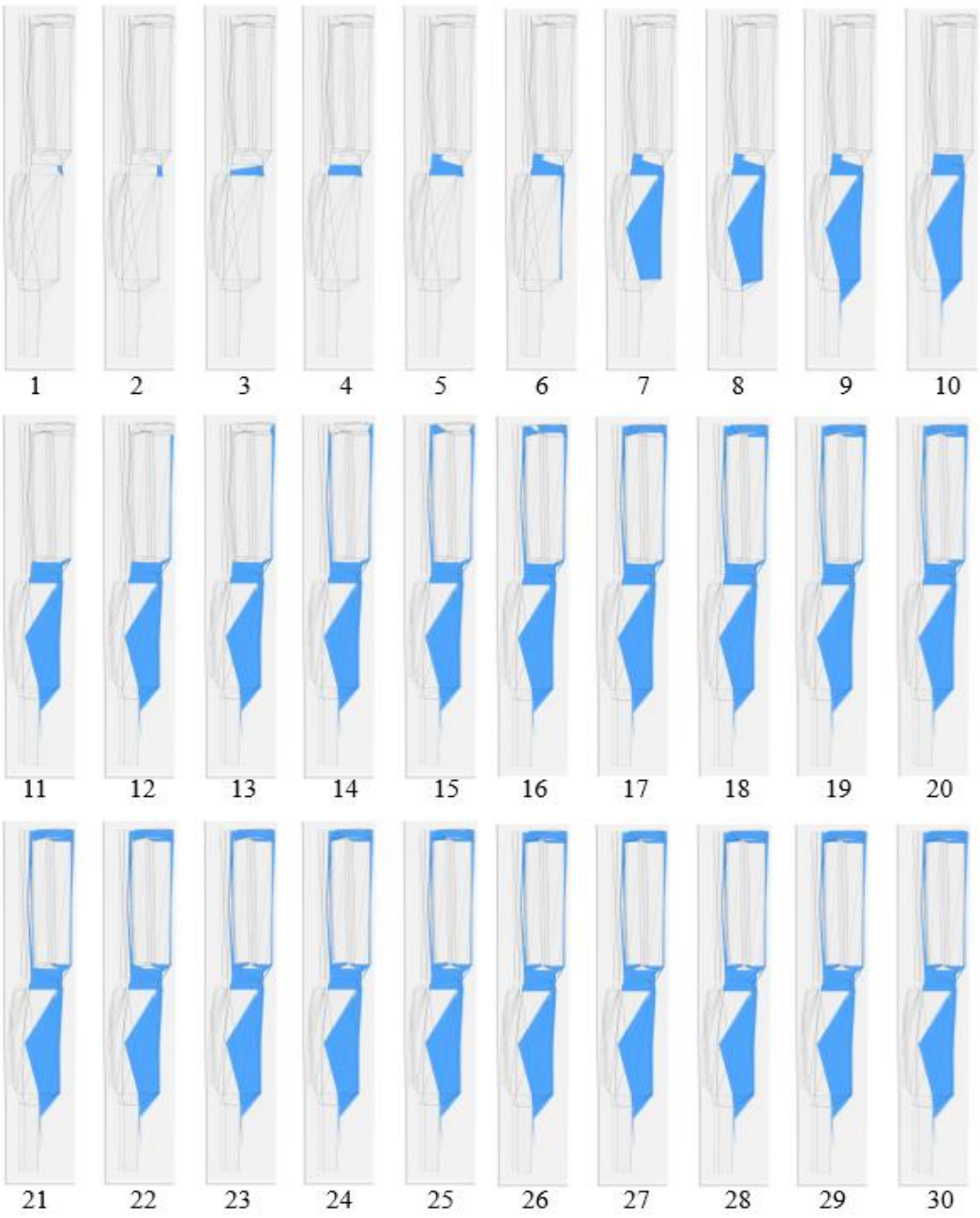


Figure 6.17 DP 2-next states - draping sequence for optimised Run 7 - steps 1 to 30

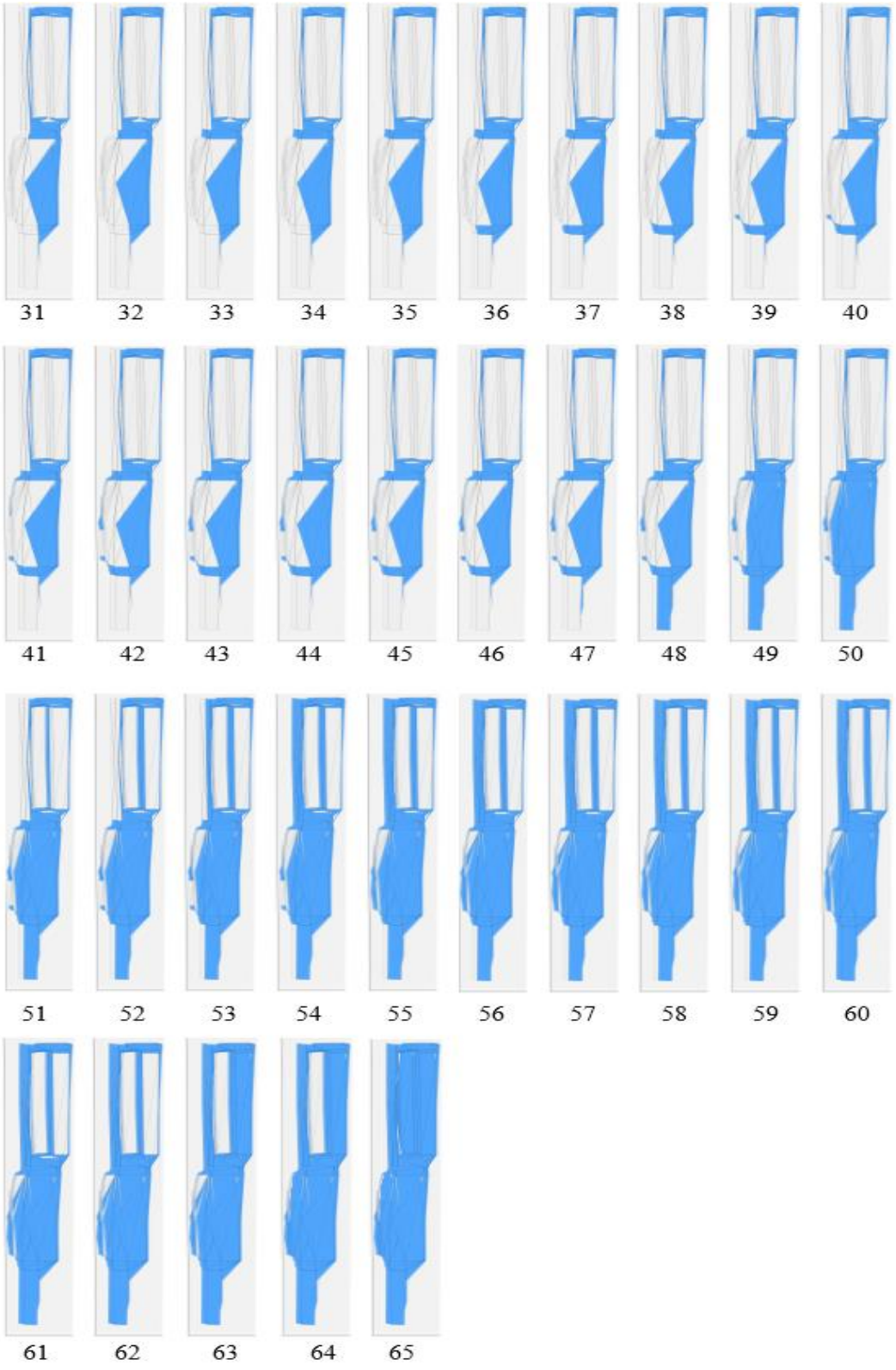


Figure 6.18 DP 2-next states - draping sequence for optimised Run 7 - steps 31 to 65

A similar value was obtained for the best draping sequence for 2-next states Run 7, with a difference of +3.3%. This optimised draping sequence starts from the central flat region of the industrial mould; this facilitates further draping operations due to low in-plane shear being transferred from draped BaS in early stages. Only 6 BaS were listed with final in-plane shear larger than 15°, with a maximum shear angle of 42° registered for BaS #59. This is an improvement with respect to the 1-next state algorithm, where 7 BaS reached in-plane shear above 15° and two BaS reached in-plane shear above 45°.

As observed in Figures 6.17 and 6.18, the lower left section with the most complex geometry was not draped until step 55 of 65, hence during the latest stages of the draping sequence. This is highly desirable, as explained in previous sections.

In total, this 2-next states Run 7 took 12 fewer minutes to execute than its 1-next counterpart, representing a decrease of approximately 12% in processing time with a similar optimised value.

6.12 Conclusions

This chapter presented the development, implementation and validation of DP algorithms for the optimization of HLU draping sequences. By evaluating these algorithms across four demonstrators of increasing geometric complexity ranging from academic benchmarks to an industrial 66-BaS part, this work has demonstrated that DP provides a computationally efficient alternative to BF methods for complex HLU draping.

The analysis of the 1-next state and 2-next states algorithms reveals several critical insights into the nature of automated draping optimization:

- Enclosure and realism: A significant finding was the necessity of the No_Enclosure constraint. The initial greedy nature of the 1-next state algorithm often led to complex scenarios where undraped regions became surrounded by draped fabric. Implementing enclosure avoidance proved essential to achieving draping sequences that reflect draping decisions similar to the ones made by experienced HLU technicians.
- In-plane shear management: Across multiple evaluation runs, for example runs 1, 2 and 5, optimised sequences demonstrated consistently the ability to defer high in-plane shear to the final stages of HLU. This strategy effectively limits shear propagation, thereby reducing the likelihood of wrinkles and minimizing the need for increased operator intervention. Experimental validation confirmed that wrinkle warnings and darting recommendations issued by the software align with physical results.

- Optimisation space: The 2-next states algorithm provided a broader search space which, in certain complex cases such as 01_Francois demonstrator, identified sequences that delayed the draping of double-curvature BaS more effectively than the 1-next state algorithm. This behavior mimics more closely the strategic decision-making of experienced technicians. However, analysis also highlighted the sensitivity of the optimisation model to enclosure effects when branching constraints are not enforced.

- Computational efficiency and scale: The integration of memoization and of the initialization algorithm proved essential for scaling the software to industrial demonstrators. While a hardware threshold of 32 GB RAM was identified for the 66-BaS industrial demonstrator, the initialization algorithm successfully narrowed the search space to a group of preferred start BaS, particularly by identifying that starting on double curvature surfaces yields higher values.
- Optimization of visualization resources: 3D visualization is a critical tool for translating optimised sequences into plybooks actionable on the shop floor. However, it was found to account for approximately 60% of total processing time. To address this, a conditioning functionality was implemented to disable visual generation during initial evaluation of multiple starting BaS. Once the algorithm identifies the most efficient starting point, visualization is reactivated for a final run. This dual-phase approach allows for rapid global optimization without sacrificing visual guidance necessary for production.
- Industrial viability: Results confirm that DP algorithms are suitable for real-time industrial use, running effectively on standard high-performance computers.

In summary, Chapter 6 validates that a model-driven, database-supported DP approach can successfully replace trial and error in HLU draping design. By quantifying the balance between cost and quality $f(C, Q)$, the software provides a repeatable and rigorous methodology for identifying superior draping sequences.

Chapter 7. Discussion and Conclusions

7.1 Introduction

This chapter discusses results presented in Chapters 5 and 6 in relation to the objectives and original contributions of this thesis. The proposed software tool enables the optimisation of dry fabric draping by generating and evaluating draping sequences based on engineering parameters obtained through database querying and evaluation of the Objective Function quantifying the cost and quality of draping the reinforcement.

The following discussion opens on a review of the modelling approach used for representing HLU as a sequence of discrete draping steps. Assumptions related to geometric parameterization, initial conditions and decision rules are discussed in terms of their impact on realism, computational efficiency and optimisation outcomes.

Then, this chapter discusses the role of BF optimisation as a validation and benchmarking tool, as well as its limitations for industrially relevant problems. The DP framework is discussed subsequently with regards to its efficiency and applicability to complex moulds. In all cases, optimisation is performed by evaluating the Objective Function. The user is also provided with wrinkle warnings and darting recommendations, enabling more informed decision-making when selecting the optimal draping sequence.

Finally, this chapter reiterates the main contributions of this work, discusses its limitations and outlines directions for future work.

7.2 Discussion of Hand Lay-up Modelling

The HLU modelling framework developed and adopted in this work enabled the automatic generation of feasible draping sequences without user intervention. By expressing the draping process as a sequence of discrete steps, the optimisation problem becomes computationally manageable and adapted for systematic evaluation using both BF and DP approaches. The formulation of the draping process made it possible to explore large solution spaces while maintaining a consistent representation of the process.

Logic based progression rules limited the generation of physically unrealistic or unfeasible draping sequences, leading to coherent optimisation outcomes across different demonstrators. This robustness facilitates the exploration of different input data, enabling direct comparison between alternative draping strategies.

From a computational perspective, discretizing the HLU process into BaS significantly reduces the problem size compared to conventional Finite Element Analysis. While standard FEA approaches typically involve thousands of elements to approximate a continuous surface, this method divides the mould into larger geometric entities, thereby lowering the dimensionality of the decision space. Clear definitions of system states, such as the draped/undraped status of a BaS and its neighbours, peripheral BaS and peripheral edges, facilitate the evaluation of both partial and complete draping sequences.

Additional constraints and checks, such as the Algorithm 6.1 No_Enclosure functionality, limited the exploration of unfeasible or undesirable draping decisions. Integration of memoization in the DP framework improved computational performance for moulds with large numbers of BaS.

In terms of result interpretation, the modelling framework allowed linking of optimised outputs with practical manufacturing actions. Draping sequences often resembled strategies that an experienced operator might explore during HLU, highlighting the practical relevance of the proposed approach. Additionally, the ability to identify and compare multiple near-optimal sequences provides flexibility in selecting a draping strategy suited to shop floor constraints.

The modeling framework is designed for balancing computational speed with physical accuracy, prioritizing its function as a rapid decision-support tool. The physical behaviour of reinforcements is derived through linear interpolation of laboratory data from Database B, ensuring sufficient precision for comparative optimization. Furthermore, complex mould geometries are represented efficiently using a 5-parameter Frustum model, which maintains geometric fidelity while enabling the rapid evaluation of draping strategies required for industrial applications.

7.3 Role of Brute Force Optimisation

BF optimisation served as the foundational benchmark for the development and validation of the draping sequence optimisation framework. As detailed in Sections 5.3 and 5.4 the exhaustive evaluation of all feasible draping sequences for demonstrators with a limited number of BaS provided direct access to the complete solution space. This enabled the identification of global optima under the defined Objective Function, fulfilling the first objective of this work: the systematic and automatic comparison of very large numbers of possible draping scenarios.

BF optimisation functioned as a critical baseline for verifying that the HLU model and the Objective Function produced physically possible evaluations. BF results established confidence in

the framework's ability to quantify cost and quality before more complex algorithms were developed. This validation represents a key original contribution of the thesis, as it constitutes the first instance in which geometric and engineering parameters queried from databases are used for evaluating the complete space of discrete draping steps and sequences.

The value of the BF analysis lies in its ability to explore all possible draping decisions for the draping of smaller demonstrators. By evaluating every possible sequence, it was possible to observe that the Objective Function is sensitive to the accumulation of in-plane shear across sequential steps. BF results confirmed that the optimal path is not merely a collection of the best individual steps, but a globally integrated sequence. This insight provided justification for the transition to DP; it demonstrated that while the solution space associated with industrial moulds may be too large for exhaustive search, it possesses a structure that can be exploited by more sophisticated optimisation algorithms.

However, BF results also revealed fundamental scalability constraints. As the number of BaS increased the factorial growth of permutations to be evaluated rendered exhaustive searches impractical for most industrially relevant moulds. For example, while 10-BaS Demonstrator 31_Benchmark was successfully solved, the exponential expansion of the decision space for larger geometries confirmed that BF is ill-suited to the rapid identification of manufacturing designs required in an industrial context. These constraints necessitated the development of the DP approach presented in Chapter 6, justifying the transition from exhaustive enumeration to more computationally efficient optimisation strategies required for addressing large-scale industrial geometries.

7.4 Discussion of Dynamic Programming Results

This section discusses results obtained using the DP optimisation approach presented in Chapter 6. First, the analysis examines the characteristics of the 1-next state formulation, highlighting general trends observed across the different optimisation runs including sensitivity to local optima, computational efficiency and typical limitations. Then, the discussion addresses 2-next states algorithm performance with emphasis on the effect of additional look-ahead on solution quality, consistency of optimisation outcomes and computational cost.

Finally, the practical implications of selecting between the 1-next state and 2-next states approaches are discussed, with attention given to trade-offs between optimality and computational effort in an industrial context.

7.4.1 Behaviour of 1-Next State Approach

The 1-next state DP approach exhibited a consistent tendency to defer high in-plane shear to the latter stages of draping sequences. As evidenced in Runs 1 and 2 for Demonstrator 31_Benchmark, this strategy ensured that wrinkle warnings were issued only during latter draping steps. By limiting shear generation and propagation in early and intermediate stages the algorithm effectively mimics expert hand lay-up strategies where complex local deformation is managed only after most of the mould surface has been draped.

The greedy nature of the 1-next state algorithm provides a significant advantage in initial optimisation speed, yet it also exhibits high sensitivity to the sequence in which draping steps are evaluated. For instance, during the optimisation of the final draping step of the Run 1, the algorithm

identified three candidate steps sharing identical Objective Function values. In such scenarios, final selection depended on the indexing order within the evaluation list or memoization matrix. Rather than representing a modelling flaw, these variations highlight the algorithm's ability to identify multiple feasible local optima. Furthermore, the No_Enclosure functionality proved essential in maintaining realistic progression; in Run 2, this condition prevented the premature draping of peripheral regions that would have isolated undraped BaS.

Results across different demonstrators confirm that the algorithm is sensitive to geometric complexity rather than just fabric orientation. Changing the initial yarn orientations from 0°/90° to ±45° had no significant impact on the final Objective Function for Runs 3 and 4. However, in Runs 5 and 6 for the 15-BaS demonstrator 01_Francois, the algorithm effectively optimised the draping sequence through double-curvature regions without triggering wrinkle warnings.

Computational efficiency remains a core strength of the 1-next DP approach. For complex cases such as 66-BaS industrial demonstrator 24_Hutchinson in Run 7, the algorithm successfully evaluated over 100,000 partial sequences in approximately 115 minutes, despite high RAM requirements. This performance was facilitated by memoization which reduced execution time by up to 35% by minimizing redundant calls to Database B. Furthermore, analysis of computing time indicates that while optimisation is efficient, a significant portion of processing - roughly 60% - is dedicated to the generation of rotated 2D views and vector visualisation. Collectively, these results validate the 1-next state DP formulation as a robust and industrially viable framework for rapid draping strategy evaluation.

7.4.2 Behaviour of 2-Next States Approach

The 2-next states approach was developed as a variation of the DP framework to introduce a limited look-ahead capability. By evaluating the combined Objective Function for draping two

sequential BaS, this method aimed at mitigating the greedy limitations of the 1-next state approach hence identify superior global draping sequences.

It was confirmed that the 2-next states algorithm represents a variation of the DP optimisation engine rather than a linear evolution of the 1-next state algorithm. A primary technical divergence is the exclusion of proven BFS branching during the main iteration, which ended up limiting the amount of possible draping steps being evaluated. It is possible that optimal 2-steps draping combinations were lost because of this condition, even though incorporating branching at this level of look-ahead would result in an exponential expansion of the decision tree, compromising the 32 GB RAM capacity of the PC used for this thesis algorithms testing.

Another functionality tested for the 2-next states algorithm was the override of no branching combination of draping steps, which proved its efficacy for Run 5, for example. This override permits BFS branching only as a fallback when no valid 2-next states pairs are available while the mould is not fully draped.

Results from Chapter 6 indicate that the 2-next states approach does not always outperform the 1-next state algorithm, largely due to the different functionalities implemented for the 2-next states algorithm. For example, the deactivation of the No_Enclosure condition occasionally allowed the algorithm to pursue sequences that enclosed undraped BaS, leading to localized suboptimal results. These constraints were necessary for ensuring that the software remained executable on a standard PC within industrially relevant timeframes.

The 2-next states algorithm functions as a specialized complementary tool rather than a replacement for the 1-next state baseline. While the 1-next state approach offers superior speed and robustness across most geometries, the 2-next states method demonstrates value in identifying

superior solutions where inter-dependency between adjacent BaS is high. The choice between these two approaches represents a trade-off between Objective Function value and visual evaluation of the optimised draping sequence. It is encouraged to run both algorithms in the search of a superior solution for every mould.

7.5 Contributions and Conclusion

The development of a discrete HLU modelling framework integrated with both BF and DP optimisation algorithms marks a significant advance in the development of preforming strategies for dry reinforcements for composites. This work demonstrates that the complex fabric draping process can be structured rigorously and evaluated through a high-speed computational framework that synthesizes geometric parameterization with material-specific mechanical data. By providing an automated engine to identify optimal draping sequences, this work effectively eliminates reliance on trial-and-error methods used in industry, replacing manual intuition with repeatable, data-driven engineering methods and parameters.

Based on validation results presented in Chapters 5 and 6, original contributions made in this thesis include:

- First framework for systematic draping sequence evaluation: This work delivers the first automated tool capable of generating and evaluating very large solution spaces of draping scenarios and sequences. Through BF analysis, the software explored the

entire solution space for smaller geometries, confirming that the Objective Function evaluates and compares manufacturing cost and quality accurately.

- Discrete mathematical representation of HLU: By decomposing complex geometries into BaS, this thesis transformed the qualitative art of manual draping into a sequence of discrete, measurable steps. This enables the calculation of the energy required for draping a BaS, hence the cost for performing such operation, and the possible variability linked to draping decisions as per the experimental data contained in Database B.
- Scalable optimisation for industrial geometries: The transition to DP optimisation addressed the factorial growth of the decision space inherent to complex moulds. The software demonstrated industrial viability by evaluating over 100 000 partial sequences for a 66-BaS demonstrator in under two hours, aided by a memoization strategy that reduced redundant database queries by 35%.
- Material-specific decision support: The software bridges the gap between laboratory material characterization and shop floor implementation. By using linear interpolation of experimental data to drive the optimisation engine, the framework ensures that every sequence suggested aligns with specific mechanical limits of the fabric.

In conclusion, this thesis marks a transition in the manufacturing of dry textile reinforcement preforms for composites, from an experience-based manual craft to a predictable data-driven engineering process. By establishing a rigorous mathematical framework that discretizes the HLU process and optimizes it through both BF and DP approaches, this work provides a scalable solution to the long-standing challenges of cost and quality attainability and inconsistency of

decision-making. The software developed herein not only eliminates the industry's traditional reliance on trial-and-error, but it also creates a foundational architecture for future addition of more materials data and optimisation algorithms. Ultimately, the systematic evaluation of draping sequences through this computational engine ensures that composite parts can be produced with a level of repeatability and efficiency that meets the evolving demands of the aerospace and automotive sectors.

7.6 Observations and Future Work

While the software achieved its primary objectives, different technical observations were made during testing, providing a roadmap for future development:

- Processing efficiency: a significant observation was that approximately 60% of processing time was dedicated to visualizations and 2D rotations. Future work should focus on optimizing the graphical engine or utilizing parallel processing to further decrease execution time for large-scale industrial moulds;
- Hardware and memory constraints: the 2-next states approach highlighted a critical trade-off between look-ahead depth and RAM capacity. Future iterations could explore more memory-efficient branching to navigate deeper decision trees without exceeding standard workstation limits;

- Algorithmic refinement: while the 1-next state DP proved robust, the 2-next states DP showed potential for identifying superior solutions in areas of high geometric inter-dependency. Future research should aim at refining the look-ahead logic to better balance the searching of near optimal solutions with computational speed.

References

[1] M. Holmes, “Aerospace looks to composites for solutions,” *Reinf. Plast.*, vol. 61, no. 4, pp. 237–241, 2017, doi: 10.1016/j.repl.2017.06.079.

[2] B. Piquet, “Special edition A350 XBW,” *FAST Airbus Tech. Mag.*, no. 1, pp. 1-32, Jun. 2013. doi: 10.21428/c53615e7.6d3c0080.

[3] V. Maymard, “A new approach to simulating the draping of prepreg composites manufactured by hand layup,” M.S. thesis, KTH Royal Inst. Technol., Stockholm, Sweden, 2017.

[4] A. Skordos, C. Aceves, and M. Sutcliffe, “Drape Optimisation in Woven Composite Manufacturing,” *Proc 5th Int. Conf. Inverse Probl. Eng. Theory Pract.*, no. July, pp. 1–10, 2005.

[5] “Draping Simulation / Draping Analysis | ESI Group.” https://cn.esi-group.com/software-solutions/virtual-manufacturing/composites/draping-simulation/draping-analysis (accessed Nov. 25, 2021).

[6] “Composites.” https://www.mscsoftware.com/application/composites (accessed Nov. 25, 2021).

[7] S. Gagne, “Formage sous vide de renforts textiles de carbone et de verre pour fabrication de pieces composites pour l’aerospatiale,” M.A.Sc. thesis, Dept. Mech. Eng., Univ. Ottawa, Ottawa, ON, Canada, 2017.

[8] J. A. Hoffer, “Development of a Draping Algorithm for Non-Structural Aerospace Composites,” M.A.Sc. thesis, Dept. Mech. Eng., Univ. Ottawa, Ottawa, ON, Canada, 2019.

[9] P. Kanz, “Characterization of Textile Draping Behaviours for Composite Manufacturing Processes,” M.A.Sc. thesis, Dept. Mech. Eng., Univ. Ottawa, Ottawa, ON, Canada, 2021.

[10] C. C. Chu, C. L. Cummings, and N. A. Teixeira, “Mechanics of Elastic Performance of Textile Materials: Part V: A Study of the Factors Affecting the Drape of Fabrics—The Development of a Drape Meter,” *Text. Res. J.*, vol. 20, no. 8, pp. 539–548, 1950, doi: 10.1177/004051755002000802.

[11] P. Boisse, J. Colmars, N. Hamila, N. Naouar, and Q. Steer, “Bending and wrinkling of composite fiber preforms and prepregs. A review and new developments in the draping simulations,” *Compos. Part B Eng.*, vol. 141, pp. 234–249, May 2018, doi: 10.1016/J.COMPOSITESB.2017.12.061.

[12] A. Cherouat and H. Borouchaki, “Present state of the art of composite fabric forming: Geometrical and mechanical approaches,” *Materials (Basel).*, vol. 2, no. 4, pp. 1835–1857, 2009, doi: 10.3390/ma2041835.

[13] “Top 10 Industrial Design Software Packages for Companies | Cad Crowd.” https://www.cadcrowd.com/blog/top-10-industrial-design-software-applications-by-install-base/ (accessed May 18, 2023).

[14] S. Sivakumar and V. Dhanalakshmi, “An approach towards the integration of CAD/CAM/CAI through STEP file using feature extraction for cylindrical parts,” *https://doi.org/10.1080/0951192X.2012.749527*, vol. 26, no. 6, pp. 561–570, Jun. 2013, doi: 10.1080/0951192X.2012.749527.

[15] B. Sarde and Y. D. Patil, “Recent Research Status on Polymer Composite Used in Concrete-An Overview,” *Mater. Today Proc.*, vol. 18, pp. 3780–3790, Jan. 2019, doi: 10.1016/J.MATPR.2019.07.316.

[16] R. Yahaya, S. M. Sapuan, M. Jawaid, Z. Leman, and E. S. Zainudin, “Mechanical performance of woven kenaf-Kevlar hybrid composites,” J. Reinf. Plast. Compos., vol. 33, no. 24, pp. 2242–2254, Dec. 2014, doi: 10.1177/0731684414559864.

[17] R. Hsissou, M. Berradi, M. El Bouchti, A. El Bachiri, and A. El Harfi, “Synthesis characterization rheological and morphological study of a new epoxy resin pentaglycidyl ether pentaphenoxy of phosphorus and their composite (PGEPPP/MDA/PN),” Polym. Bull., vol. 76, no. 9, pp. 4859–4878, Sep. 2019, doi: 10.1007/s00289-018-2639-9.

[18] R. Hsissou *et al.*, “New epoxy composite polymers as a potential anticorrosive coatings for carbon steel in 3.5% NaCl solution: Experimental and computational approaches,” *Chem. Data Collect.*, vol. 31, p. 100619, Feb. 2021, doi: 10.1016/J.CDC.2020.100619.

[19] N. El-Aouni, R. Hsissou, J. El Azzaoui, M. El Bouchti, and A. Elharfi, “Synthesis rheological and thermal studies of epoxy polymer and its composite,” *Chem. Data Collect.*, vol. 30, p. 100584, Dec. 2020, doi: 10.1016/J.CDC.2020.100584.

[20] H. P. S. A. Khalil, M. A. Tehrani, Y. Davoudpour, A. H. Bhat, M. Jawaid, and A. Hassan, “Natural fiber reinforced poly(vinyl chloride) composites: A review,” J. Reinf. Plast. Compos., vol. 32, no. 5, pp. 330–356, Mar. 2013, doi: 10.1177/0731684412458553.

[21] O. Dagdag *et al.*, “Epoxy pre-polymers as new and effective materials for corrosion inhibition of carbon steel in acidic medium: Computational and experimental studies,” *Sci. Reports 2019 91*, vol. 9, no. 1, pp. 1–14, Aug. 2019, doi: 10.1038/s41598-019-48284-0.

[22] S. Mazumdar, “Composites Manufacturing : Materials, Product, and Process Engineering,” *Compos. Manuf.*, Dec. 2001, doi: 10.1201/9781420041989.

[23] R. M. Jones, “Mechanics Of Composite Materials,” Oct. 2018, doi: 10.1201/9781498711067.

[24] S. G. Advani and E. Murat Sozer, “Process Modeling in Composites Manufacturing,” *Process Model. Compos. Manuf.*, Aug. 2002, doi: 10.1201/9780203910061.

[25] F. Robitaille and R. Gauvin, “Compaction of textile reinforcements for composites manufacturing. I: Review of experimental results,” Polym. Compos., vol. 19, no. 2, pp. 198–216, Apr. 1998, doi: 10.1002/pc.10091.

[26] U. Vaidya, N. Uddin, S. Cauthen, and L. Ramos, “Vacuum Assisted Resin Transfer Molding (VARTM) for External Strengthening of Structures,” in *Developments in Fiber-Reinforced Polymer (FRP) Composites for Civil Engineering*, 2013, pp. 77–114. doi: 10.1533/9780857098955.1.77.

[27] K.-T. Hsiao and D. Heider, “10 - Vacuum assisted resin transfer molding (VARTM) in polymer matrix composites,” in *Manufacturing techniques for polymer matrix composites (PMCs)*, Elsevier Ltd, 2012, pp. 310–347. doi: 10.1016/B978-0-85709-067-6.50010-9.

[28] N. Kuentzer, P. Simacek, S. G. Advani, and S. Walsh, “Correlation of void distribution to VARTM manufacturing techniques,” *Compos. Part A Appl. Sci. Manuf.*, vol. 38, no. 3, pp. 802–813, 2007, doi: 10.1016/j.compositesa.2006.08.005.

[29] J. M. Lawrence, S. T. Holmes, M. Louderback, A. Williams, P. Simacek, and S. G. Advani, “The Use Of Flow Simulations Of Large Complex Composite Components Using The

Vartm Process," in *The Use Of Flow Simulations Of Large Complex Composite Components Using The Vartm Process*, Jul. 2008.

[30] Y. F. Zhang, Y. Q. Liu, R. K. Du, F. F. Wang, G. Z. Zhao, and X. H. Chen, "Flow Simulation and Optimization of the Car Bumper Beam by VARTM Process," in *Advanced Materials Research*, 2013, vol. 753–755, pp. 236–240. doi: 10.4028/www.scientific.net/AMR.753-755.236.

[31] T. Fumihito, H. Kengo, S. Yasuo, N. Shigeru, K. Yasuhiro, and A. Nobuo, "Research in the Application of the VaRTM Technique to the Fabrication of Primary Aircraft Composite Structures," *Mitsubishi Heavy Ind. Ltd. Tech. Rev.*, vol. 42, Dec. 2005.

[32] S. Gangil and N. Bhatnagar, "Comparative study of resin infusion techniques for manufacturing large composite structures," in *2016 International Conference on Nascent Technologies in Engineering (ICNTE)*, 2016, pp. 1–4. doi: 10.1109/ICNTE.2016.7568882.

[33] S. Shevtsov, I. Zhilyaev, S. H. Chang, J. K. Wu, J. P. Huang, and N. Snezhina, "Experimental and Numerical Study of Vacuum Resin Infusion for Thin-Walled Composite Parts," *Appl. Sci. 2020, Vol. 10, Page 1485*, vol. 10, no. 4, p. 1485, Feb. 2020, doi: 10.3390/APP10041485.

[34] V. R. Tamakuwala, "Manufacturing of fiber reinforced polymer by using VARTM process: A review," *Mater. Today Proc.*, vol. 44, pp. 987–993, 2021, doi: 10.1016/j.matpr.2020.11.102.

[35] M. Elkington, D. Bloom, C. Ward, A. Chatzimichali, and K. Potter, "Hand layup: understanding the manual process," *Adv. Manuf. Polym. Compos. Sci.*, vol. 1, no. 3, pp. 138–151, 2015, doi: 10.1080/20550340.2015.1114801.

[36] K. G. Swift and J. D. Booker, “Plastics and Composites Processing,” *Manuf. Process Sel. Handb.*, pp. 141–174, Jan. 2013, doi: 10.1016/B978-0-08-099360-7.00005-7.

[37] M. Raji, H. Abdellaoui, H. Essabir, C. A. Kakou, R. Bouhfid, and A. El Kacem Qaiss, *Prediction of the cyclic durability of woven-hybrid composites*. 2018. doi: 10.1016/B978-0-08-102290-0.00003-9.

[38] T. G. Gutowski, Ed., *Advanced Composites Manufacturing.* New York, NY, USA: John Wiley & Sons, 1997.

[39] P. Boisse, *Advances in composites manufacturing and process design*. Cambridge, UK: Woodhead Publishing, 2015.

[40] A. B. Strong, *Fundamentals of composites manufacturing : materials, methods, and applications*, 2nd ed. Dearborn, MI: Society of Manufacturing Engineers, 2008.

[41] J. W. Gooch, “Hand Layup,” in *Encyclopedic Dictionary of Polymers*, New York, NY: Springer New York, pp. 356–356. doi: 10.1007/978-1-4419-6247-8_5773.

[42] A. G. Prodromou and J. Chen, “On the relationship between shear angle and wrinkling of textile composite preforms,” *Compos. Part A Appl. Sci. Manuf.*, vol. 28, no. 5, pp. 491–503, Jan. 1997, doi: 10.1016/S1359-835X(96)00150-9.

[43] P. Boisse, A. Gasser, and G. Hivet, “Analyses of fabric tensile behaviour: determination of the biaxial tension–strain surfaces and their use in forming simulations,” *Compos. Part A Appl. Sci. Manuf.*, vol. 32, no. 10, pp. 1395–1414, Oct. 2001, doi: 10.1016/S1359-835X(01)00039-2.

[44] P. Boisse *et al.*, “The bias-extension test for the analysis of in-plane shear properties of

textile composite reinforcements and prepregs: a review," *Int. J. Mater. Form.*, vol. 10, no. 4, pp. 473–492, 2017, doi: 10.1007/s12289-016.

[45] J. Lindberg, B. Behre, and B. Dahlberg, "Part III: Shearing and Buckling of Various Commercial Fabrics," *Text. Res. J.*, vol. 31, no. 2, pp. 99–122, 1961, doi: 10.1177/004051756103100203.

[46] P. Grosberg and B. J. Park, "The Mechanical Properties of Woven Fabrics," *Text. Res. J.*, vol. 36, no. 5, pp. 420–431, 1966, doi: 10.1177/004051756603600505.

[47] P. Grosberg, G. A. V. Leaf, and B. J. Park, "The Mechanical Properties of Woven Fabrics," *Text. Res. J.*, vol. 38, no. 11, pp. 1085–1100, 1968, doi: 10.1177/004051756803801102.

[48] S. M. Spivak and L. R. G. Treloar, "The Behavior of Fabrics in Shear," *Text. Res. J.*, vol. 38, no. 9, pp. 963–971, 1968, doi: 10.1177/004051756803800911.

[49] W.-R. Yu, P. Harrison, and A. Long, "Finite element forming simulation for non-crimp fabrics using a non-orthogonal constitutive equation," *Compos. Part A. Appl. Sci. Manuf.*, vol. 36, no. 8, pp. 1079–1093, 2005, doi: 10.1016/j.compositesa.2005.01.007.

[50] N. Hamila, P. Boisse, F. Sabourin, and M. Brunet, "A semi-discrete shell finite element for textile composite reinforcement forming simulation," *Int. J. Numer. Methods Eng.*, vol. 79, no. 12, pp. 1443–1466, 2009, doi: 10.1002/nme.2625.

[51] S. P. Haanappel, R. H. W. ten Thije, U. Sachs, B. Rietman, and R. Akkerman, "Formability analyses of uni-directional and textile reinforced thermoplastics," *Compos. Part A. Appl. Sci. Manuf.*, vol. 56, pp. 80–92, 2014, doi: 10.1016/j.compositesa.2013.09.009.

[52] J. R. Smith, U. K. Vaidya, and J. K. Johnstone, "Analytical modeling of deformed plain

woven thermoplastic composites," *Int. J. Mater. Form.*, vol. 7, no. 4, pp. 379–393, 2014, doi: 10.1007/s12289-013-1133-z.

[53] M. Hübner, J.-E. Rocher, S. Allaoui, G. Hivet, T. Gereke, and C. Cherif, "Simulation-based investigations on the drape behavior of 3D woven fabrics made of commingled yarns," *Int. J. Mater. Form.*, vol. 9, no. 5, pp. 591–599, 2016, doi: 10.1007/s12289-015-1245-8.

[54] R. H. W. ten Thije and R. Akkerman, "Solutions to intra-ply shear locking in finite element analyses of fibre reinforced materials," *Compos. Part A. Appl. Sci. Manuf.*, vol. 39, no. 7, pp. 1167–1176, 2008, doi: 10.1016/j.compositesa.2008.03.014.

[55] X. Yu, B. Cartwright, D. McGuckin, L. Ye, and Y.-W. Mai, "Intra-ply shear locking in finite element analyses of woven fabric forming processes," *Compos. Part A. Appl. Sci. Manuf.*, vol. 37, no. 5, pp. 790–803, 2006, doi: 10.1016/j.compositesa.2005.04.024.

[56] N. Hamila and P. Boisse, "Locking in simulation of composite reinforcement deformations. analysis and treatment," *Compos. Part A. Appl. Sci. Manuf.*, vol. 53, pp. 109–117, 2013.

[57] P. Causse, E. Ruiz, and F. Trochu, "Influence of preforming on the quality of curved composite parts manufactured by flexible injection," *Int. J. Mater. Form.*, vol. 6, no. 3, pp. 341–362, 2013, doi: 10.1007/s12289-012-1090-y.

[58] J. Wang, R. Paton, and J. R. Page, "The draping of woven fabric preforms and prepregs for production of polymer composite components," *Compos. Part A. Appl. Sci. Manuf.*, vol. 30, no. 6, pp. 757–765, 1999, doi: 10.1016/S1359-835X(98)00187-0.

[59] C. Eitzinger, F. Christoph, S. Ghidoni, and V. Enrico, "System Concept For Human-Robot Collaborative Draping," in *SAMPE Europe Conference 2021*, 2021, pp. 7542–7549.

[60] P. Boisse, J. Huang, and E. Guzman-Maldonado, “Analysis and modeling of wrinkling in composite forming,” *J. Compos. Sci.*, vol. 5, no. 3, pp. 1–16, 2021, doi: 10.3390/jcs5030081.

[61] C. Ward, S. Hancock, and K. Potter, “Forming Complex Shaped Components Using Drape Simulation Software: Informing Manual and Automated Production Needs,” *SAE Int. J. Aerosp.*, vol. 1, no. 1, pp. 798–810, Sep. 2008, doi: 10.4271/2008-01-2316.

[62] V. Maymard, “A new approach to simulating the draping of prepreg composites manufactured by hand layup,” 2017, Accessed: Feb. 08, 2023. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-209330

[63] S. C. Kularatna, *Developments in Advanced Composites Skills Training and Manufacturing through Virtual Reality and an Artificially Intelligent Layup Agent*. Bristol: University of Bristol, 2020.

[64] R. K. Malhan *et al.*, “Automated planning for robotic layup of composite prepreg,” *Robot. Comput. Integr. Manuf.*, vol. 67, no. July 2020, p. 102020, 2021, doi: 10.1016/j.rcim.2020.102020.

[65] S. M. Haffner, “Cost modeling and design for manufacturing guidelines for advanced composite fabrication,” M.S. thesis, Dept. Mech. Eng., Massachusetts Inst. Technol., Cambridge, MA, USA, 2002.

[66] F. Robitaille, “Introduction to composites (MCG 4190 / MCG 5177),” Lecture Notes, Dept. Mech. Eng., Univ. Ottawa, Ottawa, ON, Canada, 2015.

[67] S. G. Hancock and K. D. Potter, “The use of kinematic drape modelling to inform the hand

lay-up of complex composite components using woven reinforcements," *Compos. Part A Appl. Sci. Manuf.*, vol. 37, no. 3, pp. 413–422, 2006, doi: 10.1016/j.compositesa.2005.05.044.

[68] R. Fitas, S. Hesseler, S. Wist, and C. Greb, "Kinematic draping simulation optimization of a composite B-pillar geometry using particle swarm optimization," *Heliyon*, vol. 8, no. 11, Nov. 2022, doi: 10.1016/J.HELIYON.2022.E11525.

[69] K. Potter and C. Ward, "Draping processes for composites manufacture," *Adv. Compos. Manuf. Process Des.*, pp. 93–109, Jan. 2015, doi: 10.1016/B978-1-78242-307-2.00005-1.

[70] V. Santhanakrishnan Balakrishnan, M. Rao Yellur, J. J. Roesch, L. Ulke-Winter, H. Seidlitz, and M. R. Yellur, "Experimental and numerical investigation on draping behaviour of woven carbon fabric," *J. Reinf. Plast. Compos.* vol. 51, no. 3S, pp. 3575–3592, 2022, doi: 10.1177/15280837211038850.

[71] A. K. Pickett, T. Queckbörner, P. De Luca, and E. Haug, "An explicit finite element solution for the forming prediction of continuous fibre-reinforced thermoplastic sheets," *Compos. Manuf.*, vol. 6, no. 3, pp. 237–243, 1995, doi: 10.1016/0956-7143(95)95016-R.

[72] P. de Luca and Y. Benoit, "Numerical simulation of reinforcements forming: the missing link for the improvement of composite parts virtual prototyping," *Repair. Struct. Using Compos. Wraps*, no. September, pp. 315–321, 2003.

[73] W. R. Yu, P. Harrison, and A. Long, "Finite element forming simulation for non-crimp fabrics using a non-orthogonal constitutive equation," *Compos. Part A Appl. Sci. Manuf.*, vol. 36, no. 8, pp. 1079–1093, Aug. 2005, doi: 10.1016/J.COMPOSITESA.2005.01.007.

[74] J.-L. Daniel, D. Soulat, and P. Boisse, “Shear and tension stiffness influence in composites forming modelling.,” Apr. 2004, Accessed: Feb. 08, 2023. [Online]. Available: https://hal.science/hal-00507347

[75] P. Boisse, B. Zouari, and A. Gasser, “A mesoscopic approach for the simulation of woven fibre composite forming,” *Compos. Sci. Technol.*, vol. 65, no. 3, pp. 429–436, 2005, doi: 10.1016/j.compscitech.2004.09.024.

[76] C. Krogh *et al.*, “A simple MATLAB draping code for fiber-reinforced composites with application to optimization of manufacturing process parameters,” pp. 457–471, 2021.

[77] G. Fantoni *et al.*, “Grasping devices and methods in automated production processes,” *CIRP Ann.*, vol. 63, no. 2, pp. 679–701, Jan. 2014, doi: 10.1016/J.CIRP.2014.05.006.

[78] Maciej Serda *et al.*, “Automated composite draping: a review,” *Uniw. śląski*, vol. 7, no. 1, pp. 343–354, May 2017, doi: 10.2/JQUERY.MIN.JS.

[79] H. H. Hoos and T. Stützle, “Travelling salesman problems,” in *Stochastic Local Search*: *Foundations and Applications.* San Francisco, CA, USA: Elsevier/Morgan Kaufmann, 2005, pp. 357–416, doi: 10.1016/B978-155860872-6/50025-1.

[80] Y. Liu, L. Xu, Y. Han, X. Zeng, G. G. Yen, and H. Ishibuchi, “Evolutionary Multimodal Multiobjective Optimization for Traveling Salesman Problems,” *IEEE Trans. Evol. Comput.*, pp. 1–1, 2023, doi: 10.1109/TEVC.2023.3239546.

[81] G. Gutin and A. Punnen, “The traveling salesman problem,” *Discret. Optim.*, vol. 3, no. 1, p. 1, Mar. 2006, doi: 10.1016/J.DISOPT.2005.12.001.

[82] M. J. D. Powell, “Direct search algorithms for optimization calculations,” pp. 287–336,

1998, doi: 10.1017/S0962492900002841.

[83] Z. B. Zabinsky, “Random search algorithms,” in *Encyclopedia of Optimization*, 2nd ed., C. A. Floudas and P. M. Pardalos, Eds. Boston, MA, USA: Springer, 2009, pp. 3217–3222, doi: 10.1007/978-0-387-74759-0_555.

[84] O. Cheikhrouhou and I. Khoufi, “A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy,” *Comput. Sci. Rev.*, vol. 40, p. 100369, May 2021, doi: 10.1016/J.COSREV.2021.100369.

[85] D. Jungnickel, *Graphs, Networks and Algorithms*, Second Edition., vol. 5. Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/b138283.

[86] R. Fleischer and G. Trippen, “Experimental studies of graph traversal algorithms,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 2647, pp. 120–133, 2003, doi: 10.1007/3-540-44867-5_10/COVER.

[87] D. C. Kozen, “Depth-First and Breadth-First Search,” *Des. Anal. Algorithms*, pp. 19–24, 1992, doi: 10.1007/978-1-4612-4400-4_4.

[88] T. M. Chan, “More algorithms for all-pairs shortest paths in weighted graphs,” SIAM J. Comput., vol. 39, no. 5, pp. 2075–2089, 2010, doi: 10.1137/08071990X.

[89] M. A. Javaid, “Understanding Dijkstra’s Algorithm,” *SSRN Electron. J.*, Apr. 2013, doi: 10.2139/SSRN.2340905.

[90] H. de Fraysseix and P. Rosenstiehl, “A characterization of planar graphs by trémaux orders,” *Combinatorica*, vol. 5, no. 2, pp. 127–135, Jun. 1985, doi: 10.1007/BF02579375.

[91] H. S. Sas, P. Šimáček, and S. G. Advani, “A methodology to reduce variability during

vacuum infusion with optimized design of distribution media," *Compos. Part A Appl. Sci. Manuf.*, vol. 78, pp. 223–233, Nov. 2015, doi: 10.1016/J.COMPOSITESA.2015.08.011.

[92] F. Henriksson and K. Johansen, "On Material Substitution in Automotive BIWs-From Steel to Aluminum Body Sides," *Procedia CIRP*, vol. 50, pp. 683–688, 2016, doi: 10.1016/J.PROCIR.2016.05.028.

[93] G. P. W. Rodrigues, L. H. M. Costa, G. M. Farias, and M. A. H. de Castro, "A Depth-First Search Algorithm for Optimizing the Gravity Pipe Networks Layout," *Water Resour. Manag.*, vol. 33, no. 13, pp. 4583–4598, Oct. 2019, doi: 10.1007/S11269-019-02373-X/METRICS.

[94] V. Trigonakis, J. Lozi, T. Faltín, et al., "aDFS: An almost depth-first-search distributed graph-querying system," in Proc. 2021 USENIX Annu. Tech. Conf. (USENIX ATC 21), Jul. 2021, pp. 209–224.

[95] S. L. Khursan, A. S. Ismagilova, F. T. Ziganshina, and A. I. Akhmet'yanova, "Constructing a Complete Set of Homodesmic Reactions Using the Depth-First Search Procedure," *Russ. J. Phys. Chem. A*, vol. 95, no. 7, pp. 1386–1393, Jul. 2021, doi: 10.1134/S0036024421070141/METRICS.

[96] S. D. Pawar, K. K. Sharma, S. G. Sapate, and G. Y. Yadav, "Segmentation of pectoral muscle from digital mammograms with depth-first search algorithm towards breast density classification," *Biocybern. Biomed. Eng.*, vol. 41, no. 3, pp. 1224–1241, Jul. 2021, doi: 10.1016/J.BBE.2021.08.005.

[97] H. Li, B. Song, X. Tang, Y. Xie, and X. Zhou, "Controller optimization using data-driven constrained bat algorithm with gradient-based depth-first search strategy," *ISA Trans.*, vol.

125, pp. 212–236, Jun. 2022, doi: 10.1016/J.ISATRA.2021.06.032.

[98] R. Tarjan, “Depth-first search and linear graph algorithms,” SIAM J. Comput., vol. 1, no. 2, pp. 146–160, Jun. 1972, doi: 10.1137/0201010

[99] R. Diestel, “Graphs and Their Representation,” *Graph Theory*, vol. 173, no. 1, pp. 1–12, 2017, Accessed: Mar. 31, 2023. [Online]. Available: http://link.springer.com/10.1007/978-3-662-53622-3

[100] J. L. Gross, J. Yellen, and M. Anderson, “Graph Theory and Its Applications,” *Graph Theory Its Appl.*, Nov. 2018, doi: 10.1201/9780429425134.

[101] D. L. Poole and A. K. Mackworth, “Depth-first search,” in *Artificial Intelligence: Foundations of Computational Agents*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2017. [Online]. Available: artint.info (accessed Mar. 15, 2023).

[102] “Graph Theory Tree and Forest - javatpoint.” https://www.javatpoint.com/graph-theory-tree-and-forest (accessed Mar. 31, 2023).

[103] A. Laaksonen, *Guide to Competitive Programming: Learning and Improving Algorithms Through Contests.* Cham, Switzerland: Springer, 2017, doi: 10.1007/978-3-319-72547-5.

[104] J. Sanz-Corretge, “A procedure to design optimum composite plates using implicit decision trees,” *Struct. Multidiscip. Optim.*, vol. 56, no. 5, pp. 1169–1183, Nov. 2017, doi: 10.1007/S00158-017-1711-7/METRICS.

[105] S. Doherty, R. Wehbe, K. Plain, R. Harik, and A. Halbritter, “Off-Part Motion Optimization for an Automated Fiber Placement Machine using Travelling Salesman Problem,” *Comput. Des. Appl.*, vol. 19, no. 2, pp. 220–237, 2022, doi: 10.14733/cadaps.2022.220-237.

[106] S. Sen and A. Kumar, *Design and analysis of algorithms : a contemporary perspective*.

[107] F. Zhang *et al.*, "An adaptive breadth-first search algorithm on integrated architectures," *J. Supercomput.*, vol. 74, no. 11, pp. 6135–6155, Nov. 2018, doi: 10.1007/S11227-018-2525-0/METRICS.

[108] R. Rahim, A. S. Ahmar, A. P. Ardyanti, and D. N. Khalika, "Breadth first search approach for shortest path solution in Cartesian area," J. Phys.: Conf. Ser., vol. 1019, no. 1, Art. no. 012038, Jun. 2018, doi: 10.1088/1742-6596/1019/1/012038.

[109] L. Hao, Y. Wang, Y. Bai, and Q. Zhou, "Energy management strategy on a parallel mild hybrid electric vehicle based on breadth first search algorithm," *Energy Convers. Manage.*, vol. 243, Art. no. 114408, Sep. 2021, doi: 10.1016/j.enconman.2021.114408.

[110] M. Gath-Morad, L. E. Aguilar Melgar, R. Conroy-Dalton, and C. Hölscher, "Beyond the shortest-path: Towards cognitive occupancy modeling in BIM," *Autom. Constr.*, vol. 135, p. 104131, Mar. 2022, doi: 10.1016/J.AUTCON.2022.104131.

[111] Y. Yue and J. Gong, "An efficient implementation of shortest path algorithm based on Dijkstra algorithm," *Geomatics Inf. Sci*. Wuhan Univ., vol. 24, no. 3, pp. 208–212, Mar. 1999. [Online]. Available: http://ch.whu.edu.cn/en/article/id/4401

[112] M. Castellino, M. Rovere, M. I. Shahzad, and A. Tagliaferro, "Conductivity in carbon nanotube polymer composites: A comparison between model and experiment," *Compos. Part A Appl. Sci. Manuf.*, vol. 87, pp. 237–242, Aug. 2016, doi: 10.1016/J.COMPOSITESA.2016.05.002.

[113] A. Aryanfar, S. Medlej, A. Tarhini, and A. R. Tehrani B, "Elliptic percolation model for

predicting the electrical conductivity of graphene–polymer composites," *Soft Matter*, vol. 17, no. 8, pp. 2081–2089, Mar. 2021, doi: 10.1039/D0SM01950J.

[114] W. Shang, X. Liu, X. Wang, and X. Wang, "Fatigue Life Prediction for SiC/Al Materials Based on Path Planning Algorithm Considering Residual Stress," *Chinese J. Mech. Eng. 2023 361*, vol. 36, no. 1, pp. 1–12, Feb. 2023, doi: 10.1186/S10033-023-00843-3.

[115] S. W. G. Abusalim, R. Ibrahim, M. Zainuri Saringat, S. Jamel, and J. Abdul Wahab, "Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimization," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 917, no. 1, p. 012077, Sep. 2020, doi: 10.1088/1757-899X/917/1/012077.

[116] "What is Dijkstra's Algorithm? | Introduction to Dijkstra's Shortest Path Algorithm - GeeksforGeeks." https://www.geeksforgeeks.org/introduction-to-dijkstras-shortest-path-algorithm/ (accessed Mar. 31, 2023).

[117] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[118] M. Evans, "Process Optimisation Through Industrial Experimentation," *Optim. Manuf. Process.*, pp. 3–15, Oct. 2022, doi: 10.1201/9780138744885-2.

[119] B. A. Hassan and T. A. Rashid, "Operational framework for recent advances in backtracking search optimisation algorithm: A systematic review and performance evaluation," *Appl. Math. Comput.*, vol. 370, p. 124919, Apr. 2020, doi: 10.1016/J.AMC.2019.124919.

[120] M. Helbig and A. P. Engelbrecht, "Performance measures for dynamic multi-objective optimisation algorithms," *Inf. Sci. (Ny).*, vol. 250, pp. 61–81, Nov. 2013, doi:

10.1016/J.INS.2013.06.051.

[121] M. Jafferali, J. Venkateshwaran, and Y. J. Son, “Performance comparison of search-based simulation optimisation algorithms for operations scheduling,” *Int. J. Simul. Process Model.*, vol. 1, no. 1–2, pp. 58–71, 2005, doi: 10.1504/IJSPM.2005.007114.

[122] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[123] K. Deb, *Multi-objective optimization using evolutionary algorithms*. Chichester ; John Wiley & Sons, 2004.

[124] X.-S. Yang, *Introduction to mathematical optimization : from linear programming to metaheuristics*. Cambridge, UK: Cambridge International Science Publishing, 2008.

[125] A. Garcés, “A brief introduction to mathematical optimization,” in Mathematical Programming for Power Systems Operation, 1st ed. Hoboken, NJ, USA: Wiley, 2021, ch. 2, pp. 17–37, doi: 10.1002/9781119747291.ch2.

[126] J. Snyman, *Practical Mathematical Optimization An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorithms*, 1st ed. 2005. New York, NY: Springer US, 2005. doi: 10.1007/b105200.

[127] R. Kennedy, “Discrete and continuous optimization for motion estimation,” Ph.D. dissertation, Dept. Elect. Syst. Eng., Univ. Pennsylvania, Philadelphia, PA, USA, 2015.

[128] R. A. Polyak, *Introduction to continuous optimization*. Cham, Switzerland: Springer, 2021. doi: 10.1007/978-3-030-68713-7.

[129] K.-H. Chang, R. Cuckler, and C.-H. Chen, “An Efficient Direct Search Method for

Simulation Optimization With Conditional-Expectation-Based Objectives," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 4, pp. 1–15, 2022, doi: 10.1109/TASE.2021.3135798.

[130] S. Bandyopadhyay and S. K. Pal, *Classification and Learning Using Genetic Algorithms. Applications in Bioinformatics and Web Intelligence*. 2007.

[131] H. Abadlia, N. Smairi, and K. Ghedira, "Comparative performance evaluation of island particle swarm algorithm applied to solve constrained and unconstrained optimization problems," *J. Intell. fuzzy Syst.*, vol. 43, no. 3, pp. 2747–2763, 2022, doi: 10.3233/JIFS-213380.

[132] J. M. Sautto, "Linear and Convex Optimization: A Mathematical Approach," *Investig. operacional*, vol. 43, no. 4, p. 533, 2022.

[133] M. H. Veatch, *Linear and convex optimization : a mathematical approach*. Hoboken, NJ: Wiley, 2021.

[134] D. G. Luenberger and Y. Ye, "Part I Linear Programming," in *Linear and Nonlinear Programming*, vol. 228, Switzerland: Springer International Publishing AG, 2015.

[135] M. P. Moklyachuk, *Convex optimization : introductory course*. London: ISTE, Ltd., 2020.

[136] F. S. Lobato, *Multi-Objective Optimization Problems Concepts and Self-Adaptive Parameters with Mathematical and Engineering Applications*, 1st ed. 2017. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-58565-9.

[137] N. Saini and S. Saha, "Multi-objective optimization techniques: a survey of the state-of-the-art and applications," *Eur. Phys. journal. ST, Spec. Top.*, vol. 230, no. 10, pp. 2319–2335, 2021, doi: 10.1140/epjs/s11734-021-00206-w.

[138] P. M. Pardalos, *Non-Convex Multi-Objective Optimization*, 1st ed. 2017. Cham: Springer International Publishing, 2017. doi: 10.1007/978-3-319-61007-8.

[139] S. Bandyopadhyay and S. Saha, *Unsupervised classification: Similarity measures, classical and metaheuristic approaches, and applications*, vol. 9783642324. 2013. doi: 10.1007/978-3-642-32451-2.

[140] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003, doi: 10.1145/937503.937505.

[141] E.-G. Talbi, *Metaheuristics: From Design to Implementation*. Hoboken, NJ, USA: John Wiley & Sons, 2009.

[142] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, 2nd ed. Berlin, Germany: Springer-Verlag, 2015.

[143] M. R. Hasanzadeh and F. Keynia, "A new population initialisation method based on the Pareto 80/20 rule for meta-heuristic optimisation algorithms," *IET Softw.*, vol. 15, no. 5, pp. 323–347, Oct. 2021, doi: 10.1049/sfw2.12025.

[144] M. J. H. Heule and O. Kullmann, "The science of brute force," *Commun. ACM*, vol. 60, no. 8, pp. 70–79, 2017, doi: 10.1145/3107239.

[145] B. Konev and A. Lisitsa, "A SAT Attack on the Erdős Discrepancy Conjecture," in *Theory and Applications of Satisfiability Testing – SAT 2014*, vol. 8561, pp. 219–226. doi: 10.1007/978-3-319-09284-3_17.

[146] R. L. Graham, J. Nešetřil, and S. Butler, *The mathematics of Paul Erdős I*, Second edition.

New York, NY: Springer, 2013. doi: 10.1007/978-1-4614-7258-2.

[147] E. Lamb, “Maths proof smashes size record,” *Nature*, vol. 534, p. 18, 2016.

[148] M. J. H. Heule, O. Kullmann, and V. W. Marek, “Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9710, pp. 228–245. doi: 10.1007/978-3-319-40970-2_15.

[149] A. Lew and H. Mauch, *Dynamic Programming: A Computational Tool*. Berlin, Germany: Springer-Verlag, 2007.

[150] H. de Harder, “Choosing Fast with Dynamic Programming | by Hennie de Harder | Towards Data Science,” *Towards Data Science*, 2020. https://towardsdatascience.com/choosing-fast-with-dynamic-programming-b6916da543f4 (accessed Apr. 13, 2022).

[151] O. A. Alzubi, J. A. Alzubi, M. Alweshah, I. Qiqieh, S. Al-Shami, and M. Ramachandran, “An optimal pruning algorithm of classifier ensembles: dynamic programming approach,” *Neural Comput. Appl.*, vol. 32, no. 20, pp. 16091–16107, 2020, doi: 10.1007/s00521-020-04761-6.

[152] Z. kai Feng, W. jing Niu, C. tian Cheng, and X. yu Wu, “Optimization of large-scale hydropower system peak operation with hybrid dynamic programming and domain knowledge,” *J. Clean. Prod.*, vol. 171, pp. 390–402, 2018, doi: 10.1016/j.jclepro.2017.09.257.

[153] S. Dian *et al.*, “Modeling and Trajectory Tracking Control for Magnetic Wheeled Mobile Robots Based on Improved Dual-Heuristic Dynamic Programming,” *IEEE Trans. Ind.*

Informatics, vol. 17, no. 2, pp. 1470–1482, 2021, doi: 10.1109/TII.2020.2983841.

[154] "Chapter 3 Dynamic Programming," *Math. Sci. Eng.*, vol. 88, no. C, pp. 12–21, Jan. 1972, doi: 10.1016/S0076-5392(08)62713-7.

[155] N. Xu *et al.*, "Global optimization energy management for multi-energy source vehicles based on 'Information layer - Physical layer - Energy layer - Dynamic programming' (IPE-DP)," *Appl. Energy*, vol. 312, p. 118668, Apr. 2022, doi: 10.1016/J.APENERGY.2022.118668.

[156] D. Zhu, E. Pritchard, S. R. Dadam, V. Kumar, and Y. Xu, "Optimization of rule-based energy management strategies for hybrid vehicles using dynamic programming," *Combust. Engines*, vol. 184, no. 1, pp. 3–10, Jul. 2022, doi: 10.19206/CE-131967.

[157] J. Xu *et al.*, "Analysis and Application of Dynamic Programming," vol. 1865, no. 4, Apr. 2021, doi: 10.1088/1742-6596/1865/4/042023.

[158] S. Chakkour, "A survey on dynamic programming algorithms: Theory, applications, and advancements," Dept. Comput. Sci., Mississippi State Univ., Mississippi State, MS, USA, May 2025. [Online]. Available: researchgate.net.

[159] D. Rani and D. K. Srivastava, "Optimal operation of Mula reservoir with combined use of dynamic programming and genetic algorithm," *Sustain. Water Resour. Manag.*, vol. 2, no. 1, pp. 1–12, Mar. 2016, doi: 10.1007/S40899-015-0036-1/TABLES/2.

[160] V. Cool, N. Aage, and O. Sigmund, “A practical review on promoting connectivity in topology optimization: A practical review on promoting connectivity in topology optimization,” *Struct. Multidiscip. Optim.*, vol. 68, no. 4, p. 73, 2025, doi: 10.1007/s00158-025-04004-z.

[161] S. Sen and A. Kumar, *Optimization I: Brute Force and Greedy Strategy*. 2019. doi: 10.1017/9781108654937.005.

[162] R. Bai, B. Chen, J. Colmars, and P. Boisse, “Physics-based evaluation of the drapability of textile composite reinforcements,” *Compos. Part B Eng.*, vol. 242, p. 110089, Aug. 2022, doi: 10.1016/J.COMPOSITESB.2022.110089.

[163] J. Huang, P. Boisse, N. Hamila, I. Gnaba, D. Soulat, and P. Wang, “Experimental and numerical analysis of textile composite draping on a square box. Influence of the weave pattern,” *Compos. Struct.*, vol. 267, p. 113844, Jul. 2021, doi: 10.1016/J.COMPSTRUCT.2021.113844.

[164] “Standard Test Method for Determining the Flexural Stiffness of Medical Textiles.” https://www.astm.org/f3260-18.html (accessed Apr. 18, 2023).

[165] H. P. Gavin, “Strain energy in linear elastic solids,” Lecture Notes, Dept. Civil Environ. Eng., Duke Univ., Durham, NC, USA, 2021.

[166] R. Hulse and J. A. Cain, *Solid Mechanics*, 2nd ed. Basingstoke, U.K.: Palgrave Macmillan, 2003.

[167] P. Kanz and F. Robitaille, “Concave and convex small radius bending behavior of single and multiple layers reinforcement fabrics,” *Text. Res. J.*, vol. 94, no. 19–20, pp. 2284–2295, Oct. 2024, doi: 10.1177/00405175241246554.

[168] S. S. Skiena, *The Algorithm Design Manual*, 3rd ed. Cham, Switzerland: Springer, 2020.

[169] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.

Appendix A. Geometric Parameterization

As explained in Section 1.2, the geometric parameters are extracted through a MSVB™ Macro software that interacts with SolidWorks™, delivering the following data as per Table A.1, for the demonstrator 01_Francois, shown in Figure A.1 .

Table A.1 Geometric data extracted by macro for demonstrator 01_Francois

Base Surface #	Edge #	Edge length (mm)	Point #	*x* (mm)	*y* (mm)	*z* (mm)
1	1	858.01	1	7000	0	0
			2	7428.07	23.92	0
			3	7850	100	0
	2	2915.48	4	7000	0	0
			5	7000	750	1250
			6	7000	1500	2500
	3	858.79	7	7000	1500	2500
			8	7428.6	1519.53	2500
			9	7850	1600	2500
	4	2915.48	10	7850	100	0
			11	7850	850	1250
			12	7850	1600	2500
2	5	2915.48	13	7000	0	0
			14	7000	750	1250
			15	7000	1500	2500
	6	7000	16	0	0	0
			17	3500	0	0
			18	7000	0	0
	7	2915.48	19	0	0	0
			20	0	750	1250
			21	0	1500	2500
	8	7000	22	0	1500	2500
			23	3500	1500	2500
			24	7000	1500	2500
3	9	2915.48	25	12000	2250	0
			26	12000	3000	1250

			27	12000	3750	2500
	10	4673.86	28	7850	100	0
			29	9925	1175	0
			30	12000	2250	0
	11	2915.48	31	7850	100	0
			32	7850	850	1250
			33	7850	1600	2500
	12	4673.86	34	7850	1600	2500
			35	9925	2675	2500
			36	12000	3750	2500
4	13	858.79	37	7000	1500	2500
			38	7428.6	1519.53	2500
			39	7850	1600	2500
	14	2617.99	40	7000	1500	2500
			41	6999.99	2298.99	3518.01
			42	7000	3500	4000
	15	2694.12	43	7850	1600	2500
			44	7645.09	2445.4	3507.21
			45	7000	3500	4000
5	16	2617.99	46	7000	1500	2500
			47	6999.99	2298.99	3518.01
			48	7000	3500	4000
	17	7000	49	0	1500	2500
			50	3500	1500	2500
			51	7000	1500	2500
	18	2617.99	52	0	1500	2500
			53	0	2299.04	3517.95
			54	0	3500	4000
	19	7000	55	0	3500	4000
			56	3500	3500	4000
			57	7000	3500	4000
6	20	1035.92	58	12000	2250	0
			59	12489.18	2418.27	0
			60	13000	2500	0
	21	2915.48	61	12000	2250	0
			62	12000	3000	1250
			63	12000	3750	2500
	22	1035.92	64	12000	3750	2500
			65	12489.18	3918.27	2500
			66	13000	4000	2500
	23	2915.48	67	13000	2500	0
			68	13000	3250	1250
			69	13000	4000	2500

7	24	2333.83	70	12000	3750	2500
			71	11826.7	4436.97	3413.89
			72	11250	5250	4000
	25	4673.86	73	7850	1600	2500
			74	9925	2675	2500
			75	12000	3750	2500
	26	2694.12	76	7850	1600	2500
			77	7645.09	2445.4	3507.21
			78	7000	3500	4000
	27	4596.19	79	7000	3500	4000
			80	9125	4375	4000
			81	11250	5250	4000
8	28	7000	82	0	3500	4000
			83	3500	3500	4000
			84	7000	3500	4000
	29	7000	85	0	3500	4000
			86	0	7000	4000
			87	0	10500	4000
	30	12414.71	88	0	10500	4000
			89	5625	7875	4000
			90	11250	5250	4000
	31	4596.19	91	7000	3500	4000
			92	9125	4375	4000
			93	11250	5250	4000
9	32	7000	94	13000	2500	0
			95	16500	2500	0
			96	20000	2500	0
	33	2915.48	97	13000	2500	0
			98	13000	3250	1250
			99	13000	4000	2500
	34	7000	100	13000	4000	2500
			101	16500	4000	2500
			102	20000	4000	2500
	35	2915.48	103	20000	2500	0
			104	20000	3250	1250
			105	20000	4000	2500
10	36	2190.74	106	13000	4000	2500
			107	13000	4582.98	3417.02
			108	13000	5500	4000
	37	1035.92	109	12000	3750	2500
			110	12489.18	3918.27	2500
			111	13000	4000	2500
	38	2333.83	112	12000	3750	2500

			113	11826.7	4436.97	3413.89
			114	11250	5250	4000
	39	1787.12	115	11250	5250	4000
			116	12108.96	5487.31	4000
			117	13000	5500	4000
11	40	1787.12	118	11250	5250	4000
			119	12108.96	5487.31	4000
			120	13000	5500	4000
	41	12414.71	121	0	10500	4000
			122	5625	7875	4000
			123	11250	5250	4000
	42	13928.39	124	0	10500	4000
			125	6500	8000	4000
			126	13000	5500	4000
12	43	2190.75	127	20000	4000	2500
			128	20000	4583.01	3416.99
			129	20000	5500	4000
	44	7000	130	13000	4000	2500
			131	16500	4000	2500
			132	20000	4000	2500
	45	2190.74	133	13000	4000	2500
			134	13000	4582.98	3417.02
			135	13000	5500	4000
	46	7000	136	13000	5500	4000
			137	16500	5500	4000
			138	20000	5500	4000
13	47	7000	139	13000	5500	4000
			140	16500	5500	4000
			141	20000	5500	4000
	48	13928.39	142	0	10500	4000
			143	6500	8000	4000
			144	13000	5500	4000
	49	20000	145	0	10500	4000
			146	10000	10500	4000
			147	20000	10500	4000
	50	5000	148	20000	5500	4000
			149	20000	8000	4000
			150	20000	10500	4000
14	51	2617.99	151	20000	10500	4000
			152	20000	11700.96	3517.95
			153	20000	12500	2500
	52	20000	154	0	10500	4000
			155	10000	10500	4000

			156	20000	10500	4000
	53	2617.99	157	0	10500	4000
			158	0	11700.96	3517.95
			159	0	12500	2500
	54	20000	160	0	12500	2500
			161	10000	12500	2500
			162	20000	12500	2500
15	55	2915.48	163	20000	12500	2500
			164	20000	13250	1250
			165	20000	14000	0
	56	20000	166	0	12500	2500
			167	10000	12500	2500
			168	20000	12500	2500
	57	2915.48	169	0	12500	2500
			170	0	13250	1250
			171	0	14000	0
	58	20000	172	0	14000	0
			173	10000	14000	0
			174	20000	14000	0

Figure A1 shows the BaS labeling directly from the STEP file extracted data.

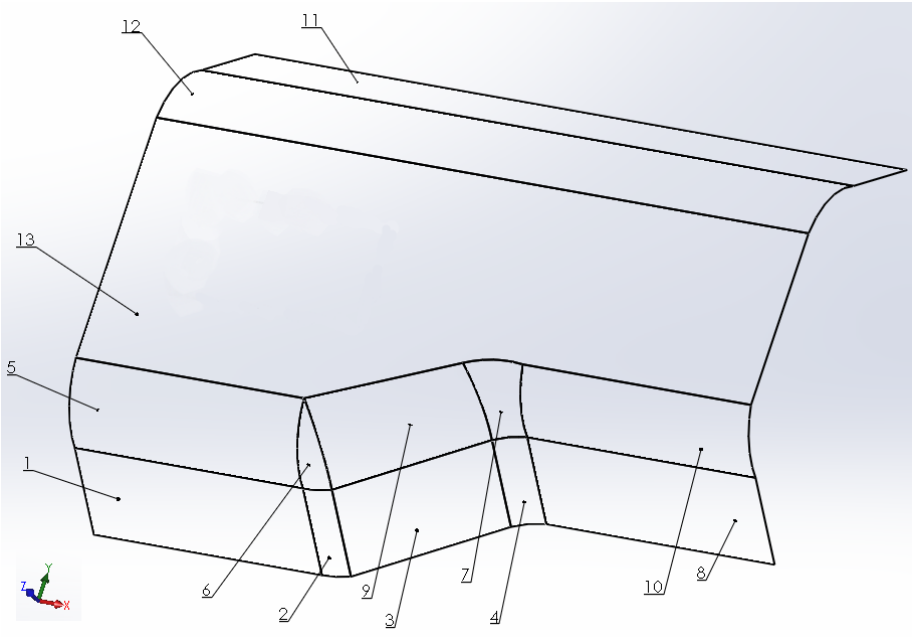


Figure A.1 As is Macro labelled BaS numbers

Table A.1 serves as a preliminary list characterized by repetitive entries and overlapping labels. By applying consistent indexing framework MinMax, Table A.2 eliminates these redundancies, ensuring that data structure and nomenclature convey specific information.

Table A.2 MinMax geometric information for demonstrator 01_Francois

Base Surface #	Edge #	Edge length (mm)	Point #	*x* (mm)	*y* (mm)	*z* (mm)
1	1	7000	1	0	0	0
			8	3500	0	0
			7	7000	0	0
	2	2915.48	7	7000	0	0
			6	7000	750	1250
			5	7000	1500	2500
	3	7000	5	7000	1500	2500
			4	3500	1500	2500
			3	0	1500	2500
	4	2915.48	3	0	1500	2500
			2	0	750	1250
			1	0	0	0
2	3	7000	3	0	1500	2500
			4	3500	1500	2500
			5	7000	1500	2500
	5	2617.99	5	7000	1500	2500
			13	7000	2299	3518
			12	7000	3500	4000
	6	7000	12	7000	3500	4000
			11	3500	3500	4000
			10	0	3500	4000
	7	2617.99	10	0	3500	4000
			9	0	2299	3518
			3	0	1500	2500
3	6	7000	10	0	3500	4000
			11	3500	3500	4000
			12	7000	3500	4000
	8	4596.19	12	7000	3500	4000
			18	9125	4375	4000
			17	11250	5250	4000

	9	12414.71	17	11250	5250	4000
			16	5625	7875	4000
			15	0	10500	4000
	10	7000	15	0	10500	4000
			14	0	7000	4000
			10	0	3500	4000
4	11	1787.12	17	11250	5250	4000
			21	12109	5487.3	4000
			20	13000	5500	4000
	12	13928.39	20	13000	5500	4000
			19	6500	8000	4000
			15	0	10500	4000
	9	12414.71	15	0	10500	4000
			16	5625	7875	4000
			17	11250	5250	4000
5	13	7000	20	13000	5500	4000
			26	16500	5500	4000
			25	20000	5500	4000
	14	5000	25	20000	5500	4000
			24	20000	8000	4000
			23	20000	10500	4000
	15	20000	23	20000	10500	4000
			22	10000	10500	4000
			15	0	10500	4000
	12	13928.39	15	0	10500	4000
			19	6500	8000	4000
			20	13000	5500	4000
6	15	20000	15	0	10500	4000
			22	10000	10500	4000
			23	20000	10500	4000
	16	2617.99	23	20000	10500	4000
			31	20000	11701	3518
			30	20000	12500	2500
	17	20000	30	20000	12500	2500
			29	10000	12500	2500
			28	0	12500	2500
	18	2617.99	28	0	12500	2500
			27	0	11701	3518
			15	0	10500	4000
7	17	20000	28	0	12500	2500
			29	10000	12500	2500
			30	20000	12500	2500
	19	2915.48	30	20000	12500	2500

			36	20000	13250	1250
			35	20000	14000	0
	20	20000	35	20000	14000	0
			34	10000	14000	0
			33	0	14000	0
	21	2915.48	33	0	14000	0
			32	0	13250	1250
			28	0	12500	2500
8	22	858.01	7	7000	0	0
			41	7428.1	23.9	0
			40	7850	100	0
	23	2915.48	40	7850	100	0
			39	7850	850	1250
			38	7850	1600	2500
	24	858.79	38	7850	1600	2500
			37	7428.6	1519.5	2500
			5	7000	1500	2500
	2	2915.48	5	7000	1500	2500
			6	7000	750	1250
			7	7000	0	0
9	24	858.79	5	7000	1500	2500
			37	7428.6	1519.5	2500
			38	7850	1600	2500
	25	2694.12	38	7850	1600	2500
			42	7645.1	2445.4	3507.2
			12	7000	3500	4000
	5	2617.99	12	7000	3500	4000
			13	7000	2299	3518
			5	7000	1500	2500
10	26	4673.86	38	7850	1600	2500
			45	9925	2675	2500
			44	12000	3750	2500
	27	2333.83	44	12000	3750	2500
			43	11826.7	4437	3413.9
			17	11250	5250	4000
	8	4596.19	17	11250	5250	4000
			18	9125	4375	4000
			12	7000	3500	4000
	25	2694.12	12	7000	3500	4000
			42	7645.1	2445.4	3507.2
			38	7850	1600	2500
11	28	4673.86	40	7850	100	0
			48	9925	1175	0

			47	12000	2250	0
	29	2915.48	47	12000	2250	0
			46	12000	3000	1250
			44	12000	3750	2500
	26	4673.86	44	12000	3750	2500
			45	9925	2675	2500
			38	7850	1600	2500
	23	2915.48	38	7850	1600	2500
			39	7850	850	1250
			40	7850	100	0
12	30	1035.92	44	12000	3750	2500
			51	12489.2	3918.3	2500
			50	13000	4000	2500
	31	2190.74	50	13000	4000	2500
			49	13000	4583	3417
			20	13000	5500	4000
	11	1787.12	20	13000	5500	4000
			21	12109	5487.3	4000
			17	11250	5250	4000
	27	2333.83	17	11250	5250	4000
			43	11826.7	4437	3413.9
			44	12000	3750	2500
13	32	1035.92	47	12000	2250	0
			54	12489.2	2418.3	0
			53	13000	2500	0
	33	2915.48	53	13000	2500	0
			52	13000	3250	1250
			50	13000	4000	2500
	30	1035.92	50	13000	4000	2500
			51	12489.2	3918.3	2500
			44	12000	3750	2500
	29	2915.48	44	12000	3750	2500
			46	12000	3000	1250
			47	12000	2250	0
14	34	7000	53	13000	2500	0
			59	16500	2500	0
			58	20000	2500	0
	35	2915.48	58	20000	2500	0
			57	20000	3250	1250
			56	20000	4000	2500
	36	7000	56	20000	4000	2500
			55	16500	4000	2500
			50	13000	4000	2500

	33	2915.48	50	13000	4000	2500
			52	13000	3250	1250
			53	13000	2500	0
15	36	7000	50	13000	4000	2500
			55	16500	4000	2500
			56	20000	4000	2500
	37	2190.75	56	20000	4000	2500
			60	20000	4583	3417
			25	20000	5500	4000
	13	7000	25	20000	5500	4000
			26	16500	5500	4000
			20	13000	5500	4000
	31	2190.74	20	13000	5500	4000
			49	13000	4583	3417
			50	13000	4000	2500

Figure A.2 shows labelled demonstrator 01_Francois, as processed by MinMax. Small grey numbers are the vertex and middle edge point numbers, and black large numbers identify the BaS.

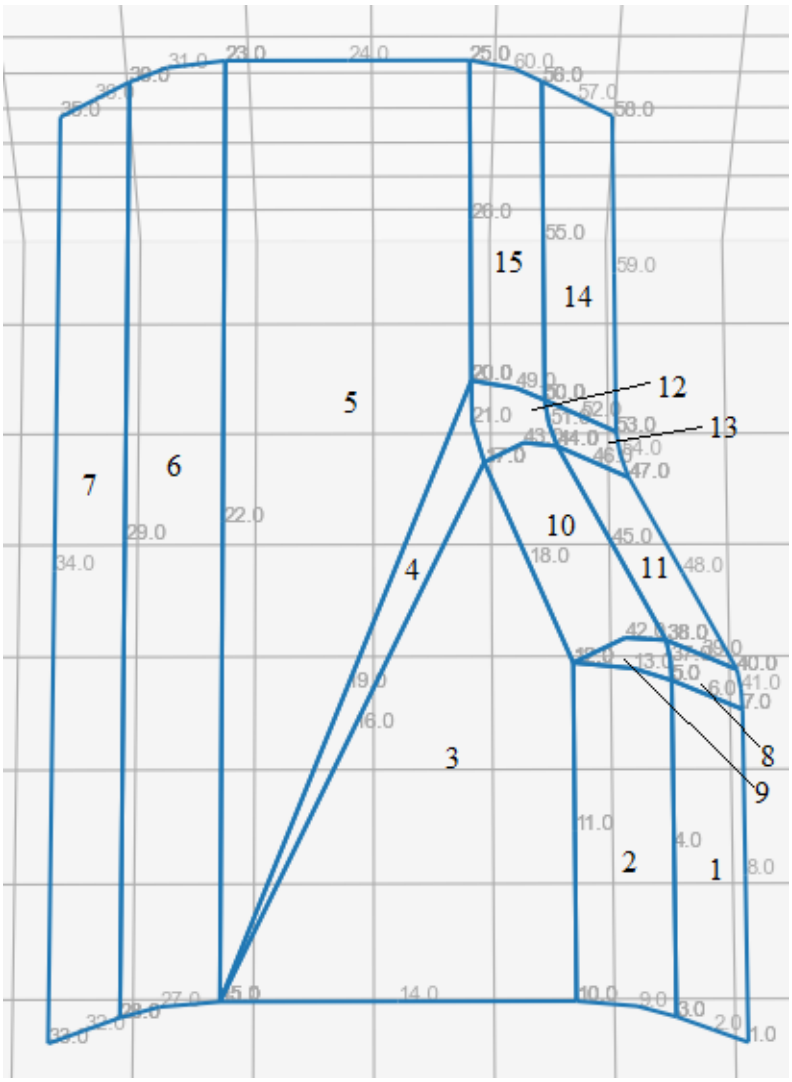


Figure A.2 Demonstrator 01_Francois relabelled by MinMax - BaS and Vertices

Appendix B. Dynamic Programming Results

This Appendix B presents results, validation and performance analysis of the 1-next state and the 2-next states DP algorithms. Evaluation presents execution time, predicted yarn orientations, wrinkle warnings and overall draping behaviour together with validation photographs for runs performed on academic demonstrators.

Small differences in resulting draping sequences may be observed between runs. Such differences can arise when two or more candidate draping steps yield identical Objective Function values. In such cases, the selected draping step depends on the listing sequence produced by the *Possible_Draping_Steps* function or on the order in which values are stored in the memoization matrix. The resulting variations reflect implementation sequencing effects rather than differences in underlying physical behaviour or modelling assumptions.

B.1 DP 1-Next State - Run 1

Testing Run 1 for 10-BaS Demonstrator 31_Benchmark, initial orientation 0°/90°, 1-next state was conducted as per Table 6.2 from Section 6.10, setting mould type as a mostly convex. In total, 372 partial draping sequences were evaluated hence 372 sub-spaces were optimised. All BaS were tested as starting BaS. Total computing time was 3 min 6 s .

As observed in Table B.1, the best draping sequences start from BaS #1 and BaS #4 and are essentially symmetric. Figure B.1 shows optimal sequence [1-5, 5-10, 10-9, 9-4, 1-2, 5-6, 6-7,

7-8, 8-3]. Figure B.2 shows resulting warp and weft orientations for this optimised draping sequence, represented by green and orange vectors respectively.

One wrinkle warning was issued by the optimisation algorithm, when draping BaS #3 entering from BaS #8. Wrinkle warnings are indicated by the letter W appearing in the sequences, at the end of specific steps for which they were issued. Similarly, darting recommendations are indicated by the letter D.

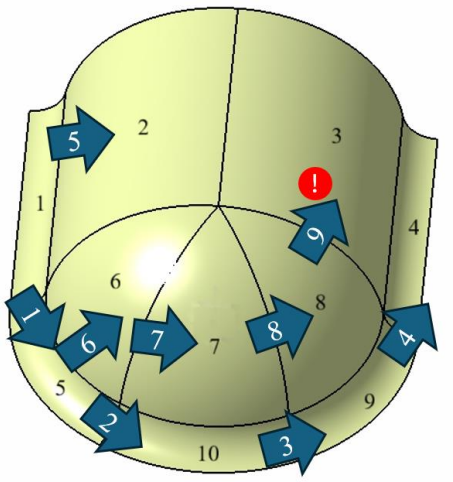


Figure B.1 DP 1-next state - draping sequence for optimised Run 1

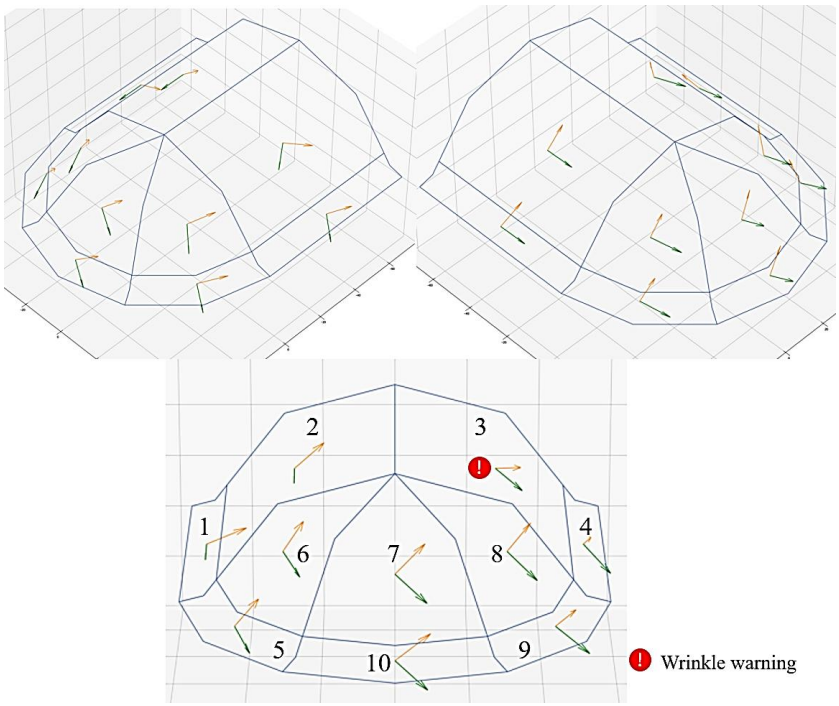


Figure B.2 DP 1-next state - resulting orientations for optimised Run 1

A review of the algorithm operation record showed that for the final draping step when BaS #3 was the only non-draped BaS, the optimisation algorithm evaluated possible draping steps [4-3], [8-3], and [2-3], all resulting in identical Objective Function $f(C, Q)$ values.

Table B.1 DP 1-next state - Run 1 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Wrinkle warnings	Optimised draping sequence:
31_Benchmark 1-next state Run 1	1	11.27	1	[1-5, 5-10, 10-9, 9-4, 1-2, 5-6, 6-7, 7-8, 8-3(W, D)]
	4	11.27	1	[4-9, 9-10, 10-5, 5-1, 1-2, 5-6, 6-7, 7-8, 8-3(W, D)]

In this run, the optimised draping sequence progressively deferred in-plane shear to the latter stages of the process, resulting in wrinkle warnings only during the final step, as demonstrated in the experimental validation shown in Figure B.3. Consequently, a darting recommendation was issued for the peripheral BaS #3.

Figure B.3 DP 1-next state - validation for optimised Run 1

Deferring in-plane shear to the final stages of the sequence limits its propagation to subsequent segments, possibly reducing the likelihood of inducing wrinkles in latter steps. This potentially minimizes operator effort and decreases variability in the quality of the final draped geometry.

Figure B.3 provides experimental validation for this sequence. Although bending and in-plane shear spring-back were characterized in the laboratory and included in Database B, predicting post-draping spring-back is outside the scope of this work. Even prior to darting, these validation results illustrate the necessity of wrinkle warnings, as the physical deformations observed match the high-strain regions identified by the algorithm.

B.2 DP 1-Next State - Run 2

This run was presented in Section 6.11.3 .

B.3 DP 1-Next State - Run 3

This section presents results for DP 1-next state optimisation algorithm applied to 10 BaS demonstrator 32_Variant, for Run 3 with initial orientation 0°/90°. All BaS were tested as starting BaS. 366 partial sequences were evaluated and optimised in 3 min 29 s . Optimisation results appear in Table B.2 as best draping sequences.

Table B.2 DP 1-next state - Run 3 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Wrinkle warnings	Optimised draping sequence:
32_Variation 1-next state Run 3	1	13.93	0	[1-5, 5-10, 10-9, 9-4, 1-2, 5-6, 6-7, 7-8, 8-3]
	4	13.93	0	[4-9, 9-10, 10-5, 5-1, 1-2, 5-6, 6-7, 7-8, 8-3]

Several wrinkle warnings were flagged during the evaluation of draping steps; however, alternative HLU draping paths were consistently available, preventing the selection of high in-plane shear steps.

The optimal draping sequence for Run 3 starting from BaS #1 is observed in Figure B.4, with corresponding warp and weft vector orientations shown in Figure B.5. The highest in-plane shear angles corresponded to BaS #7, #8, and #3 as observed.

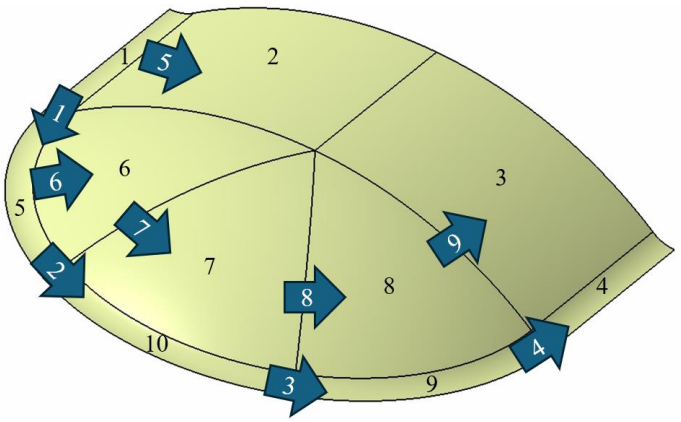


Figure B.4 DP 1-next state - draping sequence for optimised Run 3

Optimised values obtained for demonstrator 32_Variation in Run 3 were larger than those for demonstrator 31_Benchmark in Run 1 due to the surface area being draped being 51% larger. Figure B.5 shows the resulting yarn orientations for the optimised draping sequence. Differences between warp and weft orientations were observed between neighbouring BaS, notably between BaS #3 and its previously draped neighbours BaS #2 and BaS #4. As in previous runs, higher in-plane shear was deferred to the final stages of the draping sequence.

Figure B.6 shows experimental validation of the optimised draping sequence for Run 3, where resulting warp and weft orientations can be compared with those predicted by the optimisation software. It is important to note that all optimisation runs were evaluated using the maximum in-plane shear angle calculated for each BaS edge, representing a conservative, worst-case scenario for warp and weft orientations.

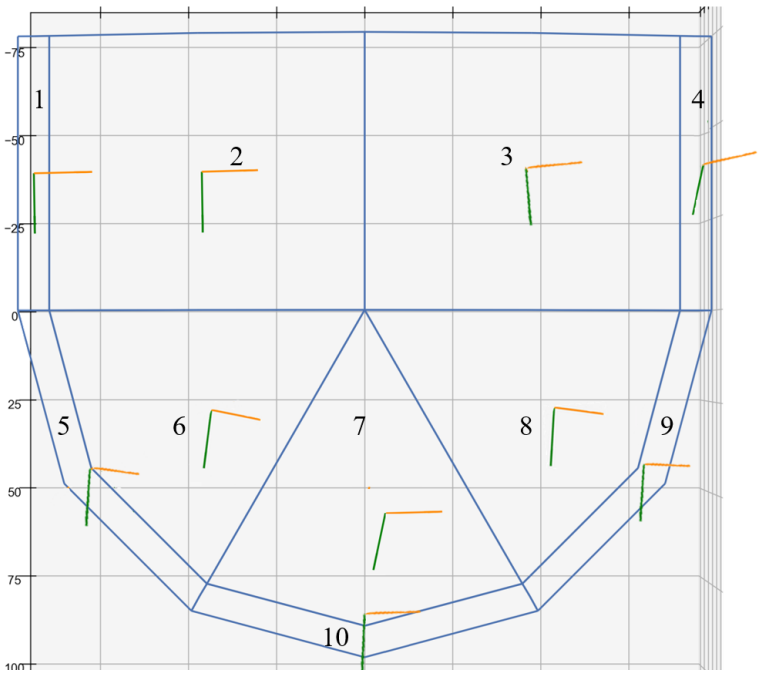


Figure B.5 DP 1-next state resulting orientations for optimised Run 3

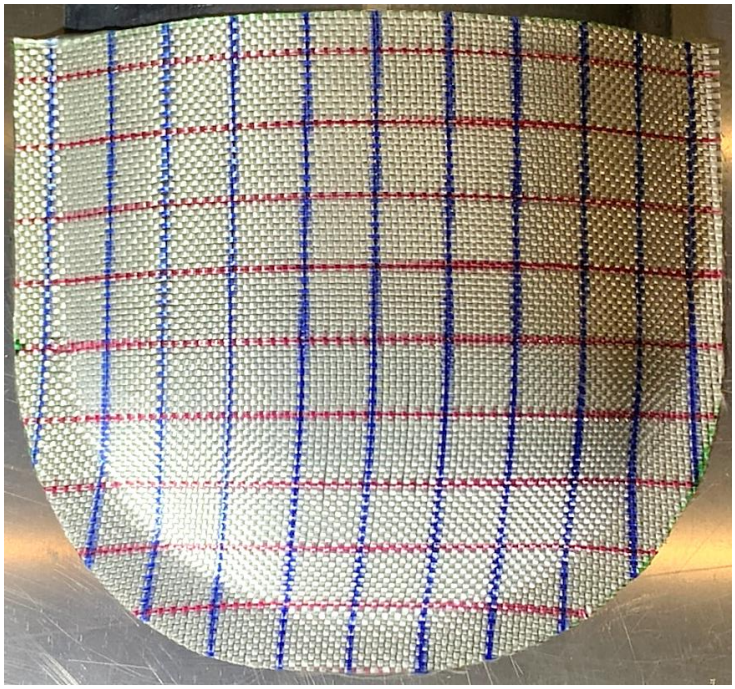


Figure B.6 DP 1-next state - validation for optimised Run 3

B.4 DP 1-Next State - Run 4

This section presents results for DP 1-next state optimisation algorithm applied to 10 BaS demonstrator 32_Variant for Run 4, initial orientation +/- 45°. All BaS were tested as starting BaS. 370 partial draping sequences were evaluated and optimised in 3 min 11 s . Results are summarized in Table B.3, the optimal draping sequence is illustrated in Figure B.7 and the corresponding yarn orientations are illustrated in Figure B.8.

No wrinkle warnings were issued when draping this demonstrator with initial yarn orientations +/- 45°. Comparing Run 3 and Run 4, the optimised values are the same when rounded to two decimal places. Therefore, no advantage in reducing in-plane shear was observed when changing the initial fabric orientation from 0°/90° to +/- 45°.

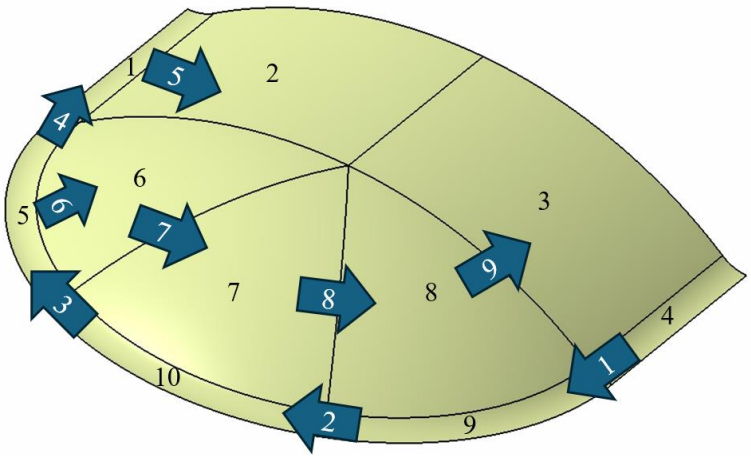


Figure B.7 DP 1-next state - draping sequence for optimised Run 4

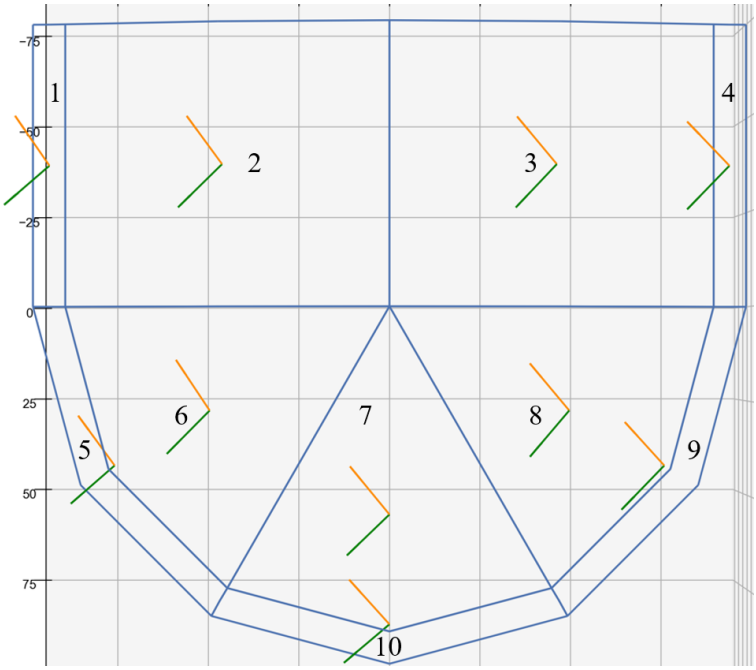


Figure B.8 DP 1-next state resulting orientations for optimised Run 4

Table B.3 DP 1-next state - Run 4 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Wrinkle warnings	Optimised draping sequence:
32_Variation 1-next state Run 4	1	13.93	0	[1-5, 5-10, 10-9, 9-4, 1-2, 5-6, 6-7, 7-8, 8-3]
	4	13.93	0	[4-9, 9-10, 10-5, 5-1, 1-2, 5-6, 6-7, 7-8, 8-3]

This outcome indicates that the 1-next-state DP approach converged to the same draping strategy under both initial orientations. Reduced in-plane shear reflects the influence of gentler curvature.

Figure B.9 shows experimental validation of the optimised draping sequence for Run 4. No significant differences were observed in optimised values or optimised draping sequences between Run 3 and Run 4. Therefore, ply orientations are not critical in this case.

Figure B.9 DP 1-next state - validation for optimised Run 4

B.5 DP 1-Next State - Run 5

This section presents results for DP 1-next state optimisation algorithm applied to 15-BaS demonstrator 01_Francois for Run 5, initial orientation 0°/90°. All BaS were tested as starting BaS. 1003 partial draping sequences were evaluated and optimised in 8 min 31 s . Results are summarized in Table B.4, the optimal draping sequence is illustrated in Figure B.10 and the corresponding yarn orientations are illustrated in Figure B.11 .

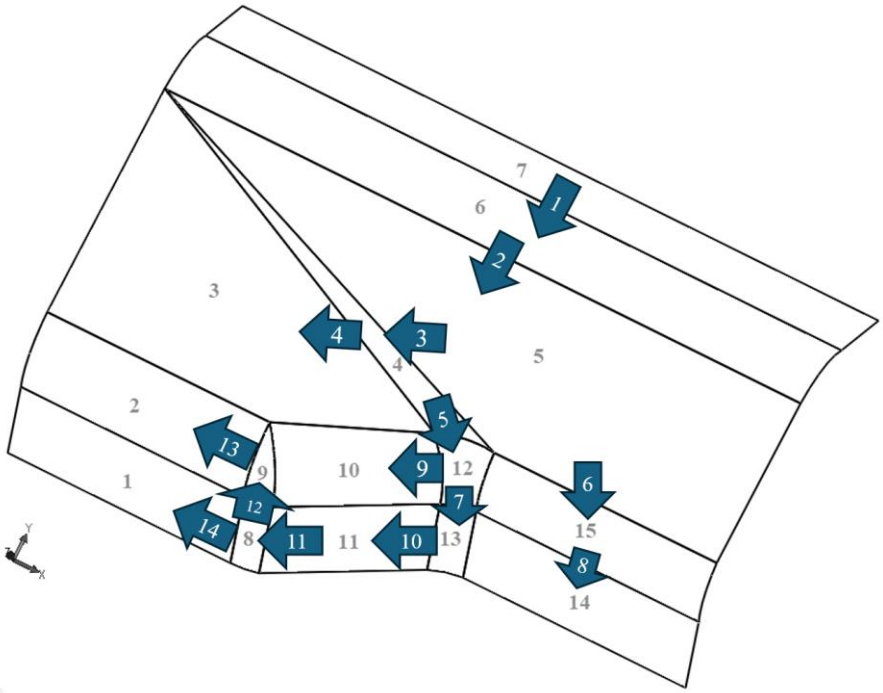


Figure B.10 DP 1-next state - draping sequence for optimised Run 5

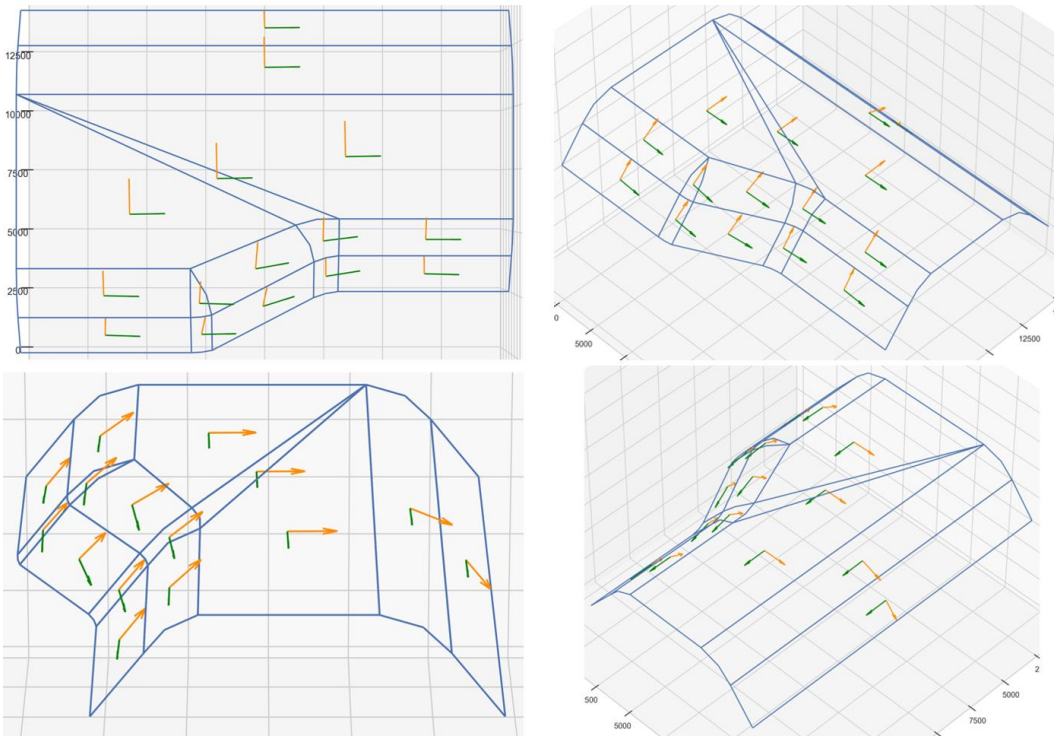


Figure B.11 DP 1-next state resulting orientations for optimised Run 5

Table B.4 DP 1-next state - Run 5 optimisation results

Demonstrator & conditions	StartingBaS	*f(C&Q)*	Optimised draping sequence:
01_Francois 1-next state Run 5	7	4.22	[7-6, 6-5, 5-4, 4-3, 4-12, 12-13, 12-10, 10-9, 10-11, 5-15, 15-14, 11-8, 9-2, 8-1]

In this run, in-plane shear was observed in double-curvature BaS #9 and #12 as shown in Figure B.11. These BaS were draped during the latter stages of the draping sequence, limiting further dispersion of in-plane shear to neighbouring regions.

Consistent with this behaviour, no wrinkle warnings were issued for the optimised draping sequence starting from BaS #7. This outcome was expected given the geometric characteristics of the double-curvature BaS in the 01_Francois demonstrator, where shear can be accommodated more effectively. Low in-plane shear was registered overall, with BaS #9, BaS #12 and BaS #13 exhibiting the highest shear values of up to 7°.

Experimental validation and a non-optimised draping sequence demonstration can be consulted in Section 6.11.4 .

B.6 DP 1-Next State - Run 6

This section presents results for DP 1-next state optimisation algorithm applied to 15-BaS demonstrator 01_Francois for Run 6, initial orientation +/- 45°. All BaS were tested as starting BaS. 914 partial draping sequences were evaluated and optimised in 5 min 7 s . Results are

summarized in Table B.5, the optimal draping sequence is illustrated in Figure B.12 and corresponding yarn orientations are illustrated in Figure B.13.

Table B.5 DP 1-next state - Run 6 optimisation results

Demonstrator & conditions	**StartingBaS**	***f(C&Q)***	**Optimised draping sequence:**
01_Francois 1-next state Run 6	7	5.03	[7-6, 6-5, 5-4, 4-3, 4-12, 12-13, 12-10, 10-9, 10-11, 5-15, 9-8, 15-14, 9-2, 8-1]

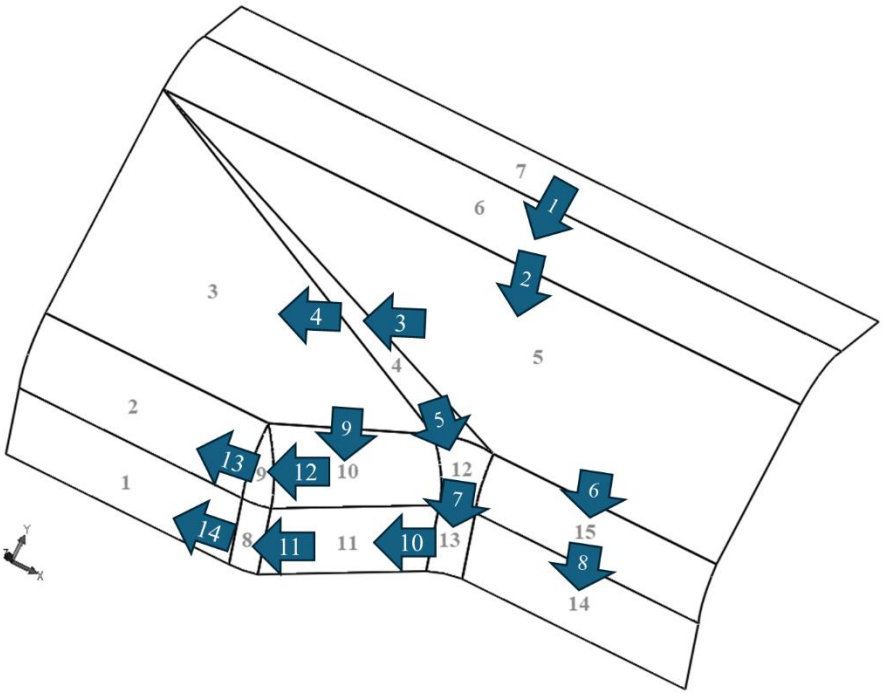


Figure B.12 DP 1-next state - draping sequence for optimised Run 6

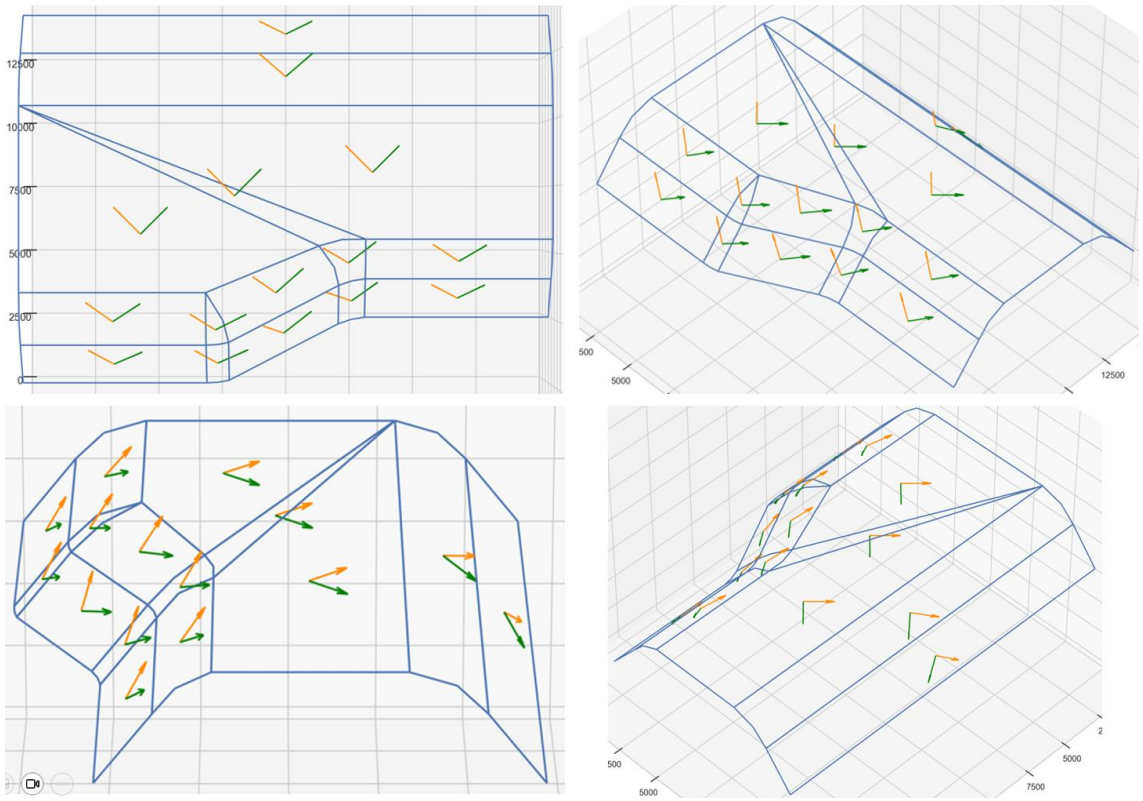


Figure B.13 DP 1-next state resulting orientations for optimised Run 6

No wrinkle warnings were issued in this run, which is consistent with warp and weft orientations observed in Figure B.13. Starting from BaS #7 proved advantageous as the progression initially led to BaS #6 as the only available option and then toward the top flat region of the mould.

Following these initial draping steps, the optimisation algorithm progressed toward double-curvature BaS #9 and #12 but only after some neighbouring BaS had been draped. This sequencing reduced the dispersion of in-plane shear to adjacent regions and limited the accumulation of shear in latter draping steps. Maximum in-plane shear was calculated for BaS #9 at close to 7°.

Figure B.14 shows experimental validation of the optimised draping sequence for this run. The observed warp and weft progression through the initial draped BaS, together with minimal

shear in regions neighbouring the double-curvature areas, closely matches the behaviour predicted by the optimisation software.

Figure B.14 DP 1-next state - validation for optimised Run 6

B.7 DP 1-Next State - Run 7

This section presents results for DP 1-next state optimisation algorithm applied to 66-BaS demonstrator 24_Hutchinson for Run 7, initial orientation 0°/90°. Querying draping sequences starting from all BaS required computational resources beyond the available 32 GB RAM required for completing a full run of the DP 1-next state algorithm. The algorithm was run first for optimising sequences starting from BaS #1 to BaS #42; more than 68 000 partial sequences were evaluated over 55 min . The algorithm was subsequently run from BaS #43 to BaS #66, evaluating an additional 37 949 partial sequences in a further 60 min 24 s .

The best draping sequence is presented in Table B.7. Optimised draping sequence for Run 7 is shown in Figure B.15 where all arrows originate at the centre of the starting draped BaS and terminate at the centre of the newly draped BaS.

Initialization, as discussed in Section 2.3.3, is recommended for this type of complex demonstrator. While this functionality is optional for Algorithm 6.1, it consists of the following heuristic considerations.

BaS are first ranked in descending order of projected area. An 80–20 Pareto rule is then applied through two complementary criteria:

- by count — selecting the top 20% BaS from the ordered list; and
- by area — selecting the subset of BaS whose cumulative area accounts for 20% of total mould area.

The two subsets are compared, and the one containing the larger number of BaS is retained as the initialization set.

In Table B.6, starting BaS recommended by the initialization functionality appear in bold characters. This information will be used for the initialization of Run 8 described in the following section.

The best draping sequence identified by the optimisation algorithm starts from BaS #2. This BaS was flagged by the initialization algorithm as a recommended starting point, confirming its usefulness towards this task in this case.

Figures B.16 and B.17 show the optimised draping sequence as a series of vignettes illustrating the sequential draping of the BaS.

Table B.6 DP 1-next state - Run 7 optimisation results

Starting BaS#	*f(C, Q)*	Starting BaS#	*f(C, Q)*	Starting BaS#	*f(C, Q)*
1	96.16	23	96.51	45	131.67
2	**84.43**	24	115.42	46	84.96
3	86.57	25	105.79	47	89.58
4	87.57	26	93.95	48	84.50
5	**85.74**	27	100.10	49	132.00
6	86.79	28	84.88	50	124.01
7	**84.97**	**29**	**109.37**	51	123.36
8	86.57	**30**	**87.07**	52	123.36
9	86.57	**31**	**102.59**	53	124.98
10	86.57	32	86.51	54	123.91
11	**86.57**	33	90.48	55	128.33
12	**86.57**	34	108.07	56	131.32
13	126.26	35	88.86	**57**	**127.66**
14	88.49	36	108.00	**58**	**171.29**
15	112.72	37	109.14	59	243.03
16	112.75	38	89.11	60	175.43
17	107.97	39	89.67	61	131.40
18	111.48	40	89.11	62	129.09
19	107.66	41	87.07	63	127.60
20	**107.76**	42	102.63	64	130.35
21	109.84	43	89.82	65	129.71
22	86.57	**44**	**89.11**	66	129.25

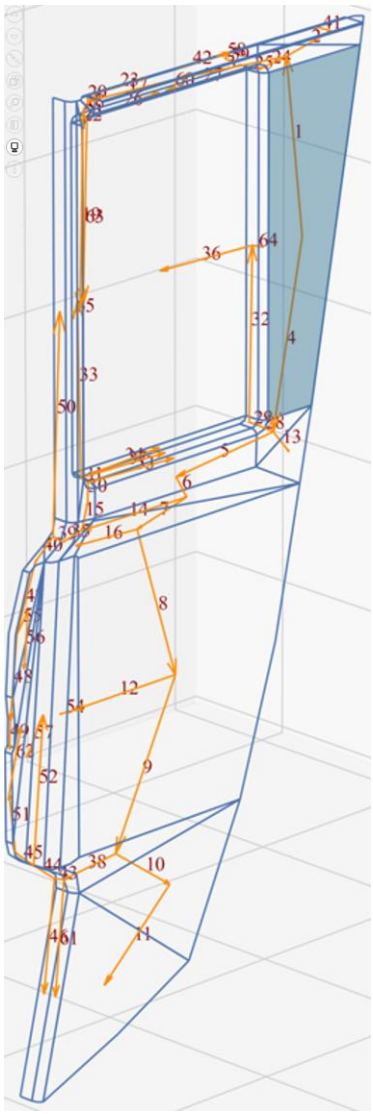


Figure B.15 DP 1-next state - draping sequence for optimised Run 7

Table B.7 DP 1-next state - Best draping sequence for Run 7

Demonstrator & conditions	**StartingBaS**	***f(C&Q)***	**Optimised draping sequence:**
24_Hutchinson 1-next state Run 7	2	84.43	[2-3, 3-1, 3-22, 2-9, 9-28, 9-8, 28-12, 12-11, 11-10, 10-30, 30-29, 29-31, 30-47, 12-40, 40-38, 38-44, 40-41, 44-39, 22-32, 32-33, 3-15, 15-18, 9-13, 33-23, 13-16, 39-37, 37-35, 18-21, 38-36, 36-34, 34-19, 40-45, 30-49, 21-20, 45-51, 51-56, 1-4, 49-54, 54-55, 55-63, 23-5, 54-53, 56-62, 62-65, 65-64 (W), 51-52, 64-66, 41-50, 62-61, 61-60, 50-57, 5-6, 6-7, 53-48, 16-24 (W), 36-26, 34-42, 37-43 (W), 18-25 (W), 37-27, 20-17, 15-14, 57-58 (W), 66-59 (W), 44-46]

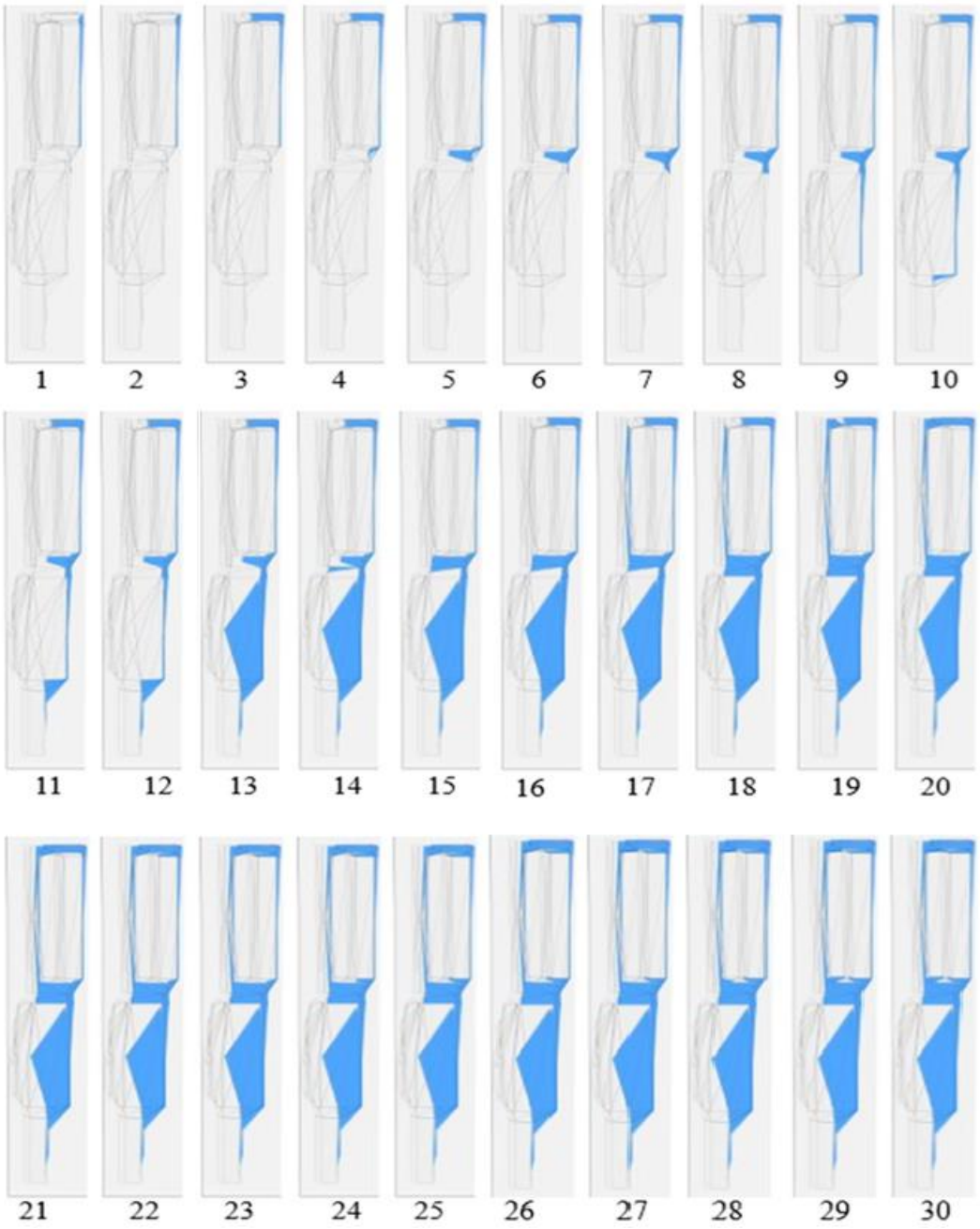


Figure B.16 DP 1-next state - draping sequence for optimised Run 7 - steps 1 to 30

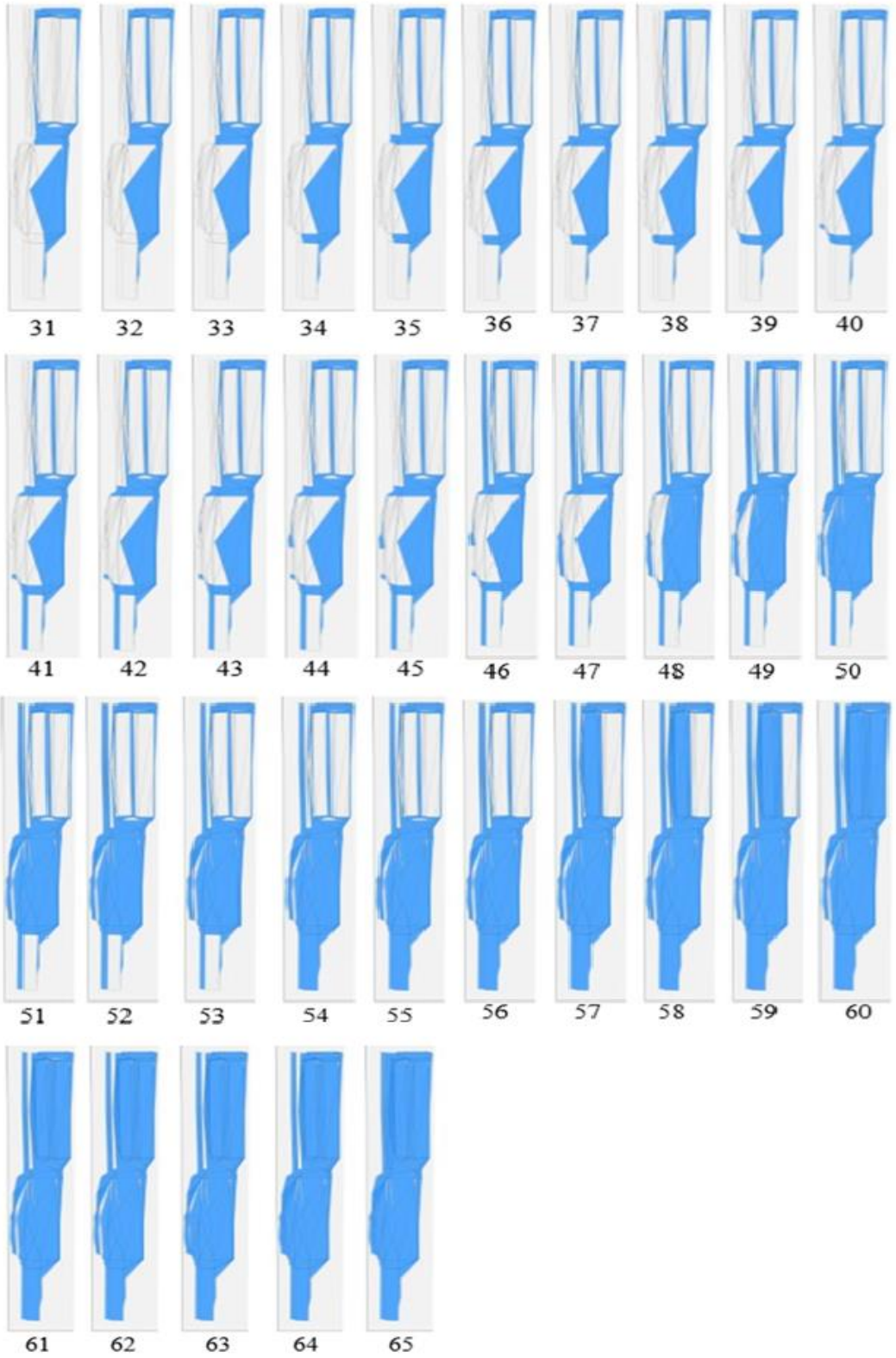


Figure B.17 DP 1-next state - draping sequence for optimised Run 7 - steps 31 to 65

In this run, a total of six wrinkle warnings were issued by the optimisation software, all occurring during the final third of the draping sequence. This behaviour reflects the consistent tendency of the algorithm to defer high in-plane shear to the latter stages of the draping process, thereby limiting shear propagation and accumulation, reducing the likelihood of inducing wrinkles in earlier draping steps.

The mould geometry for this demonstrator includes a complex depressed rectangular cavity located at the top region. In this case, the cavity was enclosed before being draped. This type of enclosure is not explicitly restricted by the optimisation algorithm, as detecting and penalising such events at every candidate draping step would increase the computational burden significantly and affect overall execution time negatively.

No experimental validation photographs are available for this industrial demonstrator subject to confidentiality constraints. However, the observed optimisation behaviour is consistent with trends identified in previous runs.

B.8 DP 1-Next State - Run 8

This section presents results for DP 1-next state optimisation algorithm applied to 66-BaS demonstrator 24_Hutchinson for Run 8, initial orientation +/- 45°. The initialization algorithm selected 14 BaS as the best starting BaS candidates. More than 22 000 partial draping sequences were evaluated and optimised in 91 min 36 s . Results are summarized in Table B.8 and the best draping sequence is presented in Table B.9. The optimal draping sequence is illustrated in Figures B.18 to B.20.

Table B.8 DP 1-next state - Run 8 optimisation results

Starting BaS#	*f(C, Q)*	Starting BaS#	*f(C, Q)*
2	87.70	30	86.38
5	107.89	31	91.96
11	86.38	44	83.98
12	86.43	57	115.32
20	105.61	58	159.66
29	88.17		

Table B.9 DP 1-next state - Best draping sequence for Run 8

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Optimised draping sequence:
24_Hutchinson 1-next state Run 8	44	83.98	[44-38, 38-28, 28-12, 12-11, 11-10, 10-30, 30-29, 10-47, 12-40, 11-41, 44-39, 39-32, 32-22, 22-3, 3-1, 32-33, 3-15, 22-23, 15-18, 39-37, 37-35, 38-36, 36-34, 30-49, 28-9, 12-8, 3-2 (W), 9-13, 13-16, 34-19, 18-21, 40-45, 19-20, 49-54, 54-55, 55-63, 1-4, 23-5, 45-51, 51-56, 56-62, 62-65, 65-64, 51-52, 64-66, 54-53, 41-50, 62-61, 50-57, 63-58, 5-6, 6-7, 53-48, 48-31, 16-24, 13-26, 16-17 (W), 17-14, 34-42, 44-43, 35-25, 37-27, 61-60, 66-59, 45-46]

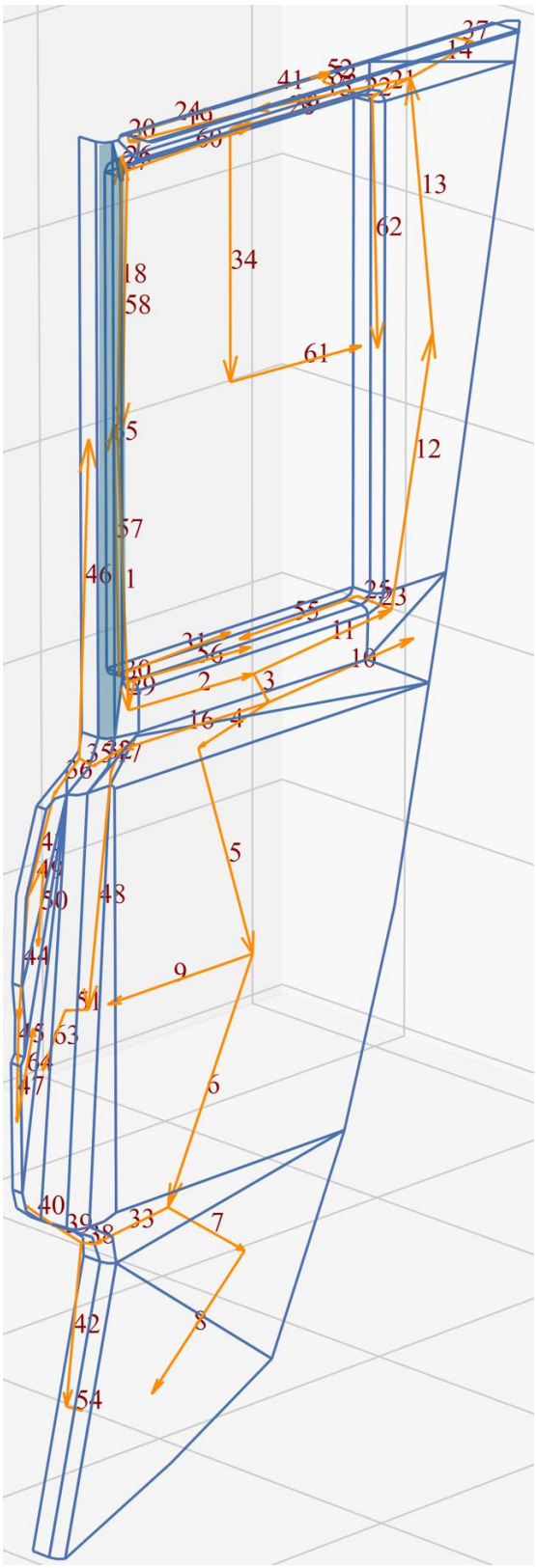


Figure B.18 DP 1-next state - draping sequence for optimised Run 8

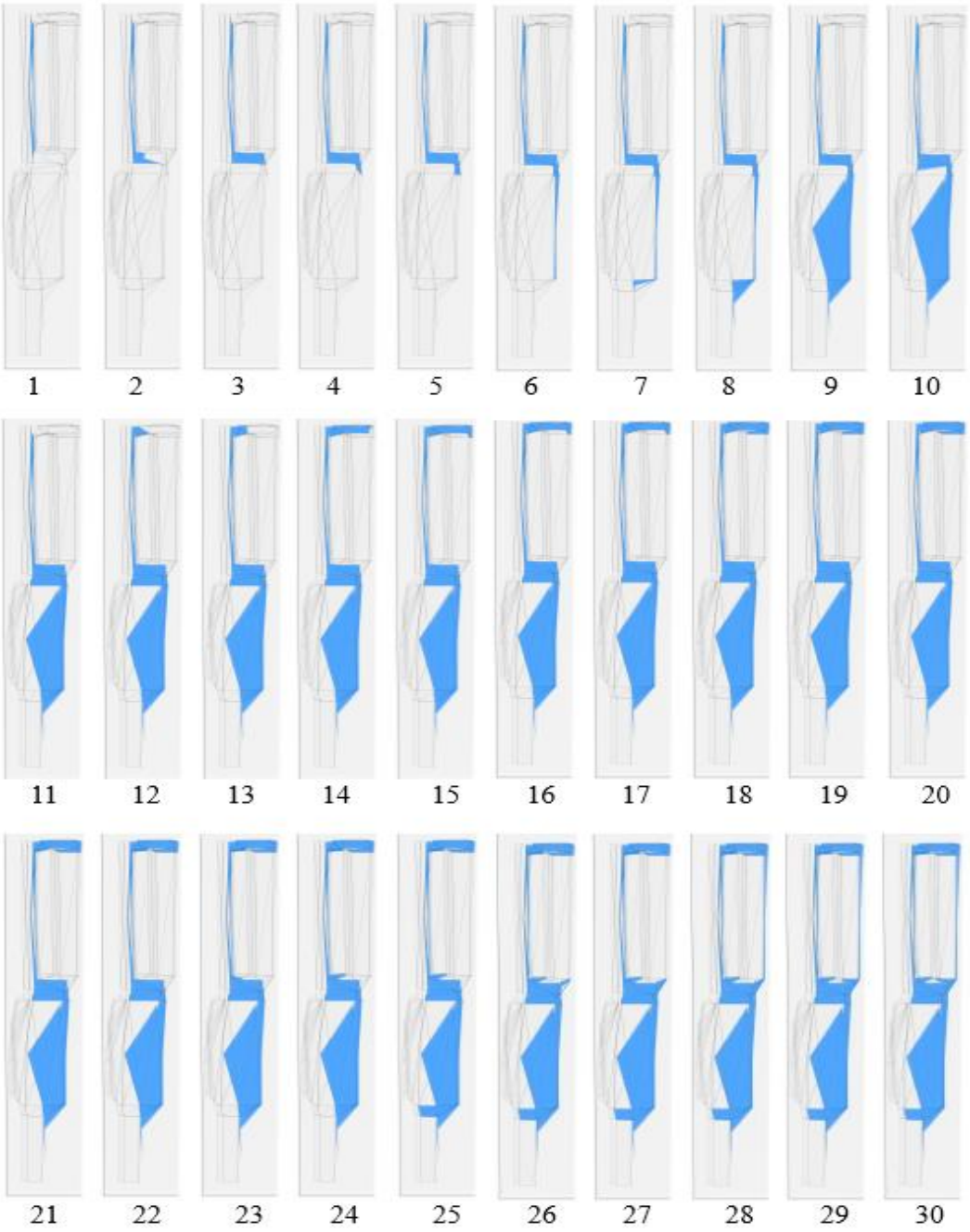


Figure B.19 DP 1-next state - draping sequence for optimised Run 8 - steps 1 to 30

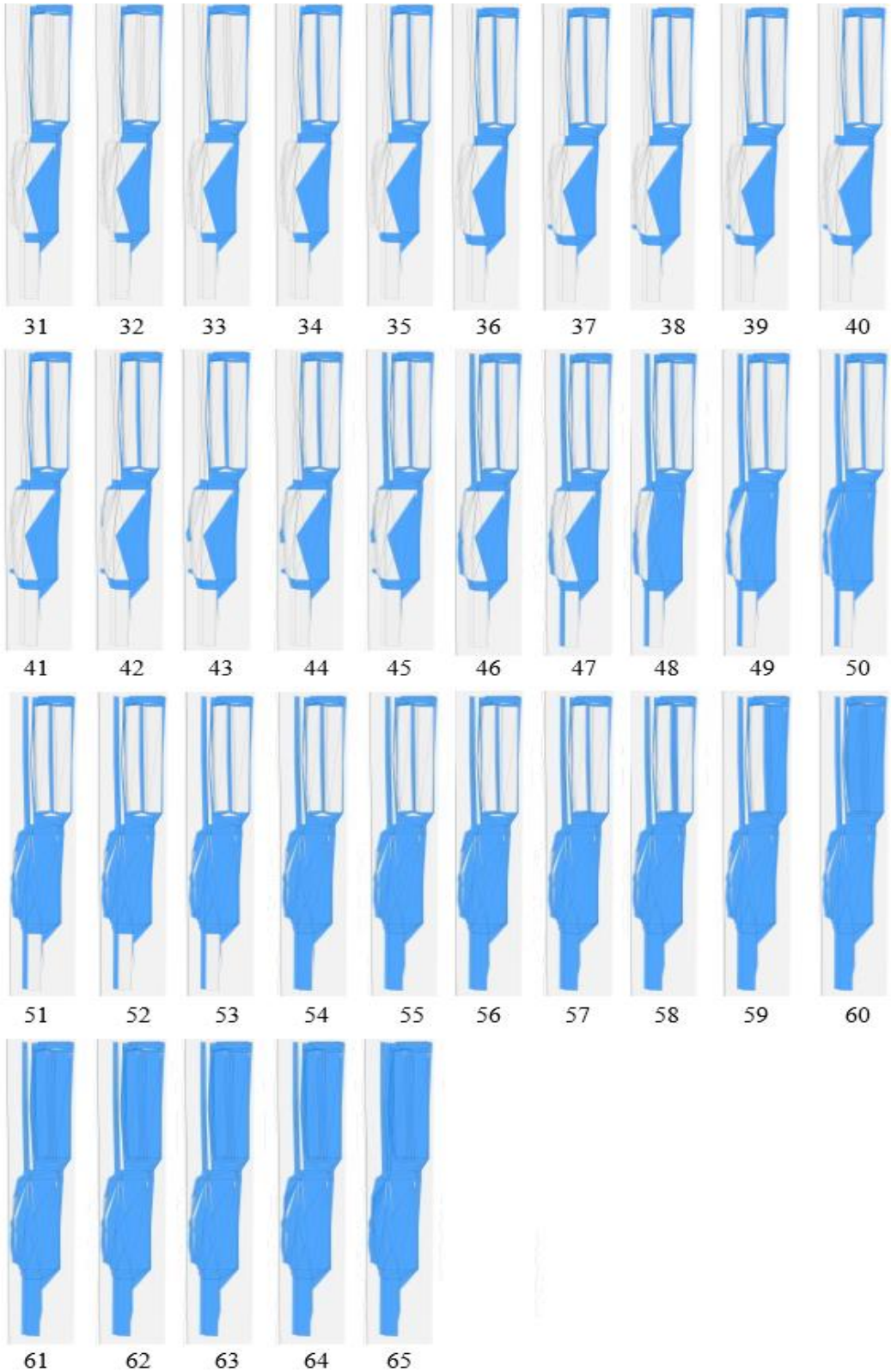


Figure B.20 DP 1-next state - draping sequence for optimised Run 8 - steps 31 to 65

As shown in Table B.9, two wrinkle warnings were issued during 1-next state optimisation for Run 8. The first occurred during the first third of the draping sequence, while the second occurred during the final third. This run generated fewer wrinkle warnings than Run 7, which is reflected in the slightly lower Objective Function value. However, this positive impact is attenuated by the occurrence of high in-plane shear at an early stage of the draping sequence.

Similar to Run 7, an enclosure event was observed during draping progression. Such an operation is undesirable in shop-floor practice, as enclosing a depressed pocket before draping can lead to displacement of the reinforcement plies and increased difficulty in controlling fabric placement.

Given the optimised values for Run 7 and Run 8 with this industrial demonstrator, it is recommended to explore the optimised draping sequences from both initial orientation configurations. Comparing these alternatives would allow further assessment of overall performance, particularly with respect to wrinkle generation and the ability to manage these effects through subsequent draping steps or darting operations.

B.9 DP 2-Next States Run 1

From Appendix B.9 to Appendix B.16 is presented the results obtained with the DP 2-next states algorithm and evaluates its computational performance for the same selected demonstrators. The same 8 runs featuring the same parameters listed in Table 6.3 were evaluated.

Given the substantially larger search space explored by the 2-next states DP algorithm, memory usage becomes a critical consideration. For this reason, wrinkle warning and darting recommendation parameters were not stored for hypothetical states, preventing excessive memory usage while preserving the core optimisation functionality.

Run 1 for 10-BaS Demonstrator 31_Benchmark, DP 2-next states, initial orientation set at 0°/90°, optimised a total of 231 partial draping sequences. All BaS were tested as starting BaS. Total computing time was 2 min 25 s . As explained in Section 6.4, BFS branching is not considered for the 2-next states DP draping steps formulation; therefore, a reduced number of partial draping steps are evaluated.

Table B.10 lists the optimised draping sequences for DP 2-next-states Run 1 which start from BaS #2 and #3 and are essentially symmetric. Figure B.21 shows optimal sequence [3-2, 2-1, 3-4, 4-9, 9-10, 10-5, 3-8, 8-7, 2-6]. Figure B.22 shows the resulting warp and weft orientations for this optimised draping sequence, represented by green and dark orange vectors respectively.

Table B.10 DP 2-next states - Run 1 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Optimised draping sequence:
31_Benchmark 2-next states Run 1	2	14.44	[2-3, 3-4, 2-1, 1-5, 5-10, 10-9, 3-8, 8-7, 2-6]
	3	14.44	[3-2, 2-1, 3-4, 4-9, 9-10, 10-5, 3-8, 8-7, 2-6]

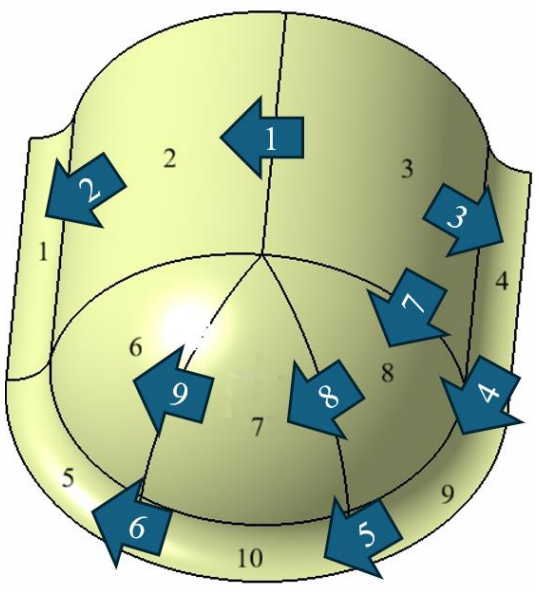


Figure B.21 DP 2-next states - draping sequence for optimised Run 1

Figure B.22 shows warp and weft orientations for BaS of Demonstrator 31_Benchmark, and how orientations are affected when draping a double curvature BaS.

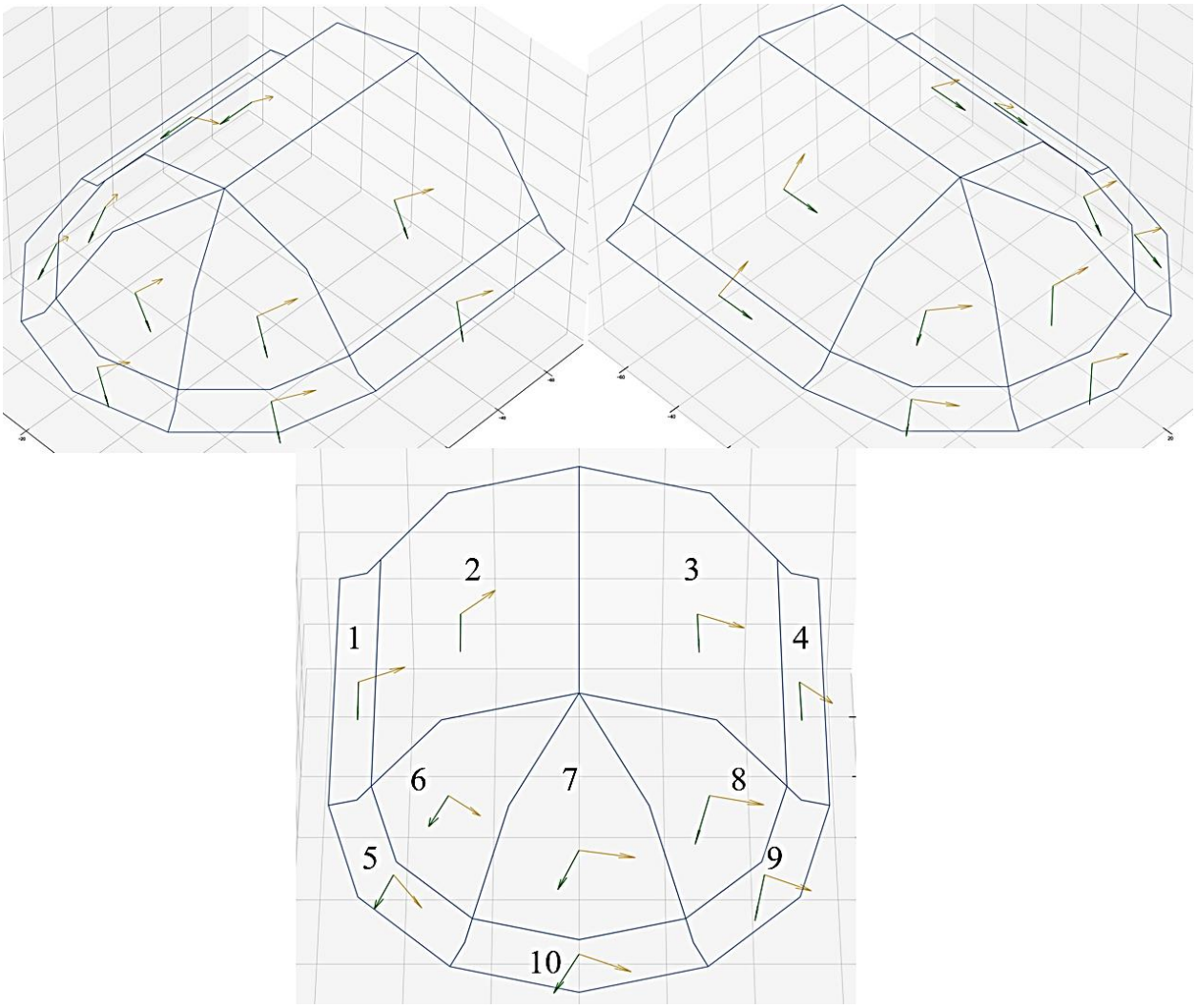


Figure B.22 DP 2-next states resulting orientations for optimised Run 1

Run 1 for the 2-next states algorithm resulted in an optimised value of 14.44, compared to 11.27 for Run 1 of the 1-next state algorithm. This increase is associated with the formation of an enclosed double-curvature region involving BaS #6, #7, and #8, which increased in-plane shear. Such configurations are undesirable as darting operations cannot be performed once the region is enclosed. From a practical perspective, enclosed areas are difficult to drape and are typically avoided in hand lay-up operations.

In addition, Figure B.22 shows large differences in yarn orientations between BaS #1 and #5, and between BaS #6 and #2, which promote wrinkle initiation.

Figure B.23 shows the laboratory validation trial corresponding to this draping sequence. The results demonstrate a clear association between the observed undesirable draping characteristics and the higher Objective Function value obtained for the 2-next states Run 1.

Figure B.23 DP 2-next states - validation for optimised Run 1

B.10 DP 2-Next States Run 2

Run 2 for 10 BaS Demonstrator 31_Benchmark, DP 2-next states, initial orientation set at +/- 45°, optimised a total of 232 partial draping sequences. All BaS were tested as starting BaS. Total computing time was 2 min 54 s .

Table B.11 lists the optimised draping sequences for DP 2-next-states Run 2, which start from BaS #2 and #3 and are essentially symmetric. Figure B.24 shows optimal sequence [3-2, 2-1, 3-4, 4-9, 9-10, 10-5, 3-8, 8-7, 2-6]. Figure B.25 shows resulting warp and weft orientations for this optimised draping sequence, represented by green and orange vectors respectively.

Table B.11 DP 2-next states - Run 2 optimisation results

Demonstrator & conditions	**Starting BaS**	***f(C&Q)***	**Optimised draping sequence:**
31_Benchmark 2-next states Run 2	2	24.475	[2-3, 3-4, 2-1, 1-5, 5-10, 10-9, 3-8, 8-7, 2-6]
	3	24.475	[3-2, 2-1, 3-4, 4-9, 9-10, 10-5, 3-8, 8-7, 2-6]

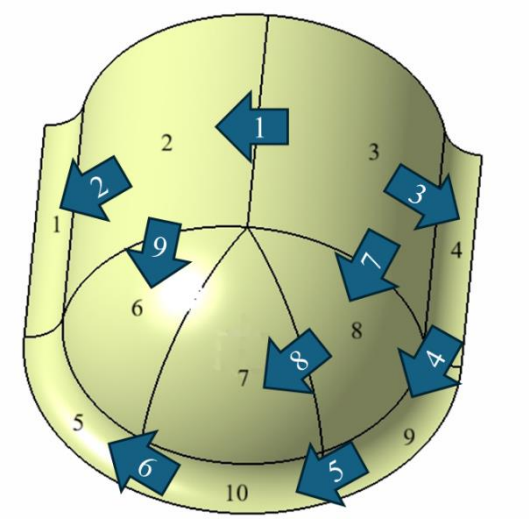


Figure B.24 DP 2-next states - draping sequence for optimised Run 2

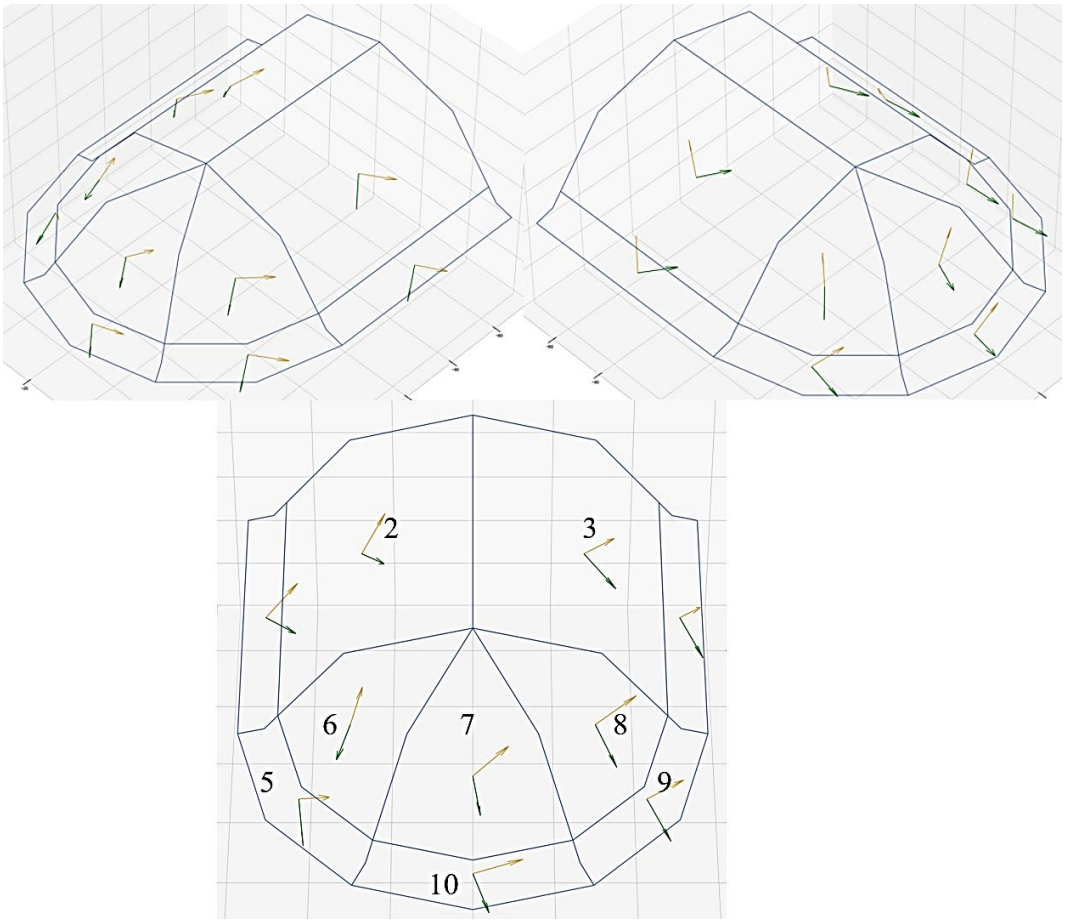


Figure B.25 DP 2-next states resulting orientations for optimised Run 2

Analysis of generated warp and weft orientations identified the source of the difference in values between 2-next states Run 1 and Run 2. For Run 2, in-plane shear angle in BaS #6 exceeded the 50° locking angle and it exceeded 40° in BaS #7 against approximately 30° and 15° for the same BaS in Run 1. This is observed easily when comparing Figure B.25 to Figure B.22.

Comparing this Run 2 results between 1-next state and 2-next states is a good performance evaluation of the Objective Function, showing clear penalization for suboptimal draping decisions in the latter case.

As detailed in Section 6.10 the observed increase in Objective Function value for Run 1 and Run 2 is associated with the formation of an enclosed region in the most critical double-curvature area of the mould in the latter case.

B.11 DP 2-Next States Run 3

Run 3 for 10 BaS Demonstrator 32_Variation, DP 2-next states, initial orientation set at 0°/90°, optimised a total of 230 partial draping sequences. All BaS were tested as starting BaS. Total computing time was 2 min 30 s .

Table B.12 lists the optimised draping sequences for the DP 2-next-states Run 3, which start from BaS #2 and BaS #3. Figure B.26 shows optimal sequence [3-2, 2-1, 3-4, 4-9, 1-5, 5-10, 3-8, 8-7, 2-6]. Figure B.27 shows resulting warp and weft orientations for this optimised draping sequence, represented by green and orange vectors respectively.

Table B.12 DP 2-next states - Run 3 optimisation results

Demonstrator & conditions	**Starting BaS**	***f(C&Q)***	**Optimised draping sequence:**
32_Variation 2-next states Run 3	2	17.82	[2-3, 3-4, 2-1, 1-5, 4-9, 9-10, 3-8, 8-7, 2-6]
	3	17.82	[3-2, 2-1, 3-4, 4-9, 1-5, 5-10, 3-8, 8-7, 2-6]

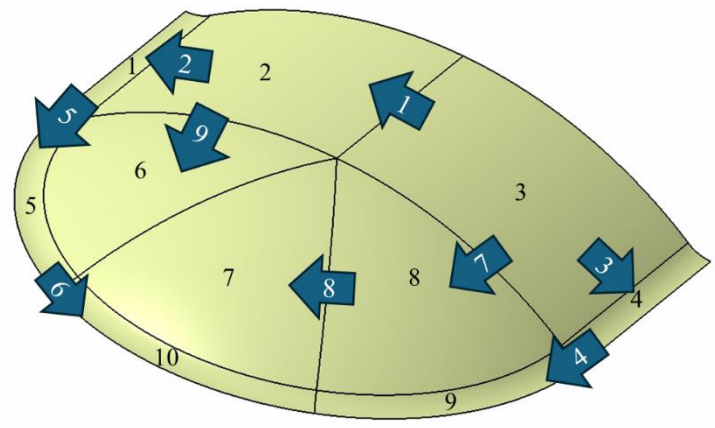


Figure B.26 DP 2-next states - draping sequence for optimised Run 3

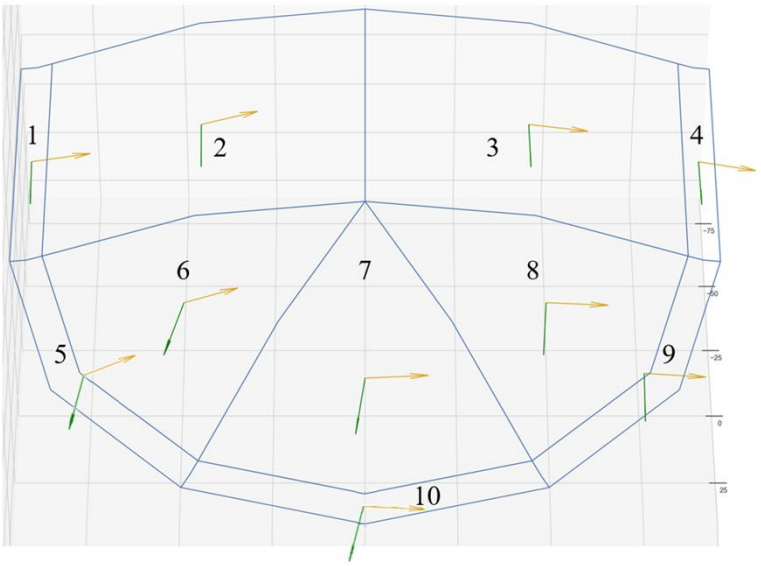


Figure B.27 DP 2-next states resulting orientations for optimised Run 3

The optimised value for 2-next states Run 3 is 28% higher than its counterpart in the 1-next state case. Similar peripheral draping is observed, resulting from the greedy nature of the DP algorithm when not constrained by the no-enclosure condition, which led to suboptimal decisions.

High in-plane shear is observed in BaS #6, reaching values up to 30°. In addition, large differences in warp and weft orientations between BaS #1 and BaS #5 are seen in Figure B.27. This condition can increase the effort required for HLU and lead to variability in ply placement.

Figure B.28 confirms the previous analysis, highlighting the entrapped region and associated in-plane shear in BaS #6 as well as regions prone to wrinkle initiation due to yarn orientation conflicts between BaS #1 and BaS #5, and between BaS #2 and BaS #6.

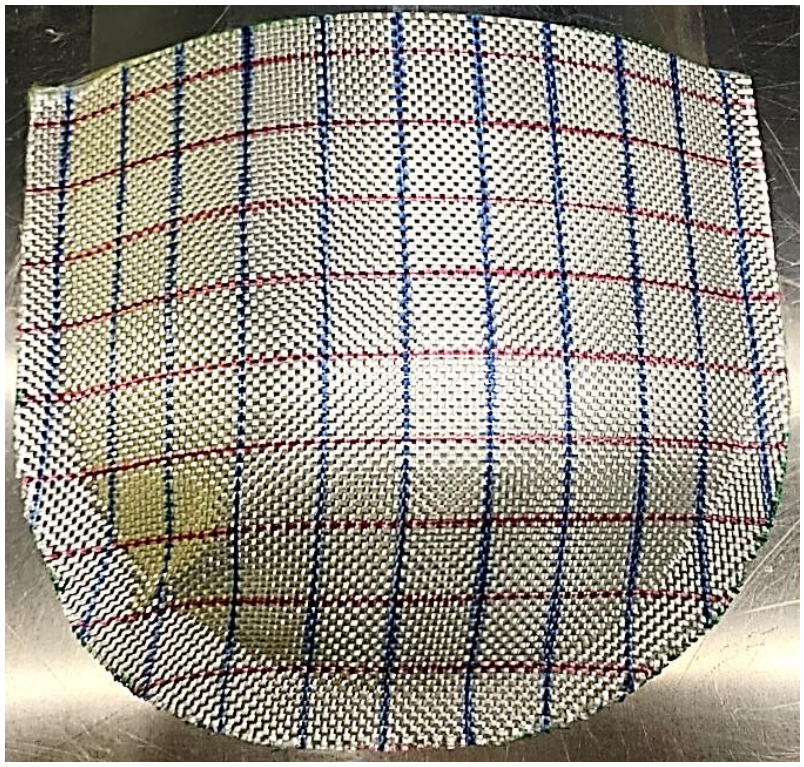


Figure B.28 DP 2-next states - validation for optimised Run 3

B.12 DP 2-Next States Run 4

This section presents results of the DP 2-next states optimisation algorithm applied to 10-BaS demonstrator 32_Variant for Run 4, initial orientation set at +/- 45°. All BaS were tested as starting BaS. 224 partial draping sequences were evaluated and optimised in 3 min . Results are

summarized in Table B.13, optimal draping sequence starting from BaS #3 is illustrated in Figure B.29 and the corresponding yarn orientations are illustrated in Figure B.28 .

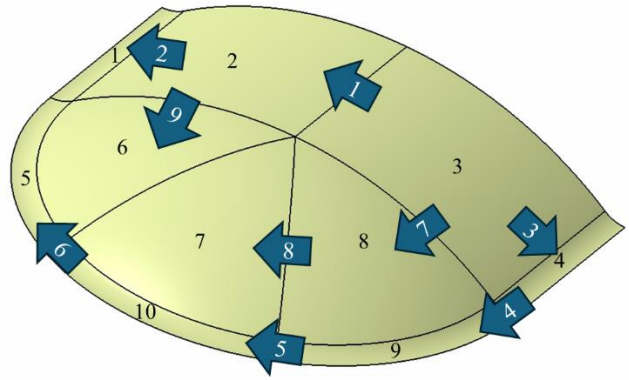


Figure B.29 DP 2-next states - draping sequence for optimised Run 4

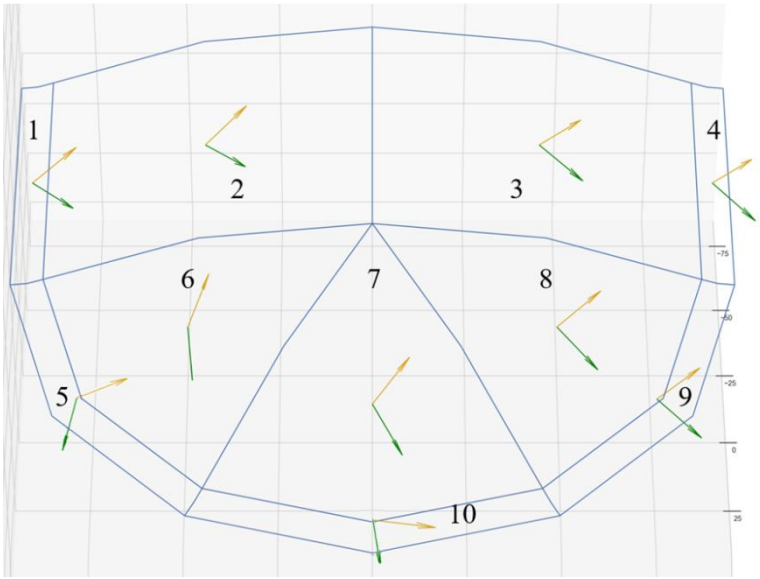


Figure B.30 DP 2-next states resulting orientations for optimised Run 4

Table B.13 DP 2-next states - Run 4 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Optimised draping sequence:
32_Variation 2-next states Run 4	2	19.20	[2-3, 3-4, 2-1, 1-5, 4-9, 9-10, 3-8, 8-7, 2-6]
	3	19.20	[3-2, 2-1, 3-4, 4-9, 9-10, 10-5, 3-8, 8-7, 2-6]

The optimised value for the 2-next states Run 4 reached 19.2, compared to 13.93 for the corresponding 1-next state case. This increase reflects the persistence of enclosure-related effects in the critical double-curvature region, which continues to drive elevated in-plane shear for BaS #6.

Differences in warp and weft orientations between adjacent BaS further contribute to the increase of energy required for draping these areas, and to increase in resulting variability. These combined effects explain the higher value obtained for this run.

B.13 DP 2-Next States Run 5

This section presents results of the DP 2-next states optimisation algorithm applied to 15-BaS demonstrator 01_Francois for Run 5, initial orientation set at 0°/90°. All BaS were tested as starting BaS. 808 partial draping sequences were evaluated and optimised in 4 min 48 s . Results are summarized in Table B.14, optimised draping sequence starting from BaS #6 and resulting yarn orientations are shown in Figures B.31 and B.32 respectively.

Table B.14 DP 2-next states - Run 5 optimisation results

Demonstrator & conditions	Starting BaS	*f(C&Q)*	Optimised draping sequence:
01_Francois 2-next states Run 5	2	3.58	[2-3, 3-4, 4-5, 5-6, 5-15, 15-14, 2-1, 1-8, 14-13, 13-12, 2-9, 9-10, 6-7, 10-11]
	6	3.58	[6-5, 5-4, 5-15, 15-14, 4-3, 3-2, 2-1, 1-8, 14-13, 13-12, 2-9, 9-10, 6-7, 10-11]

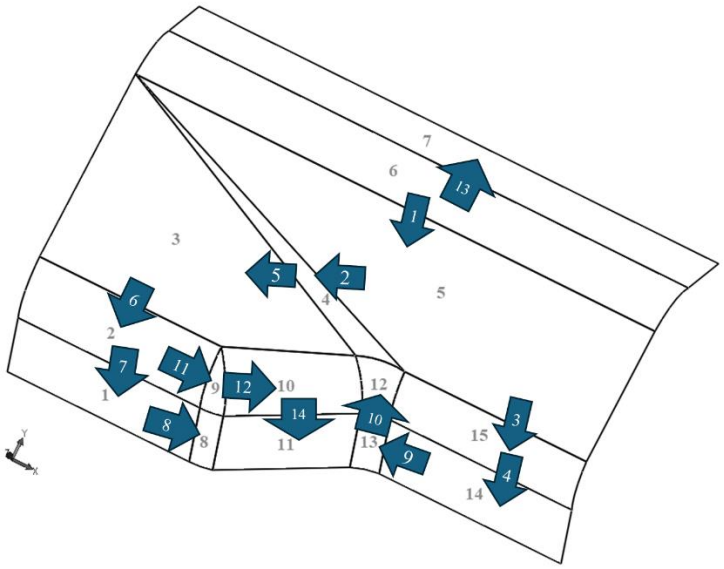


Figure B.31 DP 2-next states - draping sequence for optimised Run 5

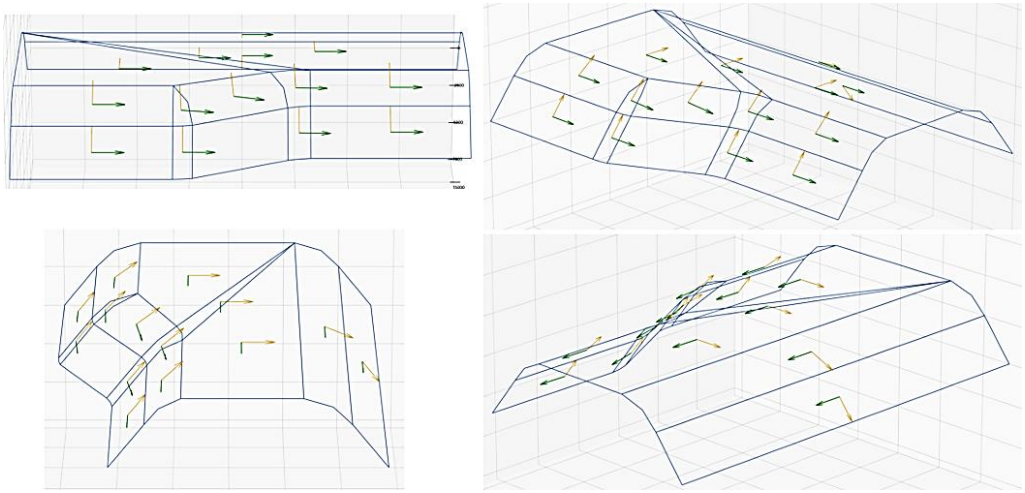


Figure B.32 DP 2-next states resulting orientations for optimised Run 5

In this run, a lower value is observed compared to its 1-next state counterpart. This can be linked to how the draping sequence progresses, delaying the draping of double-curvature BaS and prioritizing other BaS earlier in the sequence. This progression is better aligned with decisions typically made by experienced technicians on the shop floor.

It is important to note that the final two draping steps were completed through software override in the 2-next states algorithm, that bypassed the no-BFS branching condition. This override ensures compliance with the global constraint requiring all BaS to be draped, to conclude the draping sequence.

Figure B.33 shows the resulting warp and weft orientations when this sequence is executed in the laboratory. Low in-plane shear is observed from this optimised draping sequence, validating Run 5 optimisation results.

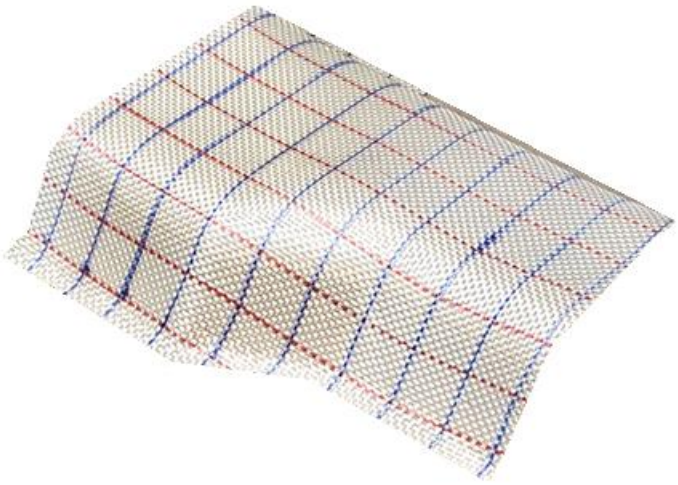


Figure B.33 DP 2-next states - validation for optimised Run 5

B.14 DP 2-Next States Run 6

This section presents results obtained using the DP 2-next states optimisation algorithm for Run 6 on the 15-BaS demonstrator 01_Francois, with the initial yarn orientation set at ±45°. Table B.15 summarizes the optimisation results, while Figures B.34 and B.35 show the optimised draping sequence obtained from starting BaS #7 and the resulting yarn orientations, respectively. All BaS were evaluated as potential starting BaS. A total of 713 partial draping sequences were evaluated and optimised in 4 min 12 s.

Table B.15 DP 2-next states - Run 6 optimisation results

Demonstrator & conditions	**Starting BaS**	***f(C&Q)***	**Optimised draping sequence:**
01_Francois 2-next states Run 6	7	5.01	[7-6, 6-5, 5-4, 4-3, 4-12, 12-13, 5-15, 15-14, 3-10, 10-11, 3-2, 2-1, 2-9, 9-8]

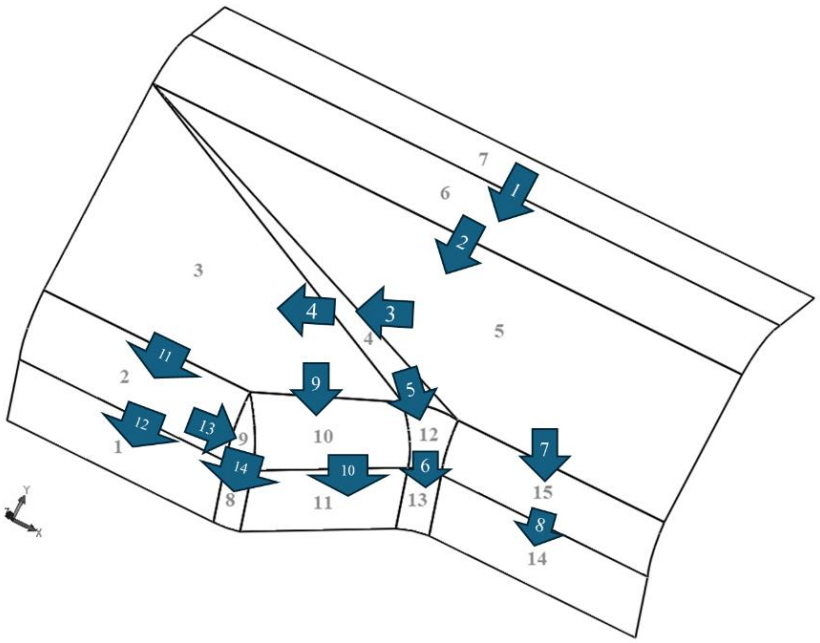


Figure B.34 DP 2-next states - draping sequence for optimised Run 6

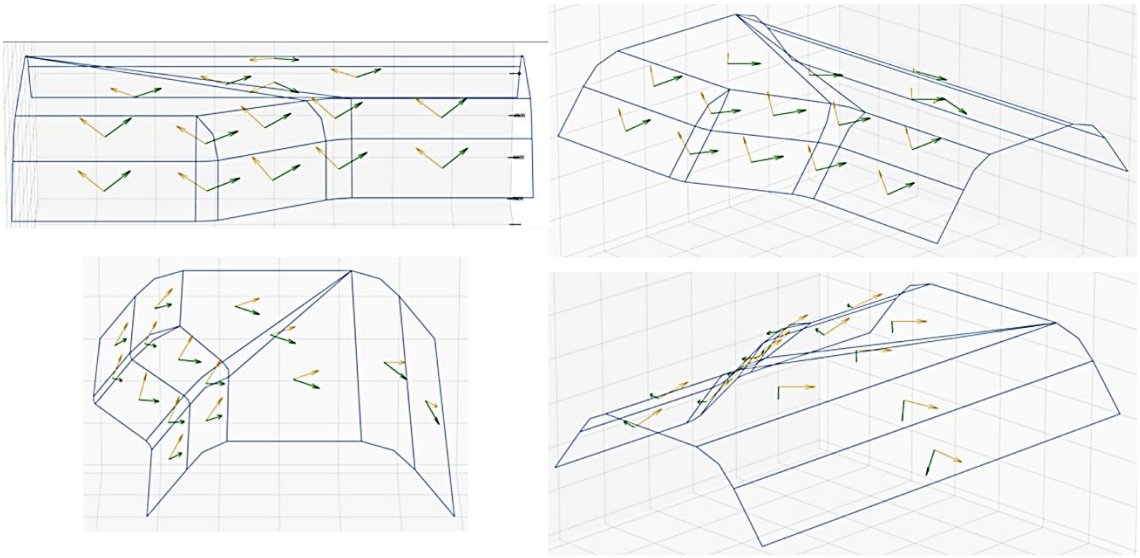


Figure B.35 DP 2-next states resulting orientations for optimised Run 6

For the 2-next states Run 6 starting from BaS 7 a value of 5.01 is obtained, compared to 5.03 for the corresponding 1-next state run. The lower value indicates an improved evaluation of the draping progression.

The resulting sequence closely resembles the decision-making process typically followed by experienced technicians on the shop floor. The draping progression reflects a practical approach, prioritizing geometrically favourable regions before addressing more critical areas.

Figure B.36 shows a magnified view of the most critical area of this demonstrator, where the two double curvature BaS are located. It is observed that in-plane shear is contained in this area, avoiding propagation.

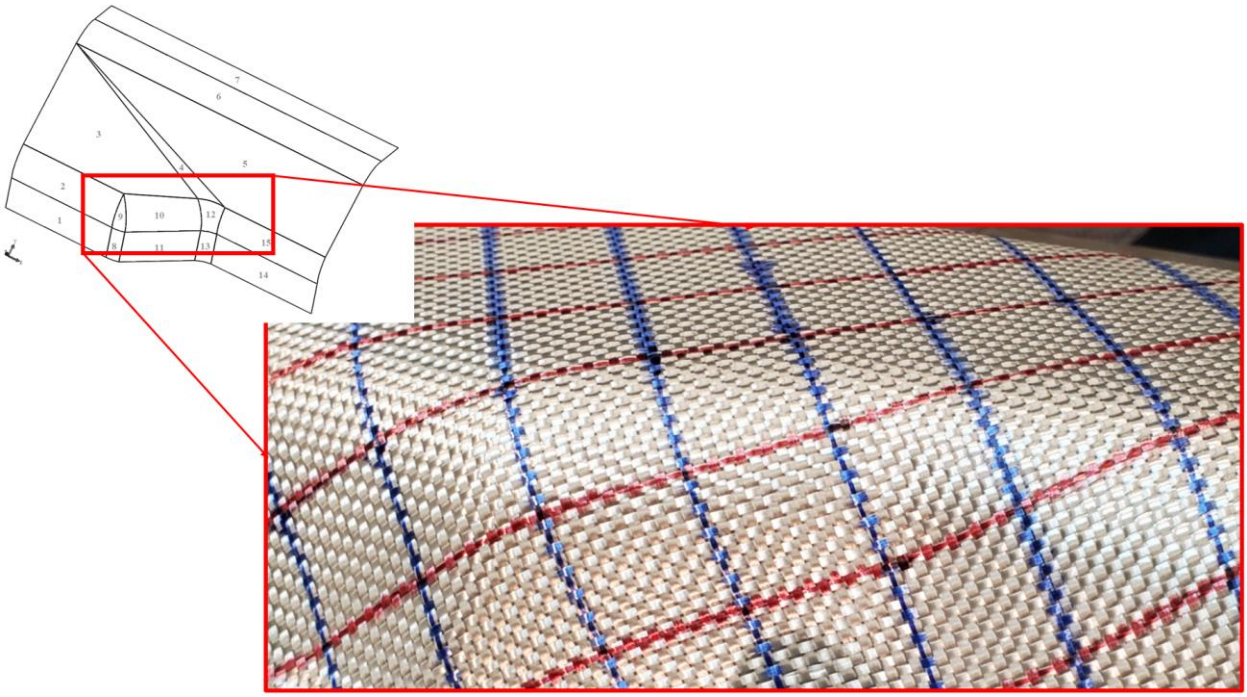


Figure B.36 DP 2-next states - validation for optimised Run 6

B.15 DP 2-Next States Run 7

This DP 2-next states Run 7 is presented in Section 6.11.5

B.16 DP 2-Next States Run 8

This section presents results of the DP 2-next states optimisation algorithm applied to the 66-BaS demonstrator 24_Hutchinson for Run 8, with the initial orientation set at ±45°. Figures B.37 to B.39 show the optimised draping sequence obtained from starting BaS #12. Table B.16 lists the corresponding draping sequence. The initialization algorithm was executed using the 11 selected starting BaS listed in Table B.8. More than 11,200 partial draping sequences were evaluated and optimised in 93 min.

Table B.16 DP 2-next states - Best draping sequence for Run 8

Demonstrator & conditions	**Starting BaS**	***f(C&Q)***	**Optimised draping sequence:**
24_Hutchinson 2-next states Run 8	12	86.10	[12-11, 11-41, 41-40, 40-38, 11-10, 10-47, 10-30, 30-29, 12-28, 28-9, 9-2, 2-3, 38-44, 44-39, 3-22, 22-32, 3-15, 15-18, 9-13, 13-16, 38-36, 36-34, 39-37, 37-35, 32-33, 33-23, 3-1, 1-4, 18-21, 21-25, 40-45, 45-51, 23-5, 5-6, 16-19, 19-24, 30-49, 49-54, 54-55, 55-63, 51-56, 56-62, 62-65, 65-64, 29-31, 31-48, 49-50, 50-57, 21-20, 20-17, 44-46, 46-52, 63-66, 66-59, 9-8, 12-8, 6-7, 48-53, 25-27, 24-26, 17-14, 2-14, 63-58, 57-58, 34-42]

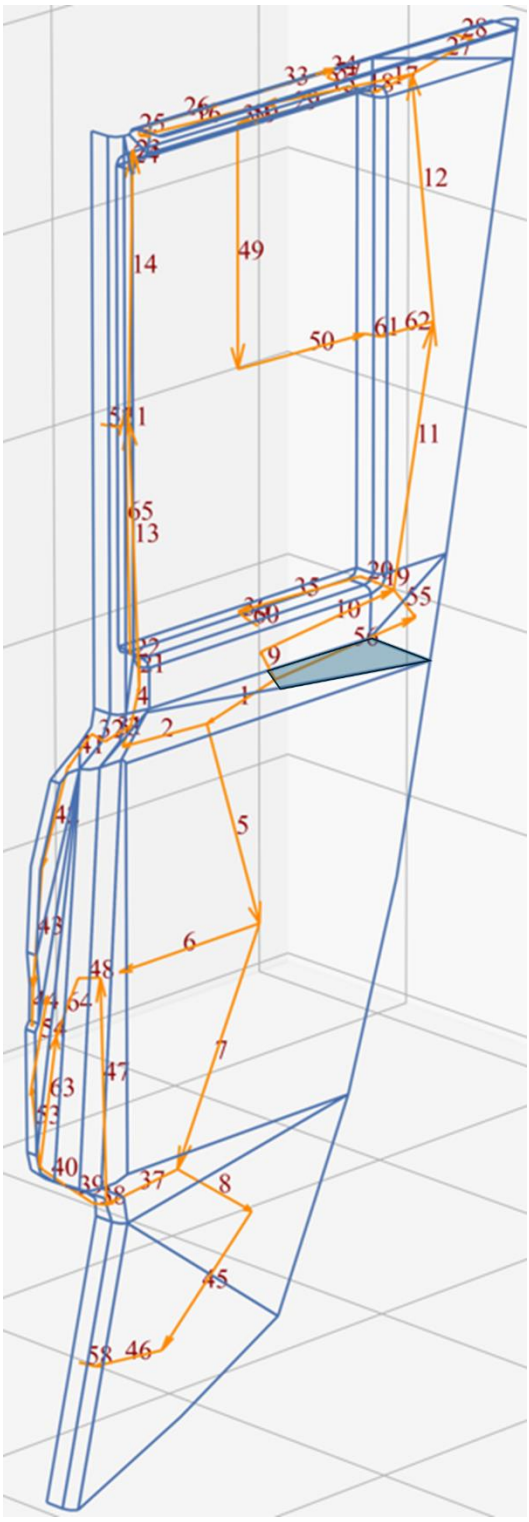


Figure B.37 DP 2-next states - draping sequence for optimised Run 8

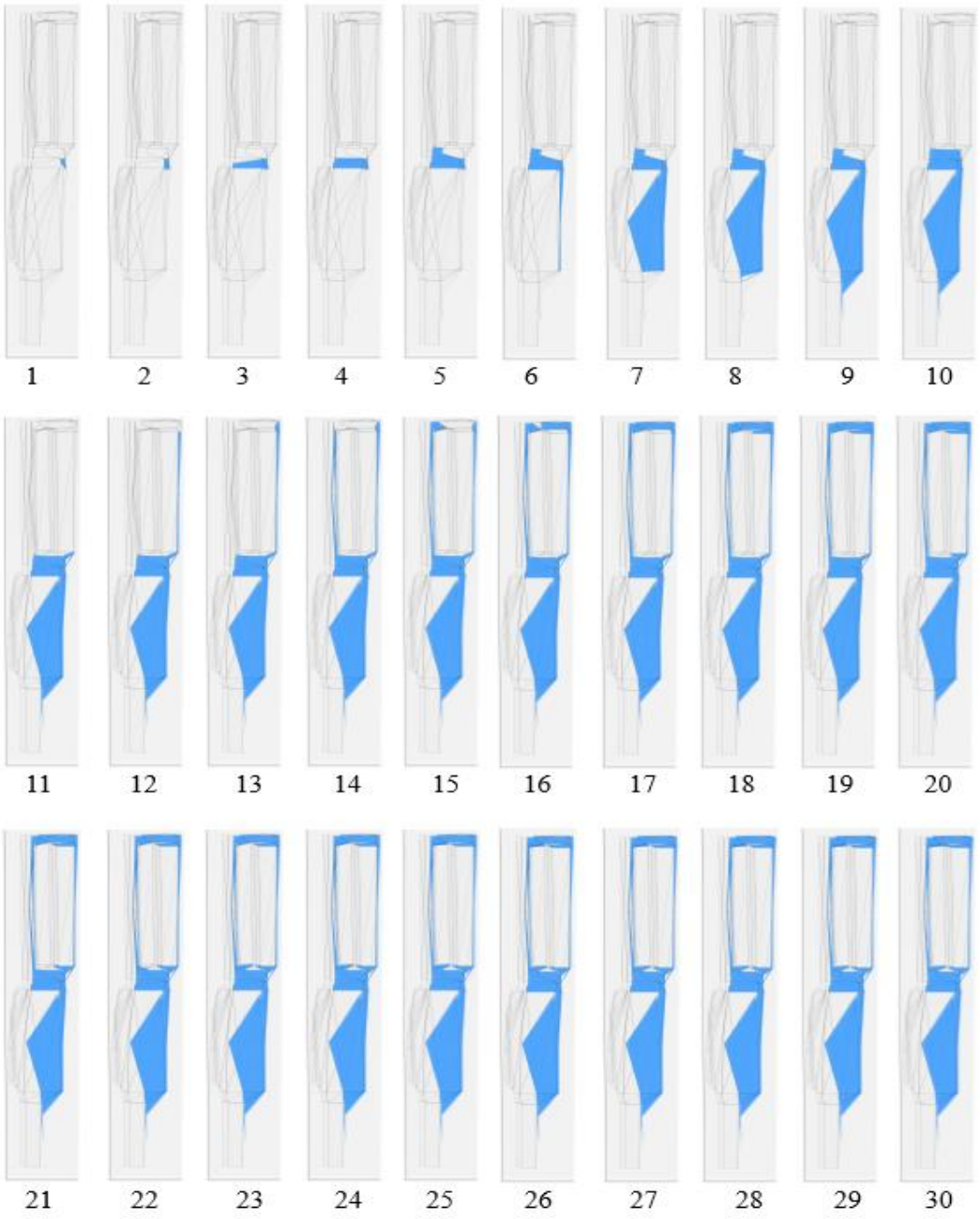


Figure B.38 DP 2-next states - draping sequence for optimised Run 8 - steps 1 to 30

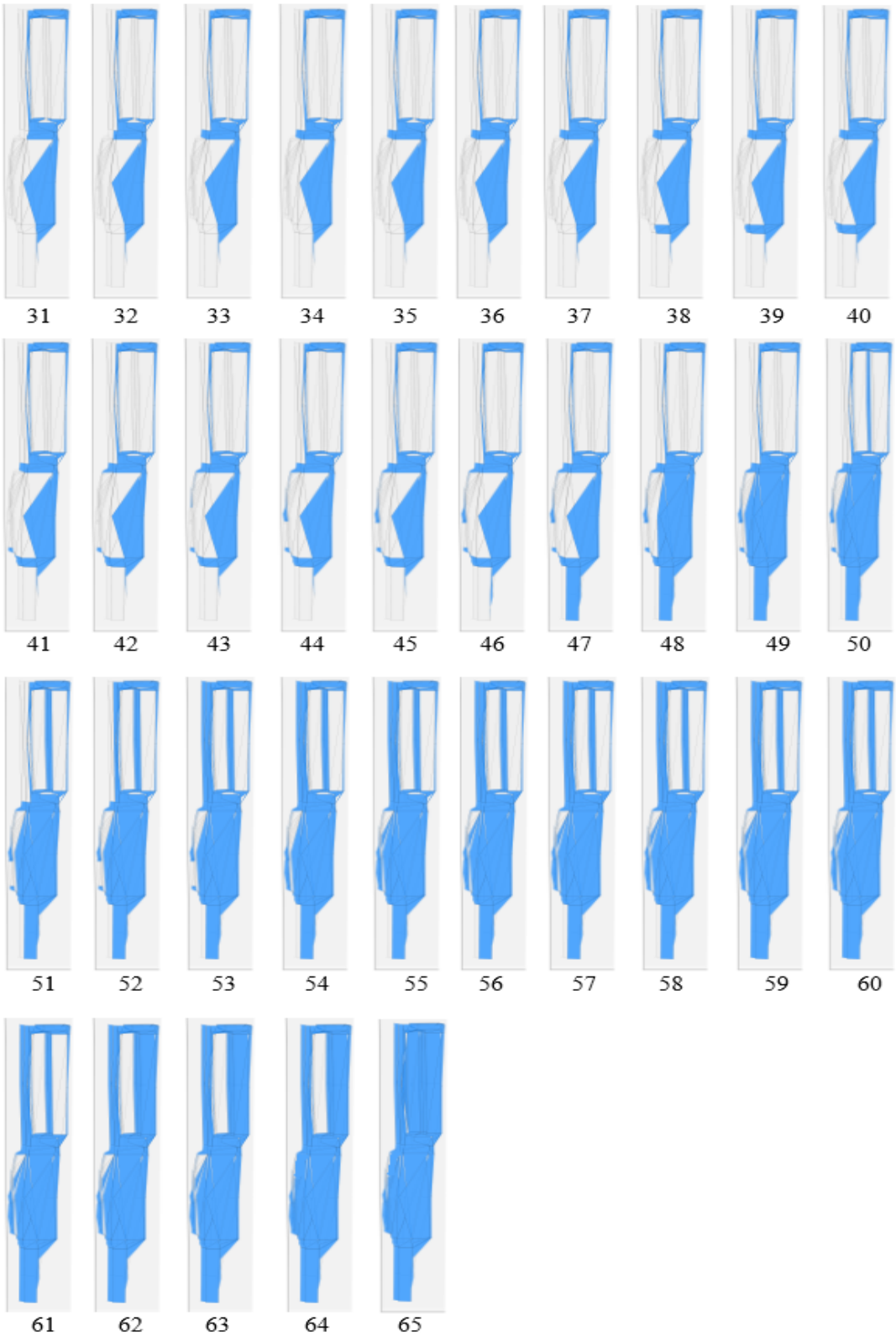


Figure B.39 DP 2-next states - draping sequence for optimised Run 8 - steps 31 to 65

Run 8, with initial warp and weft orientations of +/- 45°, reached an Objective Function value slightly better than Run 7, by 1%. The best optimised draping sequence for Run 8 is similar to that of Run 7, with both starting in the central area of the industrial mould. The same number of BaS presented final shear angles around 15°, and it was the same case of BaS #59 reaching a maximum average shear angle of 42° between warp and weft yarns.

This consistency between runs can support future decisions during the development phase of a composite part, particularly when multiple ply orientations are defined in the ply book. It suggests that different initial yarn orientations do not significantly affect overall part quality, while helping to avoid interlaminar wrinkles caused by conflicting ply orientations.

Comparing the 1-next state Run 8 with its 2-next states counterpart shows a minor increase of 2.5% in value and a difference in processing time of 1%. Therefore, it can be concluded that there are no significant differences in performance between these two approaches for this industrial demonstrator.