Automatic 3D Routing for the Physical Design of Electrical Wiring Interconnection Systems for Aircraft

Zhu, Zaoxu

DOI
10.4233/uuid:2ca107b4-202d-4638-a044-d45649b89275

Publication date
2016

Document Version
Final published version

Citation (APA)
Zhu, Z. (2016). *Automatic 3D Routing for the Physical Design of Electrical Wiring Interconnection Systems for Aircraft*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:2ca107b4-202d-4638-a044-d45649b89275

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Automatic 3D Routing for the Physical Design of Electrical Wiring Interconnection Systems for Aircraft

Automatic 3D Routing for the Physical Design of Electrical Wiring Interconnection Systems for Aircraft

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K.C.A.M. Luyben;
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op
dinsdag 20 december 2016 om 10:00 uur

door

Zaoxu Zhu
Master of Mechanical-Electronic Engineering,
Beihang University, P.R. China
geboren te Anhui, P.R. China

This dissertation has been approved by the
promotors: Prof. dr. ir. M.J.L. van Tooren

Composition of the doctoral committee:

Rector Magnificus	chairman
Prof. dr. ir. M.J.L. van Tooren	University of South Carolina
Dr.ir. G. La Rocca	Delft University of Technology

Independent members:

Prof.dr.ir. L.L.M. Veldhuis	Delft University of Technology
Prof.ir. J.J. Hopman	Delft University of Technology
Priv.-Doz. Dr.-Ing. habil. S. Rudolph	Universität Stuttgart
MEng. S. Taylor	Fokker Elmo B.V.

This research is funded by the Chinese Scholarship Council and Fokker Elmo.

Keywords: Electrical Wiring Interconnection System; Wire Harness; Design Automation; Automatic 3D Routing; Knowledge-Based Engineering; Multidisciplinary Design Optimization

ISBN: 978-94-92516-30-5

Cover photo: Azure Window
Cover designed by: Zaoxu Zhu

Printed by: Haveka, the Netherlands

Summary

The electronic equipment of an aircraft needs to be interconnected by an electrical wiring system. This system is essential to aircraft safety, and its malfunction may cause fatal results, such as the aviation accidents with TWA Flight 800 in 1996 and Swissair 111 in 1998. Immediately after these two accidents, the term Electrical Wiring Interconnection System (EWIS) was designated to the electrical wiring system to replace the original Electrical Interconnection System (EIS) to emphasize the importance of the wiring of aircraft.

The physical design (also known as geometric design) of EWIS is an important part of the design process of the aircraft wiring system. This design is responsible for the definition of wire harness routes in a 3D Digital Mock-Up (DMU) environment to guide EWIS installation and to support harness manufacture.

The physical design is a challenging process because the design needs: 1) to handle an increasingly complex EWIS system that needs to interconnect with the increasing number of electronic systems caused by modern in-flight entertainment systems, More-Electrical-Aircraft (MEA) and Full-Electrical-Aircraft (FEA), and so forth; 2) to respect many design specifications issued by authorities and aircraft OEM, and multidisciplinary design rules and best practices from the EWIS supplier to guarantee aviation safety; and 3) to tackle the design changes of the airframe and other systems in a collaborative aircraft design environment. These challenges mean that designers now work under high pressure, and that the design results are error-prone and subject to a lot of changes and limited design optimization opportunities.

Design automation is a solution to these challenges considering that the EWIS physical design is a largely repetitive process and that the EWIS components are mainly selected from catalogues. Automating the design process will release the engineers from hard work and enable them to focus on the creative work and also control the quality of the design results. However, no or very limited off-the-shelf solutions which can automate the physical design, especially the 3D routing, which is considered the most critical and time-consuming phase in the physical design, have been found in either academic or industrial domains.

In this dissertation, an automation solution is proposed for the 3D routing in the physical design. This solution is enabled by a two-step, hybrid optimization strategy. The position of wire harness clamps and breakouts can be represented by the design variables of the optimization. The harness cost which design engineers want to minimize can be represented by the optimization objective function. The various design rules can be represented by the constraint functions.

Yet this optimization cannot be solved immediately since the number and initial values of the design variables, namely the number and position of clamps, are not included in the inputs of the 3D routing. One of the inputs, i.e. the harness electrical definition, defines the topology structure of these wire harnesses including the number and gauge of wires within each wiring bundle. However, the number and position of clamps used to fix the wire harnesses on the airframe are not included. Actually, the number and position of the clamps are outputs of the 3D routing. Therefore, the two-step optimization strategy is adopted.

The first step, called the Initialization, is responsible for the generation of a preliminary harness definition including the number and initial position of the clamps and breakouts. The second step, called the Refinement, is in charge of the refinement of the preliminary harness definition to achieve the minimum cost (e.g. in Euros) harness that also satisfies design

specifications.

During the Initialization step, a road map-based, bi-level optimization method has been used.

The routing environment, including the geometric components and environmental information, such as hot and vibratory areas, is discretized to generate a road map. With this road map definition, the road map-based pathfinding algorithms are able to find a path between two points without knowing the number and initial value of the clamps. Actually, the positions of the road map vertexes on the found path are the positions of the clamps.

A typical wire harness connects with multiple start points and multiple end points. This feature is described as the multi-origin/destination and a harness that has the multi-origin/destination is described as a complex harness. Conventional road map-based pathfinding methods are only able to find a path between a single origin and destination. None of them is able to find a path for a complex harness. Therefore, bi-level optimization architecture has been proposed to handle this problem. This breaks down the complex harness into branches with the coordination of the breakouts, where wires separate from each other to connect with different origins/destinations. The global/harness-level optimization is only responsible for moving breakout points to get different breakout configurations. According to the configurations, a local/branch-level pathfinding algorithm finds the best path for each branch. Then, the costs of all the branches are sent back to the global optimizer. On the basis of the local calculation results, the global optimizer determines whether the optimization is converged and if necessary it will move the breakouts to generate new configurations for the next iteration until a satisfactory result is reached.

During the generation of the road map and the local pathfinding, the design rules that have to be satisfied in the Refinement step are also taken into account to generate more promising results for the Refinement. When the pathfinding is finished, a post-process is carried out to transfer the road map-based result, which consists of some vertexes and edges of the map, to an actual preliminary harness definition. This definition contains the number and initial values of the clamps and breakouts and is given as an input to the Refinement step.

In the Initialization, the 3D routing is simplified into a road map-based pathfinding. Due to the simplification, there is no guarantee that the preliminary harness definition, which is the optimum result of the Initialization, is also the optimum result of the actual pathfinding problem. Actually, even the feasibility cannot be guaranteed. This requires an additional step, which is called the Refinement step.

In the Refinement step, the preliminary harness definition is automatically instantiated into the actual harness geometric model by a parametric geometric modelling module. The geometric model is then analysed by some analysis tools to calculate the harness cost and check the violation of the design rules. According to the analysis results, the optimizer moves the design variables to new places to get a new harness definition, if necessary. This process iterates until the optimum result is reached. Then the 3D geometric model and datasheet including the cost, weight, and violations of design rules of this optimum harness will be exported, as the results of the 3D routing.

The automation solution of the 3D routing is supported by Multidisciplinary Design Optimization (MDO) and Knowledge-Based Engineering (KBE) technologies. The MDO technology is responsible for the systematic exploration of the routing space to achieve a feasible and optimum harness design. The KBE technology takes care of all the operations involving geometry including the generation of the road map and the post-process in the Initialization step, and the parametric geometric modelling and all the analyses involving geometry in the Refinement step.

This automation approach has been implemented into a software application and several routing cases have been executed with this application. The results have demonstrated that the method is able to handle routing cases with representative geometric complexity and design constraints, and deliver 3D harness models in full automation.

Samenvatting

De elektronische apparatuur in een vliegtuig dient onderling verbonden te worden door een elektrisch bekabelingssysteem. Dit systeem is van essentieel belang voor de veiligheid van het vliegtuig en een technische storing kan fatale gevolgen hebben, zoals in het geval van vlucht TWA 800 in 1996 en Swissair 111 in 1998. Direct na deze twee ongelukken werd de term “Electrical Wiring Interconnection System” (EWIS) geïntroduceerd om het elektrisch bekabelingssysteem aan te duiden. EWIS verving de term ”Electrical Interconnection System” (EIS) om het belang van vliegtuigbekabeling te onderstrepen.

Het fysieke ontwerp (ook wel het geometrische ontwerp) van EWIS is een belangrijk proces binnen het ontwerp van de vliegtuigbekabeling. Deze ontwerpfase is verantwoordelijk voor het definiëren van kabelbundels routes in een 3D “Digital Mock-Up (DMU)” omgeving om EWIS installaties aan te sturen en om de fabricage van kabelbundels te ondersteunen.

Dit fysieke ontwerpproces vormt een uitdaging omdat het ontwerp: 1) om moet gaan met een steeds complexer EWIS systeem dat steeds meer elektronische systemen onderling moet verbinden vanwege de moderne “in-flight entertainment systems”, “More-Electrical-Aircraft” (MEA), “Full-Electrical-Aircraft (FEA)” enzovoort, 2) de vele ontwerpeisen van de overheid, vliegtuigbouwers (OEM) en multidisciplinaire ontwerpregels en best practices van de EWIS leverancier moet respecteren om de veiligheid van de luchtvaart te waarborgen en 3) de ontwerpwijzigingen van het casco en andere systemen moet aankunnen in een samenwerkende vliegtuig-ontwerpomgeving. Deze uitdagingen betekenen dat ontwerpers tegenwoordig onder hoge druk moeten werken en dat ontwerpresultaten gevoelig zijn voor fouten en wijzigingen en er een beperkt product optimalisatie mogelijk is.

Automatisering van het ontwerpproces is een oplossing voor deze uitdagingen, gezien het feit dat het fysieke ontwerp van een EWIS grotendeels een repetitief proces is met componenten die voornamelijk uit een catalogus worden geselecteerd. Door het ontwerpproces te automatiseren worden ingenieurs in staat gesteld zich te focussen op het creatieve deel van het ontwerp en op kwaliteitscontrole. Echter, er zijn geen, of slechts enkele, kant en klare oplossingen beschikbaar die het fysieke ontwerpproces kunnen automatiseren. Zeker het proces van “3D routing”, wat de meest kritieke en tijdrovende fase is, vormt de grootste academische en industriële uitdaging.

In deze dissertatie wordt een oplossing voor de automatisering van 3D routing voorgesteld. Deze oplossing wordt mogelijk gemaakt door een tweetraps hybride optimalisatiestrategie. De posities van bekabeling klemmen en “breakouts” kunnen als variabelen gezien worden in deze optimalisatie. De minimalisatie van de kosten van de kabelbundel kunnen hierin gezien worden als het doel van de optimalisatie en deze kunnen als een wiskundige functie worden gerepresenteerd. De verschillende ontwerpregels kunnen gezien worden als de randvoorwaarden waaraan het ontwerp moet voldoen.

Echter, dit optimalisatieprobleem kan niet direct worden opgelost aangezien het aantal en positie van de klemmen niet vaststaan bij de opzet van het 3D routing proces. Eén van de uitgangswaardes van het ontwerpproces, namelijk de elektronische definitie van de kabelboom, definieert de topologie van de kabelbundels inclusief het aantal en de dikte van de draden binnen elke kabelbundel. Het aantal klemmen en de uitgangsposities van de klemmen zijn echter niet inbegrepen in de uitgangswaardes. Deze zijn zelfs de resultaten van de 3D routing. Daarom is de tweetraps optimalisatiestrategie toegepast op dit probleem.

De eerste stap in de optimalisatie, de initialisatie, is verantwoordelijk voor het genereren van

de voorlopige kabelbundeldefinitie, inclusief het aantal en de posities van de klemmen en breakouts. De tweede stap, de verfijning is verantwoordelijk voor het verfijnen van de voorlopige bundeldefinitie om een kabelbundel te ontwerpen met minimale productiekosten die voldoet aan de ontwerpeisen.

Tijdens de initialisatiestap is een bi-level optimalisatiestrategie gebruikt. De omgeving waarbinnen de kabel getrokken dient te worden, inclusief de geometrische componenten en de omgevingsinformatie, zoals warme of vibrerende zones, is gediscretiseerd om een plattegrond te genereren. Door middel van deze plattegrond, kunnen zogenaamde “pathfinding” algoritmes de optimale route vinden tussen twee punten zonder het aantal klemmen en hun initiële posities te weten. De posities van de vertexen van de gevonden route vormen zelfs de posities van de klemmen.

Een typische kabelbundel verbindt meerdere startpunten met meerdere eindpunten. Dit wordt ook wel “multi-origin/destination” genoemd en wordt beschreven als een complexe kabelbundel. Gebruikelijke “road map-based pathfinding” algoritmes kunnen slechts een pad vinden tussen een enkel startpunt en een enkel eindpunt. Geen van deze algoritmes is in staat een pad te vinden in een complexe bundel. Daarom wordt de bi-level optimalisatiestrategie voorgesteld om dit probleem op te kunnen lossen. Deze strategie splitst de complexe bundels in vertakkingen met de coördinatie van de breakouts, waar de aparte kabels verder gaan naar verschillende start- of eindpunten. De optimalisatie op bundel niveau is verantwoordelijk voor het verplaatsen van de breakouts om verschillende configuraties te verkrijgen. Per configuratie zoekt een lokaal algoritme naar het beste pad, voor elke kabelvertakking. De kosten van alle vertakkingen worden vervolgens teruggegeven aan de globale optimalisatie, die aan de hand daarvan bepaalt of het ontwerp geconvergeerd is en of het noodzakelijk is om de breakouts te verplaatsen en een nieuwe configuratie te starten voor de volgende iteratie, totdat een bevredigend resultaat is behaald.

Tijdens de generatie van de road-map en het zoeken van de beste oplossing per vertakking moet aan de ontwerpregels voldaan worden om betere startpunten voor de verfijningsstap te genereren. Wanneer een pad gevonden is, wordt door middel van post-processing een voorlopige kabelbundeldefinitie gecreëerd uit de vertexen en hoekpunten van de road-map. Deze definitie bevat het aantal en de initiële waarde van de klemmen en breakouts en wordt als input aan de verfijningsstap gegeven.

Tijdens de initialisatie is de 3D routing vereenvoudigd in “road map-based pathfinding”. Vanwege deze vereenvoudiging is er geen garantie dat de voorlopige bundeldefinitie, die het optimum is van de initialisatie, ook het optimum is van het werkelijke probleem. Zelfs de daadwerkelijke fysieke haalbaarheid van de voorlopige definitie kan niet worden gegarandeerd. Hiervoor is een extra stap benodigd, de verfijningstap genoemd.

In de verfijningsstap wordt de voorlopige bundeldefinitie automatische omgezet in een bundelgeometrie door een parametrische geometriemodule. Deze geometriemodule wordt vervolgens door analyse software doorgerekend om de kosten van de bundel te bepalen en te controleren of er geen schending van de ontwerpregels is. De optimalisatie bepaalt volgens de analyseresultaten de nieuwe waardes van de ontwerpvariabelen om, indien nodig, tot een nieuwe bundeldefinitie te komen. Dit iteratieve proces wordt herhaald tot een optimaal resultaat is gevonden. Daarna wordt voor de optimale bundel het 3D model van de geometrie en een rapport met kosten, gewicht en schendingen van ontwerpregels geëxporteerd als eindresultaat van de 3D routing.

De automatisering van de 3D routing wordt ondersteund door Multidisciplinary Design Optimization (MDO) en Knowledge-Based Engineering (KBE). De MDO technologie is

verantwoordelijk voor de systematische verkenning van de ontwerpruimte om tot een fysiek haalbare en optimale bundel ontwerp te komen. De KBE technologie verzorgt alle geometrische aanpassingen, inclusief de generatie en post-processing van de road-map in de initialisatiestap. Verder is deze technologie gebruikt voor het genereren van de parametrische geometrie en het uitvoeren van geometrische analyses in de verfijningsstap.

Deze automatisatiebenadering is geïmplementeerd in een software applicatie en verschillende ontwerpstudies voor routing zijn met deze applicatie uitgevoerd. De resultaten demonstreren dat de methode geschikt is voor routing studies met een representatieve geometrische complexiteit, met randvoorwaarden voor het ontwerp en in staat is om volledig automatisch een 3D kabelbundel te genereren.

Contents

Nomenclature

Symbols

r_1^{cover}	Inner radius of a covering
r_2^{cover}	Outer radius of a covering
th_{cover}	Thickness of a covering
$u-1$	Covering start parameter on a bundle central curve
$u-2$	Covering end parameter on a bundle central curve
L^{cover}	Length of a covering
e_{cover}	Unit length density of a covering
$centrecurve_{bundle}$	Bundle central curve
c^{clamp}	Position of a clamp centre (x, y, z)
$\overrightarrow{d^{clamp}}$	Clamp axial direction vector
d_{fix}^{clamp}	Fixing distance
h_s^{clamp}	Height of a stand-off
o_s^{clamp}	Horizontal offset distance of a stand-off measured from centre of clamp
r_s^{clamp}	Radius of a stand-off
l^{clamp}	Length of a clamp tail
$\overrightarrow{n^{clamp}}$	Clamp normal vector; perpendicular to the clamp-fixing surface
r_1^{clamp}	Clamp inner radius
r_2^{clamp}	Clamp outer radius
w^{clamp}	Clamp width
w_1^{con}	Bigger cylinder width of a connector
w_2^{con}	Smaller cylinder width of a connector
r_1^{con}	Bigger cylinder radius of a connector
r_2^{con}	Smaller cylinder radius of a connector
P_{Con}	Connector position (x, y, z)
$\overrightarrow{d_{con}}$	Connector direction vector
r_{bundle}	Bundle radius
p^{bundle}	List of the waypoints of bundle central curve

d^{bundle}	List of the tangent direction attached to the waypoints
p_i^{bundle}	i_{th} item in p^{bundle}
d_i^{bundle}	i_{th} item in d^{bundle}
r^{bundle}	Radius of the bundle cross section
D^{bundle}	Diameter of a bundle cross section
D^{wire}	Diameter of the thickest wire cross section of a bundle
$d_{fix-min}^{clamp}$	Minimum allowed fixing distance
$D_{\max}^{clamp}$	Maximum allowed clamping distance
D_{ij}^{clamp}	Clamping distance between two adjacent clamping points i and j
ij	Line segment between clamp i and j
ps_i	Fixing point of clamping point i on the fixable surface
f_j	Cost of harness branch j
f_j^b	Bundle cost of harness branch j
f_j^c	Clamp cost of harness branch j
f_j^p	Covering cost of harness branch j
e_{bundle}	Bundle density
p_u	Bundle unit weight price of wire harness
L	Bundle length
Co	Cost coefficient of a branch
Cob	Sub-cost coefficient of a bundle
Coc	Sub-cost coefficient of clamps
Cop	Sub-cost coefficient of coverings
$C_{\cos t}$	Cost of one clamp
C_m	Material cost of one clamp
C_i	Installation cost of one clamp
r_{pipe}	Radius of the pipe causing flammable zones
D^{bf}	Distance between bundle centre line and centre line of fluid pipe
$P_{D_{bf}}$	Point on ij having the shortest distance to a fluid pipe
$P_{D_{bf}}^{in-zone}$	$P_{D_{bf}}$ in a flammable zone
$D_{bf}^{in-zone}$	Distance from $P_{D_{bf}}^{in-zone}$ to the pipe centre line

T_s	Tessellation size to generate road map
s_a	Safety factor of tessellation size
T	Centigrade temperature in the hot zone
Q	Heat diffusion power of heating source
th_{panel}	Thickness of a panel transferring heat
λ	Thermal conductivity
r^{hot}	Radius of the cylinder causing the hot zone
$g(n)$	Cost to move from the source to current node n in A* algorithm
$h(n)$	Estimated cost to move from n to the destination in A* algorithm
$h^*(n)$	Actual optimal distance from current node n to target node
$f(n)$	Total cost of moving via current node n in A* algorithm
$r_{\min}^{bending}$	Actual minimum bend radius of a bundle
$r_{allowed}^{bending}$	Allowed bend radius of a bundle
ς^{bundle}	Allowed bend radius ratio of a bundle
ς^{wire}	Allowed bend radius ratio of a wire
$D_{i,j}^{actual\_sep}$	Minimum 3D distance between the central curves of branches i and j
$C(x)_{collision}$	Geometric collision constraint function
$C(x)_{clamping}$	Clamping distance constraint function
$C(x)_{fixing}$	Fixing distance constraint function
$C(x)_{bendradius}$	Bend radius constraint function
$C(x)_{sep}$	Harness separation constraint function
$C(x)_{clearance}$	Harness clearance constraint function
w_i	Weight of the constraint function i
$w_{bending}$	Weight of the bend radius constraint function
$f(x)$	Cost of a wire harness

Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AWG	American Wire Gauge
BREP	Boundary Representation
CAD	Computer-Aided Design
CDAT	Clamp Distance Analysis Tool
CSBPC	Curve Segment Bend Radius Pre-Calculation
DA	Design Automation
DEE	Design and Engineering Engine
DMU	Digital Mock-up
ECAD	Electrical Computer-Aided Design
EGS	EMI, Group, and Subsystem
EMI	Electro-Magnetic Interference
EWIS	Electrical Wiring Interconnection System
GCAT	Geometric Collision Analysis Tool
GUI	Graphical User Interface
HBRAT	Harness Bend Radius Analysis Tool
HDEE	Harness Design and Engineering Engine
HMMG	Harness Multi-Model Generator
HCAT	Harness Cost Analysis Tool
HCLAT	Harness CLearance Analysis Tool
HSAT	Harness Separation Analysis Tool
KBE	Knowledge-Based Engineering
MCAD	Mechanical Computer-Aided Design
MDA	Multidisciplinary Design Analysis
MDO	Multidisciplinary Design Optimization
MRA	Main Route Architecture
NURBS	Non-Uniform Rational Basis Spline
OEM	Original Equipment Manufacturer
PCB	Printed Circuit Board
PDM	Product Data Management
PLM	Product Lifecycle Management
VR	Virtual Reality
STEP	Standard for the Exchange of Product model data

UID	Unique Identification Number

Chapter 1 Introduction

1.1 Electrical Wiring Interconnection System – a complex aircraft system

Current aircraft accommodate many pieces of electronic equipment due to the application of the Fly-By-Wire (FBW) system and other on-board systems, such as communication, navigation, terrestrial landing aids, surveillance, indication, and the air data computer[2]. The amount of electronic equipment is still increasing due to the new generation More-Electrical-Aircraft (MEA) and Full-Electrical-Aircraft (FEA)[3], where more or full conventional flight control components will be replaced by electronic components.

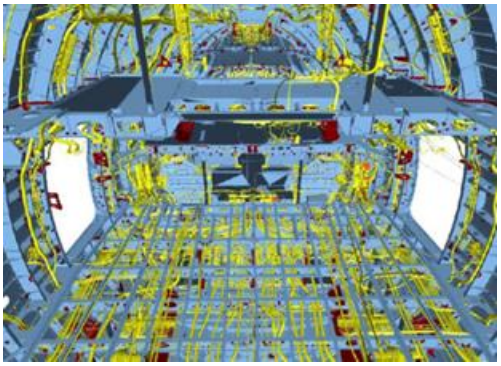


Figure 1-1 A part of EWIS inside the fuselage of the A380[1]

These electronic components need to be interconnected by a so-called Electrical Wiring Interconnection System (EWIS) to make the aircraft ready to fly. A EWIS consists of a large number of wire harnesses, circuit breakers, and accessories. The wire harnesses are considered the most complex part in a EWIS due to their total length, their limited routing space, and the amount of harness components. For instance, the EWIS of the Airbus A380 contains 530km cables, 100,000 wires and 40,300 connectors[4]. This impressive large number of harnesses has to be routed inside a relatively narrow space, while taking into account the room reserved for payloads and equipment. Figure 1-1 clearly denotes that the harnesses are only allowed to be routed along the floor and between the fuselage skin and interior panels. An integrated wire harness comprises various components, such as connectors, bundles, coverings, and clamps. These components contain many sub-components, such as wires and pins and these sub-components also have lots of different types, such as different wire colours and gauges. A detailed example of some harnesses is given in Figure 1-2.

Figure 1-2: Wire harnesses in the Russian Sukhoi Superjet 100[5]

However, although technology such as data buses decreases the total wiring length by 6km for the Boeing 737 NG compared with its predecessor[6], the scale of the EWIS of a typical civil transport aircraft is still considered to increase further due to the rise of electronic equipment caused by MEA and FEA and lack of reliable alternative interconnection solutions. The so-called Fly-By-Wireless cannot replace EWIS in the near future because of the electromagnetic susceptibility and safety reasons [7].

1.2 Electrical Wiring Interconnection System development – a challenging design process

The design process of such complex EWIS can be split into three parts, the electrical design, the physical design, and the manufacturing design. These are illustrated in Figure 1-3.

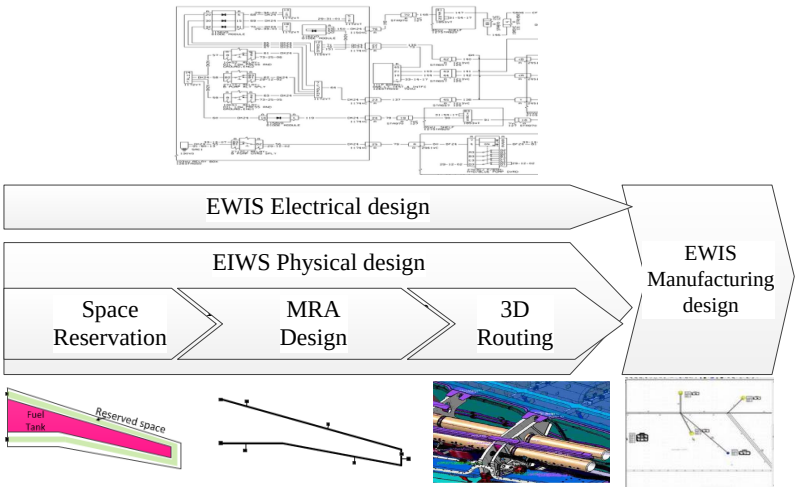


Figure 1-3: The EWIS design process

The electrical design defines electrical diagrams to denote the power and data signal interconnection between the various electronic components. The physical design, also known as the installation design, develops 3D harness routes according to the electrical diagrams and design specifications. The manufacturing design, also known as the formboard design, creates the 2D harness drawings according to the 3D harness geometric models. These 2D drawings will be used for the placement of wires in the manufacturing process. The physical design consists of three design phases, called the space allocation, Main Route Architecture (MRA) design, and 3D routing. In the space allocation, some spaces are reserved exclusively for the accommodation of wire harnesses. The reserved space connects as much electronic equipment as possible so that the designers can route the wire harnesses within the allocated space to avoid costly and time consuming renegotiations with designers of other subsystems in the 3D routing phase. In the MRA design, the MRA (i.e. the harness motorway) is generated in the allocated space. The MRA expresses the relatively detailed routes among the electronic equipment but without considering details, such as the clamping method. In the 3D-routing phase, the detailed harness wiring system required to connect various electronic components is defined, according to the allocated space and MRA in the aircraft, the electrical definition established in the electrical design phase, and the design specifications.

The EWIS physical design, especially the 3D routing, is essential to aircraft safety. The malfunction of wiring, which is the direct output of the 3D routing, can cause fatal results,

such as the accidents with TWA Flight 800 in 1996[8] and Swissair 111 in 1998[9]. Actually the authorities also gradually understood the importance of the wiring for aircraft safety after these two fatal accidents. They designated the term Electrical Wiring Interconnection System (EWIS) to the wiring system to replace the original Electrical Interconnection System (EIS) to emphasize the importance of wiring. [10]

The EWIS physical design is also very challenging because of the intrinsic structural complexity of the EWIS system and also two other reasons presented below.

1) The design process of EWIS needs to comply with the many design specifications stipulated by certification authorities, such as the FAA (Federal Aviation Administration), U.S. Department of Defense, and EASA (European Aviation Safety Agency), for safety reasons. These specifications belong to different domains, such as heat transfer, mechanical engineering, and electrical engineering. CAD (Computer-aided Design) engineers working on the EWIS design must be familiar with these specifications in order to perform their work correctly.

2) EWIS design is influenced by the frequent design changes of an aircraft. The EWIS design generally starts at the beginning of the aircraft preliminary design and goes in parallel until the very end of the aircraft development process. This design suffers from the frequent changes on the aircraft structure and the update of other systems caused by, for example, the customization of airliners. A much greater number of wiring changes than expected resulting from modifications to electrical systems and structure have led to the delay of the A380 that was almost ready to be delivered.[11] Due to the frequent design changes and the limited lead time, EWIS engineers work under high pressure and their design is error-prone. They need to be ready for last-minute changes, which are common and expected nowadays in aircraft such as the Boeing 787[12].

Such challenging EWIS design is still mostly performed manually by experienced engineers with the partial support of CAD systems.

1.3 Potential solutions for 3D harness routing

Considering the fact that a large number of harness components are selected from catalogues and that the nature of the wire harness design work is largely repetitive and mostly rule-based, in the authors' opinion, there are a lot of opportunities to automate a significant part of the design process to release the design engineers from the repetitive hard work and potentially increase their creativity. Indeed, some Electrical CAD (ECAD) software tools, such as Mentor Graphic, Zuken E^3 and Cadence are already available to support the EWIS electrical design automation. In EWIS physical design, however, current leading Mechanical CAD (MCAD) software tools used in industry are not able to generate 3D wire harness models automatically, and still demand a lot of manual work by expert designers. In academia, a number of researchers [13-17] have focused on the design automation and the computer-aided design of EWIS. However, as detailed in Sub-section 2.4.3, no or very limited mature solutions have been found to automate the EWIS physical design, especially the 3D-routing phase, which is most complex and time consuming (see Table 1-1).

In this dissertation, an innovative approach is proposed to enable the automation of EWIS 3D routing. This approach is based on a hypothesis that the harness 3D-routing problem can be modelled and solved as an optimization problem. The objective function to minimize represents a cost function that accounts for both the cost of the wire harness bundles and the cost of the harness accessories, such as clamps, connectors, and protection layers, required to fix the bundles in the aircraft and to protect the harness in harsh aircraft areas (e.g. hot, vibratory, wet areas). The design variables represent the coordinates of the wire harness

clamping points, namely the position of the various fixing elements used to fasten the cables to the aircraft structure. The optimization parameters represent the position of the various points where the harness has to connect (e.g. the production breaks where a harness is connected to another harness, or the receptacles of the various electronic systems) and by the required number and gauge of the wires or cables. The various design rules, such as the minimum allowed bend radius of cables, maximum allowed distance between adjacent clamping points, minimum allowed distance between a cable and its support structure, and requirements on the cable protection sleeve for certain environmental conditions are formulated as constraints for the optimization problem.

Table 1-1: Labour figures on the physical design for wire harnesses*

Design phase	Staffing level	Lead time	Design effort proportion (% of lead time)		
			Actual design	Trade studies	Integration
Space allocation	20%	2 months	70 %	10 %	20 %
MRA design	30%	4 months	25 %	25 %	50 %
3D routing	100%	8-10 months	80 %	5 %	15 %

Actual design – the work that engineers do to the final (3D routing) design
Trade studies – find out the best design from competing design options
Integration – a non-design activity that entails coordination, discussion, and agreements between the EWIS design group and other aircraft-design involved integrating parties, such as Hydraulics, Fuel, and Structures
***This table contains experience-based figures and is courtesy of Fokker Elmo**

The challenge of solving the optimization problem described above is that the number and initial values of the design variables, i.e. the number and initial positions of the harness clamping points is not known a priori. Indeed, the number and positions of clamping points are the output of the harness routing and actually the goal of the optimization problem. In order to handle this challenge, a two-step, hybrid optimization process has been devised. In the first step, called *Initialization*, a grid of potential clamping points and their connections, addressed as a road map, is generated in front of the structural elements where the harness is allowed to attach. Then an optimization algorithm is applied to route a simplified harness model on such a grid. As the results of this Initialization step, a preliminary harness definition that includes the number and position of clamping points and breakout points is obtained. This step is similar to car navigation where a road map is built and a pathfinding is applied on this road map. At this point, as the number of design variables and their initial values are known (i.e. the number of clamping points and breakout points and their coordinates), the second optimization process can be applied. In this second step, called *Refinement*, a detailed geometric model of the wire harness is used, and the design variables are varied continuously in order to minimize the cost objective function, while satisfying all the constraints.

1.4 Aim of this research

The aim of this research is to answer the question:

Whether the EWIS 3D routing can be automated, namely: whether the previously proposed approach is feasible.

In order to answer this question, a detailed automatic routing approach needs to be developed. The development of the approach mainly includes the development of the Initialization approach, the Refinement approach, and the harness geometric modelling capability.

In order to achieve full automation, these approaches need to be developed and implemented into a software application. The software application needs to be able to automate the generation of 3D wire harness models, by capturing and systematically reusing the experts' knowledge; and to automatically update wire harness models when changes occur in the routing environment and/or in the electrical design phase of the EWIS.

1.5 Outline of dissertation

The structure of this dissertation is illustrated in Figure 1-4.

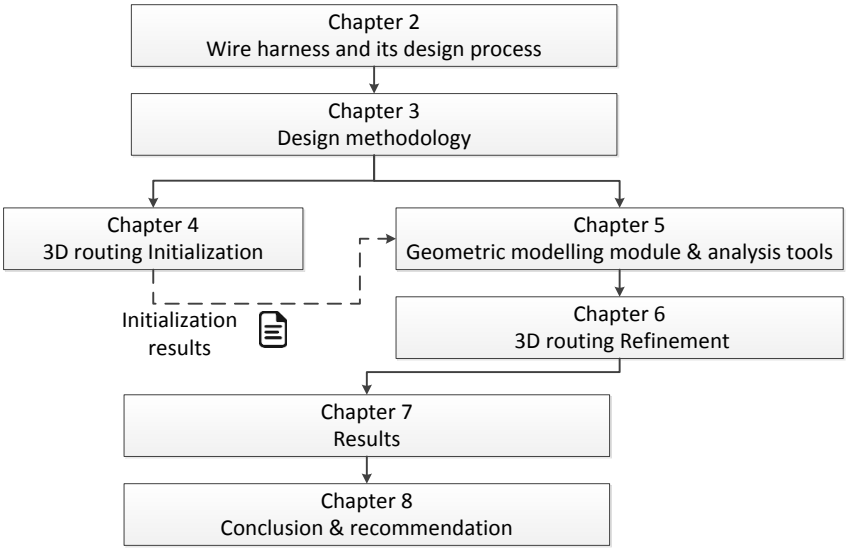


Figure 1-4: Structure of the dissertation

Chapter 2 first presents an overview of EWIS, its design process, and some typical design rules that needs to be respected by the design engineers. Then it discusses the limitations of the current harness design method and proposes a potential solution.

Chapter 3 introduces the innovative approach used in this research to enable the automation of 3D harness routing. It firstly presents an optimization problem definition representing the 3D harness routing problem. Then a hybrid, two-step optimization (i.e. the Initialization and Refinement) strategy used to solve the optimization problem is elaborated. The advanced design methods that support the previous hybrid optimization are discussed. Finally, a framework is presented to support the software implementation of this innovative approach.

Chapter 4 covers the harness Initialization step. It first introduces some available pathfinding methods and then details the method used in this research to support the harness Initialization.

Chapter 5 presents the development process of harness parametric modelling modules that are used to automatically generate harness geometric models, and also very briefly introduces various analysis tools used to analyse the performance of these previously generated harness models to support the harness Refinement.

Chapter 6 covers the harness Refinement step. It first discusses the limitations of harness Initialization, namely the motivation of the Refinement. Then the detailed harness optimization problem definition is given. The solution of the optimization problem is

presented next and finally the technical implementation of the solution is given.

Chapter 7 proves that the proposed approach is able to automate 3D harness routing via implementation of some representative routing cases.

Chapter 8 concludes this research and gives recommendations for future work.

Chapter 2 Wire harness and its design process

Automating the 3D-routing phase of the Aircraft EWIS (Electrical Wiring Interconnection System) is the objective of this research. Understanding the wire harnesses and their design, especially the 3D harness routing, is the starting point. This chapter first presents an overview of wire harnesses and their development process, including some of the typical design rules that need to be followed by designers. Then the 3D-routing phase is detailed and the challenges in this phase are discussed. Finally, the potential solution of the challenges is presented.

2.1 Overview of the aircraft EWIS and its development process

2.1.1 Overview of the aircraft EWIS

An aircraft EWIS consists of a set of wire harnesses, circuit breakers, clamps and brackets, and additional protection. They are introduced in the following parts.

2.1.1.1 Wire harness

Wire harnesses are used to transfer signals and power between different pieces of equipment, such as the electronic devices, sensors, and the power plant. A wire harness consists of various components, such as a bundle, bundle protections, clamps, brackets, and connectors. The breakout is also considered as a harness component here since it is an important point where wires separate from each other, although it is not an actual component. A harness overview is presented in Figure 2-1 with two annotated pictures.

Figure 2-1: A harness installed in an aircraft (left); and a harness with an independent view (right) [18]

Clamps and brackets can be considered either as a part of wire harnesses since they are used only when harnesses are routed or as independent components since they are not manufactured together with the harness. Here, they are considered as parts of harnesses, since this research focuses on the harness design and in the design process, the placement of clamps and brackets will significantly impact the harness design result.

2.1.1.2 Circuit breaker/fuse

Any well-designed electrical system is protected by a circuit-protective device. According to design rule AC 43.13-1B, electrical wires also needs to be protected with circuit breakers or fuses located as close as possible to the electrical power source bus [19]. The function of a

circuit breaker and fuse is to prevent secondary disasters caused by electrical breakdown such as short circuits. Circuit breakers with some connected wires are shown in Figure 2-2.

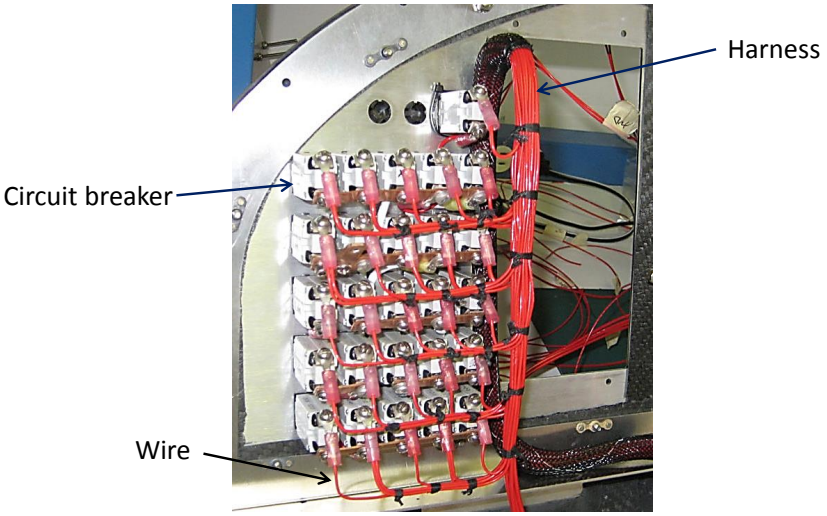


Figure 2-2: Circuit breakers connected by wires[20]

2.1.1.3 Additional protection

Additional protection components protect harnesses from danger. They are designed together with the harnesses but are not considered as a harness part. One example of the additional protection is the grommet, as shown in Figure 2-3. This grommet installed on the edge of a lightening hole is used to prevent abrasion between the harness and a sharp edge.

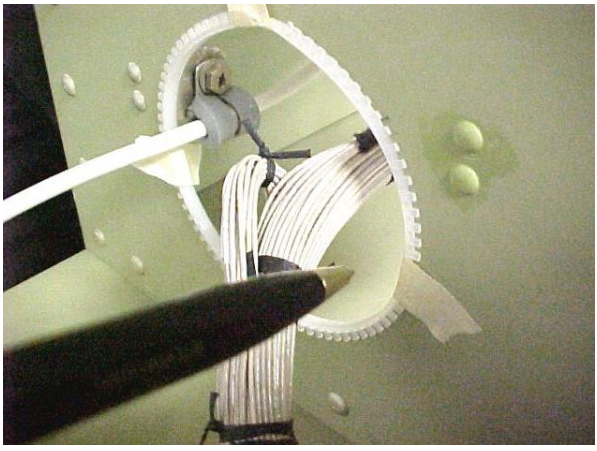


Figure 2-3: An anti-chafing grommet on the edge of a lightening hole[21]

2.1.2 Overview of the entire EWIS development

The design of an aircraft is generally divided into three phases, namely: the conceptual design, preliminary design, and detailed design. The conceptual design defines the overall aircraft performance goals and generates a conceptual aircraft configuration; the preliminary design refines the baseline design concept via multidisciplinary parametric analysis and freezes the global design with the possibility to change a few details; the detailed design

divides the entire aircraft into detail, evaluates the performances accurately, fine-tunes the design, and releases the production drawings[22].

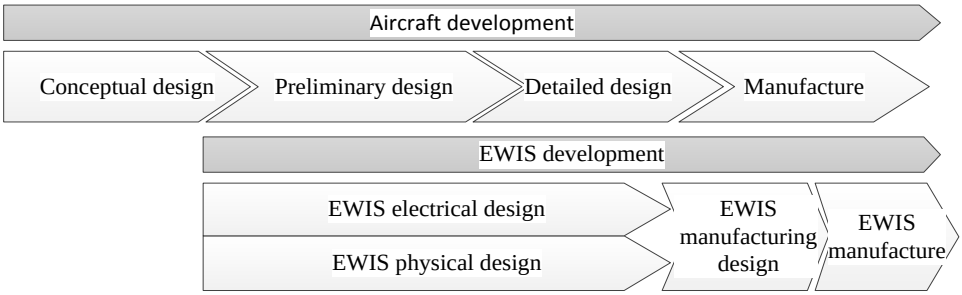


Figure 2-4: Overview of the EIWS development (below) in the overall aircraft development process (top)

The development of EWIS, shown in Figure 2-4, starts between the conceptual design and the preliminary design and goes in parallel with the aircraft development until its end. The EWIS development contains three design parts, namely: electrical design, physical design, and manufacturing design. The electrical design generates the electrical diagram that shows the logical connectivity of the harnesses (see picture A in Figure 2-5). It works out which wires/cables should be routed among the electrical and electronic equipment. The EWIS physical design answers how to route the wires/cables among the equipment. It converts the electrical definition to the actual harness geometric models that are part of the aircraft Digital Mock-up (DMU). These geometric models, as shown in picture B of Figure 2-5, are necessary to support the harness's manufacture and installation. The manufacturing design, also known as the formboard design, creates the 2D harness drawings (see picture C in Figure 2-5) according to the 3D harness geometric models. These 2D drawings will be used for the placement of wires in the manufacturing process[23].

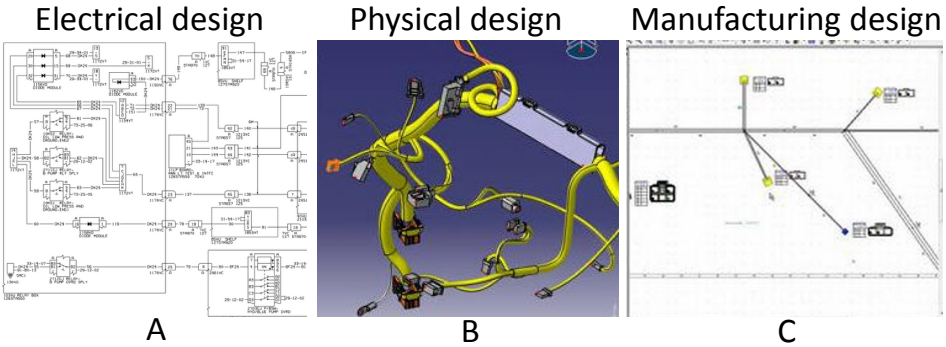


Figure 2-5: Results of the electrical design, physical design, and manufacturing design

In practice, the development efficiency of such complex systems as the aircraft EWIS can be increased by using the design automation method. Indeed, some off-the-shelf Electrical CAD (ECAD) software tools, such as Mentor Graphic's Capital tools, Zuken E^3 series, and Cadence tools, are already able to support the automation of the EWIS electrical design. Using these kinds of tools, the design engineers can automatically generate electrical definitions and integrate these abstract definitions into the physical environment to generate a more detailed wiring design. The automation of EWIS manufacturing design, namely transferring 3D harness models to 2D drawing has been studied by Van den Berg[23].

However, current mainstream Mechanical CAD (MCAD) software tools are not able to

automate the physical harness design yet when considering the representative design requirements (see Section 2.4). Surprisingly, automating the physical harness design has not received much academic attention in spite of the technical challenges and associated improvement opportunities. Ritchie et al. combined Virtual Reality (VR) with CAD to get an immersive design environment for wire harness aiming at increasing the user experience and ultimately increasing the productivity.[17] Conru formulates the harness path planning as a graph search problem and provides a genetic algorithm-based approach to place the harness breakouts on the vertexes of the 3D graph that is generated manually and represents the general shape of the routing space[14]. Van der Velden et al. developed a 3D volume mesh-based harness-routing method to automate the harness path planning. Nevertheless, in these publications, no or very limited solutions are presented that enable the automation of the physical harness design when taking into account typical design requirements.

Therefore, the automation of the physical harness-design process is studied in this research. Understanding the physical design process is the starting point of its automation. In order to better understand this process, the characteristics of the aircraft EWIS are presented first in Section 2.2 before detailing the physical harness design in Section 2.3.

2.2 EWIS hierarchy structure and basic components

2.2.1 Overview of the EWIS hierarchy structure

A complete aircraft EWIS is too huge to be designed, manufactured, and installed as a whole. In practice, such a complex system is decomposed into various EWIS parts located in relatively enclosed small spaces to facilitate the manufacture and installation. The enclosed spaces are known as wiring zones and their enclosed EWIS parts are referred to as zone EWIS. The environmental conditions, such as heat, vibration, and wetness, may be different in different zones and consequently the characteristics of different zone EWISs may also be different due to the respective protection and support requirements.

The harnesses in different zones connect with each other via so-called production breaks, shown in Figure 2-6[24]. These production breaks generally are the only interfaces between the adjacent zones.

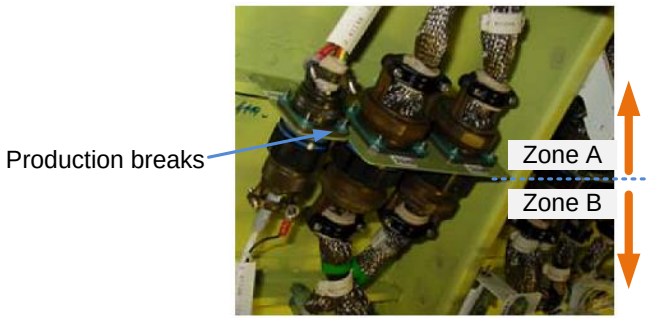


Figure 2-6: Example of production breaks

A zone EIWS comprises some wire harnesses that connect with receptacles of the production breaks and the equipment in this zone. A typical wire harness connects with multiple start and end receptacles. This feature is referred to as multi-destination (multi-origin). A multi-destination harness contains some breakouts and the breakouts can break down the entire harness into various branches. A harness having only one origin and one destination is called a simple harness.

A branch is defined as a bundle located between breakouts and/or connectors and other harness components, such as connectors, protection layers, and clamps on this bundle. The harness components such as a bundle and connectors can be further broken down into sub-components, such as wires and pins. The entire EWIS hierarchy structure is illustrated in Figure 2-7.

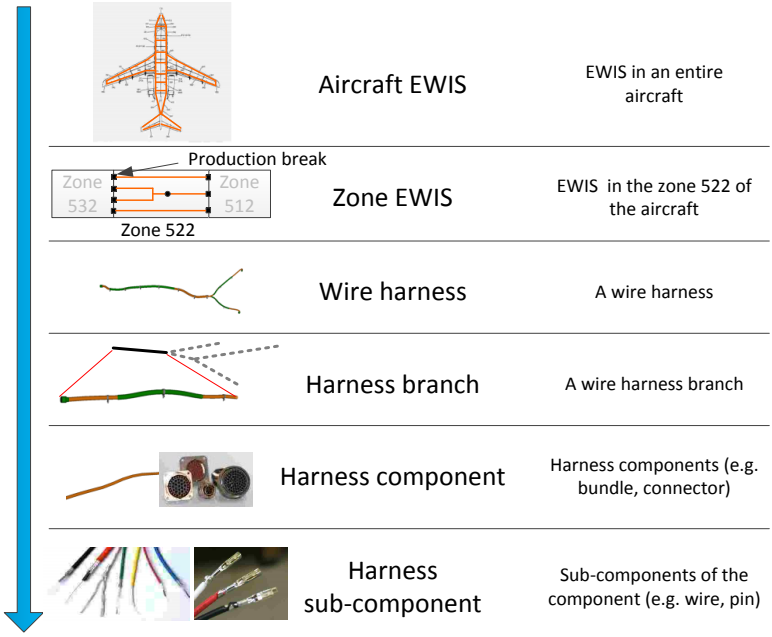


Figure 2-7: Hierarchy structure of EWIS

The harness components are considered as the basic components of the hierarchy structure. These components significantly influence the position, shape, weight, cost and so on of a harness and they constitute the harness branch, harness, and ultimately the entire aircraft EWIS. In the next sub-section, the basic components are introduced.

2.2.2 Introduction of the basic harness components

- **Harness bundle**

A harness bundle is a set of wires grouped together. The shape of the bundle cross section is determined by the wires and the wire combination form. In practice, many combination forms exist and three of them are illustrated in Figure 2-8.

Wires can differ in terms of colour, length and gauge. The colour of a wire depends on its functions and colour code system, which uses colours to designate the wire function. Its length depends on the position of the wiring points to be connected with and the routing environment. The wire size, namely its diameter or cross-sectional area, is determined by the wire length and the circuit voltage. The size is generally chosen from some recommended values listed in design standards, such as the American Wire Gauge (AWG), shown in Appendix A.

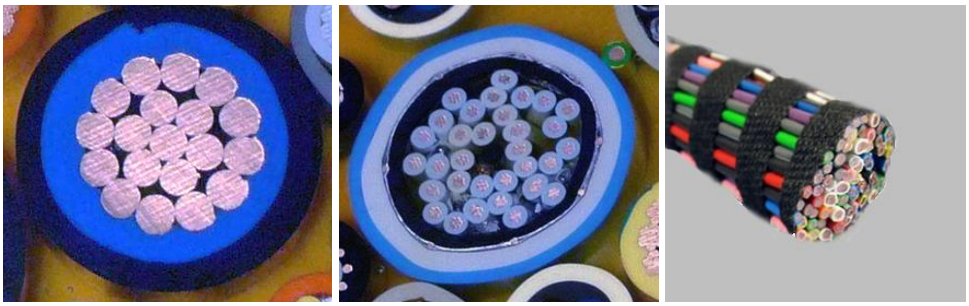


Figure 2-8: Different harness cross sections

- **Breakout point**

The wire harness splits into two or more branches at breakout points. There are three types of breakouts existing in aircraft wire harnesses, namely: Y, T and complex types.[21]

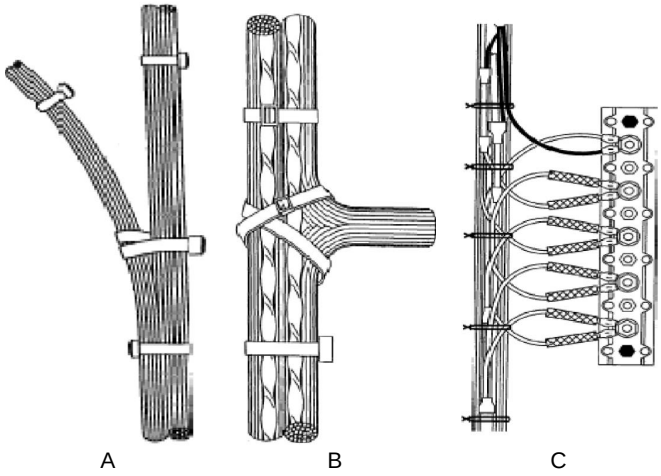


Figure 2-9: Breakouts types (A: type Y; B: type T; C: complex type)

A breakout is not a physical component of the harness but an important feature that significantly affects the shape of a harness. Therefore, here it is also considered as a component.

During the electrical design, the logical relationship of breakouts (i.e. which branches share a breakout) is worked out during the generation of the harness electrical definition. In the physical design, the 3D position of breakouts that influence the start and/or end points of branches and consequently influence the length of each bundle will be solved.

- **Connector**

In order to facilitate the connection between the harness and equipment, connectors are used together with the receptacles at production breaks and electrical equipment. During the physical harness design, the connectors will be placed in the receptacles which represent the start and end points of the harness. These connectors are generally selected from catalogues by designers. The FAA Advisory Circular 43.13-1B[19] introduces different standard

connector types. Some of them are shown in Figure 2-10.

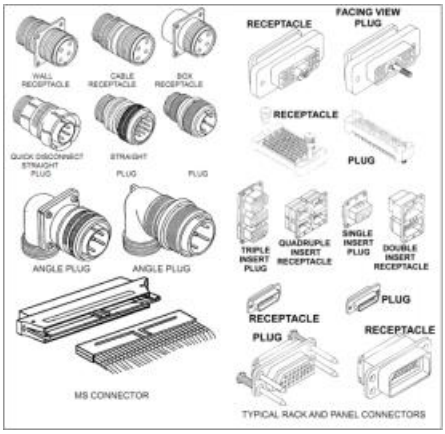


Figure 2-10: Different connector types [19]

- **Support components**

The harness support components include the primary and secondary supports.

The main functions of the primary support are to carry the weight of the harness, to fix the harnesses along the designated path, to keep proper slack to allow maintenance, and to prevent mechanical strain. It also provides the required airframe-to-harness clearance to prevent harness damage, such as chafing with the aircraft structure or being cut off by movable parts. The support also needs to guarantee that a failure of the harnesses, such as a broken wire, will not cause secondary damage, such as an electrical arc, on the aircraft.

The primary support includes three components, namely: clamps, brackets, and stand-offs. Clamps spaced at intervals provide primary support directly to the harness. The stand-off keeps clearance between a harness and the airframe or other components to protect the harness from potential damage. The bracket has a similar function to the stand-off. In addition, it bridges the angle difference between the normal direction of the airframe member (Dir A) and the direction that a clamp is fixed with (Dir B), as shown in Figure 2-11 (left-hand side).

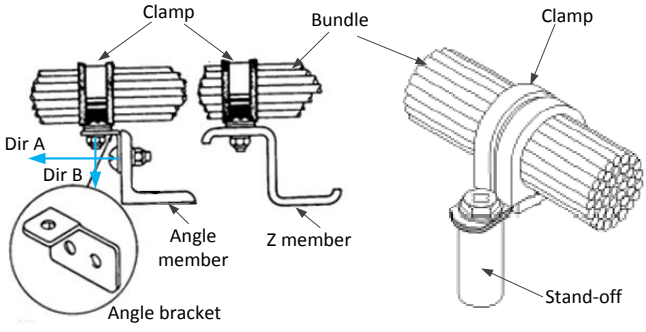


Figure 2-11: Application of clamps and brackets[19](Left) and stand-off [21] (Right)

The secondary supports, such as tying tape, tie wraps, plastic tape straps, insulation tape and protective outer covering[25], are used together with the primary supports to provide

additional support to the harness. An application example of the primary and secondary supports is illustrated in Figure 2-12.

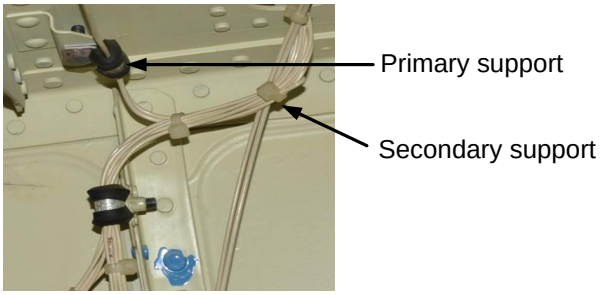


Figure 2-12: Primary and second support on a harness

- **Protection and shielding**

Wire harnesses can be routed with or without a protection layer (i.e. protected or open harnesses). Open harnesses installed inside the aircraft without any protection are easy to maintain, are lightweight, and have a low cost. These open harnesses can be found for example in the fuselage interior where temperature and pressure are well regulated (see Figure 2-12). Protected harnesses are used in hazardous zones, such as high temperature and wet zones. In these areas, the harness needs to be protected by braid, tape, conduit, plastic tubing (shown in Figure 2-13), and so on to avoid degeneration, abrasion, corrosion, or other type of damage.

Using these types of protection and shielding will increase the total weight and cost of a harness. Provided that a harness does not have to be routed through the hazardous zones, a detour can be taken to avoid the use of these types of protection but sometimes at the cost of a longer harness.

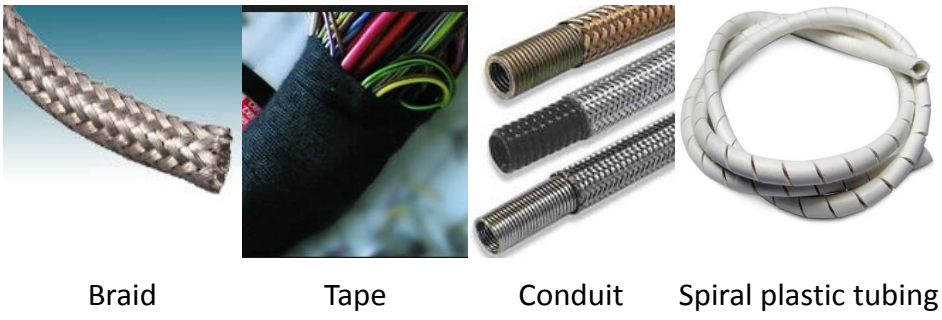


Figure 2-13: Examples of harness protection

2.3 The EWIS physical design process and related design rules

The aim of the physical harness design is to transfer the harness electrical definition into the actual harness geometric models. As shown in Figure 2-14, the input of the physical design includes the harness electrical definition, design specifications, and the routing environment. The electrical definition tells which wires (including the wire number and gauge) are routed from the start connectors to the end connectors (from-to format). The number and logical position (not the 3D position) of the breakout are also defined. The design specifications state

the design rules that need to be followed by the designer. The routing environment includes geometric models such as aircraft DMU where harnesses are going to be routed and some environmental information such as the high temperature areas (i.e. the hot zone) or the vibration areas (i.e. the vibratory zone).

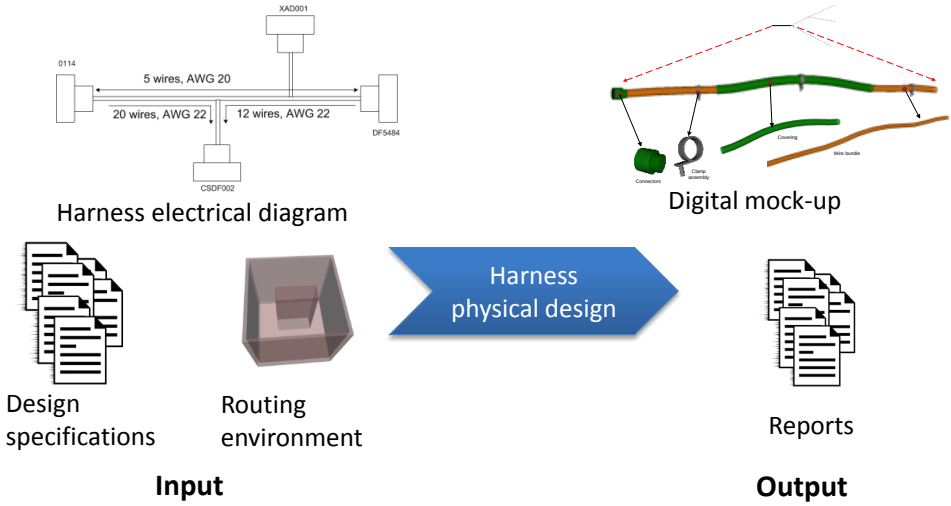


Figure 2-14: Inputs and outputs of the harness physical design

The EWIS physical design takes care of fixing the harness defined in the electrical diagram to the routing environment while at the same time satisfying the design rules. The design outputs include the harness geometric model which reflects the electrical definition but also contains more detailed information such as where to put the clamps and what the 3D position of the breakout is. In addition, the design report that contains the harness information, such as the length of wires and the weight of the harness, will also be exported.

2.3.1 Three phases in the EWIS physical design

The physical design is carried out in three sequential phases, namely: 1) Space reservation, 2) Main Route Architecture (MRA) design, and 3) 3D routing, as shown in Figure 2-15. From the space reservation to the 3D routing, the design engineers gradually transfer logic connectivity requirements to the actual geometric path of the harnesses.

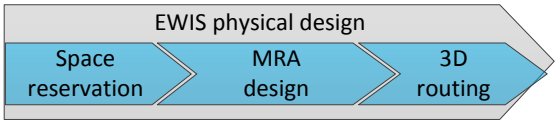


Figure 2-15: The three phases of EWIS physical design

2.3.1.1 Space reservation

At the very beginning of the wire harness design, harness design engineers work together with the OEM and other sub-system engineers to allocate space in the aircraft model for wire harness accommodation, according to the aircraft conceptual configuration, domain rules, and their own experience. In the space reservation phase many details, such as airframe components, are not available yet, and therefore they cannot be taken into account. Only the conceptual aircraft configuration (e.g. the wing area) and knowledge of previous similar

aircraft (e.g. leading and trailing edges are preferred for harnesses to route) will be used to support the design. Figure 2-16 shows a reserved space in an aircraft wing.

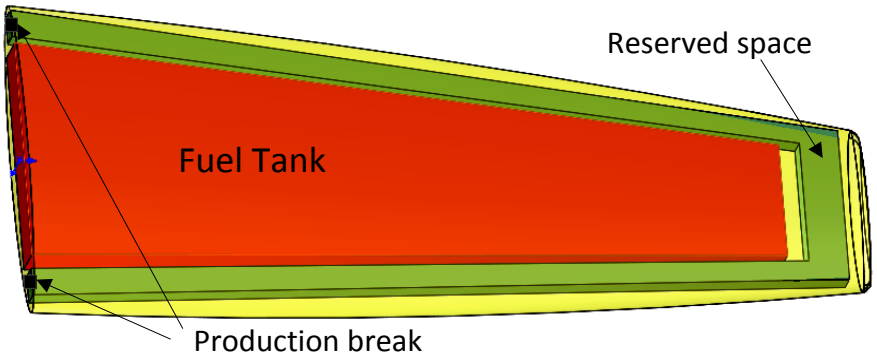


Figure 2-16: A reserved space inside an aircraft wing

2.3.1.2 Main Route Architecture (MRA) design

The MRA can be considered as the harness motorway inside an aircraft and will be the input for the 3D harness routing. During the MRA design, more detailed information of an aircraft is provided by the OEM and the suppliers of other systems. For instance, the layout of avionic systems provides the position of the receptacles. The MRA is generated according to these positions, the design specification, the previously reserved space, and also the aircraft DMU. During the MRA design phase, different harnesses that need to be segregated are physically separated from each other to meet the redundancy requirements for safety. The MRA indicates the general position of harnesses inside the aircraft and provides the estimation of harness length. With this estimated harness length and the voltage drop charts, the gauge of wires can be selected and accordingly the cross-sectional diameter of harness bundles can be calculated. Figure 2-17 illustrates a harness MRA inside a wing.

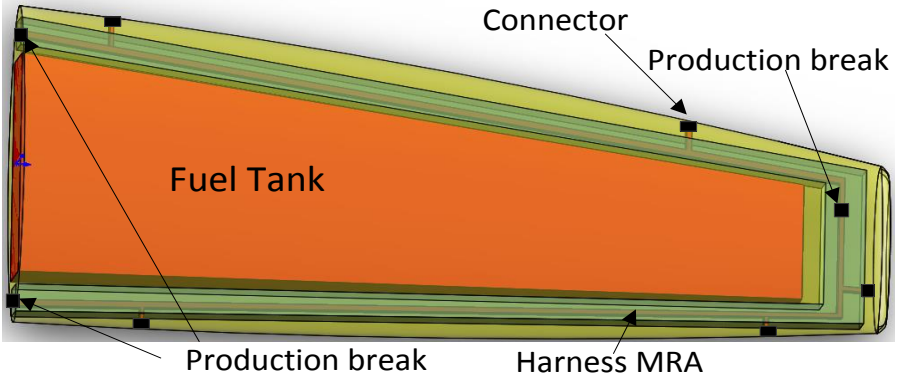


Figure 2-17: A MRA inside an aircraft wing

2.3.1.3 3D routing

Detailed 3D wire harnesses, shown in Figure 2-18, will be generated in this phase to support the manufacture and the installation.

In the 3D harness routing phase, very detailed information of the aircraft such as the thickness of ribs and spars, the position of lightening holes, and the position of equipment is available. In addition, the allocated space and the MRA are given as inputs in this phase. Design engineers will generate the wire harness DMU within the reserved space while at the same time satisfying design rules such as the bend-radius violation- free rule. These detailed harness models will then be exported to support the manufacture and the installation. As well as the geometric model, the design report containing the harness information, such as weight and cost, will be generated.

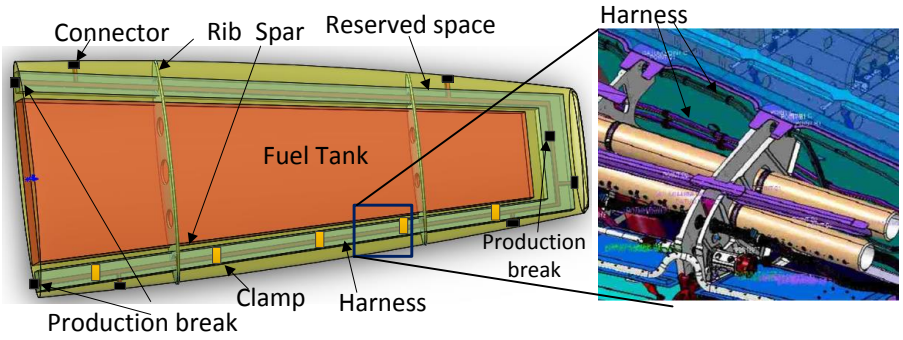


Figure 2-18: The 3D-routing result in an aircraft wing

2.3.2 Typical EWIS design rules

During the harness physical design, many design rules need to be followed by the design engineers. Understanding these rules facilitates a further understanding of the physical design. These design rules are described in the design specifications (see some examples in Table 2-1) issued by the authorities to guarantee the safety of the aircraft.

Table 2-1: Examples of design specifications from the authorities

Index	Source	Title
MIL-C-27500	US Military standards	Cable, Power, Electrical and Cable Special Purpose
MIL-I-3190	US Military standards	Insulation Sleeving, Electrical, Flexible, Coated,
MIL-T-43435	US Military standards	Tape, Lacing and Tying
AC 43.13-1B, Chapter 11	FAA	Acceptable Methods, Techniques, and Practices on Aircraft Electrical Systems
AC 25-16	FAA	Electrical Fault and Fire Prevention and Protection

Not all the rules will be handled in this research. Instead, only the most general and representative ones are selected. General rules are the rules which need to be satisfied in any part of the routing environment, such as the fuselage, wing, and tail. Representative means the solution of a rule also works for similar rules. For example, the geometric envelope of moving parts in Sub-section 2.3.2.4 can be for any geometric component which needs to keep a certain clearance from harnesses. In the following part of this sub-section, these selected rules are presented.

2.3.2.1 The bundle diameter calculation method

The diameter is an essential parameter of a bundle. It is not only used to generate the bundle geometric model but also used to check the violation of design rules such as the bend radius violation-free rule (see Sub-section 2.3.2.2). In principle, this value is given as an input to 3D routing. However, sometimes the input of 3D routing may only include the number and gauge of wires. Therefore, a bundle diameter calculation method is needed.

The method to calculate the bundle diameter on the basis of its wires/cable is complex since there are many wire/cable types, the deformation of these cables, and many combinations of these wires/cables. The accurate calculation method needs some years' research which includes a large number of experiments and simulations[26]. It is unrealistic to include such complex and time-consuming work in this automatic 3D-routing research.

Therefore, a simplified bundle diameter estimation approach presented in the specification MIL-C-27500[27] is adopted here. This approach is based on an assumption that all the wires of a bundle have the same gauge. It is able to estimate the diameter of a bundle that has up to 10 wires, as shown in Figure 2-19. For example, if a bundle has only 1 wire, the diameter of the bundle is 1 times the wire diameter (i.e. the bundle diameter ratio); if a bundle has 4 wires, the bundle diameter is 2.414 times the wire diameter. These bundle diameters are also the diameters of the so-called *Clear Hole*, which are the smallest circular hole that the bundle can go through.

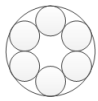
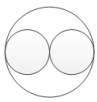
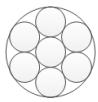
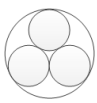
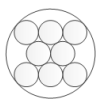
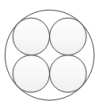
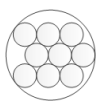
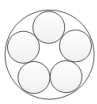
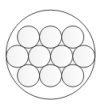
Number of wires	Bundle diameter ratio	Illustration	Number of wires	Bundle diameter ratio	Illustration
1	1		6	3	
2	2		7	3	
3	2.155		8	3.646	
4	2.414		9	3.8	
5	2.701		10	4	

Figure 2-19: Relations between the number of wires and the bundle diameters

The diameter of one wire is calculated by Equation (2.1)[28]. *wiregauge*, the input of this function, is the wire gauge selected from wire gauge table in Appendix A.

$$DiameterOfWire = e^{(2.1104 - 0.11594 \times wiregauge)} mm \quad (2.1)$$

Then the diameter of the bundle can be calculated as a product of the wire diameter and the

bundle diameter ratio. For example, if a bundle contains 4 wires whose gauge are 10, then the bundle diameter is $2.414 \times e^{(2.1104 - 0.11594 \times 10)} = 2.414 \times 2.588 = 6.247\text{mm}$.

2.3.2.2 The bend radius-violation check rule

A wire or a bundle must not be bent beyond its allowed limits to avoid damage. The minimum allowed bend radius of a wire mainly depends on the diameter (e.g. thin or thick) and the material (e.g. rigid or flexible) of the given wire.

Design specifications, such as MIL-W-5088L [25] and Aircraft EWIS Best Practices [21], point out that "*the minimum bend radius of a bundle shall be ten (10) times the outside diameter of the largest included wire or cable*", and "*the minimum bend radius, as measured on the inside radius of a protected harness, shall be six (6) times its (bundle's) outer diameter. In no case shall the bend radius of a protected harness be less than ten times the diameter of the largest included wire or cable*". "*For flexible type coaxial cables, the radius of the bend (measured on the inside) shall not be less than six (6) times the outside diameter. For semi-rigid types, the radius (measured on the inside radius) shall not be less than ten (10) times the outside diameter*". In Figure 2-20, the terms in the above design rules are illustrated.

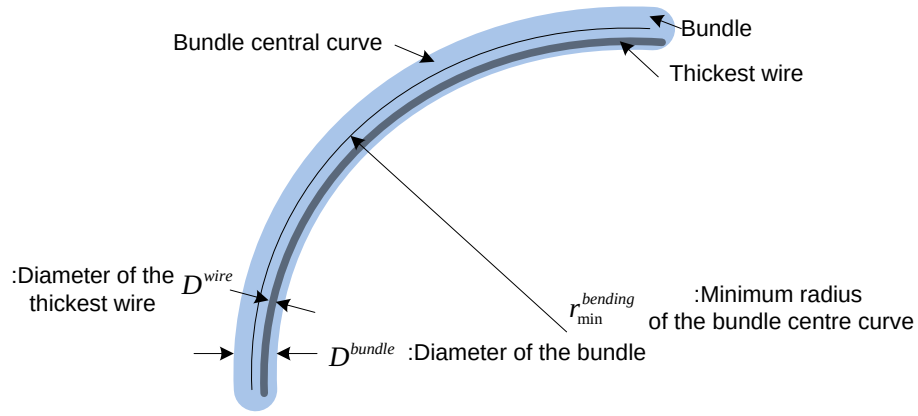


Figure 2-20: Illustration of the bend radius-related terms

The previous design rules will be satisfied if the inequality

$$r_{\min}^{bending} - D^{bundle} / 2 \geq \max(\varsigma^{bundle} D^{bundle}, \varsigma^{wire} D^{wire})$$

is satisfied. Here, $r_{\min}^{bending}$ is the minimum bend radius of the bundle central curve; $D^{bundle} / 2 = r_{\min}^{bending}$ is the bundle radius. The left side of the inequality is the bend radius that measured on the harness inside.

D^{bundle} and D^{wire} are the diameter of the bundle and the thickest wire respectively. ς^{wire} and ς^{bundle} are the allowed bend radius ratio of the wire and the bundle, and they depend on the wire and bundle materials, as shown in Table 2-2. $\max(\,)$ is a logic function to find the maximum number from the two inputs.

Table 2-2: Allowed bend radius ratios

Material		ς^{wire}	ς^{bundle}
Normal wire		10	6
Coaxial cable	Flexible	6	6
	Semi-rigid	10	10

2.3.2.3 The geometric collision-free and geometric attraction rule

A collision between geometries means that they intersect each other. The collisions considered here are the following three types, namely: 1) a collision between harnesses and aircraft components, 2) a collision between different branches of the same harness, and 3) a collision between different harnesses. These three collision types are illustrated in Figure 2-21. None of them is allowed in the 3D routing and collision avoidance is a major part of the routing procedure.

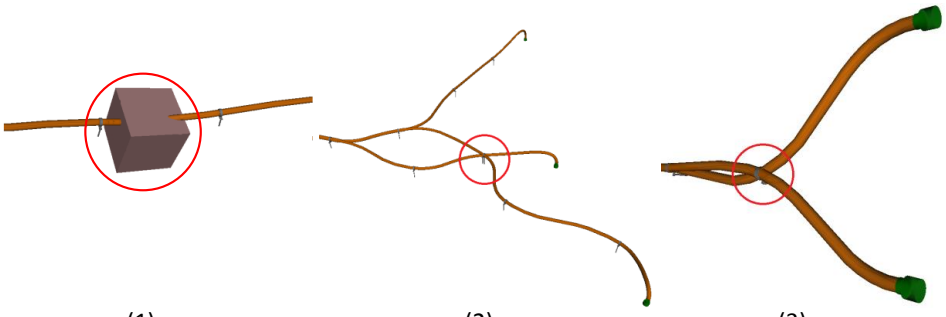


Figure 2-21: Three geometric-collision types: (1) a collision between harness and geometric structure, (2) a collision between different branches, (3) a collision between different harnesses

Geometric attraction means wire harnesses should be routed in proximity of the fixable structure (i.e. the structure that harnesses are allowed to be fixed on), because the harnesses have to be fixed on some of the components and also because the space in the centre of the fuselage is reserved for passengers and cargos. However, not all these geometric components are suitable for harnesses to be attached on. For example, wire harnesses are generally not allowed to be fixed on the equipment and aircraft systems. The OEM will also provide a list to specify the (non-) fixable components of the airframe.

2.3.2.4 Clearance requirement between a harness and moving parts and systems

The "*clearance between EWIS and moving parts and systems shall be at least N mm* [1] *throughout the full range of movement*". "*full range of movement*" is defined by a 3D mock-up of a dynamic envelope which represents all the possible positions of the moving part, as shown in Figure 2-22. Compliance to this rule is checked by the measurement of the minimum distance between the harness and the dynamic envelope.

[1] The actual number is confidential to Fokker Elmo.

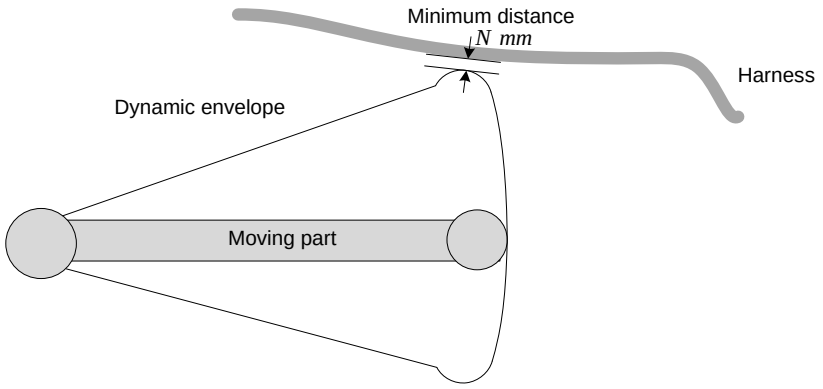


Figure 2-22: The minimum distance between a harness and the dynamic envelope of a moving part

2.3.2.5 Rules on the use of clamps

Wire harnesses need to be fixed on the airframe at a proper *clamping distance* and *fixing distance*, shown in Figure 2-23, to satisfy the design specifications.

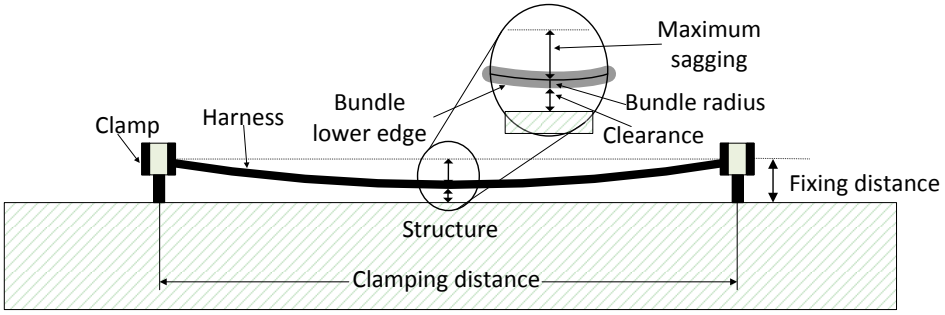


Figure 2-23: The harness clamping distance and fixing distance

The fixing distance is defined as the distance between the harness central curve at a clamp and its support structure. Due to the natural deflection, a harness bundle section between two clamps always has sag. The position with the maximum sagging size is in the middle of the bundle section. The maximum sagging size (measured from the bundle central curve) depends on the clamping distance and it can be found from tables such as Table 2-3.

Table 2-3: Relation between the maximum sagging size and the clamping distance

Clamping distance (inches)	3	4	5	…	20	21	22	23	24	…
Maximum sagging (inches)	0.01	0.02	0.04	…	0.4	0.44	0.46	0.48	0.5	..

The data in this table is imaginary on the basis of confidential company data from Fokker Elmo

In addition to the sagging, a harness also needs to keep a minimum clearance (measured from the bundle lower edge) from the structure. Therefore, the minimum fixing distance can be represented by Equation (2.2), where x is the clamping distance and $sag(x)$ reflects Table 2-3. The *clearance* is the minimum allowed distance between the bundle and the structure.

$$min\ fixing\ distance = sag(x) + harness\ radius + clearance \tag{2.2}$$

When the minimum fixing distance (including the clearance) is satisfied, the actual fixing distance should be as small as possible to limit the size and weight of the clamps, stand-off, and brackets.

The *clamping distance* is the distance between two adjacent clamping points. The actual clamping distance should not exceed the maximum allowed clamping distance, which depends on the harness material and the routing environment and given as an input. For example, according to design specification MIL-W-5088L[25], the clamping distance of a normal harness routed in a non-vibration zone should not exceed 24 inches. For rigid harnesses this distance is extended to 42 inches. For conventional aircraft with wing-fixed engines, the clamping distance inside the wings is smaller due to the vibration caused by the engines.

The clamping points here mean not only the clamps, but also the connectors and breakout. The distance between a connector and its adjacent clamp mustn't exceed the allowed value in order to provide the harness enough support. It is the same at breakout points. Actually, in practice, the allowed distance between a breakout and its adjacent clamp is smaller than the allowed distance between clamps. In order to simply the distance check between clamps, breakouts or connector, the same allowed (clamping) distance is used. This means not only the distance between two clamps but also the distance between a connector/breakout and a clamp should be smaller than the allowed clamping distance.

2.3.2.6 Rules applying to the grey areas

In an aircraft, there are some hazardous areas, such as wet, hot, and vibratory areas. Harnesses routed in these zones are prone to more risk. Hence domain rules are given to guide the harness design in these zones to avoid potential damage to the harness. These domain rules generally require using more protective coverings and/or clamps to better protect/support harnesses. Hot zones and flammable zones are two typical area types and they influence the use of protection coverings and clamps respectively. These areas are neither forbidden (i.e. black), such as the space placing geometric components; nor free (i.e. white), such as the pressured, room temperature area, where there is no extra protection or support for harnesses to go through. Therefore these zones are also referred to as the grey areas.

1) Rules for hot zones:

Hot zones exist around the high-temperature equipment such as resistors, exhaust stacks, and heating ducts. A harness exposed to a high temperature suffers from deterioration and deformation. Therefore, it is necessary to "*insulate wires that must run through hot areas with a high-temperature insulation material...*"[25]. This rule also hints that if the harness does not have to be routed through the hot areas, they can be routed with a detour around hot zones to avoid using insulation material, but this may be at the cost of a longer harness.

2) Rules for flammable zones:

Inside an aircraft, some harnesses are routed above the pipes used to transport flammable liquid. Contact between broken wires which belong to the harness bundle and the pipe may cause arc faults which are unwanted by design engineers. Since the arc fault may cause an on-board fire, the area which is influenced by this arc fault is called a flammable zone here.

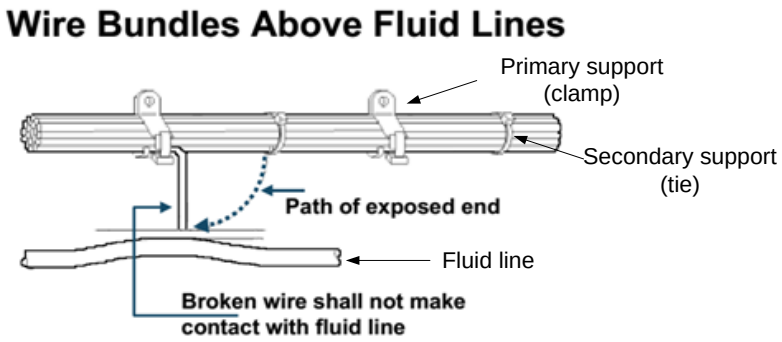


Figure 2-24: A wire bundle above a fluid line

In order to avoid potential contact in the flammable zone, the wire harness routed in this zone should be intensively clamped to make sure that if "*a wire breaks, the broken wire will not contact hydraulic lines, oxygen lines, pneumatic lines, or other equipment whose subsequent failure caused by arcing could cause further damage*"[21], as illustrated in Figure 2-24.

Extra clamps affect the total cost and weight of the harness. If the harness is not related to the components above a fluid pipe, it is possible to take a detour to avoid using extra clamps but at the cost of a longer harness and a few additional clamps to guide the detour.

2.3.2.7 Rules on 3D routing in the reserved space

In the space reservation phase, some spaces are dedicated for harnesses. Harnesses are preferably routed inside the reserved space. As shown in Figure 2-25, path B which is mostly located in the reserved space will be selected by engineers even though it is longer than the shortest path A.

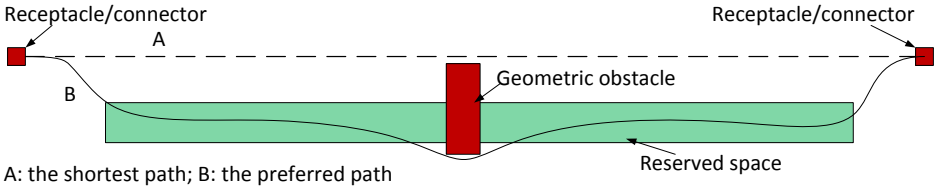


Figure 2-25: Routing preference for the reserved space

Since routing the entire path of a harness inside the reserved space is not always possible (see path B in Figure 2-25), the previous preference is represented by a penalty (or bonus) function. The harness section routed in a reserved space will have a lower cost, calculated with Equation (2.3).

$$f_{cost} = \varsigma_{res} f_{actualcost} \tag{2.3}$$

Here, $f_{actualcost}$ is the actual harness cost; ς_{res} is the reserved space coefficient which is smaller than 1 to reflect the bonus of routing in the reserved space; and f_{cost} is the final harness cost used in the 3D routing only. The harness cost in the final output report is still $f_{actualcost}$.

2.3.2.8 Rules of the segregation and separation

As defined by Fokker Elmo, **segregation** is "*the act of sorting and grouping of signals in*

order to isolate these on grounds of system independence, continued airworthiness requirements, and susceptibility or emission of different EM radiation levels", and the **separation** is "*the act of physically adding distance between segregated groups of wiring in order to ensure system independence and safety/reliability requirements, ...*".

The segregation is handled during the electrical design. In this process, the so-called segregation (EGS) code will be assigned to each wire. EGS represents three digits, namely E is the EMC digit, G is the Group digit, and S is the Subsystem digit. These three digits are defined with different codes, as shown in Table 2-4. An EGS code looks like <A><1><X>. Wires with identical segregation codes are deemed compatible and can be bundled together in one bundle and one wire harness. Consequently, a harness has the same segregation code as its wires.

The separation is handled during the physical design. The harnesses having the identical segregation codes can be placed next to each other. Otherwise they need to be routed at a certain distance away from each other. The minimum allowed distance between the harnesses with different EGS codes can be found in the separation table given in Table 2-4.

Table 2-4: Separation table (unit: mm)

		EMC			Group			Subsystem		
		A	**B**	**C**	**1**	**2**	**3**	**X**	**Y**	**Z**
EMC	**A**	0	10	50						
	B	10	0	25						
	C	50	25	0						
Group	**1**				0	300	300			
	2				300	0	300			
	3				300	300	0			
Sub-system	**X**							0	25	50
	Y							25	25	40
	Z							50	40	0

***This table is courtesy of Fokker Elmo. The codes and values in the table are purely fictional.**

This is the combined table of the EGS digits. In this table, 0 at the intersection of different codes means the codes are compatible. The other values indicate the minimum allowed distance between harnesses with the two codes. The three minimum distance values of digits E, G, and S need to be calculated respectively and the final minimum allowed distance is the biggest of the three distances. For instance, the EGS code of bundle A and bundle B is <A><2><Z> and <C><3><X>. Then the minimum allowed distance between the two bundles is $max(50,300,50) = 300mm$.

2.4 The actual 3D harness routing phase and its limitations

The physical harness design consists of three phases. The space allocation and MRA design belong to the conceptual harness design and they need a relatively low staffing level and short lead time. The 3D routing is a detailed design and needs a higher staffing level and more time. Indeed, according to Table 1-1, the man-hour ($staffinglevel \times leadtime$) of 3D routing is 20 and 6.7 times the space allocation and the MRA design respectively. The automation of the physical harness design can be achieved to a large extent by the automation of 3D harness routing. In addition, the 3D-routing phase contains as much as 80% of the actual design. This actual design is largely rule-based and repetitive and is also suitable to be automated.

In this section, the 3D harness routing phase is detailed first. The challenges of the current

3D-routing method will then be presented. After that, the proposed design automation method, which is a potential solution to the challenges, is discussed, and then the state and limitations of the solution is studied.

2.4.1 The actual 3D harness routing phase

During the 3D harness routing phase, engineers route harnesses per zone and gradually achieve an entire aircraft EWIS. Except the dedicated methods to handle the different environmental conditions in different zones (e.g. heat in Zone A and moisture in Zone B), the design methods in these zones are very similar due to the similarity of the wiring zones themselves. This research focuses on the 3D routing in one wiring zone.

The 3D routing in a wiring zone is to fix the geometric model of harnesses that connect the production breaks and avionics on the fixable structures without any violation of the design rules. This 3D-routing phase can be broken down into the four activities shown in Figure 2-26, where the input/output of each activity is also included.

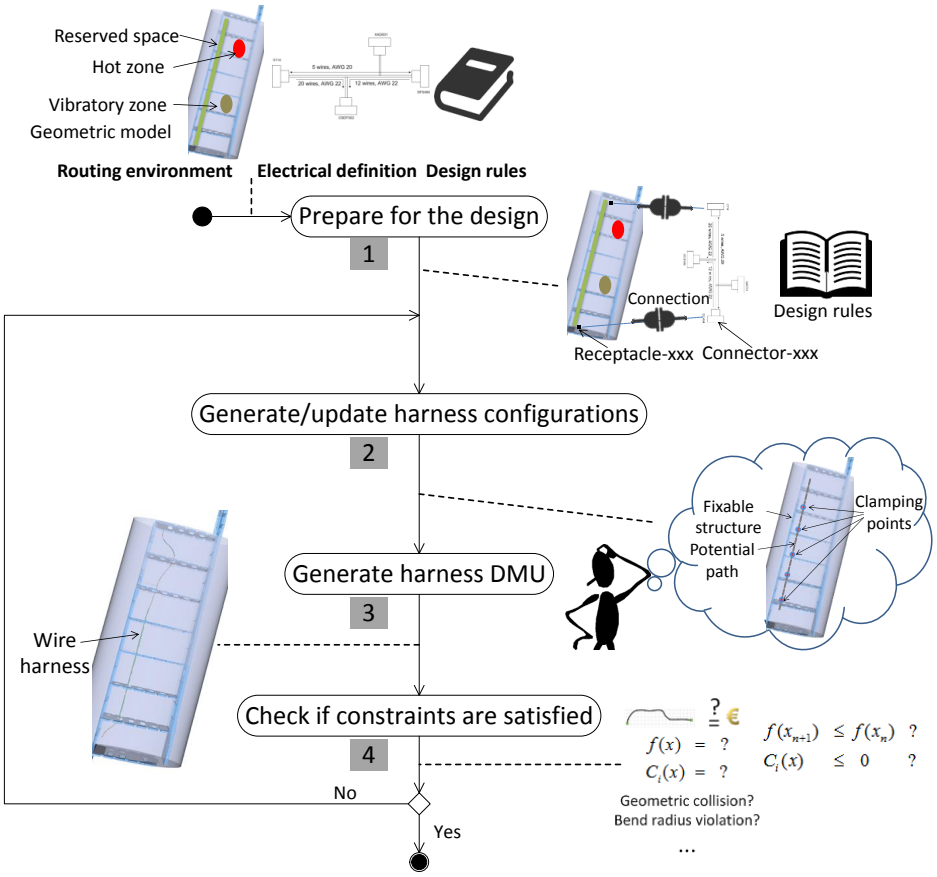


Figure 2-26: Workflow of 3D harness routing and the input/output of each activity

1) Preparation of the design

In this activity, the routing environment (including geometric model and the environmental information, such as temperature, vibration level), harness electrical definition are identified.

The design rules, in most cases, are also about to be referred to by the engineers. The receptacles that harnesses connect with will be recognized by designers.

2) Generation/update of harness configurations

The second activity is to generate or update the harness configurations. Engineers build a virtual harness as their mental activity rather than generate the actual harness geometric model within the CAD software. This part contains the tasks of generating new harness configurations and updating existing configurations.

The first task starts from the electrical definition. The engineers work out the 3D position and direction of the breakouts and clamps. Accordingly, they estimate the position and shape of the harness. This task is divided further into 3 steps.

- The first step is to identify the fixable structures in the routing environment.
- The second step is to find the potential harness path in front of the fixable structure surfaces, taking into account the clearance between the path and these surfaces to avoid a potential chafing between them. In this step, the potential positions for the clamps and breakouts are roughly considered to enable the further harness fastening.
- The third step is to find the exact position for the clamps and breakouts. In this step, the clamping distance and fixing distance need to be satisfied. The breakouts need to be placed at proper positions to minimize the harness weight and cost.

The second task is similar to the first task except it starts from existing harnesses. If the previously generated harness does not satisfy the design rules according to the performance check (see Activity 4), the previous design will be adjusted to generate a new harness configuration.

In the both tasks, all the design constraints, such as geometric collision-free and bend radius violation-free, are considered by the design engineers in order to generate a feasible harness DMU. The grey areas are also considered. The engineers will implement a trade-off study to find a better solution which will be either going through these areas with extra protection/supports, or taking a detour to avoid the areas but at the cost of a longer harness and additional clamps too.

3) Generation of a harness DMU

The third activity is to generate a harness DMU which is the final output of the physical harness design. This process transfers the aforementioned harness configuration to an actual harness geometric model using the CAD system. For example, the harness path defined by some parameters (e.g. waypoints, diameter) will be modelled to an actual curved tube, and the connector defined by the Unique Identification Number (UID), such as UID-1234, will be transferred to a simple cylinder.

In current manual design processes the second and third activities are implemented almost simultaneously by engineers. The engineers draw 3D harness models according to the harness configurations and then they may modify the configurations according to the visual/intuitive feedback from the generated harness models if the models are not satisfactory. Here, the second and third activities are formalized separately to support the 3D-routing automation.

4) Checking the satisfaction of the design constraints

Although the design engineers have already taken into account the design constraints in the second activity, the design results cannot always be right first time. In addition, updating the routing environment and the harness electrical definition may also cause violation of design

rules, such as geometric collision. Hence, checks on the satisfaction of the design rules, such as the clamping-distance check, bend-radius violation check, and geometric-collision check are carried out on the generated harness DMU. If any of the design rules is violated, the second activity will be triggered again. Designers might need to iterate the design many times before getting a satisfactory result.

2.4.2 Challenges in the current 3D-routing phase

As already introduced in Chapter 1, the 3D-harness routing is very complex and challenging due to the 1) increasingly complex EWISs, 2) increasing number of design specifications/constraints, and 3) increasing number of design changes to the airframe and other systems. Therefore it is very difficult for the design to be right first time and many design loops are needed to handle the frequent changes to the airframe and other systems. Facing such challenges, design engineers often work under pressure and their designs are error-prone.

2.4.3 Potential solutions to the 3D-routing challenges and their current state

The previous challenges can be solved by automating the 4 activities of the 3D routing, especially the last three. Many design specifications can be checked automatically, this will release the work load of design engineers. The design change of the airframe and other systems can be detected by the routing tool. If design rules are violated due to these changes, the harness configuration can be modified automatically with smart 3D-routing algorithms and accordingly a new harness DMU can be generated. These activities can iterate many times until the design is converged and the iteration can be carried out 24/7. Therefore, the design automation method can significantly increase the routing efficiency and decrease the workload of engineers.

Indeed, these opportunities have already been noticed and there have been a few achievements, as shown below; however an automation method for the entire 3D-routing phase has not yet been found.

Activity 1) is already automated. Importing the harness electrical definition (in format of datasheet) and the routing environment (including the geometric model and the environmental information) into 3D routing tool and identifying the receptacles can be automated by some PDM/PLM[1] tools.

Activities 3) and 4) can be partly automated. For instance, some harness geometric models can be generated automatically by the harness parametric modelling module defined by Van den Berg for the harness manufacturing design[23]. However, a wide use for this method has not yet been found. Some analyses, such as the harness geometric-collision check and calculating the harness cost according to pre-given data, can be achieved automatically in most mainstream MCAD tools. The internal software tool of Fokker Elmo can automatically check the harness bend radius violation. However, a large number of design rules, such as the clamping distance and fixing distance violation, segregation and separation rules, and clearance between harnesses and moving parts cannot yet be detected automatically by currently available tools.

[1] Product Data Management (PDM)/Product Lifecycle Management (PLM), such as Siemens Teamcenter

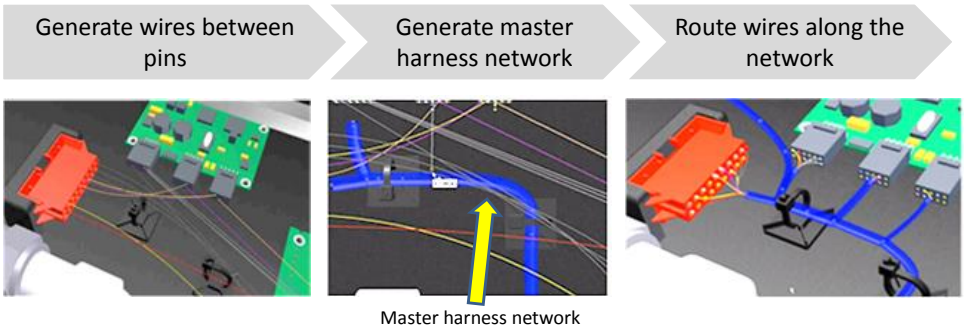


Figure 2-27: A typical 3D-routing method with current representative MCAD tools
The pictures in this figure are screenshots from a webcast[29] that introduces the 3D routing function of Autodesk Inventor 10. The 3D routing functionality of other CAD tools, such as Solid Edge ST4, Autodesk Inventor 2013, and Catia V5/V6, which are the foremost CAD tools, is very similar to this one.

The automation of Activity 2), namely generating a harness configuration while at the same time considering typical design rules, has not been achieved by any currently available tools. Automation of this activity is very difficult since it is an extremely challenging task to enable a computer to think like an experienced engineer. Representing the routing environment, the harnesses themselves, and the many design rules in a software tool are already difficult. Yet the more challenging task is to automatically place the harness models properly in the routing environment while satisfying all the design rules. As shown in Figure 2-27, design engineers can automatically place the wires along the so-called master harness network in the 3D-routing task carried out with current mainstream CAD tools. However, this network needs to be generated manually. Besides, these tools cannot automatically move the network to avoid the violation of design constraints, such as geometric collision and bend-radius violation.

In the academic domain, Conru formulates the 3D harness routing as a graphic search problem and provides a Genetic Algorithm-based approach to place the harness breakouts on the vertexes of the 3D graph that represents the general shape of the routing space[14]. This method can find a harness configuration. However, the 3D graph is generated manually and typical design rules, such as clamping distance and bend radius, are not considered. Van der Velden et al. have developed a 3D volume mesh-based harness-routing method to automate the 3D routing. In this research, the geometric model of the routing environment is transformed into a set of cubes and then a pathfinding algorithm is used to find the harness path taking into account some design constraints[13]. This method can generate the configuration of a single harness (i.e. one start and one end point) automatically. However, it cannot route multi-destination harnesses. Typical design rules, such as EMI-free and bend radius violation-free, are also not considered.

2.5 Automation of the 3D harness routing

Current 3D harness routing includes creative tasks and repetitive tasks. Creative tasks are those that tackle the design problems never met before. This work needs the experience, knowledge, creation, and even intuition of engineers. It takes only a small part of the total design and is very difficult to be automated. Repetitive tasks are to repetitively generate many harness geometric models using already known knowledge. These repetitive tasks occur a lot during the entire design process and they are easier to be automated.

Considering the challenges in 3D-routing phase, the automation opportunities, and the current state of the automation level, the research presented in this dissertation will propose an

innovative approach to automating the repetitive tasks of the four activities in the 3D-routing phase, as shown in Figure 2-28. Details of the automatic 3D-routing approach will be presented in the following chapters.

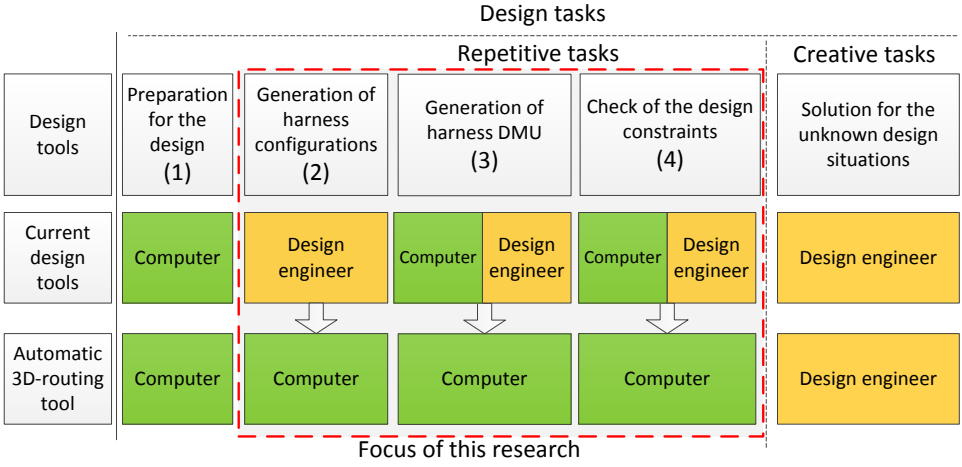


Figure 2-28: Improvement of the automation level of the 3D harness routing

Chapter 3 Methodology to enable automatic 3D routing of wire harnesses

In order to automate the 3D routing of wire harnesses, an innovative approach has been proposed. The approach is based on a hypothesis that the 3D-routing problem can be modelled and solved as a hybrid optimization problem. In this chapter this innovative approach is elaborated.

In Section 3.1, the approach, i.e. solving the 3D harness-routing problem as an optimization problem, is explained. In Section 3.2, the challenges in using this approach are discussed first and then a hybrid optimization strategy is proposed to handle the challenge. The proposed approach needs to be implemented into a software application to validate the feasibility of the approach. The technical implementation is based on a combination of Multidisciplinary Design Optimization (MDO) and Knowledge-Based Engineering (KBE) technology. Section 3.3 introduces these two technologies and then discusses the use of them to support harness optimization. A computational framework developed to support the implementation of the KBE software application is presented in Section 3.4.

3.1 Automation approach to 3D harness routing

3D harness routing can be modelled and solved as an optimization problem. Design engineers use the position of the clamps and breakouts (see Figure 3-1) as design variables[1] to control the shape, position, and therefore the cost of a harness. During this process, design rules must be respected in order to generate feasible design results. Optimization is a process to seek the value of the design variables that lead to the minimum value of the objective function, while satisfying the constraint functions. The design variables, objective function, and the constraint functions in the harness optimization are able to represent the position of clamps & breakouts, cost of harness, and design rules in the manual 3D routing respectively, as shown in Table 3-1.

Table 3-1: Counterparts of the manual 3D routing and the harness optimization

Manual 3D routing	**3D routing as an optimization problem**
Coordinates of clamps & breakouts	Design variables
Cost of harness	Objective function
Design rules	Constraint functions

[1] In practice, both the position and direction of clamps and breakouts are used to define harnesses. However, in order to control the scale of the optimization problem, i.e. the number of design variables, only their positions are adopted. Their directions are calculated according to their positions and the design environment. The calculation process is elaborated in Chapter 4.

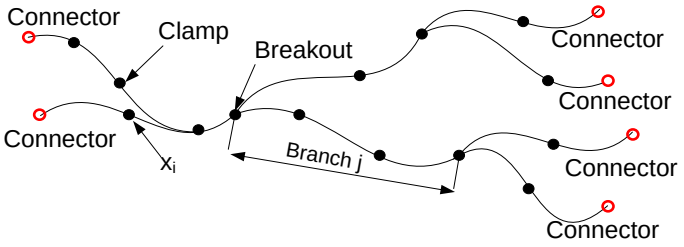


Figure 3-1: Simplified representation of a harness used for 3D routing

The mathematical definition of the harness optimization is shown below.

$$\min_{x} f(x)$$

$$f(x) = \sum_{j=1}^{M} L_j(x) Co_j(x)$$

$$\text{with respect to } x$$

$$\text{subject to } C_i(x) \le 0$$

The objective function $f(x)$ is the actual cost (e.g. in Euros) for materials and installation of the entire harness. The material cost is the cost of a harness and its accessories, such as clamps and protection covers. The installation cost means the cost to place each clamp on fixable structures. $f(x)$ is represented by the summation of each branch cost $f_j(x)$, where j is the index of each branch (Figure 3-1). x is the vector of design variables and it represents the clamps and breakout positions. These design variables are also waypoints to generate the harness central curve. The cost $f_j(x)$ is computed as the product of L_j and $Co_j(x)$. L_j is the length of branch j and is a function of the design variables x. $Co_j(x)$ is the cost coefficient of branch j and depends on its bundle diameter, the amount of clamping elements, and required protective layers for the grey areas. The coefficients may be different in different routing zones and/or when applying different design specifications. $C_i(x)$ is one of the constraint functions to account for each design rule that has to be satisfied (see Sub-section 2.3.2). Details of the optimization problem definition will be presented in Chapter 6.

3.2 A hybrid optimization strategy to enable the harness optimization

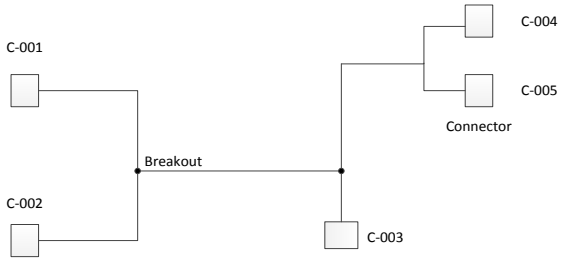


Figure 3-2: Example of a harness electrical definition

The challenge in solving the optimization problem described above is that the number of design variables, namely the number of the harness clamp points, is not known *a priori*. The harness electrical definition is an input of the 3D routing. As shown in Figure 3-2, it includes the connectivity among the receptacles (i.e. connectors) and breakouts as well as the number

and logical location of the breakouts. However it does not include any information about the clamps. Indeed, the number and position of clamps are the outputs of the harness 3D-routing. These are actually the goal of the optimization problem.

Besides the number of clamps, the 3D position of the clamps and breakouts (i.e. the initial value of design variables) is also not known. A large number of optimization algorithms are sensitive to the initial values. Without suitable initial values, the optimization is very difficult or even not able to converge.

In order to handle these challenges, a two-step, hybrid optimization strategy has been devised and illustrated in Figure 3-3. These two steps use different optimization methods to compensate for each other's drawbacks.

1) Initialization

In the first step, called *Initialization*, a discrete optimization method is applied. A grid of potential clamp points is generated in front of the structural elements where harnesses are allowed to be attached. The grid points connect with each other, building a so-called road map whose vertexes can be used for potential clamp points and whose edges form the potential path for the harnesses. Then an optimization approach is applied to route a simplified harness model on such a road map. As the results of this step, a preliminary routing of the harness is obtained together with the number and position of its waypoints (i.e. the number of clamping and breakout points and their coordinates). This process is similar to a car navigation process in which a pathfinding algorithm finds the path based on a map that represents the actual road network of the real world.

Since the aim of the Initialization is to provide the second optimization step with a preliminary harness definition that is promising to satisfy all the design rules by the end of the second optimization, all the wire harness features such as multi-destinations and all the 3D-routing rules such as geometric collision-free and bend radius violation-free are considered in the Initialization process. However, since the result of the Initialization is a preliminary harness definition, all the routing rules and the harness representation are simplified in order to be able to use the discrete optimization method and to increase the calculation efficiency. Details of this step can be found in Chapter 4.

2) Refinement

At this point, with the number of design variables and their initial values known, the second optimization step, called *Refinement*, can be applied. In this step the optimizer explores the solution space based on the initial values and the accurate representation of wire harnesses, the routing environment and design specifications. The preliminary harness digital model is firstly generated according to the preliminary harness definition from the Initialization. Then the harness model is analysed by various analysis tools. These tools include a module to calculate the harness cost and some other modules, such as the tool to check bend-radius violation and the tool to check geometric collision. The calculation results will be sent to the optimizer for the optimality and feasibility check. If necessary, the optimizer moves the design variables one by one to generate new harness configurations in order to minimize the cost objective function, while satisfying all the design constraints. The optimization step iterates until a stop condition is reached. Details of this step can be found in Chapter 6.

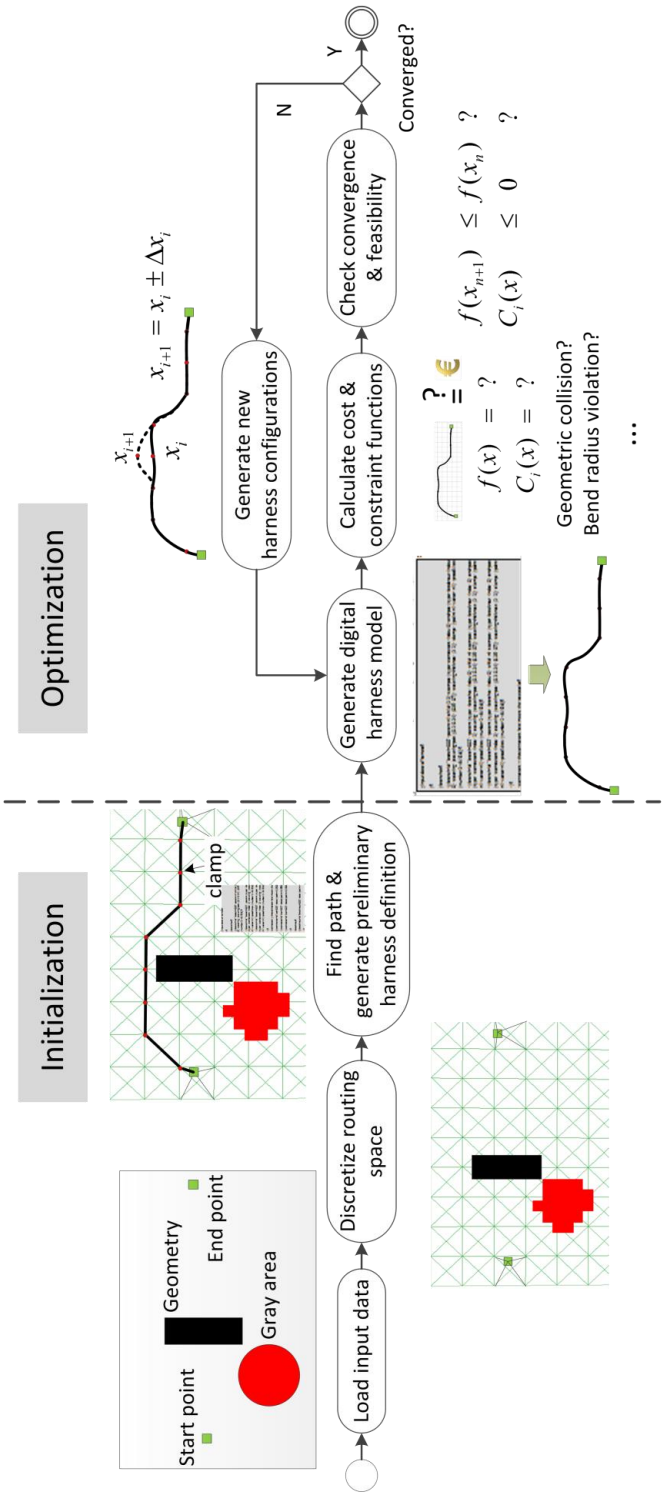


Figure 3-3: Illustration of the hybrid optimization approach

3.3 Multidisciplinary Design Optimization and Knowledge-Based Engineering to support the proposed design approach

The previously proposed hybrid optimization approach needs to be implemented into a software tool to validate the feasibility of this approach. This technical implementation is built on a combination of Multidisciplinary Design Optimization (MDO)[30] and Knowledge-Based Engineering(KBE)[31] technology. MDO is used to systematically and automatically explore the space provided by the 3D-routing problem to discover the minimum cost solution that complies with the multiple design constraints. KBE technology is exploited to enable the automation of the 3D geometric models generation, manipulations, and the performance checks. This technology also enables the capture of the typical rule-based approaches of the wire harness design. In this section the use of the MDO and KBE approaches to support the 3D harness routing is discussed.

3.3.1 Using MDO to support 3D harness routing

The Multidisciplinary Design Optimization method has been proposed to support the design of complex systems for several decades. It can simultaneously take into account different disciplines and support design optimization. The advantages to adopting MDO are significant, as it "*enables the efficiency of designs to be optimised and supports trade-off studies between the design objectives of diverse disciplines*"[32]. Since then, much academic and industrial interest has been attracted and a lot of related research has been executed in different areas[33-36]. In the domain of EWIS 3D routing, however, the application of MDO is relatively modest even though 3D harness routing is fitting the application field characteristics of MDO.

3D harness routing involves multiple design rules, such as the selected rules presented in Chapter 2. In the previously proposed Refinement method, these rules are represented by multiple design constraints. These constraints need to be checked in each optimization loop and sent back to the optimizer to support the trade-off in order to avoid conflicting results and to get the minimum-cost solution. Hence the MDO method is adopted to support the optimization.

3D routing of harnesses involves geometry. Various analyses, such as the check of geometric collision and the bend-radius violation, need to manipulate the actual geometric models of the wire harness and the routing environment, such as the aircraft Digital Mock-up (DMU). The DMU of the wire harness needs to be generated as one of the 3D-routing outputs. Therefore, the MDO method used to automate the 3D-routing phase needs to be capable of handling different manipulations involving geometry.

Three MDO architectures involving geometry are discussed by Vandenbrande et al.. These three architectures are addressed as Geometry-less MDO architecture (the lowest fidelity), grid-perturbation MDO architecture, and Integrated Geometry Generation MDO architecture (the highest fidelity) in accordance with the fidelity of geometric models used to perform the Multidisciplinary Design Analysis (MDA). The higher the fidelity is, the more details are included. The highest fidelity architecture uses the actual geometric model to perform the analysis. Details of these three architectures can be found in [37].

The hybrid optimization method is supported by a combination of two different geometric fidelity architectures, as shown in Figure 3-4.

In the Initialization step, the Geometry-less MDO architecture is adopted. In this phase the routing environment, design rules, and harness itself are represented by the simplified

mathematic expressions in the optimization loop. The various analyses are based on these simplified expressions. When the initialization is finished, the output will be transformed by a processor into the required input definition of the optimization.

The Refinement step adopts the Integrated Geometry Generation MDO architecture. It reads the output of the processor to generate an actual harness geometric model. The MDA is implemented based on this actual model, the actual routing environment and the actual design rules. Then the results are sent back to the optimizer for the next iteration. This process iterates until one of the stop conditions is satisfied.

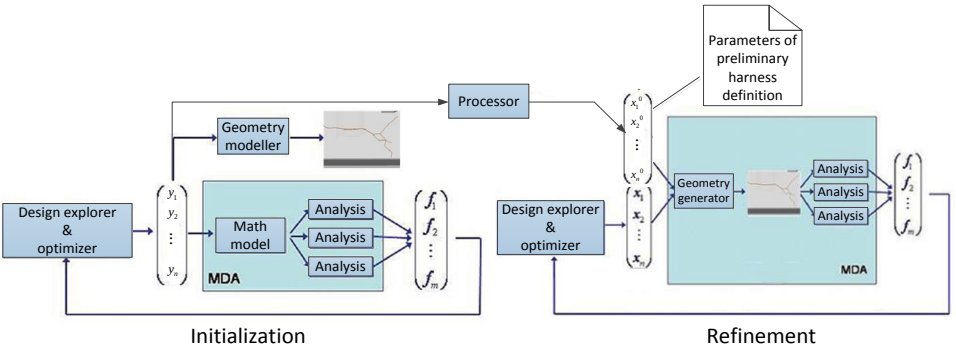


Figure 3-4: Combination of two MDO architectures to support automatic 3D routing

3.3.2 KBE to support wire harness routing optimization

Aiming for a full automation of the optimization process, the two operations in the previous MDO architecture, namely 1) geometric model generation and 2) the geometry-involved analysis method need to be automated. The automation requirements make the KBE a suitable solution to support the optimization process.

Knowledge-Based Engineering is a methodology that is able to automate certain steps of the design process by using a software tool, referred to as the KBE system, to capture and store the involved product and design knowledge. The introduction and some applications of the KBE method can be found in [22, 24, 38, 39].

The KBE capabilities that support the two operations are presented in Table 3-2.

Table 3-2: Comparison of the MDO requirements and KBE capabilities

MDO requirements	Automatic geometric model generation	Automatic geometry-involved analysis (MDA)
KBE capabilities	Capable of capturing product knowledge Object-oriented (OO) CAD -- Geometric modelling	CAD – Geometric model manipulation Capable of capturing design knowledge

KBE is not a novel revolutionary technology but a concept combining Artificial Intelligence (AI) and CAD to create generative model. Each KBE system is built around a computer language that enables object-oriented programming as well as functional programming. It is different from basic language itself through an intimate connection to a CAD expression. The AI root enables KBE to capture and reuse the harness product and design process knowledge.

Because of the OO feature of the KBE system, the captured product knowledge can be used to build the harness parametric model to enable the automatic geometric modelling. The design process knowledge is used to guide the development of a software application that can automatically generate the geometric model and analyse the performance of the model. The intimate compiling to CAD enables harness parametric models to be instantiated into various geometric instances. It also enables the manipulation of geometric models to extract information for MDA. According to the two MDO requirements, these KBE features are detailed below respectively.

3.3.2.1 KBE to automate geometry generation

The programming language used to develop the KBE application is object-oriented. Object-oriented, which is one of the programming paradigms, has two important concepts, Class and Object. Class is an abstract concept. It contains the attributes used to describe the object and associated methods. Object is an instance of a class.

```
;;class
(define-class test-box (box)
  :input-attributes   (height   length   width   centre )
)
;;objects/instances
(test-box-1 (make-object   test-box :height  1   :length 1 :width 1 :centre #(0 0 0))
.....
(test-box-n (make-object   test-box :height  5  :length 4 :width 6  :centre #(5 7 8))
```

Figure 3-5: Object-Oriented paradigm

The OO feature enables the product knowledge to be formalized, stored and reused by the KBE application. For instance, the knowledge "*a test-box contains the attributes of height, length, width, and centre*" is formalized and stored in the class definition in Figure 3-5. This knowledge can be reused to generate different test-box instances, such as test-box-1 until test-box-n.

```
;;class
(define-class test-box (box)
  :input-attributes   (height   length   width   centre )
)
;;objects/instances

(test-box-1 (make-object   test-box :height  1   :length 1 :width 1 :centre #(0 0 0))
.....

 (test-box-n (make-object   test-box :height  5  :length 4 :width 6  :centre #(5 7 8))
```

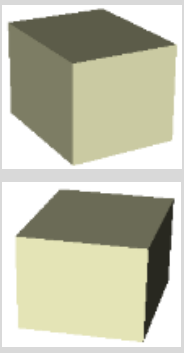


Figure 3-6: Geometric instantiation of a class

In the KBE system, the CAD capability is enabled by employing a geometric kernel. The geometric kernel contains the definitions of geometry, such as box, sphere, cylinder, and

lofted-surface. In the previous example, as an inheritor of the class box, the user-defined test-box class can be automatically instantiated to generate the actual geometric models shown in Figure 3-6 rather than only the geometric description shown in Figure 3-5.

3.3.2.2 KBE to automate geometric model-involved analysis

The analyses in harness optimization involve geometry. The execution of these analyses needs two types of information, namely the property of the geometry and the formalized rules. They are enabled by the geometric manipulation capability and rule-based feature of KBE technology.

The KBE system manipulates the geometric model in its geometric kernel to get the required properties of the geometry. For instance, it can get the volume of a previously defined test-box by calling a function in the KBE system. Due to the rule-based feature, the KBE system can transfer the design rules that are generally described in natural language to formal representations. For instance, the logic rule "*the volume of a test-box needs to be smaller than* $1m^3$ " can be formalized into a logic condition shown in Figure 3-7. With these two capabilities, the KBE system is able to automatically implement the geometry-involved analyses.

```
If volume of box < 1
  Rule satisfied
Else
  Rule violated
```

Figure 3-7: A formalized logic rule

Besides handling the above mentioned logic rule, the KBE system is also able to handle other types of rules. The two most useful rules are mathematical rules and communication rules.

The mathematical rules, which include all the mathematical operations, are used to implement the arithmetic. The mathematical expression $r_{allowed} = 10D$ represents the design rule: "*bundle minimum-allowed bend radius equals to 10 times the bundle diameter*".

The communication rules[40] are used to define the interface standard when the KBE application needs to be used in collaboration with external tools, such as the STEP and IGES for geometric model exchange and XML for data exchange.

3.3.2.3 Introduction of the KBE system

In this research a commercial KBE system called GDL[41] is adopted to support the hybrid optimization of 3D harness routing. GDL stands for General-Purpose Declarative Language and it is a superset of ANSI Common LISP[42], mainly consisting of automatic code expanding extensions to Common Lisp implemented in the form of macros.

GDL is object-oriented. It has previously mentioned OO features. In addition, it is able to

• Connect with its geometric kernel to implement the geometric model generation and operation;

• Capture the procedures and rules used to solve repetitive tasks in engineering fields;

• Connect with external optimizers, such as Matlab, to evaluate many designs or engineering alternatives and perform various kinds of optimizations within specified design spaces;

• Visualize geometric models including the wireframe, surface, and solid geometric objects.

The code-editing environment and development User Interface (UI) of GDL are shown in

Figure 3-8.

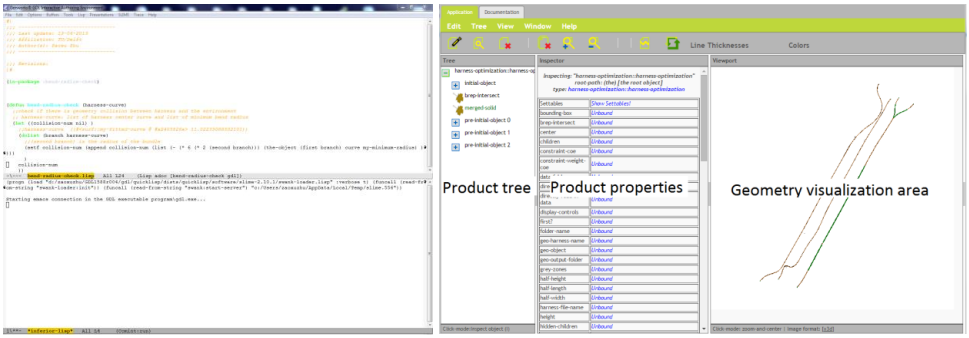


Figure 3-8: The editing environment (left) and development user-interface (right) of GDL

3.4 Harness Design and Engineering Engine (HDEE) – development of an MDO framework to enable automatic 3D routing

KBE technology has been used to support the two-step optimization process by reading in the routing environment (i.e. aircraft DMU), building harness geometric models, measuring harness properties such as bundle length and bundle-bend radius, and checking for violation of design constraints. A KBE software application has been developed using the commercial KBE system to realize these design activities. The computational framework developed to support the implementation of the KBE application is called Harness Design and Engineering Engine (HDEE) and is based on the general Design and Engineering Engine (DEE) concept [38, 43-46] developed at Delft University of Technology. In this section, the general DEE concept is presented first and then the HDEE is introduced.

3.4.1 Overview of the DEE template

The DEE is the template for advanced design platforms that facilitate and accelerate the design process of complex systems, such as aircraft, aircraft components, and aircraft systems through the automation of non-creative and repetitive design activities. The architecture of the DEE is illustrated in Figure 3-9. Several specific DEE implementations for the design of aircraft or aircraft components/systems are described in the literature. These implementations include, but are not limited to, aircraft DEE that supports the design and optimization of aircraft[22], Movable DEE (MDEE) that supports the design of movable aircraft structures [39], harness DEE that supports harness manufacturing design[23], and Airframe DEE (ADEE) that supports airframe structure design[47].

The DEE framework consists of four components that support the various steps of the design and optimization of a geometric product. Starting from the set of top level requirements, the ***INITIATOR*** provides starting values for instantiation of the product model for the following optimization step by means of empirical rules and fast analysis tools. These values will then be sent to the next step to support the generation of the geometric model. In the next step, ***MULTI MODEL GENERATOR (MMG)*** is responsible for generating geometric models and the extraction of specific data from such models for subsequent analyses. The analyses of the geometric product are performed by various ***ANALYSIS TOOLS***. These tools evaluate the design from different aspects, such as aerodynamic performance and structural strength. The analysis results are sent to the ***OPTIMIZER*** to support decision making. The ***OPTIMIZER*** checks the objective function and the constraint functions and decides on the next vector of design variables if necessary.

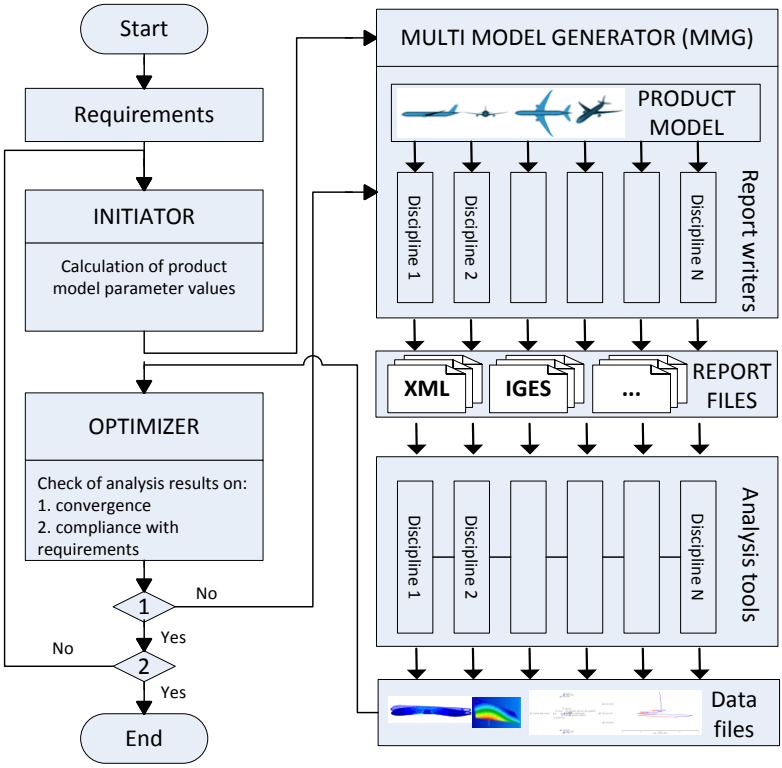


Figure 3-9: The Design and Engineering Engine (DEE) Template

3.4.2 Introduction of HDEE

In this research, a specific DEE implementation, the Harness DEE (HDEE), is developed to support the two-step harness optimization. The HDEE, shown in Figure 3-10, consists of four components, namely: ***INITIATOR***, ***MULTI MODEL GENERATOR (MMG)***, ***ANALYSIS TOOLS***, and ***OPTIMIZER***. These components are in charge of harness Initialization and Refinement.

The HDEE Initiator is the software implementation of the harness Initialization. It uses a discrete optimization method to automatically generate the preliminary harness definition, which includes the amount and position of breakouts and clamps. The wire harness, routing environment, and the design rules are simplified in order to quickly generate the preliminary definition. For instance, the harness bundle is represented by polyline rather than the actual curved pipe. Due to the simplification, the *INITIATOR* cannot guarantee fully feasible solutions although these solutions are promising for the following optimization step. The details of the Initialization are detailed in Chapter 4.

The remainder of HDEE is responsible for the harness Refinement step that includes the geometric model generation, execution of various analyses, and the optimization. Based on suitable optimization algorithms, the preliminary harness definition is adjusted in order to achieve the optimum and most feasible result.

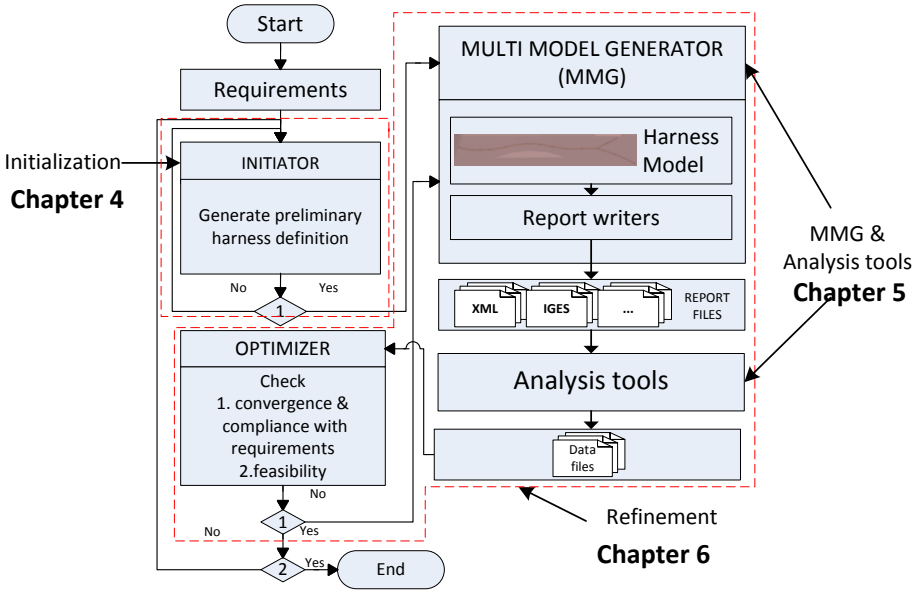


Figure 3-10: Illustration of harness Design and Engineering Engine (DEE)

The harness model generation is carried out by the MMG. In addition, the MMG also extracts specific harness data from the harness models and gives this data as inputs to the analysis tools. Detailed MMG development and implementation is given in Chapter 5.

During 3D harness routing some analyses need to be implemented to check the satisfaction of design rules and to calculate the cost of the harness. Current checks are implemented manually by design engineers and there are no off-the-shelf products available to automate these checks. Hence, a number of analysis tools have been developed in this research. These analysis tools are adopted here to analyse the harness performance which are then fed to the optimizer. The most relevant tools are

1) **Clamp Distance Analysis Tool (CDAT) – to check whether the use of a clamp satisfies the design rules;**
2) **Harness Bend Radius Analysis Tool (HBRAT) − to check whether the harness bend radius is bigger than the minimum allowed bend radius;**
3) **Geometric Collision Analysis Tool (GCAT) − to check the geometric collision between the wire harness and other geometric components;**
4) **Harness CLearance Analysis Tool (HCLAT) − to check whether the harness has enough clearance from the geometric components;**
5) **Harness Separation Analysis Tool(HSAT) − to check whether two harnesses have enough distance from each other;**
6) **Harness Cost Analysis Tool (HCAT) – to calculate the cost of the harness considering the so-called grey areas, such as hot and flammable zones and the reserved spaces.**

Detailed development and functionalities of these analysis tools will be provided in Chapter 5 and Chapter 6.

According to the analysis results, the Optimizer checks the convergence of the design and the satisfaction of constraints, and if necessary it generates a new harness configuration to start another optimization loop until the design has converged.

Chapter 4 Development of the 3D-routing Initialization

As mentioned in Chapter 3, the harness 3D-routing problem can be modelled and solved as an optimization problem. However, because the number and initial values of the design variables are unknown, a conventional optimization method cannot be used directly to support the design automation. Therefore, a two-step, hybrid optimization method (i.e. Initialization and Refinement) has been proposed. The Initialization, presented in this chapter, aims to generate a preliminary harness definition which includes the number and initial values of design variables to enable the following Refinement process, which will be discussed in Chapter 6.

Just as with the input for the current manual routing process, the input of the Initialization includes: 1) the routing environment, 2) the harness electrical diagram, and 3) the design specifications. The output of the Initialization is the preliminary harness definition. This definition already includes the number and position of the clamps and breakouts which are necessary for the following Refinement. An example output of a harness can be found in Figure G-2 in Appendix G.

The Initialization method proposed here has been implemented into a software tool called the *Initiator* to automate the generation of the preliminary harness definition. The position of the Initiator in DEE is presented in Figure 4-1.

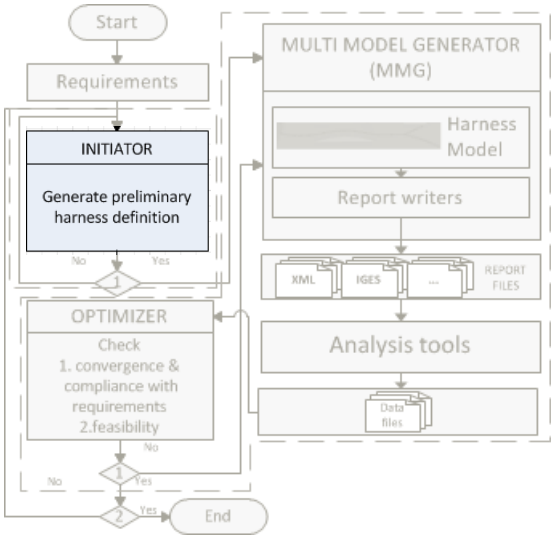


Figure 4-1: Position of the Initiator in the Harness DEE

In Section 4.1, the harness 3D-routing features and some typical pathfinding methods are studied first. Then an innovative 3D-routing strategy for the Initialization is proposed. In Section 4.2 an approach to represent the 3D-routing environment where harnesses will be placed is presented. A 3D pathfinding method which is developed on the basis of the space representation is detailed in Section 4.3. In Section 4.4, a post-process that transfers the simplified harness representation used in the previous pathfinding to the actual harness

definition is presented. Finally, conclusions are given in Section 4.5.

4.1 Development of the pathfinding strategy for the Initialization

In the Initialization, an optimization method that does not rely on the number of design variables (i.e. clamps) needs to be adopted to quickly generate preliminary harness configurations. This method needs to take into account the important features (see Table 4-1) of the wire harness and its design process in order to generate the most realistic and promising harnesses for the following Refinement step.

Table 4-1: Typical features of wire harnesses and considerations of reference for the design process[1]

Multi-original/ multi-destination	One wire harness connects more than one start and one end point; consequently the harness is divided into different branches by breakouts
Fixing distance	Distance between the harness and the attached structure at clamping points; this should be bigger than a certain value to avoid abrasion
Clamping distance	Distance between two adjacent clamps; this should be smaller than a certain value to provide enough support to harness weight and to prevent contact(chafing) between harness and airframe
Geometric obstacle	Geometric components in the routing environments; harnesses are not allowed to go through
Fixable structure	Geometric components that harnesses is allowed to be fixed on
Bend radius	Radius of a curved harness bundle; this should be larger than a certain value to avoid internal damage of the harness due to high bending strains of the harness
Grey areas	Areas that harnesses are allowed to go through but where they need extra protection
Reserved space	Areas that harnesses are intended to go through

In order to successfully handle all these features, typical pathfinding methods that do not rely on the design variables are studied and presented first. However, none of these methods are capable of handling the above features. Therefore, in the second part of this section, an innovative pathfinding strategy inspired by these methods and able to handle all these features is proposed.

4.1.1 Typical pathfinding methods

Pathfinding is the technique of finding a path between a given start and end point that taken into account constraints and one or more objective. The pathfinding is able to find a path from the start point to the end point in a routing space by using a systematic searching method, which does generally not need to know the waypoints (i.e. clamps and breakout for

[1] Details of these features are already presented in Chapter 2.

3D harness routing) *a priori*. Many pathfinding applications can be found in the domains of Computer Games pathfinding, Unmanned Aerial Vehicle (UAV) path planning, Map Navigation, Printed Circuit Board (PCB) design, and so forth.

Based on the routing space representation and the moving method of an agent (e.g. a car or a UAV), the pathfinding methods can be divided into 1) continuous pathfinding methods and 2) discrete pathfinding methods.

4.1.1.1 The continuous pathfinding method

Continuous pathfinding requires that the routing space, which generally includes feasible areas as well as geometric obstacles, is represented by continuous mathematic expressions. An example of the geometric obstacle representation is given in Equation (4.1)[48]. Different geometric obstacles shown in Figure 4-2 can be represented by this function by using different p values. The agent, such as a vehicle, can be moved to any position in the routing space provided there is no violation of the design constraints.

$$h(x,y) = \log((\frac{x-x_c}{a})^p + (\frac{y-y_c}{b})^p) - p\log c = 0 \tag{4.1}$$

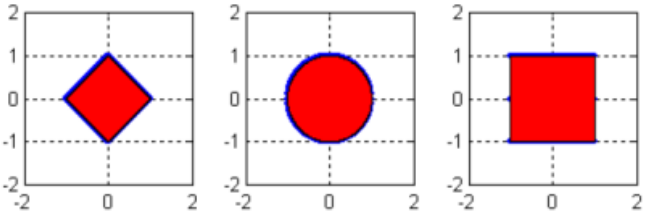


Figure 4-2: Geometric obstacles that can be represented by Equation (4.1) (p=1, 2, 100 respectively)[48]

There are a few studies available on continuous pathfinding. For instance, Matthew G. Earl et al. studied a method to handle the vehicle trajectory-generation problem considering obstacle-avoidance requirements[49]; Qi Gong et al. used a pseudospectral method to plan a path for different types of vehicles[50]; Arthur Richards and others used a Mixed-integer linear programming algorithm to support the aircraft trajectory planning[51].

These studies all focus on finding a path from a single start point to a single end point. No solutions to multi-destination/origin pathfinding have been found. In addition, the clamping distance and fixing distance are not considered in these methods since vehicles do not need to be clamped on the geometric components. The geometric obstacles are represented by simplified models, such as cylinders and cubes, in these researches. None of these researches has a solution on the representation of geometric models as complex as an aircraft.

4.1.1.2 Discrete pathfinding method

For discrete pathfinding, the routing space is represented by a discrete mathematic representation and the agent can only be moved to certain positions such as nodes or waypoints generated during the discretization. Popular discretization approaches are 1) grid (including 2D and 3D grid) generation, 2) road-map generation, and 3) navigation-mesh generation. For example, the original routing space ((a) in Figure 4-3) can be discretized by these approaches to different mathematical representations respectively, as shown in (b), (c) and (d) in Figure 4-3.

Grid generation approach discretizes the routing environment into many small squares(2D) [1] or cubes(3D). This method was elaborated by Lee in 1961[52] and is generally used in some obsolete computer games and PCB design software. The agent in this routing environment can only be moved from the centre/vertex of current square/cube to adjacent ones.

The road map is generated by discretizing the routing environment into some nodes that connect with each other via map edges. The application of this space representation method can be found in current car navigation systems, some computer games, and so forth. [53]. The agent in the routing environment can be moved from the current node along an edge to the adjacent nodes.

The navigation mesh represents the routing space with some convex polygons (including triangles). Its research and application are mainly found in computer game domains[54, 55]. The agent can move freely in the navigation mesh but a post-process[56] is needed to generate an optimum path between the start and end points.

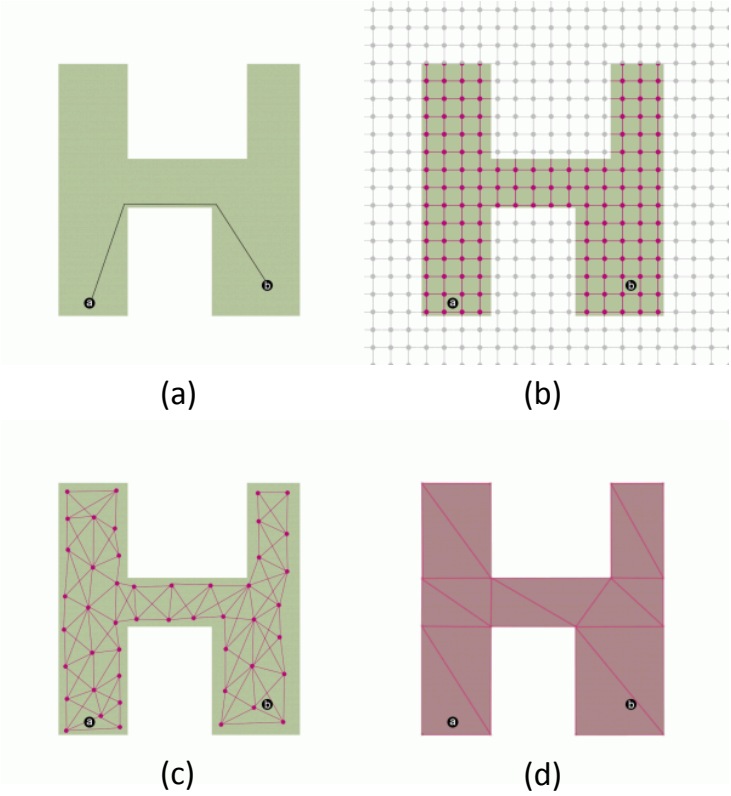


Figure 4-3: Discrete space representations of the routing environment: (a) original routing space, (b) 2D-grid representation, (c) road map representation, (d) navigation mesh[57]

Pathfinding algorithms are responsible for systematically exploring the discrete routing space to find a path between start and end points. Popular pathfinding algorithms are breadth-first, depth-first, best-first, and so forth[58]. In the best-first category, the A* (A star) algorithm,

[1] In addition to square, other 2D-grid shapes such as hexagonal can also be used to represent the routing space.

proposed by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968, is one of the most popular algorithms and it can find an optimum path between the start and end nodes. It explores the routing space by expanding the most promising position selected according to the evaluation of the cost function. It also takes advantage of the position of the destination (i.e. so-called heuristic information) to increase the efficiency.

Some academic researchers have studied harness pathfinding with the discrete pathfinding method. Conru solves the 3D harness routing as a graph search problem and uses a genetic algorithm to place the harness breakouts on the vertexes of a 3D graph that is generated manually to represent the routing space[14]. This method is able to handle the multi-destination problem but does not solve the placement of clamps or handle other design rules. Van der Velden et al. developed a harness routing method based on a 3D volume mesh to automate the harness routing. In their research, the geometric model of a routing environment is transformed into a set of cubes and then the A* algorithm is used to find the harness path, taking into account some constraints of wire harness design[13]. However, multi-destination, placement of clamps, and other design rules (e.g. bend radius) are not tackled here.

4.1.2 A pathfinding strategy for the Initialization

The above-mentioned methods reflect the current state of pathfinding research. However, none or very few methods in both continuous and discrete domains can handle all the typical features in Table 4-1. In this Initialization step, in order to quickly generate a harness preliminary definition and also handle these typical features, namely the multi-destination, placement of clamps, and other typical design rules, a **bi-level, road-map-based pathfinding method** is proposed.

- **Bi-level routing strategy**

This bi-level strategy is proposed solely to handle the multi-destination/origin problem of 3D harness routing. The pathfinding problem of an entire harness is broken down into global (harness)-level optimization and local (branch)-level pathfinding. The bi-level architecture is illustrated in Figure 4-4.

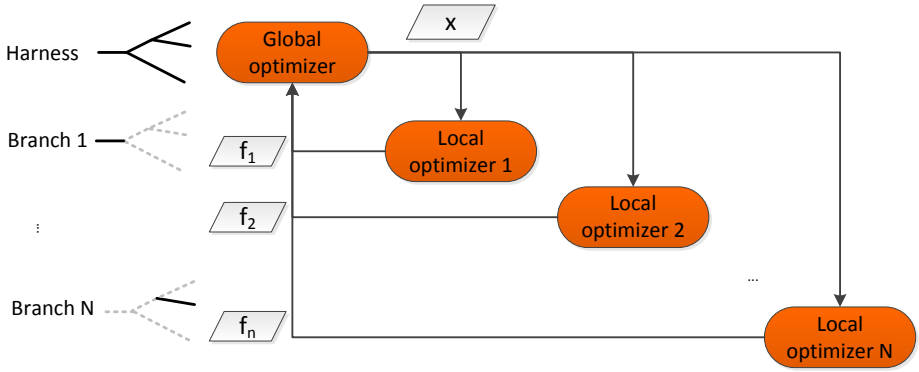


Figure 4-4: Architecture of the bi-level pathfinding

The global-level optimization only moves the breakout points x (i.e. design variables at the global level) to get different breakout configurations. These breakouts indicate the start and end position of each branch. According to the breakout positions, the local-level pathfinding method finds the best path for each branch respectively. When the local pathfinding is finished, the cost of each branch f_i is sent back to the global level. Then the global optimizer

determines whether the optimization is converged. If necessary, it will move the breakout points to generate new configurations for the next iteration. This process iterates until one of the stop criteria, such as the design is converged or the maximum iteration times is reached, is met.

- **The road-map-based pathfinding method**

The aim of the Initialization is to quickly generate a harness preliminary definition including the number and position of clamps for the following refinement.

The calculations that involve the geometric models, such as geometric collision check, is a challenge for the use of the continuous pathfinding method. It is very difficult to represent very complex routing environments such as aircrafts by algebraic formulas such as Equation (4.1). The NURBS representation of geometric models of the routing environment needs to be used for the calculations. These calculations generally are more time-consuming than the algebraic calculation. Moreover, these calculations need to be carried out many times in a complete pathfinding process. As the result, the continuous pathfinding method used for a complex routing environment is not efficient. Therefore, a discrete pathfinding method is adopted here for the pathfinding.

The grid-routing space representation is less flexible since it only allows the harness to move vertically, horizontally, or in some cases diagonally. In addition, the 3D grids exist everywhere in the routing environment and even at the geometric obstacle. This area is actually not accessible for the harness and ignoring this area will save the memory and time consumption.

The navigation mesh is very concise and the pathfinding that is based on it is efficient. However, the points to accommodate the clamps and breakouts (i.e. the design variables in the second pathfinding step) are not given directly in this routing environment, and some design constraints such as bend radius violation cannot be handled during the pathfinding. Therefore a post-process is needed to generate a path for each branch and the design constraints such as bend radius violation needs to be handled here. The design constraints analysis results very likely will influence the pathfinding results. However, this influence cannot be handled when using the navigation mesh since the pathfinding and constraint analyses are separate.

The road-map nodes can be used to place clamps and breakouts and the road-map edges represent the harness branches. During the road-map generation, some design rules, such as clamping distance, fixing distance, and geometric collision-free, can already be considered in a simplified but reasonable way. The road map of a routing environment only needs to be generated once but can be (re)used many times to route different harnesses. For these reasons, the road map is adopted to represent the routing environment.

Based on the road-map representation, the global optimizer can systematically move the breakouts to different nodes to get different harness configurations. Meanwhile, the local pathfinding can use mesh-based pathfinding algorithms, such as A*, to find a path between two nodes. A* does not need to know the design variables *a priori* and actually the number and initial value of the design variables are its output.

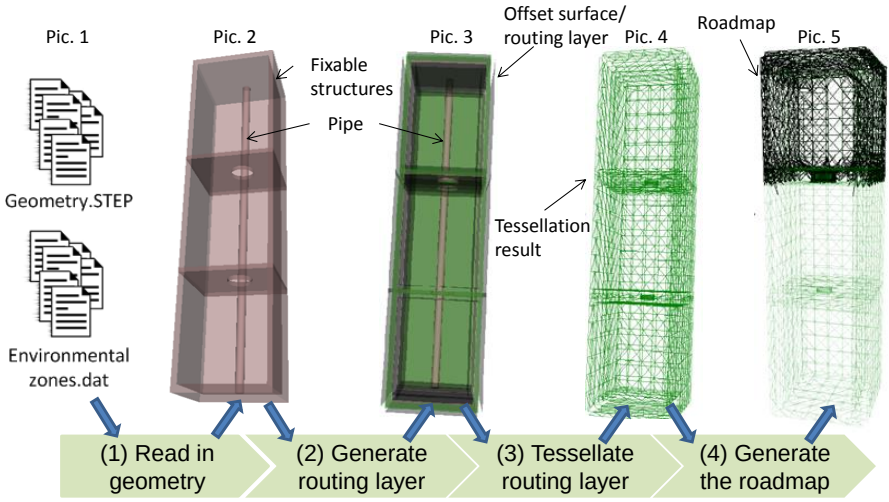
Details of the space representation and the pathfinding method will be elaborated in the following two sections.

4.2 Routing environment representation for the Initialization

The road map representation is selected to represent the routing environment. When

generating the road map, a number of the rules shown in Table 4-1 (i.e. fixable structure, clamping distance, fixing distance, geometric collision-free, and grey zones) are taken in to account. The bend radius and harness multi-origins/destinations will be handled during the pathfinding process presented in Section 4.3.

The generation of the road map includes four steps: 1) geometric model read-in, 2) generation of the routing layer, 3) tessellation of the routing layer, and 4) generation of a road map. These four steps are illustrated in Figure 4-5, where the required input and output of these steps are presented and the design rules handled in certain steps are listed. The final road map is ready to apply the routing algorithm on.



Solved design rules	none	fixable structure; fixing distance	clamping distance; flammable zone	geometry collisions; grey zones

Figure 4-5: The four steps to generating a road map

4.2.1 Geometric model read-in

The inputs of the harness routing environment include the aircraft geometric model defined in a STEP data exchange file[1] and the definition of the environmental zones, such as hot zones and flammable zones. The geometric model is read in by the built-in STEP-file-import function of the KBE system and this model can be operated on the KBE system. The zone information, such as the position and temperature of a heat source, is defined in data sheets. The pre-defined function in the KBE system can load the data sheets and generate a mathematical representation for these zones. Details of the hot zone and flammable zone definition are presented in Appendix C and Appendix D respectively.

Sometimes the geometric model contains too many details for 3D routing. These details, such as rivet holes on skins or thread on an actuator, are not necessary for the 3D routing and they also significantly slow down the 3D routing. Therefore, some geometric simplification methods are developed to simplify these complex geometric models to delete the unnecessary geometric components. The simplified models instead of the actual geometric models will be

[1] STEP is a neutral geometric data exchange file and is defined in ISO 10303-21.

used as the routing environments in both the Initialization and Refinement steps. The details of the simplification are presented in Appendix B.

4.2.2 Generation of the routing layer

Wire harnesses are not allowed to go anywhere inside an aircraft, such as the centre of the fuselage, which is reserved for passengers; and they need to be fixed on the surfaces of the airframe. The fixing distance of the harnesses should be neither very large to avoid the extra cost nor very small to avoid the chaffing between harnesses and structures. In order to keep a certain fixing distance, the so-called routing layer is proposed. A routing layer, which is an artificially-added auxiliary geometry, is an offset of all the fixable geometric surfaces. The routing The offset distance is the same as the fixing distance, which may be different in different routing environments. All the clamping points of the harnesses will be placed on these surfaces. Therefore, the fixing-distance rule will be satisfied.

The offset surfaces do not exist in front of all the geometric surfaces because not all the geometric components in the routing environment are suitable or allowed to have harnesses fixed to them. For example, it is not allowed to drill in the outer skin or in hydraulic pipes to fix a harness. The structures which are allowed to have harnesses fixed to them are called fixable structures and the other geometric components are the non-fixable structures for the wire harness. The non-fixable structures are excluded from the generation of the routing layer. Therefore, they will be excluded from the following mesh and road-map generation process. This guarantees that there is no road-map node located in front of the non-fixable structure and consequently fixing on non-fixable structures will be avoided to a large extent.

The routing layer is illustrated in picture (3) of Figure 4-5. The green surfaces are the offset surfaces and the pipe which is a non-fixable structure is excluded from the routing layer generation.

4.2.3 Tessellation of the routing layer

The aim of this step is to generate a set of points that will be used as waypoints to generate the road map on the offset surfaces. The generation of these points is enabled by a tessellation function of the KBE system.

The tessellation is a method to transform a surface into a set of facets, such as triangles, squares, and hexagons to represent this surface. Triangular tessellation is used here and the offset surfaces are tessellated into a set of isosceles right triangles as shown in Figure 4-6. The short edge is called *Tessellation Size* and the long edge therefore equals to $\sqrt{2}\,Tessellation\,size$. The vertexes of these triangles will be extracted and used as the waypoints to generate the road map in the next step.

The tessellation size is determined by the clamping distance. The selection of a suitable tessellation size is a trade-off process. On the one hand, there should be as few waypoints as possible to decrease the memory consumption and the following road-map calculation time and therefore the size should be as large as possible. On the other hand, since the clamps will be placed on these waypoints, the tessellation size should not be so large that it exceeds the clamping distance.

A suitable tessellation size is illustrated on the left-hand side of Figure 4-6. As shown in this picture, only 8 adjacent points of each vertex should be located inside the read circle (or a sphere for a curved surface) whose radius is the maximum allowed clamping distance. The diagonal movement is allowed here to avoid a sharp (perpendicular) turn. In this scenario, putting clamps on adjacent vertexes will not violate the clamping distance. One of the unsuitable tessellation sizes (too small) is illustrated on the right-hand side of Figure 4-6.

Although putting the clamps on the adjacent vertexes will not violate the clamping distance, more than the necessary number of vertexes will be generated and the number of map edges, which depends on the number of the vertexes (see next sub-section), will also be increased significantly. Consequently, the following pathfinding time will also be increased since more waypoints need to be explored.

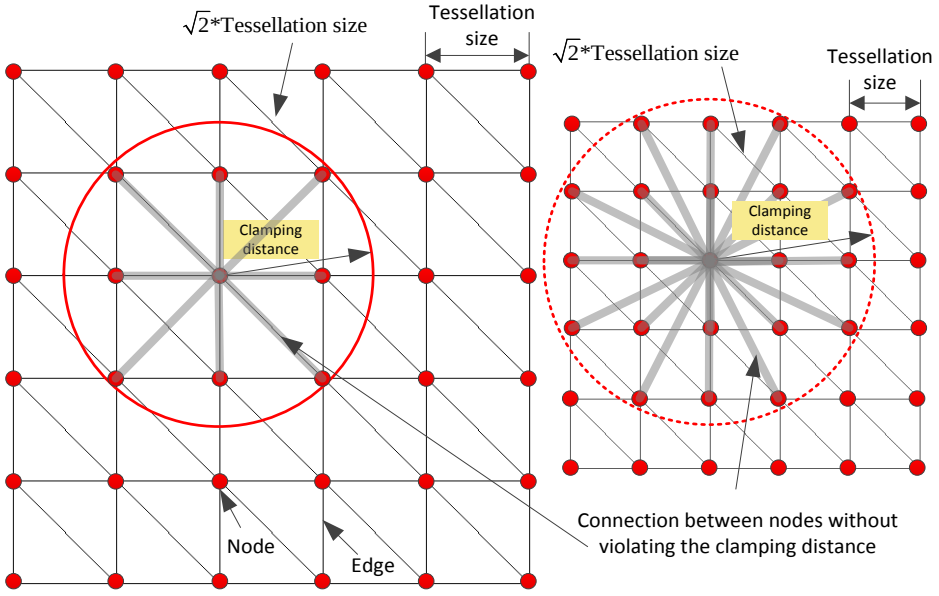


Figure 4-6: Selection of a suitable tessellation size (left: suitable tessellation size; right: too small tessellation size)

According to the 8-adjacent-vertex principle, the tessellation size calculation method is presented in Equation (4.2), where T_s is the tessellation size; $D_{\max}^{clamp}$ is the maximum allowed clamping distance; and s_a is a safety factor adopted to handle the calculation error in the computer system.

$$T_s = \frac{1}{s_a * \sqrt{2}} * D_{\max}^{clamp} = \frac{1}{1.1 * \sqrt{2}} * D_{\max}^{clamp} \approx 0.64 * D_{\max}^{clamp} \tag{4.2}$$

Using the given tessellation size, the built-in function in GDL is able to tessellate the offset surfaces. Picture (4) of Figure 4-5 shows the result of this.

The tessellation size for a routing environment is not uniform and $D_{\max}^{clamp}$ is smaller in the flammable zones than in non-flammable areas and therefore the tessellation size is also smaller there according to Equation (4.2). Smaller tessellation size means more waypoints will be generated. The variable tessellation result is shown in Figure 4-9.

4.2.4 Generation of a road map

In this step, a road map will be generated based on the waypoints extracted from the tessellation result. These points, located on different offset surfaces, will be connected with each other to generate the road map for an entire routing environment.

4.2.4.1 The road-map definition

The road map generated in this step is an undirected weighted graph, where the agent can move in both directions along the map edge and the cost to move along the map's edges depends on the edge's weight. This map is a graph in terms of the data structure definition in computer science and it is defined as $G=(V,E,W)$, as shown in Figure 4-7. V is the vertex set and E represents the edges' set. The cost to move a harness along a map edge depends on the length of the edge, the environmental information (e.g. hot or not), and the harness itself (e.g. its diameter and material). When defining the road map, the length of the edges and environmental information are available but the properties of harness are not. The cost of different harnesses on the same edge may be different. This means that the weight of each edge cannot be defined by a constant number. Instead, W is defined by some parameters (see next sub-section). During the pathfinding, these parameters together with harness properties will determine the weight of each edge in E.

$G=(V,E,W)$;; V is the vertex set; E is the edge set; W is the weight parameters set

$V=\{V_1, V_2, V_3, \ldots, V_n\}$, where $V_i = (x_{V_1}\ y_{V_1}\ z_{V_1})$;; each vertex is a 3D point in the routing environment.

$E=\{E_{ij}, E_{jk}, E_{mn}, \ldots, E_{ns}\}$;; E_{ij} are the edge between vertex V_i and V_j

$W=\{W_{ij}, W_{jk}, W_{mn}, \ldots, W_{ns}\}$;; W_{ij} is the weight parameters of E_{ij}.

$$\begin{aligned} \textit{Connection List} = &((V_1\ (E_{1i}\ W_{1i})\ (E_{1j}\ W_{1j})\ (E_{1k}\ W_{1k})) \\ &(V_2\ (E_{2l}\ W_{2l})\ (E_{2m}\ W_{2m})) \\ &\ldots \\ &(V_n\ (E_{2p}\ W_{np})\ (E_{2q}\ W_{nq}))) \end{aligned}$$

;; *Connection List* shows the connection between the vertexes

Figure 4-7: The road map definition

4.2.4.2 The road-map generation

The geometric collision, the clamping distance, hot and flammable zones, and reserved spaces are considered during the road-map generation.

The first step is to extract the vertexes from all the offset surfaces to form a vertex set V. Then the connectivity between any two vertexes in V is checked. Two vertexes will connect with each other to form a map edge if the edge between the two vertexes does not go through any geometric component and is not bigger than the maximum allowed clamping distance. The vertexes on the adjacent offset surfaces can also connect with each other if the edge between them does not have geometric collision and does not violate the clamping distance. Therefore, vertexes from the different offset surfaces will be connected with each other to form a road map of the entire routing environment.

If there is a connection, then the hot zones, flammable zones, reserved spaces and the distance between these points will determine the weight parameters of the map edge and consequently influence the cost of a harness routed along the edge. The weight parameters is defined as $W=(:L\ 1:e_{cover}\ 2:D_{\max}^{clamp}\ 3:\varsigma_{res}\ 0.5)$, where L is the edge length defined as the Euclidean distance between the two ends of the edge; e_{cover} is the unit length density of the

covering, $D_{\max}^{clamp}$ is the maximum allowed clamping distance, and ς_{res} is the reserved space coefficient.

e_{cover} is proposed to handle the existence of hot zones. A covering is needed when a harness is routed in a hot zone and it is assumed that more covering material will be used if the temperature increases. Therefore the unit length density e_{cover} increases as well. Appendix C presents the definition of a hot zone used in this research. The method to calculate the e_{cover} according to the temperature of the zone is also presented here.

$D_{\max}^{clamp}$ is used to support the calculation of the clamp cost and also handle the existence of flammable zones since more clamps need to be used when a harness is routed in a flammable zone (see the design rule in Sub-section 2.3.2.6). In non-flammable areas, a normal value of $D_{\max}^{clamp}$ (e.g. 24 inches in the fuselage) is given as an input. In flammable zones, it will be smaller and its actual value depends on the distance between the map edge and the flammable pipe causing the flammable zone. The definition of the flammable zone and the method to calculate the $D_{\max}^{clamp}$ in the zone is given in Appendix D.

```
;;;**********************************
;;;Pseudocode of road map generation
;;;**********************************
Begin
 Vertex - list = (#(x_V1 y_V1 z_V1 ) #(x_V2 y_V2 z_V2 ) ... #(x_Vm y_Vm z_Vm ) )
  ;;extract vertexes from routing environment tessellation
 n = length(vertex - list)
 Set Adjacent - list = ()
 Loop for i from 0 to n - 1
  Set current - list = (i)
   Loop for j from i to n - 1
    If (Distance(V_i ,V_j ) <= D_max^clamp ) and ( GeometeryCollision (V_iV_j ) == nil)
      ;;;;;;;;Hot zone;;;;;;;;
      If (edge of V_iV_j is not in hot zone)
        TempCoveringCoe = 0
      else
        TempCoveringCoe = N   ;;value of N depends on temperature of the hot zone
      End
      ;;;;;;;;Flammable zone;;;;;;;;
      If (edge of V_iV_j is not in flammable zone)
        TempDclamp = 24inches
      else
        TempDclamp = M   ;;M depends on the distance between V_iV_j and the pipe
      End
      If (edge of V_iV_j is in reserved space)
        TempRes = 0.5   ;;reserved space coeffient is given as input
      else
        TempRes = 1
      End
      Append (j, (: L 3DDdistance(V_i ,V_j ) : CoveringCoe TempCoveringCoe : Dclamp TempDclamp
     : ResCoe TempRes))
      to current - list
    End
  Append current - list to Adjacent - list
  End
 End
End
```

Figure 4-8: Pseudo code for the road-map generation

It is preferable to route harnesses in reserved spaces. ς_{res} is proposed to handle this preference. In the non-reserved area $\varsigma_{res} = 1$. In the reserved area, ς_{res} equals a number smaller than 1. The actual number is given as an input by the design engineer.

The pseudo code shown in Figure 4-8 reflects the above mentioned road map generation process.

After the previous process, a road map is fully defined. The road map for a typical 3D-routing environment which includes a hot zone, a flammable zone, a reserved space and a geometric obstacle, is illustrated in Figure 4-9. The edge of a road map partly located in the grey areas is considered to be located in the grey areas and the edge of a road map partly located in the reserved space is not considered to be located in the reserved space.

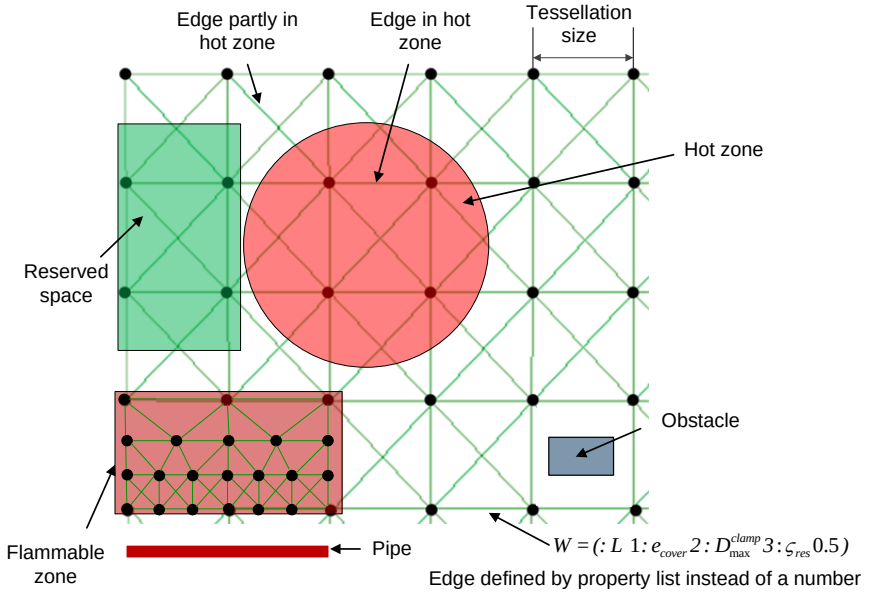


Figure 4-9: Road map of a typical routing environment

In practice, some zones may overlap with each other. If the overlapped zones are the same type (e.g. the hot zone), the worst scenario (e.g. the highest temperature) caused by these zones will be used for the weight parameter calculation. If the zones are different types, then the different parameters influenced by these zones will be calculated separately.

4.3 The harness pathfinding approach

The input for 3D harness routing includes the harness electrical diagram. This diagram which is generally defined as a schematic diagram shows the connectivity among all the receptacles that the harness connects with. The logical position of breakouts is also defined in this diagram. The aim of harness pathfinding is to transform this electrical definition (see Figure G-1 in Appendix G) into a detailed definition (see Figure G-2 in Appendix G) which includes the actual 3D position of the breakouts as well as the number and 3D position of the clamps. The transformation is enabled by the bi-level, road map-based pathfinding strategy.

As mentioned in Sub-section 4.1.2, the bi-level pathfinding approach is an iterative process. In each optimization loop, the pathfinding of an entire harness will be decomposed by the breakouts to a global-level (harness level) optimization and local-level (branch level) pathfinding. Figure 4-10 illustrates the decomposition in one pathfinding loop.

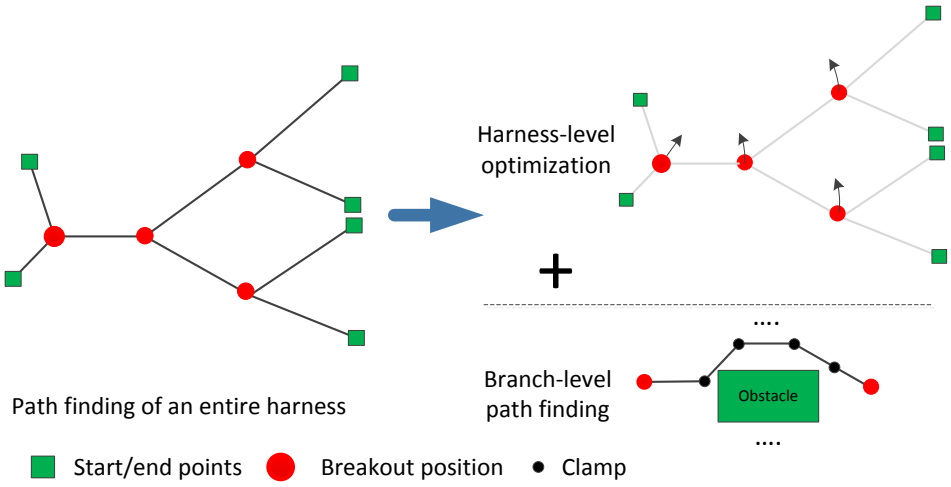


Figure 4-10: Decomposition of the entire harness routing into bi-level path planning

4.3.1 The local, branch-level pathfinding method

The local-level pathfinding is a point-to-point pathfinding problem. Its objective is to find the optimum path (e.g. the path of the minimum monetary cost) between the given start and end points (i.e. breakouts or receptacles) along the road map while satisfying the bend-radius constrains. This pathfinding is suitable to apply many road map-based pathfinding methods. Among these pathfinding methods, the A* (A star) algorithm is considered the most efficient.

4.3.1.1 Introduction of the A* algorithm

A* (A star) is a very popular algorithm that can find a path with the lowest cost between start and end nodes. It was proposed by Peter Hart, Nils Nilsson, and Bertram Raphael in 1968. Since its proposal, A* has been used in many domains, such as computer games, UAV path planning, and car navigation. The pseudocode for the A* pathfinding algorithm is shown in Figure F-1 in Appendix F.

In the A* pathfinding process, the start and end point need to be specified at the beginning. The so-called empty *Openlist* and *Closedlist* are needed to store the adjacent points and visited points respectively during the pathfinding process. The first step of pathfinding is to evaluate the cost of the start point and put the start point into the *Openlist*. The evaluation function is given in Equation (4.3).

$$f(n) = g(n) + h(n) \tag{4.3}$$

Here, n is the current node. Now it is the start point and might be any node in the searching space; $g(n)$ means the movement cost from the source to the current node n on the road map; and $h(n)$ is the estimated movement cost from the current node on the map to the destination node. $f(n)$ is the total cost of the current node n.

Then, the minimum cost point in the *Openlist* will be found and set as "*parent node*". This node will be deleted from the *Openlist* and added to *Closedlist*. Of course, since *Openlist* only has the start node at this moment, the start node is the minimum cost point.

In the next step, the cost of the adjacent points of the *parent node* will be evaluated with Equation (4.3). These adjacent points, their *parent node,* and their cost $f(n)$ will be added to

the *Openlist*. In this process, the adjacent node of the "*parent node*" might either already have been detected before and stored in the *Closedlist* or the node is unreachable. In this case, this node will be ignored. If the adjacent node has also been detected before but stored in the *Openlist*, the program will recalculate the cost of the node based on the current *parent node* and compare the current cost with the previous value. If the current cost of this node is smaller than the previous one, the *parent node* of this node will be updated to the current *parent node* and its cost will be updated to the smaller value. This update ensures that there is always a lowest-cost path between the start point and current *parent node* and consequently guarantees that the lowest-cost path between the start point and end point will be found. This process continues until the minimum cost node in *Openlist* is the end node.

The A* algorithm has three properties i.e. *Convergence*, *Admissibility*, and *Monotonicity*. These three properties of the original A* algorithm are already proved[59] and generally acknowledged. They are presented here to provide the theoretical preparation for the reasonable update of the cost-calculation method of the A* algorithm below.

Convergence means that if the start point and end points have a connection(s), the A* algorithm can always find a path between them within a limited number of steps in both a finite and infinite search space such as a finite graph and infinite graph.

Admissibility guarantees the optimum result of the pathfinding. It specifies that the heuristic information on each node $h(n)$ should always be smaller than or equal to $h^*(n)$, aiming for an optimum path. $h^*(n)$ is the optimal distance from the current node n to the target node. If $h(n) > h^*(n)$, then the admissibility will disappear and the A* algorithm turns to the greedy best-first search algorithm[60]. The value of heuristic information $h(n)$ determines the searching efficiency of the A* algorithm. With a larger $h(n)$, the pathfinding will explore fewer points and be highly efficient. Therefore, the $h(n)$ should be as close as possible to $h^*(n)$, although in most cases it is not easy to find the $h^*(n)$ for each node.

Monotonicity means that 1) $h(t) = 0$ and 2) if $n+1$ is the child node of n, then $h(n) - h(n+i) \le c^*(n, n+i)$. t is the target point and $c^*(n, n+i)$ is the real cost of moving the agent from n to $n+1$. From the monotonicity we can conclude that the $f(n)$ for all the nodes from the start point to the end point is non-decreased, i.e. $f(n) \le f(n+i)$.

4.3.1.2 Update of the cost-calculation method to handle the harness routing problem

As shown in Chapter 3, the cost of a harness branch depends on the branch length and cost coefficients which are influenced by the bundle diameter, the amount of clamps, and the required protective layers for the grey areas. The original cost function of the A* algorithm generally only takes care of the edge length. Therefore, the cost function needs to be updated.

4.3.1.2.1 Update of the cost-evaluation function

The updated evaluation function also consists of two parts, $g(n)$ and $h(n)$. The calculation method for them is defined in Equation (4.4). In order to keep consistency between the Initialization and the Refinement of the two-step optimization, Equation (4.4) is adapted from the actual harness cost calculation function defined in Equations (6.1), (6.2), (6.3), and (6.5) in Chapter 6.

$$
\begin{aligned}
& f(r_{bundle}, p_u, e_{bundle}, C_m, C_i L, D_{\max}^{clamp}, e_{cover}, th_{cover}) \\
& = LCo = L(Cob + Coc + Cop) \\
& = L(\pi {r_{bundle}}^2 p_u e_{bundle} + \frac{(C_{m\,j}^{\ k} + C_i)}{D_{\max\ j}^{clamp\,k}} + \pi p_u e_{cover}(2 r_1^{cover} th_{cover} + {th_{cover}}^2))
\end{aligned}
\tag{4.4}
$$

In this function, each parameter has the same meaning as the definition in Chapter 6.

Cob	-cost coefficient of the bundle
Coc	-cost coefficient of the covering
Cop	-cost coefficient of the protection
L	-length of bundle; the summation of the length of the road-map edges
r_{bundle}	-radius of the bundle cross section
p_u	-unit price
e_{bundle}	-density of the bundle material
C_m	-material cost of a clamp
C_i	-installation cost of a clamp
D_{max}^{clamp}	-maximum allowed clamping distance
e_{cover}	-density of the covering material
th_{cover}	-thickness of the covering
r_1^{cover}	-inner radius of the covering; the same as r_{bundle}

A bundle has a uniform radius and r_{bundle} can be calculated before routing. Unit price p_u, bundle density e_{bundle}, and the thickness of covering th_{cover} are given as inputs and these do not change during the routing process. The material and installation cost of all the clamps on the same bundle are assumed to be the same since the same bundle needs the same type clamps. Both of them can also be calculated beforehand.

During the pathfinding, the actual variables of the harness cost function are 1) the length of segment L, 2) the density of the covering material e_{cover}, and 3) the maximum allowed clamping distance D_{max}^{clamp}. These three parameters are already stored in the weight parameters list of all the edges of the road map. $g(n)$, the actual cost of moving a harness along an edge, can be conveniently calculated using the parameters list together with Equation (4.4).

4.3.1.2.2 Discussion of admissibility of the updated function

When calculating $h(n)$ by using with Equation (4.4), estimations of L, e_{cover}, and $D_{\max}^{clamp}$ are needed since the information of the unexplored edges of the current node n is unknown by the routing algorithm. When estimating the value of the edge property, the admissibility of the A* algorithm needs to be considered, aiming for the optimum result. The $h(n)$ calculated according to the estimated L, e_{cover} ,and $D_{\max}^{clamp}$ must not be bigger than the actual moving cost $h^*(n)$. According to the definition in Equation(4.4), the heuristic value $h(n) = LCob + LCoc + LCop = f_b + f_c + f_p$, where f_b, f_c, and f_p are the bundle cost, clamp cost, and protection cost respectively. Therefore, $h(n) \le h^*(n)$ can be guaranteed by making

sure that $f_b \le f_b^*$, $f_c \le f_c^*$, and $f_p \le f_p^*$ respectively.

- **Admissibility discussion on the bundle cost f_b**

The bundle cost is

$$f_b = L \times Cob = L \times 2\pi r_{bundle}^{\ 2} p_u D \tag{4.5}$$

It is determined by the length L and the sub-cost coefficient Cob. Cob is uniform for an entire bundle and therefore it is a constant for the calculation. The estimated length L is set as the 3D distance between the current node n and the target point. Since the 3D distance is never larger than the real length L^* and $Cob \ge 0$, it can be concluded that $L \times Cob \le L^* \times Cob$, namely $h_b(n) \le h_b^*(n)$.

- **Admissibility discussion on the clamp cost f_c**

The clamp cost is

$$f_c = L \times Coc = L \times \frac{(C_{mj}^{\ k} + C_i)}{D_{\max\ j}^{clamp\,k}} \tag{4.6}$$

L is the length of the harness. The estimated distance L is set as the 3D distance between the current node n and the target point, therefore $L \le L^*$. $\frac{(C_{mj}^{\ k} + C_i)}{D_{\max\ j}^{clamp\,k}}$ is the sub cost coefficient of the clamps. $C_{mj}^{\ k} + C_i$ is the total cost of a clamp and it is identical for all the clamps on a bundle section Coc of a bundle section only depends on the maximum allowed clamping distance $D_{\max}^{clamp}$. $D_{\max}^{clamp}$ is not uniform. In the flammable area, a smaller clamping distance will be applied so consequently Coc increases. This means using the maximum allowed clamping distance in a wiring zone (e.g. 24 inches inside fuselages) will guarantee $Coc \le Coc^*$, where Coc and Coc^* are the estimated and actual clamp cost coefficient respectively.

The heuristic cost and actual cost of the clamp from the current node n to the target node are $h_c(n) = L \times Coc$ and $h_c^*(n) = L^* \times Coc^*$ respectively. Because $0 \le L \le L^*$ and $0 \le Coc \le Coc^*$, we can conclude that $h_c(n) \le h_c^*(n)$. The admissibility of the clamp cost is maintained when using the maximum $D_{\max}^{clamp}$ and the 3D distance L to evaluate the heuristic information.

- **Admissibility discussion on the covering cost f_p**

The covering/protection cost is

$$f_p = L \times Cop = L \times \pi p_u e_{cover} (2r_1^{cover} th_{cover} + th_{cover}^{\ 2}) \tag{4.7}$$

L is the length of the harness. $\pi p_u e_{cover} (2r_1^{cover} th_{cover} + th_{cover}^{\ 2})$ is the sub cost coefficient of the coverings. The covering is only used in the hot zone and the thickness th_{cover} for all the coverings is the same and the density of the covering material e_{cover} depends on the temperature of the routing environment. In hot zones, $e_{cover} > 0$ (see Appendix C) and in normal, non-hot areas, $e_{cover} = 0$. When estimating the Cop, the lowest e_{cover} in the entire routing environment is used. Therefore the estimated Cop is not smaller than 0 and not larger than the actual Cop^*. Here, the estimated distance L is also set as the 3D distance between the current node n and target point, therefore $L \le L^*$.

The heuristic cost and the actual cost of the covering from the current node n to the target

node are $h_p(n) = L \times Cop$ and $h_p{*}(n) = L{*} \times Cop{*}$ respectively. Because $0 \le L \le L{*}$ and $0 \le Cop \le Cop{*}$, we can conclude that $h_p(n) \le h_p{*}(n)$. The admissibility of the covering part is maintained when using the minimum e_{cover} and the 3D distance L to evaluate the heuristic information.

From the previous descriptions, it can be concluded that $h(n) = h_b(n) + h_c(n) + h_p(n) \le h{*}(n) = h_b{*}(n) + h_c{*}(n) + h_p{*}(n)$, namely the cost calculation method used here has Admissibility.

4.3.1.3 The method to handle a preference to route harnesses in a reserved space

It is preferable to route harnesses in reserved spaces. As shown in Sub-section 2.3.2.7, this preference is transferred into the reserved space coefficient ς_{res} which is smaller than 1. An auxiliary harness cost $f_{auxcost}$ is calculated by multiplying ς_{res} by the actual harness cost, namely $f_{auxcost} = \varsigma_{res} f_{actualcost}$.

When calculating $g(n)$ for the current node, the actual ς_{res} stored on the map edges and the actual harness cost calculated by Equation (4.4) will be used. When estimating the heuristic information, the minimum ς_{res} of all the reserved spaces and the admissible heuristic cost $h(n)$ calculated in the last sub-section are used. Since $min(\varsigma_{res}) \le \varsigma_{res}$ and $h(n) \le h{*}(n)$, $\min(\varsigma_{res}) h(n) \le \varsigma_{res} h{*}(n)$. The admissibility A* is still maintained.

The auxiliary cost $f_{auxcost}$ is only used in the harness 3D-routing phase (including Initialization and Refinement) to support the decision-making of the optimizers and it does not influence the actual harness cost.

4.3.1.4 Solution of the bend radius in the harness pathfinding

When routing a harness, the bend radius of a harness $r_{\min}^{bending}$ needs to be not smaller than the allowed bend radius $r_{allowed}^{bending}$, i.e. $r_{\min}^{bending} \ge r_{allowed}^{bending}$. This rule has to be satisfied in the Refinement step presented in Chapter 6. In order to provide a promising preliminary harness definition to the Refinement, it is also considered here.

In the Refinement, the central curve of a harness branch is used to evaluate the actual minimum bend radius. This method is accurate and the same as the current manual design process. However, it is not possible to apply it here since the harness central curve is not defined yet; even the waypoints of the curve are not fully known during the local-level pathfinding process. Even if some waypoints of the curve can be estimated, the geometry-based bend-radius evaluation method is still not suitable for this phase since 1) the geometric model-involved calculation is very time-consuming and 2) the harness will still be modified in the Refinement phase. It is too early to use such a time-consuming, high-fidelity geometric model-involved method here. The bend-radius check in the Initialization step should be road map-based and able to quickly generate a result.

The road map-based check method that prevents sharp turns can be found in other routing domains, such as computer games. For instance, the turning radius shown in Figure 4-11 (similar to the bend radius) is handled by limiting the angle between the previous moving path (i.e. the path from $n-1$ to n) and the next moving path (i.e. the path from n to $n+1$). In this example, only turns with 0 or 45 degrees are allowed.

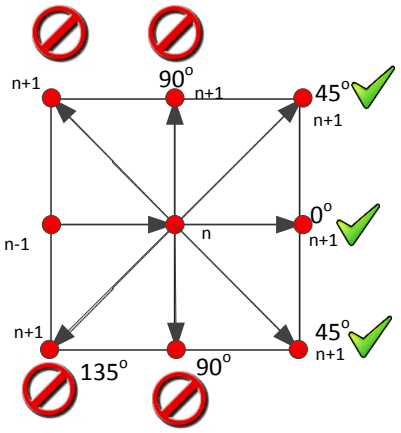


Figure 4-11: Illustration of a turning-radius check method used in computer games

In the case of harness routing, the situation is more complex. Whether the turn of a harness bundle is allowed is not only dependent on the angle between the two line segments $L_{n-1,n}$ and $L_{n,n+1}$, but also dependent on the length of the two line segments, the diameter of the bundle, and the other waypoints.

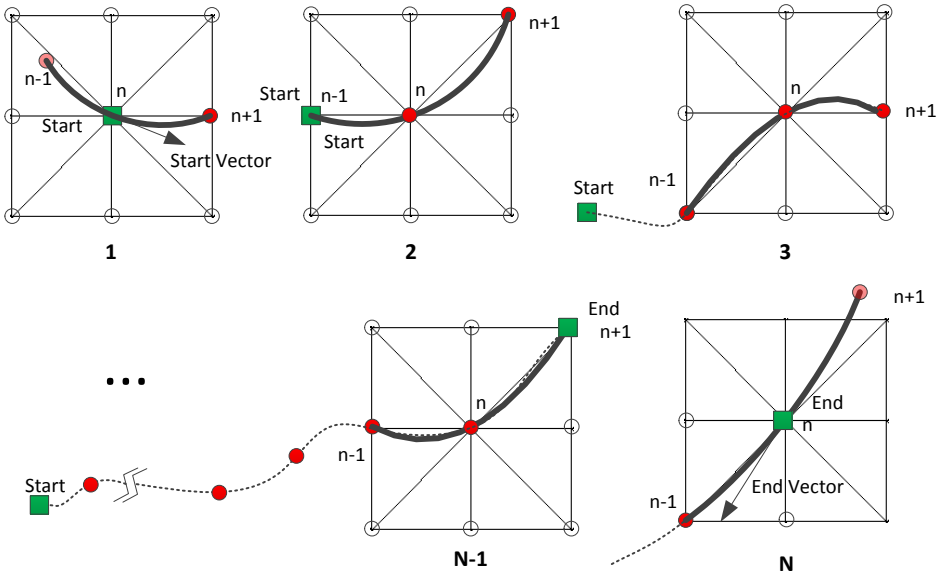


Figure 4-12: Illustration of the minimum bend radius check in the Initialization

In order to address the bend-radius check challenge, an approach called the Curve Segment Bend Radius Pre-Calculation (CSBPC) is proposed. This approach uses the current node n, the previous node $n-1$ and the next node $n+1$ to generate a curve with the *fitted-curve* class in the KBE system; then it measures the minimum bend radius of the curve and checks whether the node $n+1$ is satisfying the inequality $r_{\min}^{bending} \geq r_{allowed}^{bending}$. If so, the harness can go to

the next node $n+1$. The process of the CSBPC approach is illustrated in Figure 4-12.

The check starts from the start point. The position and direction (i.e. the direction of the receptacle) of the start point are given as inputs. Since at the start point only two points are available (see diagram 1 in Figure 4-12), an auxiliary point $n-1$ is built using the direction vector of the start point. The auxiliary point is the point generated by moving the start point along the reversed direction of the start direction by a distance that equals the Euclidean distance between point n and $n+1$. These 3 points are used together to generate a fitted curve. The minimum bend radius of this curve $r_{\min}^{bending}$ will be used as the actual bend radius of this bundle to determine whether the point $n+1$ is feasible for the harness to go. If $r_{\min}^{bending} < r_{allowed}^{bending}$, the point $n+1$ will not be taken as the next waypoint. Otherwise this point is a feasible point in terms of the bend radius; however, whether this point will be selected as the next point is also determined by the other terms of the A* algorithm.

This process continues until the end point is a feasible point and the other A* stop criteria are satisfied. The feasibility of the end point means accessibility of the end point and the smoothness of the curve at the end point. As shown in diagram N-1 of Figure 4-12, the accessibility is checked with the position of the last three points, namely whether the harness can go to the end point $n+1$ from the point n according to $r_{\min}^{bending} \geq r_{allowed}^{bending}$. If so, the end point will become the current point n and an auxiliary point will be built using the direction of the end point. The process is similar to building the auxiliary point at the start point and the result is shown in diagram N of Figure 4-12. If the auxiliary point is feasible for the end point, then the smoothness of the harness at the end point is guaranteed. Otherwise the A* algorithm will explore other points and try to access the end point from other directions.

The CSBPC method checks not only the angle between the ingoing and outgoing paths but also the length of the curve going through the 3 points. This can prevent a very thick harness being routed along 3 very close points, and consequently avoiding a sharp turn.

In the previous description, the minimum bend radius is calculated by using the built-in function of the *fitted-curve* of the KBE system. This calculation involves a geometric model and is relatively slower than, for example, an algebraic calculation. This time-consuming feature makes this check unsuitable for the Initialization step and therefore an improvement on the calculation efficiency is needed.

During the development of this method, it was noticed that the minimum bend radius of the 3-point fitted curve generated with the KBE system depends on the norm of the incoming path vector $vector-1$ and outgoing path vector $vector-2$ and the angle ∂ between the two vectors, as illustrated in Figure 4-13.

When the angle ∂ is fixed, the minimum bend radius of the fitted curve is linear to $\|vector-1\|+\|vector-2\|$ (i.e. the norm summation of the two vectors). The proportion of $\|vector-1\|$ and $\|vector-2\|$ determines the position where the minimum bend radius occurs but not the minimum bend radius value itself.

When $\|vector-1\|+\|vector-2\|$ is fixed then the minimum bend radius depends on the angle ∂. The relation between $r_{\min}^{bending}$ and ∂ is shown in Figure 4-14, where $\|vector-1\|+\|vector-2\|$ is set to the unit length (i.e. $1mm$). This figure is built on the basis of a large number of calculations of the actual fitted curves.

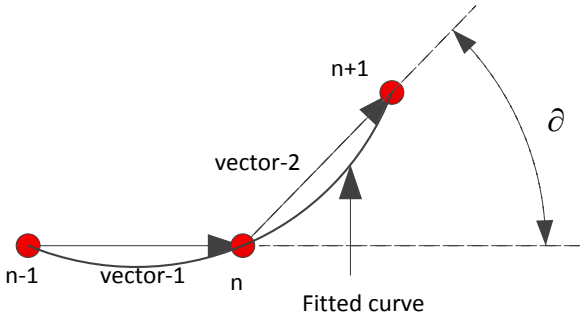


Figure 4-13: Illustration of a fitted curve and its 3 waypoints

According to the two relations, Equation (4.8) is proposed to calculate the minimum bend radius.

$$r_{\min}^{bending} = (\|vector-1\| + \|vector-2\|)R_{bm}(\partial) \tag{4.8}$$

$\|vector-1\|$ and $\|vector-2\|$ are the distances between point $n-1, n$ and $n, n+1$ respectively. $R_{bm}(\partial)$ is the function that calculates the minimum bend radius of unit length $\|vector-1\| + \|vector-2\|$, according to the data presented in Figure 4-14.

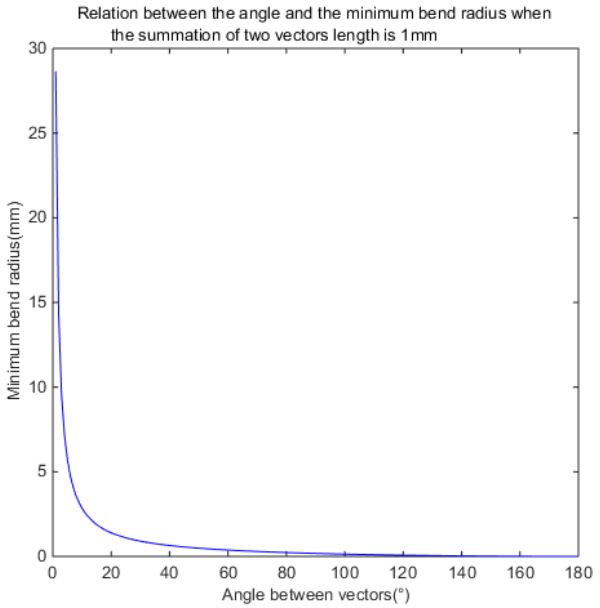


Figure 4-14: Relation between the angle and the minimum bend radius of a fitted curve

During the 3D pathfinding, there is no guarantee that the minimum bend radius of the 3-point curve is the same as the one in the actual bundle. Therefore, as shown in Equation (4.9), a coefficient Coe_{pre} is added to Equation (4.8) to provide the design engineers with more control of the automatic routing process.

$$r_{\min}^{bending} = (\|vector-1\| + \|vector-2\|)R_{bm}(\partial)Coe_{pre} \tag{4.9}$$

This coefficient can be 1) increased to loosen the bend radius constraint, aiming to find a better preliminary result in terms of the objective function, or 2) decreased to tighten the constraint to provide a more promising preliminary result that, with high probability, is or can be transferred to a feasible solution in the Refinement. In the first scenario, the bend-radius check only excludes very unrealistic results. In the second scenario, only very promising results are kept for the following Refinement. Here, the default value of Coe_{pre} is 1.

4.3.2 The global, harness-level optimization method

The global-level pathfinding is responsible for moving the breakouts to get different harness configurations and, according to the breakout configurations, calculating the entire harness cost. The entire harness cost is calculated with the global-level objective function, which is defined as the summation of all the branch costs, as shown in Equation (4.10).

$$f(x) = \sum_{i=1}^{N_b} f_i(x) \tag{4.10}$$

The design variables x represent the breakouts, which are a set of 3D points in the searching space. f_i is the cost of harness branch i. It is calculated by the local pathfinding. N_b is the number of all the branches. The number of breakouts as well as branches is given as inputs.

The inputs of the pathfinding include a road map and harness electrical definition. Firstly, the road map is updated to include the receptacles that the harness needs to connect with. The coordinate of the receptacles are added to the original map with connections to their adjacent vertexes. The electrical definition shows the logical connection among all the branches. According to this, the harness can be decomposed into branches. Both the updated map and topology structure are illustrated in Figure 4-15. In addition, the data needed to calculate the harness cost, such as the bundle diameter, is also included in the electrical definition.

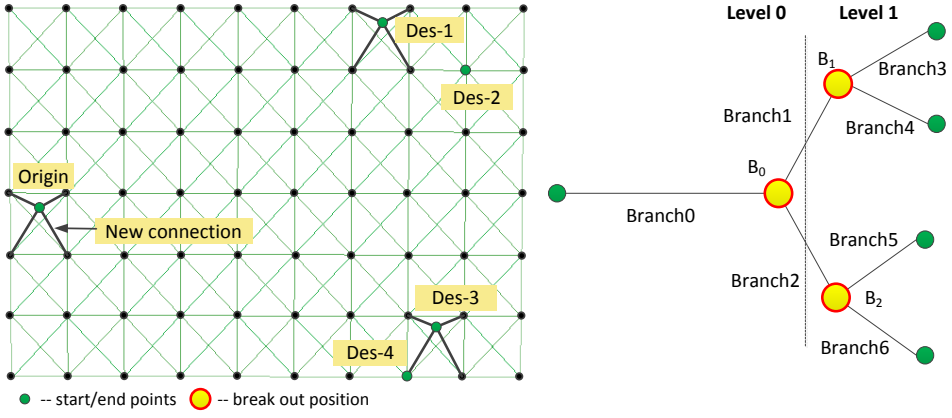


Figure 4-15: Updated map (left) and harness topology structure (right)

4.3.2.1 Introduction of the Hill Climbing algorithm

The global-level optimization uses the Hill Climbing algorithm. Hill Climbing is a mathematical optimization method which belongs to the local search family. It is a heuristic method and uses the information from the explored area to accelerate the search process. In general, the algorithm starts generating random values for the design variables and calculates

the objective function according to these values as a benchmark. Then it changes each single element of the design-variable vector one by one to get a set of the new design-variable combinations, as shown in Table 4-2. At the same time it evaluates the function cost and finds the best combination from the set. If the best one is better than the benchmark in terms of the objective function, this one will be set as the new benchmark and the design-variable combination will be accepted. The accepted design variables will be used as the start position of the design variables in the next loop. This process iterates until no further improvements can be found. More on the Hill Climbing algorithm and its applications can be found in [59, 61].

Table 4-2: Generation of the new design-variable combination

Design- variable vector	$X_1,$	$X_2,$	... X_n
New design-variable combinations	$X_1 + \Delta X_1,$	$X_2,$	... X_n
	$X_1 - \Delta X_1,$	$X_2,$	... X_n
	$X_1,$	$X_2 + \Delta X_2,$	... X_n
		...	
	$X_1,$	$X_2,$	... $X_n - \Delta X_n$

4.3.2.2 Development of the pathfinding approach

The algorithm used here is slightly different to the original Hill Climbing in terms of the 1) placement of design variables (i.e. breakouts) and 2) method to move design variables in the first few steps in order to avoid the geometric model of different branches colliding with each other.

1) Placement of the breakouts

The number of breakouts is an input of 3D routing. However, the position of the breakouts, namely the initial value of the design variables is not known *a priori*. Normally, the Hill Climbing algorithm generates the random value for the design variables that do not have initial values. However, in 3D routing, these may cause the crossover (i.e. geometric collision) between different branches, as shown in left-hand side of Figure 4-16. In order to handle this issue, the initial values of design variables are set to the coordinate of the start point of the global-level pathfinding (see right-hand side of Figure 4-16).

Sometimes the start point where the breakouts are placed is the same as the start point (i.e. origin) of the harness, as shown in diagram (a) in Figure 4-17. However, this single origin cannot be found in a harness that has multi-origins and multi-destinations, as shown in diagram (b) in Figure 4-17. In this case, it is specified that any origin can be the start point of the search and other origin(s) will become the destination(s), as shown in diagram (c) in Figure 4-17. Then the multi-origin and multi-destination pathfinding problem becomes the multi-destination problem.

When using this method, the wire flow direction (from the origin to the destination) may be different from the search direction (see the branch connected with Destination-3 in diagram (c) in Figure 4-17). The different directions have no influence on the road-map-based bi-level pathfinding. However, when transferring the pathfinding result from polylines to a smooth harness, an extra process is needed to guarantee the smoothness of the harness at the breakouts. This process will be detailed in Section 4.4.

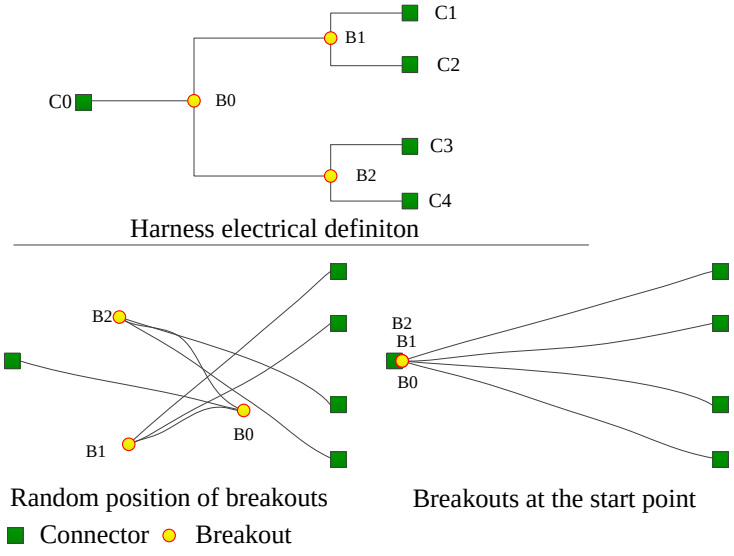


Figure 4-16: Breakouts are placed randomly (left) and at the start point (right)

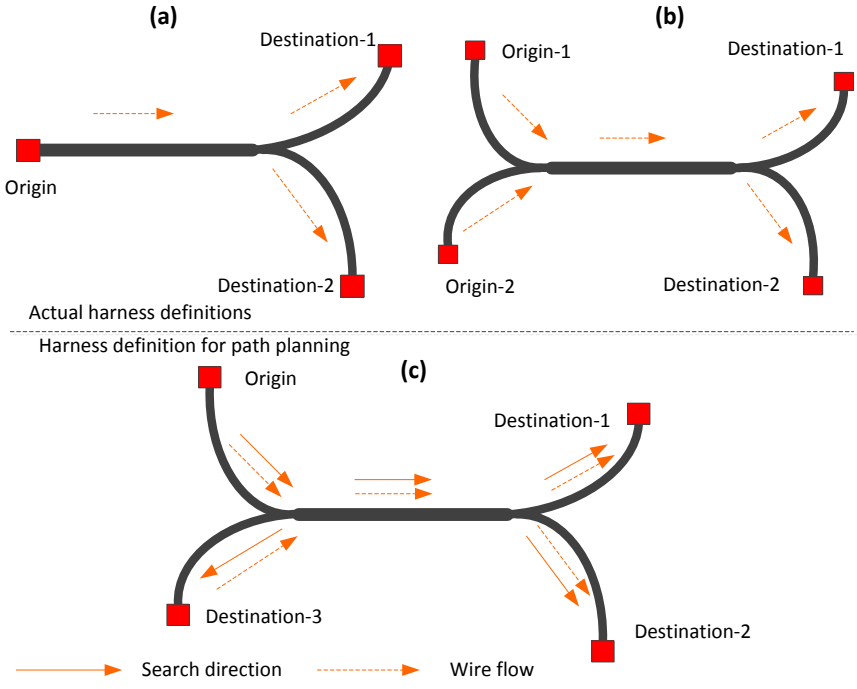


Figure 4-17: Different harness definitions (a): multi-destinations; (b): actual multi-destinations & multi-origins; (c): multi-destinations for path planning

2) Strategy to move the breakouts

The original Hill Climbing algorithm moves all the design variables (i.e. breakout points) in each optimization loop. Moving the design variables in the modified Hill Climbing algorithm depends on the breakout levels. The definition of the breakout levels is illustrated on the right-hand side of Figure 4-15. The breakouts that are one branch away from the origin are level-0 (the highest level); breakouts which are two branches away from origin are level-1; and the breakouts which are one branch to destinations are the lowest level (the biggest level number). In the first loop, only the lowest-level breakouts are moved away from the start point. In the second loop the lowest level and second lowest level breakouts will be moved. Gradually, all the breakouts will be moved in each optimization loop. Similar to the case shown in the left-hand side of Figure 4-16, this method can avoid the crossover (i.e. geometric collision) between different branches during the pathfinding process.

4.3.2.3 Illustration of the pathfinding steps

Now the pathfinding can start. The pseudocode of the Hill Clamping algorithm is given in Figure F-2 in Appendix F. The snapshots of typical pathfinding loops are presented in Figure 4-18 to illustrate the pathfinding method. The electrical definition of this harness is shown in Figure 4-15.

Firstly, the initial values of design variables (breakouts) are set to the coordinate of the origin, as shown in diagram A of Figure 4-18. The initial harness cost is calculated based on this breakout configuration as the initial benchmark. Here, the lengths of branches 0, 1 and 2 are zero. Consequently, their cost equals 0 as well. The total cost of this harness configuration depends on the cost of branches 3, 4, 5, and 6, which are calculated by the local pathfinding program. The result will be inserted into a list called *breakouts-com-cost-lst*, which is defined in the pseudocode presented in Figure F-2 in Appendix F.

Then the iteration starts. In each loop, firstly, the minimum cost of the previous breakout combinations will be extracted from *breakouts-com-cost-lst* and compared with the benchmark cost which is infinite at the beginning. If the benchmark cost is bigger than or equal to the minimum cost, the minimum cost will be assigned to the benchmark cost and the loop continues. Otherwise the search stops. Provided the search is continuous, then each breakout will be transferred to its adjacent nodes to form a new harness configuration and the new harness cost will be calculated accordingly. When a breakout is moved, other breakouts stay in their position. When all the adjacent nodes of the current breakout are explored, this breakout will stay in its original position again and the next breakout will be moved to its adjacent nodes and the cost of the harness configurations will be evaluated. This process continues until all the breakouts have explored their adjacent nodes. Then one loop finishes.

During this process, since all the breakout configurations are stored in *breakouts-com-cost-lst*, it is possible that one set of breakouts has already been explored before. If this happens, the following evaluation of this breakout configuration will stop and the process moves to the next breakout configuration. The combination of breakouts and the harness cost will be stored in the *breakouts-com-cost-lst*. During the storing process, the list is sorted from small to large in terms of harness cost. Therefore, the first item in the *breakouts-com-cost-lst* always has the minimum cost. The minimum cost breakout combination will be set as the original breakout positions for the next loop. The snapshot B in Figure 4-18 shows the result of the first loop. In this loop only breakouts 1 and 2 are moved since they are the lowest level and breakout 0 stays at the start point. This snapshot contains two steps. Firstly, the breakouts explore their adjacent nodes respectively. Then, the algorithm chooses the breakout combination that has the minimum cost as the original position for the next loop. A similar process also happens in

snapshots C, D, and E.

In snapshot F, the breakouts are still moved to their adjacent places. However, the new breakout configurations do not decrease the result of the cost function any more, and therefore the pathfinding stops.

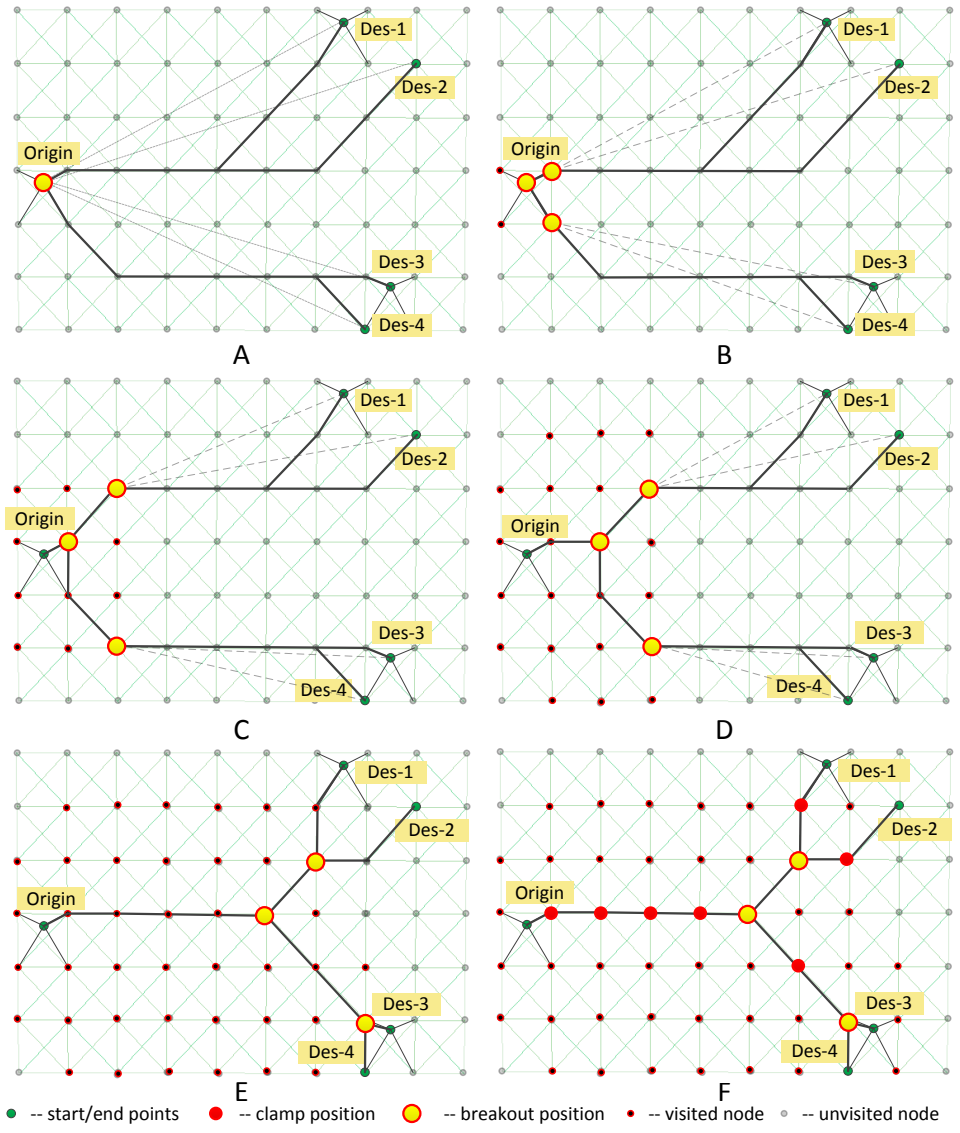


Figure 4-18: Snapshots of the harness global pathfinding process

From diagrams A to D, some branches going from breakouts to destinations share the same waypoints and part of their paths overlap with each other. Although they share the same path, these paths and their cost calculation are independent of each other. In Figure 4-18, the connectivity between breakouts and destinations is illustrated by the dashed line.

With this Hill Climbing method, the positions of breakouts with minimum cost can be found. When this process has finished, the position of the breakouts as well as the waypoints used to place clamps will be added to the harness definition. An example of this new definition is given in Figure G-2 in Appendix G.

4.4 The post-process in the Initiator to generate a detailed harness definition

With the previously calculated position of the breakouts and clamps, some polylines can be defined to represent the harness, as shown on the left-hand side of Figure 4-19. However, the polyline form is not sufficient to represent the actual smooth harness presented on the right-hand side of Figure 4-19. Therefore, a post-process is needed to generate an actual harness from the 3D position of the breakouts and clamps.

The actual harness definition will be given in Chapter 5 when discussing the harness parametric modelling module. This definition and the input requirement of the parametric modelling module will be the guidelines for this post-process since the output of this post-process will be used as the input of the parametric modelling module to generate harness geometric models, according to the Harness DEE framework. An example of the required input definition of the parametric modelling module is given in Figure G-3 in Appendix G. Although this definition contains lots of parameters, a large proportion of them are either given as inputs (e.g. positions and directions of connections) or already calculated in the previous steps (e.g. the 3D positon of clamps). The parameters that need to be calculated in the post-process only relate to the clamp and breakout.

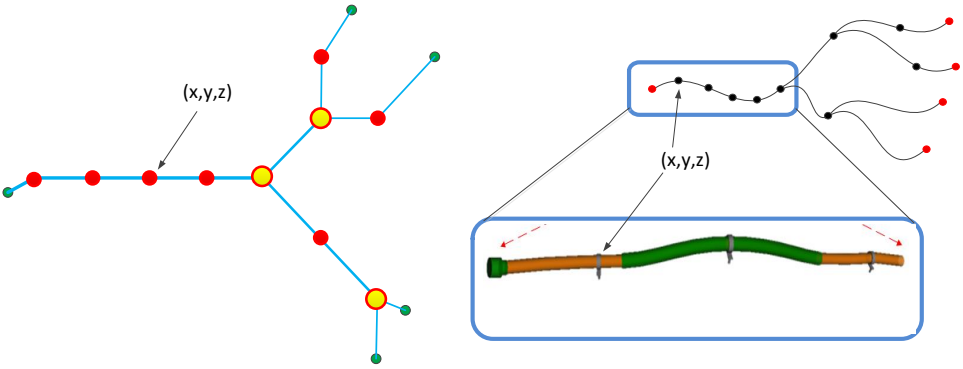


Figure 4-19: Polyline representation (left) and actual representation (right) of the same harness

4.4.1 Calculating the parameters of clamp assemblies

The parameters of the full clamp assembly definition are illustrated in Figure 5-8 in Chapter 5. The clamp direction vector $\overrightarrow{d^{clamp}}$, fixing distance d_{fix}^{clamp}, and clamp normal vector $\overrightarrow{n^{clamp}}$ are calculated here. The clamp position c^{clamp}, the geometric model of the routing environment, and the list of fixable structures are the input for this calculation. The calculation of the three parameters is implemented per clamp assembly on each branch and it includes two steps:

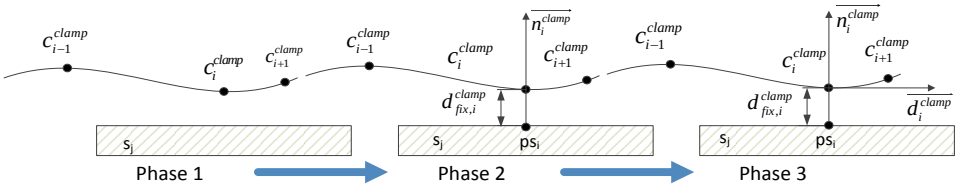


Figure 4-20: Process for getting the full definition of a clamp assembly

- **Step 1**

The first step is to find the surface where the clamp assembly is going to be placed. According to the fixable structure list and the routing environment, the structure surface set which contains all the fixable surfaces is defined as $Str = \{str_j \mid j = 1,2,...N\}$.

We specify that the clamp assembly is always perpendicular to a surface, and that the fixable surface which is closest to the point c_i^{clamp} and does not violate the minimum fixing distance, will be selected for the clamp assembly to be placed on. The built-in "**dropped-point**" function of the *surface* class in the KBE system is adopted to find the closest surface. This function can find the point ps_i which is the normal projection of the point c_i^{clamp} on the surface, if the ps_i exists. The ps_i is the closest point to c_i^{clamp} on this surface and the fixing distance $d_{fix,i}^{clamp}$ equals the distance between c_i^{clamp} and ps_i, namely $\left\| c_i^{clamp} - ps_i \right\|$. If the given 3D point cannot drop normally to the surface, the "**dropped-point**" function will return "nil" and this surface will be excluded for c_i^{clamp}. For each c_i^{clamp}, this check is implemented for all the fixable surfaces. The vector $\overrightarrow{c_i^{clamp} - ps_i}$ is the normal vector of the surface at point ps_i and it is set as the clamp normal vector $\overrightarrow{n_i^{clamp}}$.

In this method, the clamping point coming from the offset surface of a fixable surface is allowed to be fixed on other fixable surfaces. The same placement strategy is also used in the Refinement step. This strategy guarantees the free movement of the clamping point in the 3D-routing space. Therefore, the constraint of a road map which actually does not exist in the 3D-routing problem definition can be eliminated.

After the first step, the fixable structure str_j, the fixing point on the structure ps_i, the normal vector $\overrightarrow{n^{clamp}}$, and the fixing distance d_{fix}^{clamp} are all found. With these parameters, the clamp is partly defined. The definition is shown in phase 2 of Figure 4-20. In this phase, the clamp is already attached to the structure but it can still be rotated along the normal vector $\overrightarrow{n_i^{clamp}}$. Therefore, a second step is proposed to find the direction vector $\overrightarrow{d_i^{clamp}}$, to fully define the clamp assembly.

- **Step 2**

The direction vector of clamp i is calculated with its two adjacent waypoints $i-1$ and $i+1$ in order to get a smooth harness and also avoid bend-radius violation. c_{i-1}^{clamp} and c_{i+1}^{clamp} are the coordinate of the two adjacent waypoints (including the connector and breakout).

In this step, an auxiliary plane $\partial - plane$ is defined by point c_i^{clamp} and normal vector $\overrightarrow{n_i^{clamp}}$ (*point-normal* form plane). If $\overrightarrow{c_{i-1}^{clamp} c_{i+1}^{clamp}}$ locates on or is parallel to this plane (i.e. $\overrightarrow{c_{i-1}^{clamp} c_{i+1}^{clamp}}$ is

perpendicular to $\overrightarrow{n_i^{clamp}}$), then $\overrightarrow{d_i^{clamp}}$ equals $\overrightarrow{c_{i-1}^{clamp}c_{i+1}^{clamp}}$, as shown in the left-hand side of Figure 4-21. If $\overrightarrow{c_{i-1}^{clamp}c_{i+1}^{clamp}}$ diagonally intersects the $\partial - plane$, then $\overrightarrow{d_i^{clamp}}$ equals the projection vector of $\overrightarrow{c_{i-1}^{clamp}c_{i+1}^{clamp}}$ on $\partial - plane$, as shown on the right-hand side of Figure 4-21. This projection method maintains the perpendicularity between $\overrightarrow{n_i^{clamp}}$ and $\overrightarrow{d_i^{clamp}}$.

After this step, the clamp is fully defined. A full definition is illustrated in Phase 3 in Figure 4-20.

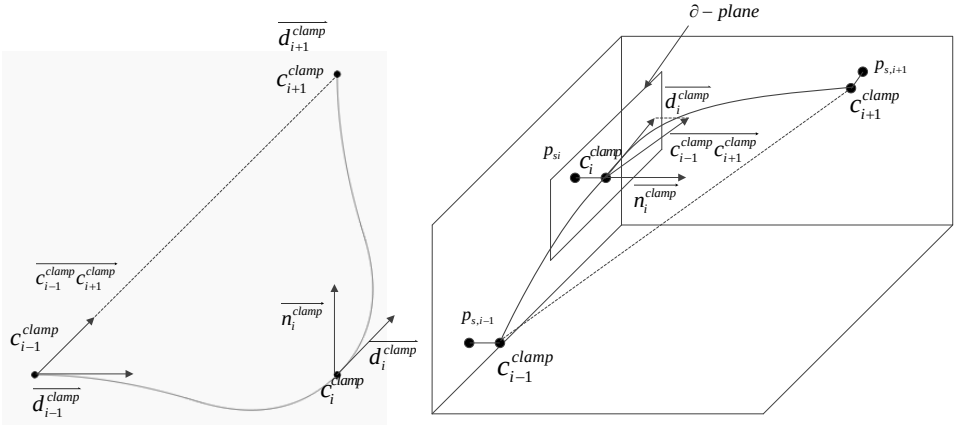


Figure 4-21: Calculation of methods of clamp tangent direction, Left: $\overrightarrow{c_{i-1}^{clamp}c_{i+1}^{clamp}}$ is perpendicular to $\overrightarrow{n_i^{clamp}}$; right: $\overrightarrow{c_{i-1}^{clamp}c_{i+1}^{clamp}}$ is not perpendicular to $\overrightarrow{n_i^{clamp}}$

c_0^{clamp} and c_N^{clamp} are the end waypoints of the fitted curve. They might be the central position of a connector or a breakout. If the point comes from a connector, the direction of the receptacle that the connector connects with will be used as the direction at this waypoint. However, if the point is at a breakout, the method to calculate the direction at this waypoint will be different and this will be presented next.

4.4.2 Calculating the parameters of breakouts

The full breakout definition, illustrated in Figure 5-10, includes as parameters the position, direction, type, and case. The position, direction, and type belong to the definition of breakout itself, and the case has been proposed to define the property of the adjacent branches at this breakout. The property specifies how the breakout direction should be used to define the start/end directions of the branches. In order to facilitate the explanation of the breakout parameter calculation, a part of Figure 5-10 is presented here, in Figure 4-22.

The breakout position is already calculated in the previous pathfinding process. The type (i.e. type Y or T) is determined by the number of wires of its adjacent branches and is calculated with the logic rule presented in Figure 5-11 and presented in Figure 4-22. The type, wire flow direction, and search direction together determine the case at the breakout. The definition and the calculation of the four different cases presented in Figure 4-22 can be found in the subsection of the breakout definition presented in Chapter 5. The different cases will support the definition of the properties of the adjacent branches at the start/end points.

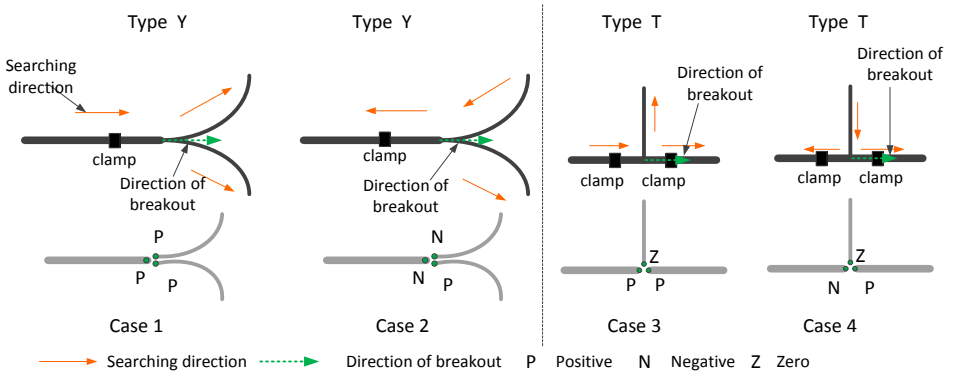


Figure 4-22: Definition of breakouts and their adjacent branches

The breakout is not a geometric component. The purpose of defining the breakout is to guarantee the connectivity (by the 3D point) and smoothness (by the direction) of the adjacent branches at the breakout. Therefore, the properties of the start/end points of adjacent branches is defined here. P (Positive) means the direction at the start/end points of branches is the same as the breakout direction; N (Negative) means the direction at the start/end points of branches is the same as the reserved breakout direction; the Z (Zero) means the direction of this branch does not relate to the breakout direction but is calculated separately (see Figure 5-13). The property information, such as $(:break-out-type\ :type\ "Y"\ :dir\ "positive"$ will be added to the branch definition (see the example in Figure G-3 in Appendix G). With this definition, a harness with smooth breakouts can already be generated by the harness parametric modelling module presented in Chapter 5.

4.5 Conclusion

The Initialization is the first step of the two-step pathfinding strategy. It uses a design optimization method to support the design automation. In this step, a road-map-based, bi-level optimization approach is adopted.

The road-map-based space representation transfers the geometric DMU that is understood by design engineers to a road map that is understandable for the computer program. The edges of the road map are represented by not only the 3D distance used to represent the length of a harness but also a coefficient related by environmental zones. This representation is able to handle the environmental information. In principle, the number of properties of map edges can be increased without limit and therefore this method provides the scalability to include more environmental information, such as reserved space where it is preferable to route harnesses. When the road map is fully generated, some design rules relating to clamps, grey areas, and geometric collision have already been tackled to a large extent.

The bi-level optimization approach decomposes the wire harness into branches to tackle the multi-destinations of the harness. It transfers the pathfinding of an entire harness into the pathfinding of individual branches (i.e. local level) and the coordination of them (i.e. global level). The classic and efficient A* algorithm can be used to support the local-level pathfinding.

When the pathfinding is finished, a post-process is carried out to generate the detailed harness definition for the parametric geometric modelling and the following Refinement step.

The harness Initialization process is implemented into a software tool with the GDL (see the

KBE system in Sub-section 3.3.2.3). The workflow of this tool, presented in Figure 4-23, reflects the previously mentioned steps.

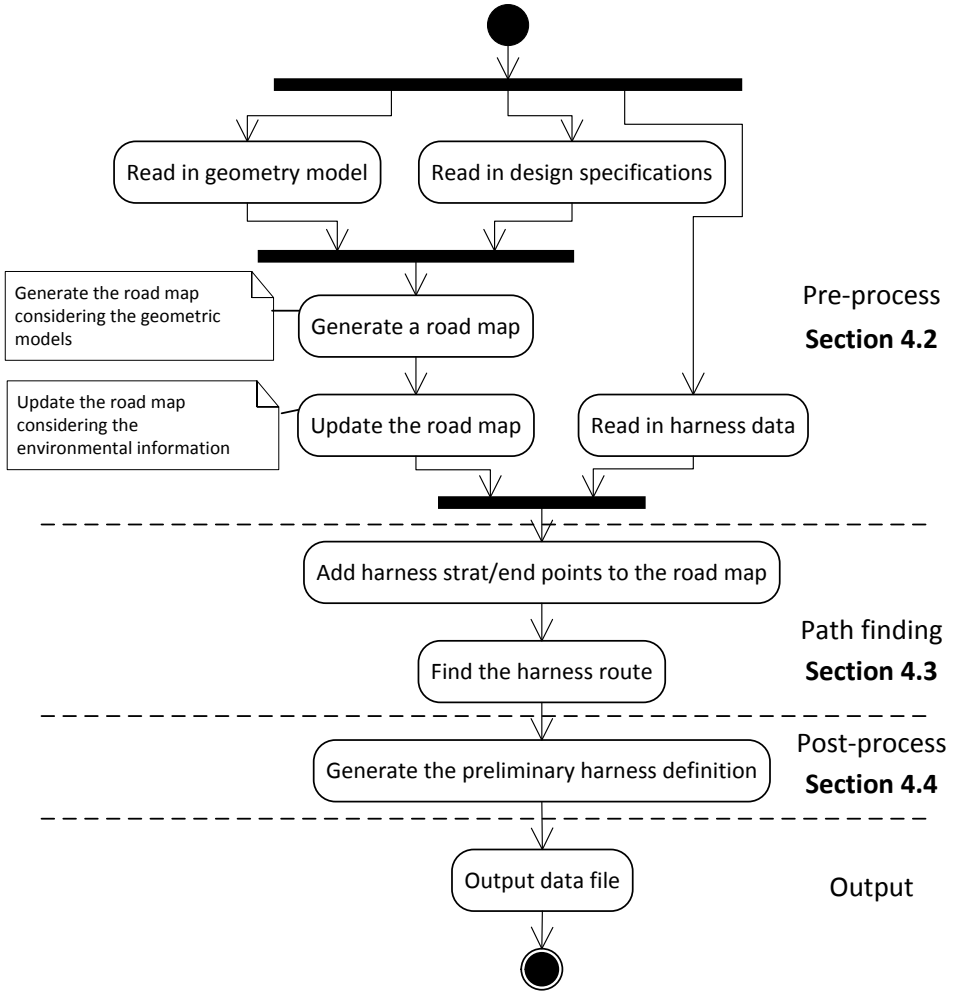


Figure 4-23: Workflow of the Initialization process

Chapter 5 Development of the HDEE geometric modelling module and analysis tools

In the harness Refinement step presented in Chapter 6, the optimizer will generate many harness configurations as candidates for the optimum result. These configurations will be modelled to actual harness geometric models for the geometry-involved analyses in each optimization loop.

Both the geometric modelling and analysis methods in the current harness 3D-routing process are not suitable for the Refinement step since these methods rely on the Graphical User Interface (GUI)-based, continuous interaction between design engineers and Computer-Aided Design (CAD)/Computer-Aided Engineering (CAE) tools. Therefore, the automatic geometric modelling and geometry-involved analysis capabilities need to be provided.

This geometric modelling capability is enabled by the harness parametric modelling module presented in this chapter. The module, referred to as the Harness Multi Model Generator (HMMG), is able to automatically generate harness geometric models and prepare the dedicated harness geometric data for the following analyses. The analytical capability is enabled by various analysis tools presented here. These tools use the data of harness geometric models to calculate the harness cost and check for the violation of design rules, such as bend-radius violation and geometric collision-free. In each optimization loop, the HMMG is able to generate geometric models according to the output of the optimizer and the analysis tools are able to evaluate the performance of these models. Then the analysis results will be sent back to the optimizer to support the decision-making for the next loop. The positions of the HMMG and the analysis tools in the harness DEE are illustrated in Figure 5-1.

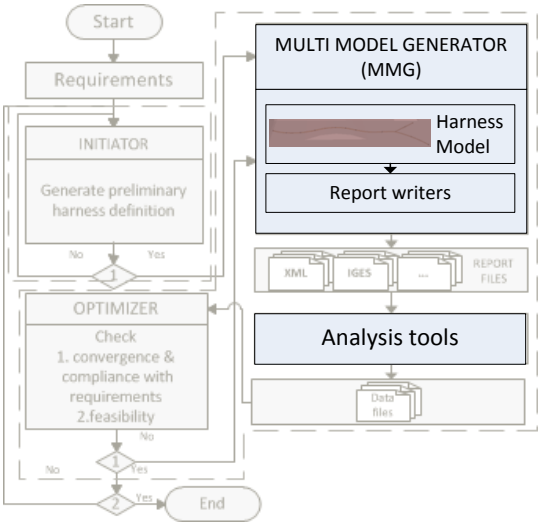


Figure 5-1: Position of the MMG and the analysis tools in the Harness DEE

In this chapter, Section 5.1 discusses the method to use one graphic topology structure, namely the full binary tree to represent various harness geometric models in order to facilitate the development of the harness parametric modelling module. Section 5.2 gives the full harness definition to support the parametric modelling. Section 5.3 presents the development of the harness parametric modelling module (MMG) and Section 5.4 briefly introduces the analysis tools and the preparation of the required input data for these tools.

5.1 Introduction of the full binary-tree structure used to represent all the harnesses

The aircraft harnesses are very different in the way their branches connect to each other and therefore in their topology structure. The four typical graphic topology structures, namely the bus, star, full binary tree, and simple branch, are shown in Figure 5-2. The solid red spots indicate harness connectors and the hollow circles represent the breakouts. Both of them are nodes in the topology structures. Branches are located between two adjacent nodes.

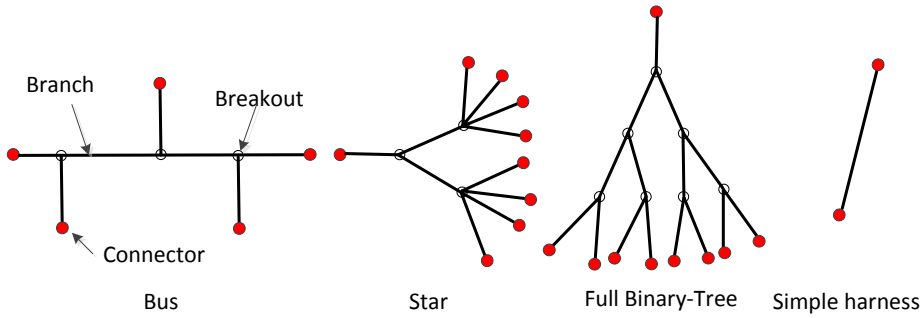


Figure 5-2: Four typical graphic topology structures of wire harnesses

The harness topology structure determines the method of harness parametric modelling. In principle, four topology structures could need four different parametric modelling modules. However, it has been found that the bus, star, and simple harness structures can all be represented by the full binary-tree structure. In other words, the parametric modelling module defined according to the full binary tree can generate a geometric model of harnesses that belong to any of the other three topology structures. In the following part of this section, the methods to transform the full binary tree to the other three topology structures are detailed.

- **From the full binary tree to the star**

The typical feature of the star topology structure is that four or more branches share the same breakout point while the full binary tree always has three branches at any breakout. By moving its breakouts, the full binary-tree structure can be transformed into the star structure. Figure 5-3 shows a transformation example.

In this figure, $C-i$, $B-i$, and $R-i$ represents the *ith* item of the connectors, branches, and breakouts respectively. Moving $R-4$ and $R-5$ to $R-2$ and moving $R-6$ and $R-7$ to $R-3$ (see the dashed blue curves), will generate a star topology structure. In this case, the breakouts $R-4$, $R-5$, and $R-2$ overlap with each other and any of them is able to represent the new breakout. Here, the higher level breakout, i.e. the one closer to the start connector (i.e. the connector with the smaller index number), is adopted to indicate the new breakout point. As the result, $R-2$ is chosen in this diagram. The same process is used to merge breakouts $R-6$, $R-7$, and $R-3$.

As shown on the right-hand side of Figure 5-3, branch $B-2$ connects with branches $B-8$, $B-9$, $B-10$, and $B-11$ directly; branch $B-3$ connects with branches $B-12$, $B-13$,

$B-14$, and $B-15$ directly. The branches $B-4$, $B-5$ and $B-6$, $B-7$ shrink to a point and their length becomes 0. They will not exist in the final harness geometric model. Therefore, the star harness model will be generated by the harness parametric model built according to the full binary-tree structure, by merging some breakouts.

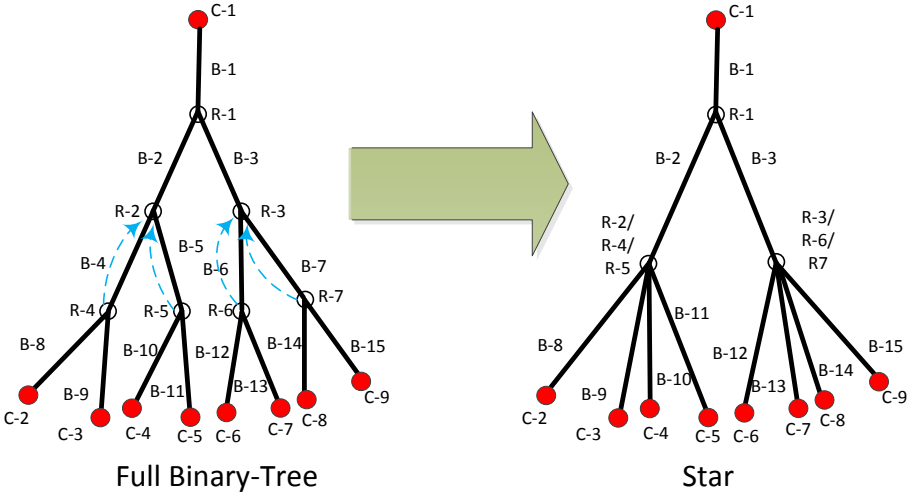


Figure 5-3: From the full binary tree to the star

- **From the full binary tree to the bus**

The transformation from the full binary tree to the bus topology structure is illustrated in Figure 5-4. Strictly speaking, this is not a transformation but just a reshaping. The entire full binary tree and the bus in Figure 5-4 have the same connectivity. The connectors, breakouts and branches are the same in both structures. This example indicates that the harness parametric model built according to the full binary-tree structure is able to generate a bus harness model.

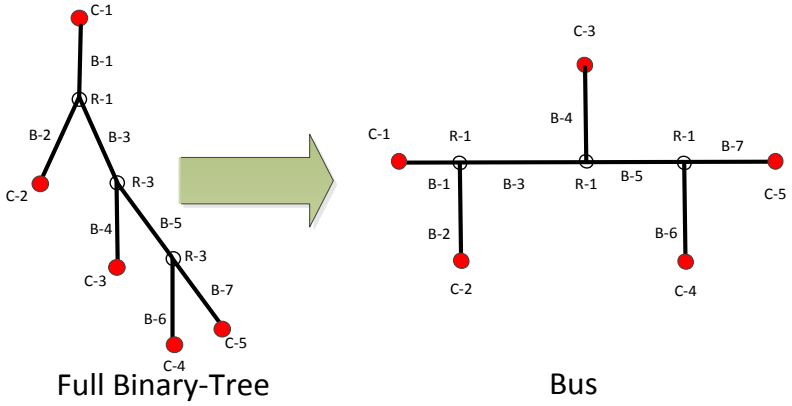


Figure 5-4: From the full binary tree to the bus

- **From the full binary tree to the simple harness**

A single branch harness is a full binary tree that only has one branch. A full binary tree that

does not have any children can be used to represent the single branch structure.

5.2 Harness top-down decomposition and bottom-up definition

The previous section demonstrates that the harness parametric modelling module based on the full binary-tree structure is capable of generating harness geometric models for all of the four structures. In order to define the parametric modelling module, the features and important components of the binary-tree harness are studied. In this section, the hierarchical decomposition of the wire harness is introduced first. Then the bottom-up harness definition is presented.

5.2.1 Harness top-down decomposition

The wire harness decomposition here is based on the EWIS hierarchical structure illustrated in Figure 2-7, and concerns the harness level, branch level, and component level. The decomposition of the entire harness until the component level is illustrated in Figure 5-5.

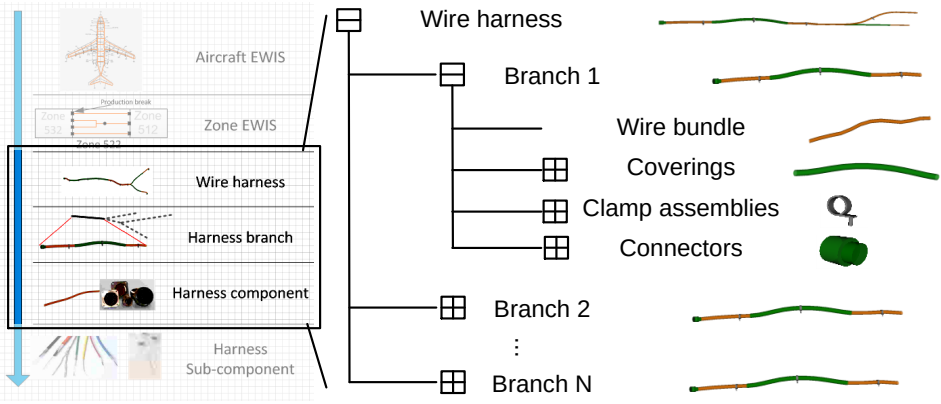


Figure 5-5: Hierarchical decomposition of a wire harness

A harness is decomposed into different branches by breakout points. Each branch is independent of the others except for the coordination of these breakouts. A branch must contain a wire bundle, which is the essential part of the branch, to carry the electrical power or signal, and may contain protective covering, clamp assemblies, and connectors. Connectors are located at the end of the bundle. The number of connectors/breakouts on a branch can be zero (a branch between two breakouts), one (a branch between an external receptacle and a breakout), or two (a simple harness). Protective coverings are located just outside the bundle and they have the same central curve as the bundle. Clamps are placed outside the bundle or the protective covering. Their central position is located on the harness central curve and their direction is the same as the tangential direction of the curve at the clamp central point. Details of these component definitions are presented in the next sub-section.

Although the components still contain subcomponents, the decomposition extent shown in Figure 5-5 is sufficient for the automatic 3D routing. For instance, a bundle can still be decomposed into wires. However, during the 3D-routing phase, design engineers model a bundle as a tube instead of as a set of wires. Hence, it is not necessary to include such details as the subcomponents.

5.2.2 Harness bottom-up definition

According to the hierarchical structure shown in Subsection 5.2.1, the definition of a wire harness is divided into three levels, namely: the component level, branch level, and harness level. The component-level definition gives the geometric details of the components; the branch-level definition shows the organization of these components to form a branch; and the harness-level definition shows the inner-connection of the branches and the inter-connection between the harness and external components, such as the avionics and other harnesses. These three level definitions are presented below.

5.2.2.1 Definition of the harness components

The definition of the four components, namely the bundle, protective covering, clamp assembly and connector, is elaborated here first. In addition, the definition of the breakout is also given here since the breakout defines the position where branches separate from each other as well as the direction of branches at the breakout point and consequently it determines the shape of the entire harness, although it is not a physical component of a wire harness.

5.2.2.1.1 Definition of a bundle

A bundle is an assembly of wires and is represented by a curved circular/elliptical solid pipe in the current manual design process. The definition of a bundle consists of the definition of its central curve and its cross-sectional shape.

The central curve is defined as a curvature continuous (G2) Non-Uniform Rational Basis Spline (NURBS, a fitted curve). NURBS is commonly used in the mainstream CAD systems. The input of the spline function is a set of waypoints $p_0^{bundle}, p_1^{bundle} \ldots p_{n+1}^{bundle}$ that are interpolated by the spline and the directions of the curve at these points $d_0^{bundle}, d_1^{bundle} \ldots d_{n+1}^{bundle}$. Both the waypoint positions and the directions attached to the waypoints are used to control the shape of the spline curve as illustrated in the upper part of Figure 5-6.

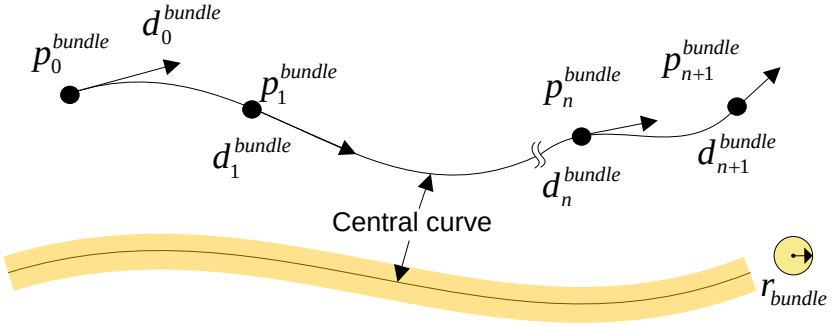


Figure 5-6: Definition of a wire bundle

In practice, the bundle cross section can be any shape, such as an ellipse and a circle, in accordance with the shape of the encompassed wires. A circle is the most common form and this is the only cross-sectional shape employed here in order to simplify the harness geometric modelling process. Therefore, only the radius of this circle needs to be defined. The lower part of Figure 5-6 shows its definition.

According to the above description, the parameters necessary to define a bundle are listed below:

p^{bundle}	List of the bundle central curve waypoints; p_i^{bundle} is the ith item on the list
d^{bundle}	List of the tangent direction attached to the waypoints; d_i^{bundle} is the ith item on the list
r^{bundle}	Radius of the circular bundle cross section

5.2.2.1.2 Definition of a covering

Coverings are used outside the bundle outer surface and a covering has the same central curve shape as the bundle which it protects. The start and end positions of the covering central curve are located on the bundle central curve. Therefore, it is convenient to define the covering using the bundle central curve. To this end, the so-called u parameter is introduced. u proportionally denotes a point on a curve, and it is located in interval [0 1]. 0 and 1 represent the start and end of the curve respectively. As shown in Figure 5-7, $u-1$ and $u-2$ are the start and end parameters of the covering central curve at the bundle central curve respectively. The covering central curve is the same as the bundle central curve part that locates between $u-1$ and $u-2$. $u-1$ of an existing covering is always smaller than $u-2$. When $u-1=u-2$, the covering shrinks to a circular ring, and therefore it does not exist outside of the bundle.

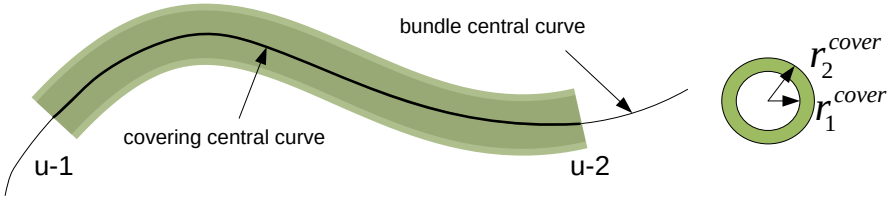


Figure 5-7: Definition of a bundle covering

The cross section of the covering is defined by the covering inner radius r_1^{cover} and outer radius r_2^{cover}. r_1^{cover} equals the wire bundle radius. r_2^{cover} equals $r_1^{cover}+th_{cover}$. th_{cover} is the thickness of the covering, which varies for different covering types.

In order to get the full definition of a bundle covering, the following parameters are needed as input:

r_1^{cover}	Covering inner radius
th_{cover}	Covering thickness
$u-1$	Covering start parameter on the bundle central curve
$u-2$	Covering end parameter on the bundle central curve
$centrecurve_{bundle}$	Bundle central curve

5.2.2.1.3 Definition of a clamp assembly

A clamp assembly, which is also generically referred to as a clamp, includes a clamp and a stand-off. The position and direction of the clamp determine the position and shape of the harness at the clamping point. The stand-off provides support to the clamp. Figure 5-8 illustrates the definition of a clamp assembly.

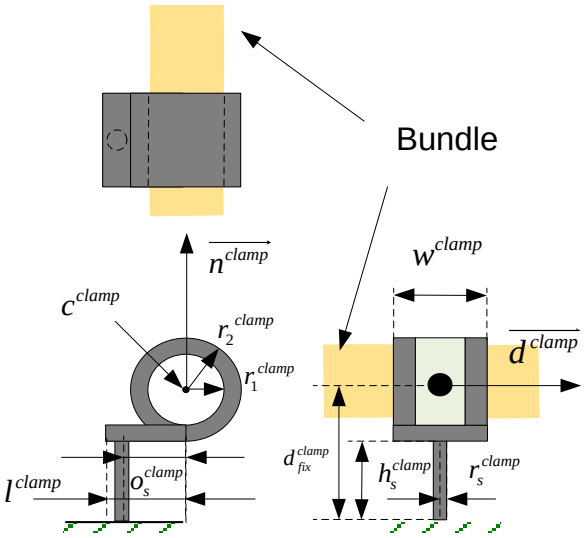


Figure 5-8: Definition of a clamp assembly

The parameters given for the clamp-assembly definition are the following.

c^{clamp} Position of the clamp centre (x, y, z)

$\overrightarrow{d^{clamp}}$ Clamp axial direction vector

d_{fix}^{clamp} Fixing distance

h_s^{clamp} Height of the stand-off

l^{clamp} Length of the clamp tail

$\overrightarrow{n^{clamp}}$ Clamp normal vector; perpendicular to the clamp-fixing surface

o_s^{clamp} Horizontal offset distance of the stand-off from the centre of the clamp

r_1^{clamp} Clamp inner radius

r_2^{clamp} Clamp outer radius

r_s^{clamp} Radius of the stand-off

w^{clamp} Clamp width

5.2.2.1.4 Definition of a connector

Various connectors can be found on wire harnesses. Since the main function of a connector in 3D routing is to provide the position and direction of the start/end positions of a harness, only the cycle connector, which is sufficient to have the position and direction information, is adopted to simplify the geometric modelling process. The radial direction (i.e. the so-called master keyway) of the circular connector is not included here since it will not influence the 3D routing result.

Although an actual circular connector is complex and contains various geometric parameters, the most relevant parameters of 3D routing are the position and direction. The geometric

shape of connectors does not have a noticeable influence on the 3D-routing result. Hence the connector is simplified into two cylinders, shown in Figure 5-9. The larger cylinder is intended to connect with the receptacle and the smaller one connects with the branch directly.

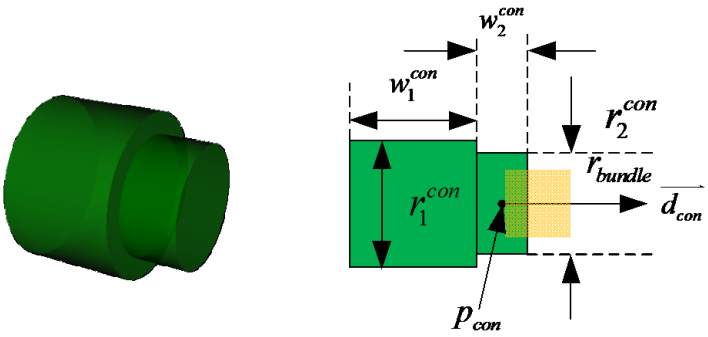


Figure 5-9: Definition of a connector

The geometric parameters to enable the full definition of this connector are given below.

w_1^{con} Larger cylinder length

w_2^{con} Smaller cylinder length

r_1^{con} Larger cylinder radius

r_2^{con} Smaller cylinder radius

P_{Con} Connector position, (x, y, z)

$\overrightarrow{d_{con}}$ Connector direction vector

r_{bundle} Bundle radius

5.2.2.1.5 Definition of breakouts

The full definition of a breakout includes the type, position, direction, and case. The breakout type, position, and direction belonging to the definition of breakout itself are given first. The breakout cases are caused by the 3D-routing strategy and determine the shape of the branches connecting to the breakouts. They are discussed subsequently.

- **Definition of breakout types, position, and direction**

There are three breakout types, namely the Y, T, and Complex. As shown in Figure 2-9, the Complex type can be seen as a combination of some Y-type breakouts. Hence it is not included or defined here.

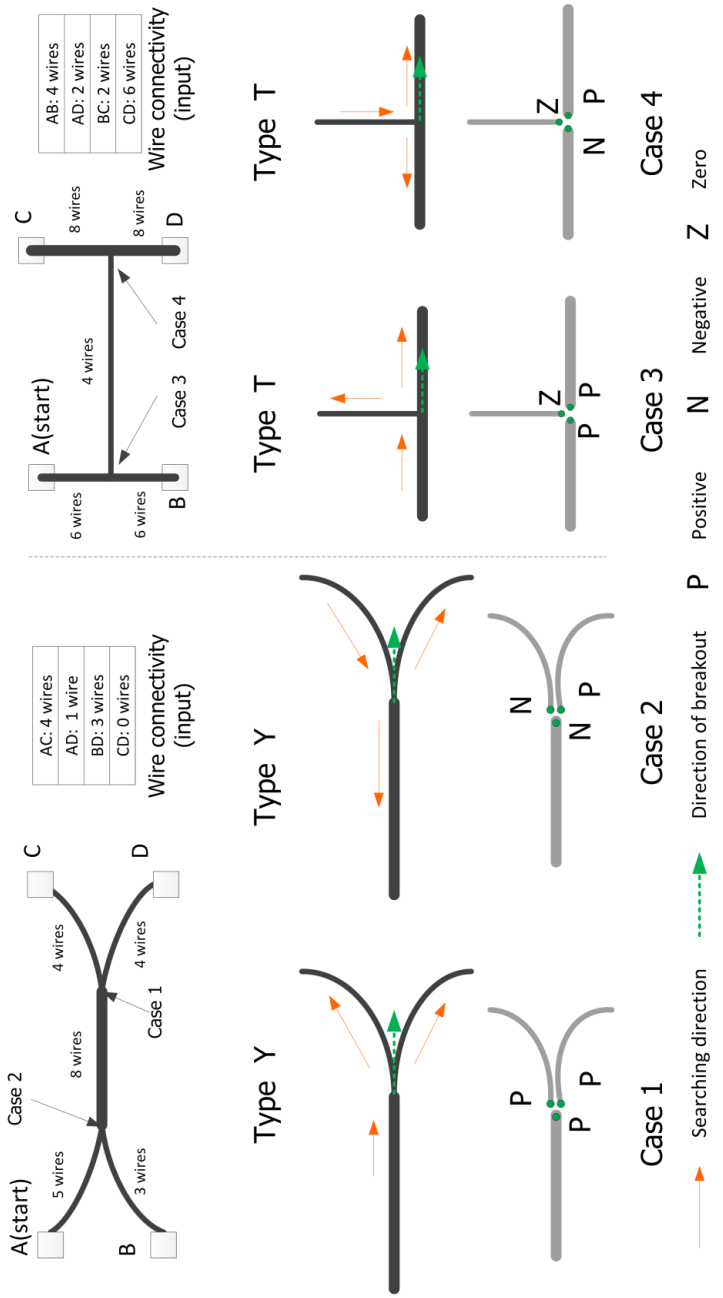


Figure 5-10: Illustration of breakout types (top) and cases (bottom)

The breakout type is determined by the wire number of the adjacent bundles at the breakout point. These types are defined by the logic rule below.

If (($wn_{B-1} = wn_{B-2} + wn_{B-3}$) or ($wn_{B-2} = wn_{B-1} + wn_{B-3}$) or ($wn_{B-3} = wn_{B-1} + wn_{B-2}$)
    BreakoutType = "Y" ;;see Case 1 and Case 2 in Figrue 5-10
Else
    If (($wn_{B-1} = wn_{B-2}$) or ($wn_{B-1} = wn_{B-3}$) or ($wn_{B-2} = wn_{B-3}$))
        BreakoutType = "T" ;;see Case 3 and Case 4 in Figrue 5-10
    Else
        BreakoutType = "unknown"

wn - the number of wires of a branch; B-1, B-2, and B-3 - the branches ID

Figure 5-11: Logic rule to determine the type of breakout

If the wire number of any branch at the breakout is the summation of another two adjacent branches, the breakout is type Y. If the wire number of any branch is not the same as the summation of the other two adjacent branches and the wire number of one branch equals the number of one of the other two branches, the breakout is type T. Otherwise the breakout type is unknown. Examples of types Y and T are presented in Figure 5-10.

The breakout position is the 3D point where the adjacent branches connect with each other. The concept of the breakout direction does not exist in current harness design practice. The directions of the start/end points of the branches at this point are given by engineers manually. This manual work may cause a manufacturing problem which is that the harness cannot be completely fitted to a 2D form-board if the directions of the 3 branches are not coplanar. In this case, extra work is needed for the manufacturing design. Therefore, the breakout direction is proposed as a reference to constrain the start and end directions of adjacent branches.

The direction of a Y-shape breakout is set as the same as the end direction or the reversed start direction of the thickest branch. This direction always points from the thick branch to the thin branches. The direction of a T-shape breakout is set as the same as the end direction of one of the two thicker branches. This direction always points from one thick branch to another thick branch. These direction definitions are illustrated in Figure 5-10.

- **Definition of breakout cases**

In the 3D-routing method proposed here, the pathfinding of a multi-origin and multi-destination harness starts from one start point (e.g. the connector A in the example shown in Figure 5-12). Then it flows from the start point via breakouts to the destinations (points B, C and D in Figure 5-12). The position and direction of the clamps are saved to a list in accordance with the search sequence. The start and/or end of the list also includes the position and direction of the breakouts, as well as the clamps, to support the generation of branch central curves. As mentioned before, the direction of type-Y breakout points from the thick harness to the thin harnesses. The direct use of the breakout direction together with the clamp direction will lead to a sharp turn in the branch central curve in some scenarios, as illustrated by the dotted curves in Figure 5-12. This sharp turn is not allowed by design rules. The type-T breakout direction is the same as one of the thick branches. The thin branch is perpendicular to the breakout direction. Therefore the thin branch cannot use the breakout direction directly to generate the branch central curve. In order to handle these problems, the breakout cases are proposed and defined here.

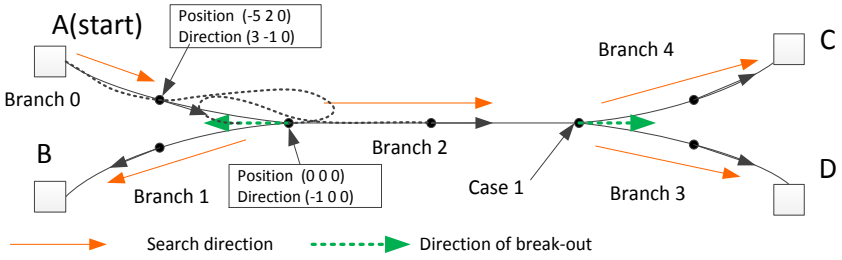


Figure 5-12: Illustration of the sharp turn problems

The breakout Case depends on the breakout type and the search direction at the breakout position. The 2 cases of the Y and T-shape breakouts will be defined. In different breakout cases, the **Property** which determines the direction of adjacent branches at the breakout is also defined here since the purpose to define the breakout is to support the generation of smooth harness branches. For the Y-shape breakout, the tangent continuity (G1) of the adjacent central curves at the breakout point will be guaranteed. The definition of the breakout cases and the **Property** of the adjacent branches are given below.

Case 1:

Case 1 is when the search direction goes from the thick branch to thin branches at a Y-shape breakout. Using the breakout direction together with the clamp direction of the three adjacent branches enables smooth branch central curves. Hence, the breakout properties of the three adjacent branches are all P (positive). P means that nothing needs to be done when the branch uses the breakout direction to generate its central curve.

Case 2:

Case 2 is when the search direction goes from one of the two thin branches to the other two branches at a Y shape breakout. In Case 2, using the breakout direction together with the clamp direction of the three adjacent branches leads to a significant sharp turn on two branch central curves (see Figure 5-12). Hence the breakout property of the incoming thin branch and the thick branch are all N (Negative) and the other thin branch is P. N means the breakout direction needs to be reversed when the branch uses this direction to generate its central curve. The direction reversion guarantees the smoothness of the branches at the breakout point.

Case 3:

Case 3 is when the search direction goes from one of the two thick branches to another thick branch and the thin branch at T-shape breakout. It is specified that the breakout direction is the same as the end point direction of the incoming branch. In Case 3, using the breakout direction together with the clamp direction of the two thick branches enables smooth central curves of the two branches. Hence the breakout properties of these two adjacent branches are all P (positive). The thin branch is perpendicular to the thick branches. The breakout property of this thin branch is Z (Zero). Z means neither P nor N but perpendicular to the breakout direction. In order to determine the perpendicular direction, an auxiliary plane is defined. The plane, shown in Figure 5-13, is defined by the breakout point and breakout direction (*point-normal* form plane). The thin branch direction at the breakout point is defined as the projection of the vector going from breakout to the first clamp on the thin branch on the auxiliary plane. This definition is illustrated on the left-hand side of Figure 5-13.

Case 4:

Case 4 is when the search direction goes from the thin branch to the two thick branches at the T-shape breakout. It is specified that the breakout direction is the same as the start direction of one of the two thick branches. In Case 4, using the breakout direction together with the clamp direction of the two thick branches enables only one smooth branch central curve. The breakout property of this smooth thick branch at the breakout is P and consequently the breakout property of the other thick branch is N. Similar to Case 3, the breakout property of the thin branch at the breakout point is Zero and the branch direction at the breakout is defined as the projection of the vector pointing from the last clamp of the thin branch to the breakout point on the auxiliary plane. This definition is also illustrated on the right-hand side of Figure 5-13.

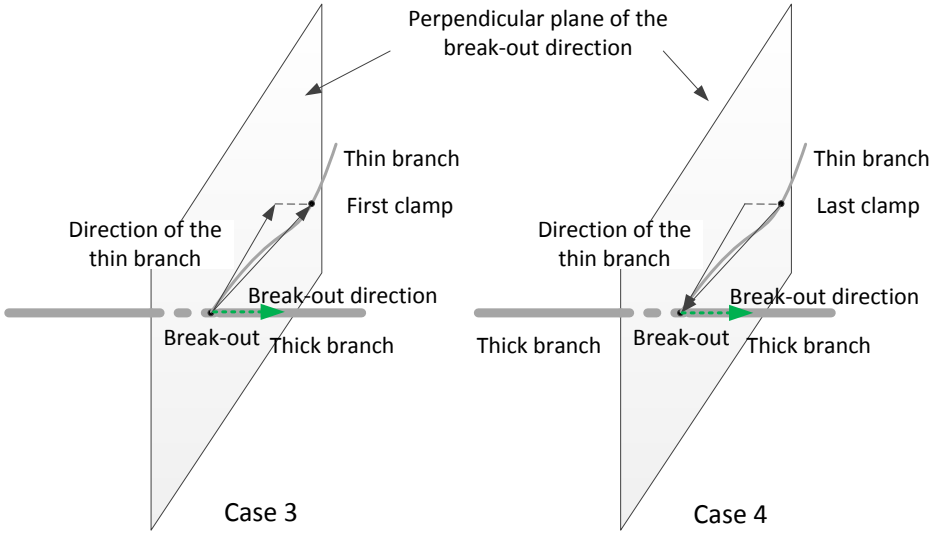


Figure 5-13: The thin branch direction at the breakout point of cases 3 and 4

In the harness design practice, the concept of breakout cases does not exist. In this research, due to the 3D-routing strategy, different breakout scenarios need different geometric modelling methods. Therefore, the breakout case concept is proposed and different cases are explicitly defined. The case definitions will be used as the standard for all the breakout-related calculations.

5.2.2.2 Definition of harness branches

A harness generally has multiple branches and a typical harness branch contains the start and end components (i.e. breakout or connector), a bundle, coverings, and clamps, as illustrated in Figure 5-14.

Each branch of a harness has an identification number (i.e. ID) and the ID will be used to define the relationship among branches (e.g. branch 001 connects with branch 002 and branch 003) in the harness definition.

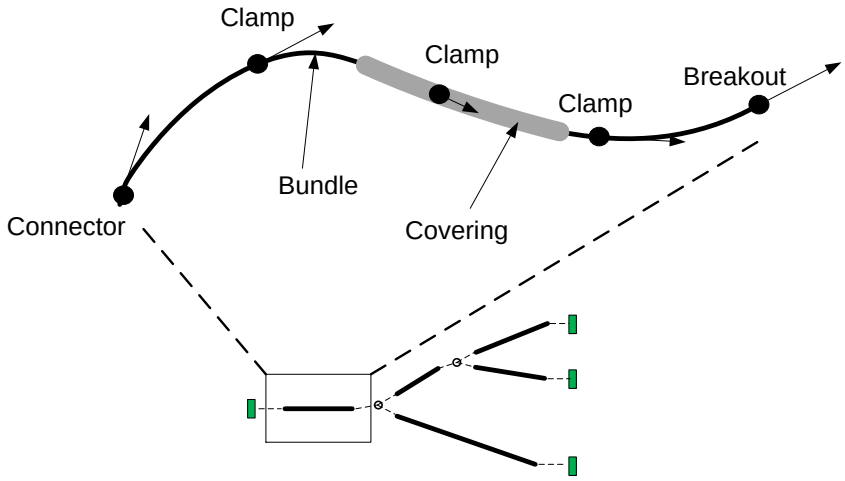


Figure 5-14: Decomposition of a branch definition

The start and end components are the interface of each branch with other branches or external equipment. According to the branch position in the harness, four different component combinations exist, as illustrated in Figure 5-15. Therefore, the start and end components are defined not only by their position and direction, but also by the component type (i.e. breakout or connector). All the parameters needed to define these components are presented in Table 5-1. **:start-poi** and **:end-poi** define the position and direction of the start and end components respectively without knowing the component types which are defined in **:start-type** and **:end-type**. The parameter **:start/end-breakout-type** defines the breakout type and property of the start/end components if the component is a breakout. Otherwise the parameter is set to *nil* which means this parameter is not applicable to the current branch.

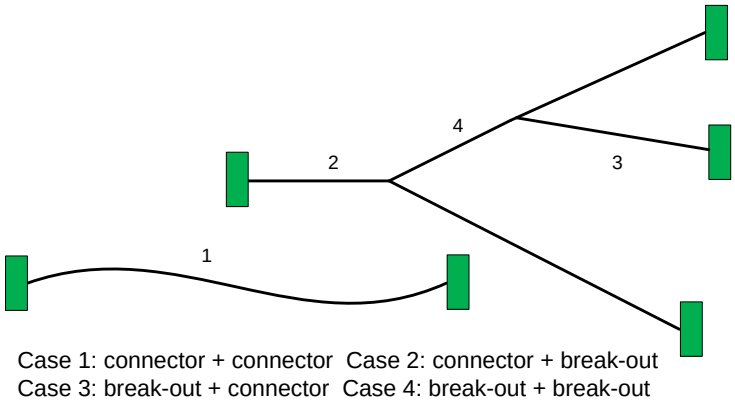


Figure 5-15: Possible location of harness branches

Between the start and end components, a bundle is located. As described before, a bundle is defined by the bundle central curve and cross-sectional shape. The central curve is defined by a set of waypoints and the tangent directions attached to these waypoints. The **:start-poi**, **:clamps**, and **:end-poi** in the branch definition provide the parameters to define the waypoints and the directions. The number of wires and wire gauge of the bundle

(**:proofwires**) in the branch definition will be used to calculate the diameter of the bundle cross section.

Table 5-1: Parameters to define a harness branch

• Branch ID definition :branch-id "branch-000" ;; unique ID number of a branch
• Parameters for start and end component definition :start-poi (:point #(28660.0 -5.5 5526.0) :vector #(0.5 8.8 0.8)) ;;3D point and direction of the start component :end-poi (:point #(28780.0 0.0 5800.0) :vector #(-1.0 0.0 -1.0)) ;;3D point and direction of the end component :start-type :breakout ;; start component type (connector or breakout) :end-type :connector ;; end component type (connector or breakout) :start-breakout-type (:type "Y" :dir "positive") ;; breakout type of start component provided the component is a breakout :end-breakout-type nil ;; breakout type of end component provided the component is a breakout
• Parameters for connector definition :connectors-positions (#(28780.0 0.0 5800.0)) ;; 3D points of connector(s) provided the start and/or end component are connectors :connectors-directions (#(-1.0 0.0 -1.0)) ;;direction of connector(s) provided the start and/or end component are connectors
• Parameter for clamp definition :clamps (:point (#(27476.6 -44.8 3846.1) …) :vector (#(0.5 8.0 0.8) …) :normal (#(-5.0 1.0 -7.2)…) :stand-height (25.1 …)) ;; clamp data, including 3D point, direction vector and normal vector
• Parameter for covering definition :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) ;; the property list of start and end position of covering(s) and its thickness
• Parameter to calculate bundle diameter :proofwires (:number 4 :AWG 10) ;; the property of wires in the bundle

A branch can have one or more clamps. The **:clamps** in the branch definition list the necessary parameters (i.e. the centre position, direction vector, normal vector, and the height) to generate a full definition of all the clamps on the branch.

Coverings are placed around the bundle. Here, its inner radius is considered the same as the bundle radius. The parameter **:covering** defines the start and end points of the coverings on the bundle in proportional format (i.e. u-1 and u-2) and also their thickness.

The previous branch definition is uniform for all branches. This uniform definition can be instantiated to different actual branches. Although they are defined and instantiated

independently, the geometric connectivity among the adjacent branch models is maintained because the position and direction of the breakout shared by these branches are included in the start/end components of these branches. The assignment of the breakout value to the start/end components of the related branches will be discussed in the following harness definition.

5.2.2.3 Definition of a whole harness

A harness is an aggregation of branches. These branches interconnect with each other at the breakouts and connect with external components, such as the avionic system or other harnesses, via the connectors. A typical harness diagram which indicates the branch interconnections is presented on the left-hand side of Figure 5-16.

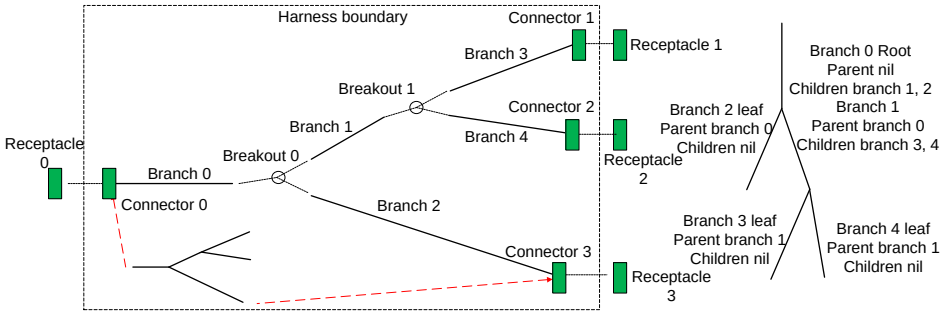


Figure 5-16: Decomposition of a typical harness (left) and its parent-child relationship (right)

The harness definition, shown in Figure 5-17, mainly focuses on these interconnections among branches and between the harness and external components while the previous branch definition focuses on the geometric features of one branch.

The interconnection among branches is defined as parent-child relation format[62], which is illustrated on the right-hand side of Figure 5-16. The low-level branch is a child of the high-level branch. High level means the branch is closer to the root. A node that does not have a parent is called a root and a node that does have a child is called a leaf. The adoption of the parent-child definition is caused by the 3D-routing strategy, namely searching the harness path from a start point (root) to destinations (leaves). The parameters :**parent** and :**child** of each branch define this relationship. In the harness definition shown in Figure 5-16, branch 0 has no parent and has two child branches 1 and 2, and is the root. The :**parent** and :**child** of this branch is **nil** and (1 2) respectively. **nil** means it does not exist and (1 2) are the index of the branches. Branches 2, 3 and 4 do not have any children. They are leaves.

The interconnection between a branch and external components is defined by :**start-poi**, and :**end-poi**, as shown in Figure 5-17. These two parameters indicate the type (i.e. connector or breakout) and index (i.e. nth item in the connector or breakout list) of the start and end components of the branch. If the component is a breakout, the breakout type (i.e. Y or T) and property (i.e. positive, negative, or zero) are also included in these parameters.

The connectors and breakouts lists are also included in the harness definition. The connector list defines the ID, position, and direction of all the connectors. Similarly, the breakout list defines the ID, position, and direction of all the breakouts. The sequence to index the ID of the connector, breakout, and branch starts from the root of the harness. The components that are closer to the root have a smaller index number.

A connector is always used in pairs with a receptacle of avionics or production breaks. The position and direction of the connector are the same as the position and direction of the

receptacle which is given as inputs. The same position and direction guarantee that the harness geometric models and the receptacle have a tight connection. The breakout list only defines the position and direction of the breakout here. The type and case of the breakout will be calculated during the geometric modelling process.

```
((:branches
    ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index
0) :end-poi (:breakout-type (:type "Y" :dir "positive") :type :breakout :index 0) :covering
(:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point (#(500.0 -150.0 -
1299.9) …) :vector (#(0.9 1.5 -0.3) …):normal (#(0.0 -1.0 -4.5) …) :stand-height
(50.0 …)) :proofwires (:number 2 :AWG 8))
    (:branch-id "branch-001" :parent (:type :branches :index 0) :child nil :start-poi (:breakout-
type (:type "Y" :dir "positive") :type :breakout :index 0) :end-poi (:type :connectors :index
1) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point
(#(3700.0 -150.0 -900.0) …) :vector (#(0.7 4.8 0.7) …) :normal (#(9.0 -1.0 -2.2) …) :stand-height
(50.0 …)) :proofwires (:number 1 :AWG 8))
    (:branch-id "branch-002" :parent (:type :branches :index 0) :child nil :start-poi (:breakout-
type (:type "Y" :dir "positive") :type :breakout :index 0) :end-poi (:type :connectors :index
2) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point
(#(3700.0 -150.0 -1700.0)) :vector (#(0.7 9.8 -0.6)) :normal (#(9.5 -1.0 -4.5)) :stand-height
(50.000000000000014)) :proofwires (:number 1 :AWG 8)))
:connectors
    ((:connector-id "con-000" :data (:point #(160.0 -130.0 -1000.0) :vector #(0.0 0.0 1.0)))
    (:connector-id "con-001" :data (:point #(4530.0 -130.0 -500.0) :vector #(0.0 -1.0 0.0)))
    (:connector-id "con-002" :data (:point #(4130.0 -130.0 -2000.0) :vector #(0.0 -1.0 0.0))))
:breakout
    ((:breakout-id "breakout-000" :data (:point #(3300.0 -150.0 -1299.9) :vector #(1.0 0.0
0.0))))))
```

Figure 5-17: A typical harness definition

5.3 Development of the harness parametric modelling module

A wire harness geometric model is an aggregation of geometric models of harness branches which consist of the geometric models of various harness components. Therefore, the hierarchical method as shown in Figure 5-18 is adopted to support the development of the harness parametric modelling module.

The component parametric modelling modules are considered as the bricks of the development. Here, the concept of High Level Primitives (HLPs)[45] is adopted. HLPs, which are software classes in the MMG, are parametric modelling modules of the harness components. One HLP represents a type of geometric components that have a sufficient level of commonality, namely a type of connectors, bundles, coverings, and clamps. All four of the HLPs enable the full definition of the branch parametric modelling module and the harness parametric modelling module, as shown in Figure 5-18.

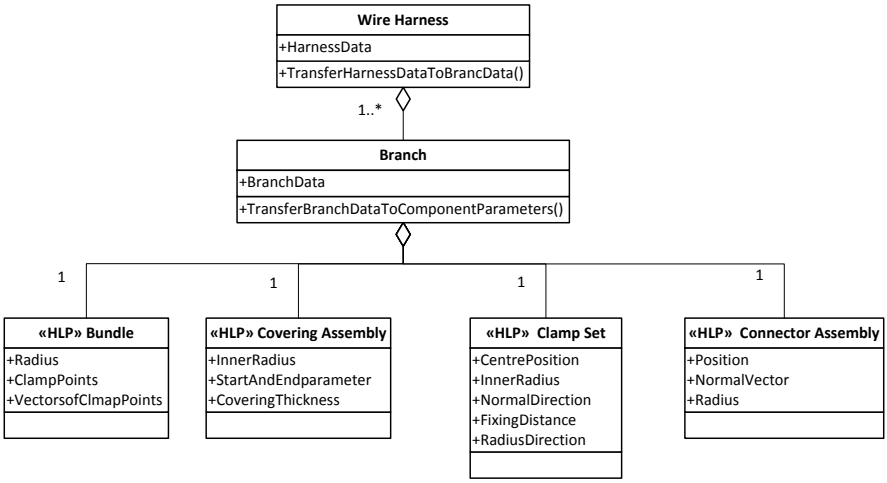


Figure 5-18: UML class diagram of the wire harness product model

In this section, the development of HLPs of harness components is presented first. According to these HLPs, the development of the branch parametric modelling module is introduced. Finally, the development of the harness parametric modelling module is presented.

5.3.1 Development of harness component HLPs

The components that are necessary to build a harness model are the bundle, covering, clamp assembly and connector. The breakout is an important component that determines the harness shape. However, it is an abstract concept rather than a concrete geometric component. Therefore, it is not considered during the development of the component HLPs.

5.3.1.1 The bundle HLP

As shown in Figure 5-19, the bundle HLP includes the classes *Fitted Curve* and *Swept Circle* which are used to generate the central curve and the bundle respectively.

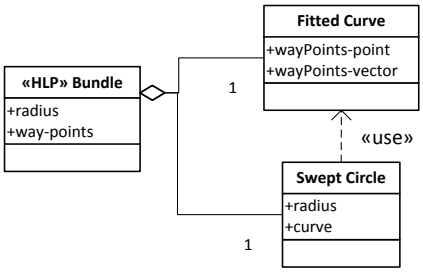


Figure 5-19: UML class diagram of the Bundle HLP

The central curve is defined by a set of waypoints and the direction attached to each of the points. These points represent the clamps, breakouts and/or connectors and are the input of the *Bundle* HLP. Based on these waypoints, the central curve can be generated using the class *Fitted Curve*.

In order to define the bundle, first the radius of the bundle needs to be known. The radius is one of the inputs of the *Bundle* HLP and is calculated at the branch parametric modelling

module (see the next sub-section) since the radius value is also used by other components. After defining the cross-section circle radius and the bundle central curve, the bundle defined as a *Swept Circle* can be generated by sweeping the cross-section circle along the central curve.

The geometric representation of both the central curve and the swept object can be found in Figure 5-6.

5.3.1.2 The Covering Assembly HLP

A branch can contain multiple coverings. These coverings are described as the covering assembly. A covering is defined by its central curve and the cross-sectional shape. Since the covering is always used outside and has the same central curve as a bundle, the covering central curve is defined by the bundle central curve and the start parameter $u-1$ and end parameter $u-2$. In the development of *Covering Assembly* HLP shown in Figure 5-20, the class *Trimmed Curve* is defined first to get the curve located between $u-1$ and $u-2$. The actual trimmed curve overlaps with the bundle central curve and its start and end points are $u-1$ and $u-2$ respectively. Since the geometric model instantiated from the *Trimmed Curve* cannot be the central curve of a swept object in this KBE system, the class *Fitted Curve* is used to define the central curve. This fitted curve is built with the waypoints obtained from the trimmed curve and therefore it has the same geometric shape and position as the actual trimmed curve without considering the interpolation error. After the definition of the central curve, the circular ring which is illustrated in Figure 5-7 can be swept along the curve to get the geometric model of a bundle covering. Since the KBE system only supports the surface sweep but not the solid sweep, in this research the outer surface rather than the actual solid pipe is used to represent the covering geometric model.

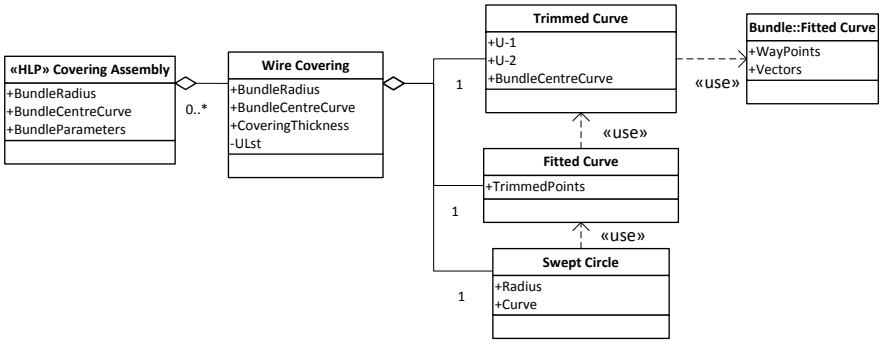


Figure 5-20: UML class diagram of the Covering Assembly HLP

5.3.1.3 Clamp Set HLP

A bundle needs to be supported by a set of clamp assemblies, which is described as the clamp set. The class used to define the clamp set is called the *Clamp Set* HLP, shown on the left-hand side of Figure 5-21. An exploded view of the clamp assembly is given on the right-hand side. The upper and lower parts belong to the clamp and they are defined by *Cone Solid* and *Box Solid* respectively. The stand-off is also defined by *Cone Solid*. These components constitute the clamp assembly.

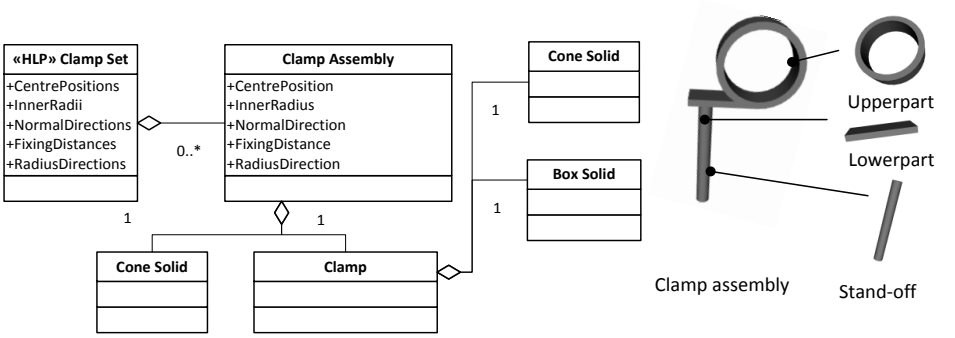


Figure 5-21: UML diagram of the Clamp Set HLP (left) and exploded view of a clamp assembly (right)

A clamp is always used outside the bundle or bundle covering. The clamp inner radius depends on the bundle or the covering outer radius and is an important parameter to define the clamp. In addition to the inner radius, the clamp geometric model also contains other parameters that significantly influence the position and geometric shape of the clamp. These parameters are presented in Table 5-2. They are calculated at the branch parametric modelling module (see the next sub-section) and given as inputs of the *Clamp Set* HLP.

Table 5-2: Input parameters for clamp assembly

Clamp centre position	c^{clamp}	Fixing distance	d_{fix}^{clamp}
Clamp inner radius	r_1^{clamp}	Clamp normal vector	$\overrightarrow{n^{clamp}}$
Clamp axial vector	$\overrightarrow{d^{clamp}}$		

The rest of the parameters that are shown in Figure 5-8 are also necessary to fully define a clamp assembly but are not necessary for the 3D routing since they do not influence the harness geometric shape. Hence, these parameters are calculated based on some reasonable assumptions. The calculation formulas are presented in Table 5-3.

Table 5-3: Secondary parameters to define the clamp assembly

Clamp outer radius	$r_2^{clamp} = 1.2 * r_1^{clamp}$	Clamp width	$w^{clamp} = 0.8 * r_1^{clamp}$
Height of stand-off	$h_s^{clamp} = d_{fix} - r_2^{clamp}$	Offset distance	$o_s = 1.2 * r_2^{clamp}$
Clamp tail length	$l^{clamp} = 1.6 * r_2^{clamp}$	Stand-off radius	$r_s^{clamp} = w^{calmp} / 4$

5.3.1.4 Connector Assembly HLP

Connectors are located at the bundle ends. A branch can contain up to 2 connectors. These connectors are described as connector assembly. The class used to define the assembly is called *Connector Assembly* HLP, shown in Figure 5-22. The *Connector Assembly* HLP is an aggregation of up to two instances of the *Connector* and the *Connector* is defined by two cylinders (*Cone Solid*). The geometric definition of the connector can be found in Figure 5-9.

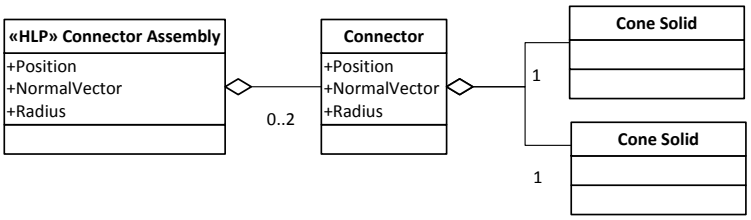


Figure 5-22: UML class diagram of the Connector Assembly HLP

Although all parameters in the geometric definition are needed to generate the entire geometric model, only some of them influence the harness geometric shape and the 3D-routing result. These important parameters are calculated at the branch parametric modelling module (see the next sub-section) and given as input here. The remaining parameters are calculated based on some reasonable assumptions.

The input parameters necessary for connector definition are given in Table 5-4.

Table 5-4: Input parameters for a connector

Bundle radius	r_{bundle}	Connector direction vector	$\overrightarrow{d_{con}}$
Connector position	P_{Con}		

The remaining connector parameters are given in Table 5-5.

Table 5-5: Secondary parameters to define a connector

Larger cylinder radius	$r_1^{con} = 1.5 * r_{branch}$	Larger cylinder width	$w_1^{con} = 2 * r_{branch}$
Smaller cylinder radius	$r_2^{con} = 1.1 * r_{branch}$	Smaller cylinder width	$w_2^{con} = 1 * r_{branch}$

5.3.2 Development of the branch parametric modelling module

The branch parametric modelling module is the aggregation of harness component HLPs. It also contains component check functions and parameter calculation functions. The component check functions detect whether certain components (e.g. the connector, clamp, and covering) exist on a certain branch and the parameter calculation functions calculate the required parameters for the HLP of the existing components. Since the parameters of some components (e.g. the inner radius of the covering) depend on the parameters of other components (e.g. the outer radius of the bundle), the component existence check and parameter calculation need to be carried out according to a certain sequence, as shown in the UML activity diagram in Figure 5-23.

In this sub-section, the generation of bundle parameters is detailed first since the bundle is the most important component and always exists in a branch and then the existence check and parameter calculation of other components are introduced.

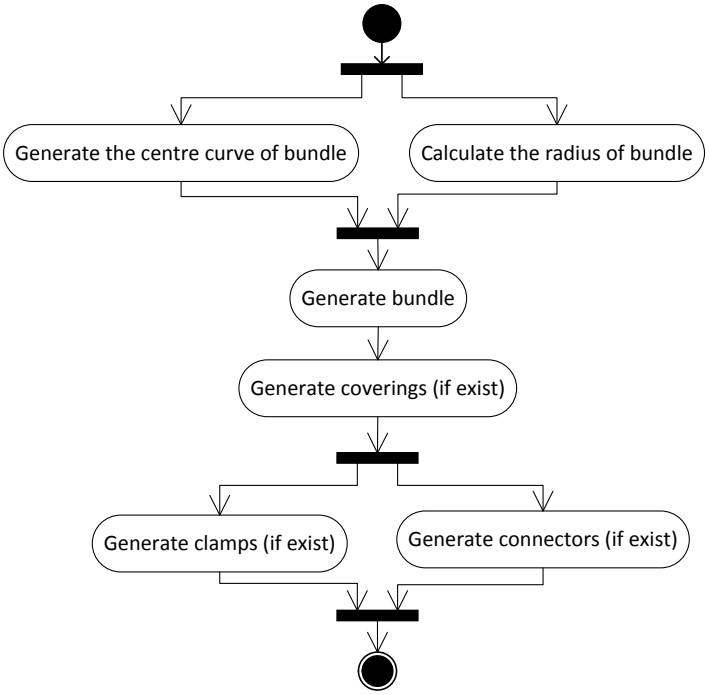


Figure 5-23: UML activity diagram of a harness branch generation

5.3.2.1 Calculation of the parameters for the bundle HLP

All the input parameters of the *Bundle* HLP need to be calculated here. These parameters include the waypoints, direction attached to these points, and the bundle radius.

The waypoints and attached directions are used to generate the bundle central curve (fitted curve). They come from two different sources: 1) the clamp and 2) the bundle ends (i.e. connector/breakout). The clamp and bundle ends are already defined in the branch input file (see Table 5-1). The position and direction of the clamps as well as the position of the two ends will be passed down directly to the bundle HLP. However, the direction of the two ends defined in the input file cannot be used directly and two logical calculations, detailed below, are needed to handle the connector and breakout respectively.

The end types (i.e. connector or breakout) can be recognized from **:start-type** and **:end-type**. In a practical definition, the connector direction which is the same as the receptacle direction always points from aircraft systems to harnesses. As shown in Figure 5-24, the original direction of the end-point connector will lead to a sharp turn (i.e. the dashed curve) when it is used directly as the direction attached to the curve end point. In order to get a smooth and correct curve, this direction needs to be reversed.

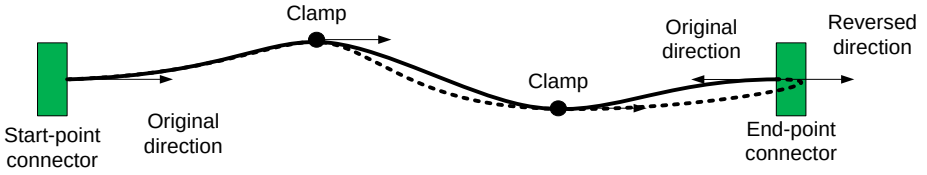


Figure 5-24: Illustration of the sharp turned (dashed curve) and the correct (solid curve) bundle centre curve

In order to handle this operation, a logical calculation is developed and presented below.

```
If (connector is a start point)
  DirOfPoint = DirOfRec;   ;; the direction of point (DirOfPoint) equals the direction
                           ;; of the receptacle(DirOfPoint) which the connector connects with
Else  ;;connector is an end point
  DirOfPoint = reverse(DirOfRec);  ;; the direction of point equals the reversed direction
                                   ;; of the receptacle which the connector connects with
```

When the start/end points are a breakout then another logical calculation presented below is applied in order to get a smooth bundle central curve. This calculation is based on the breakout definition, illustrated in Figure 5-10.

```
If (BreakoutType = "Y")
   If (ProOfPoint = "Pos")            ;;property of the point is Positive
      DirOfPoint = DirOfBre;
   Else                               ;;property of the point is Negative
      DirOfPoint = reverse(DirOfBre);
Else if (BreakoutType = "T")
   If (ProOfPoint = "Pos")
      DirOfPoint = DirOfBre;
   Else if (ProOfPoint = "Neg")
      DirOfPoint = reverse(DirOfBre);
   Else if (ProOfPoint = "Zero")
      DirOfPoint = DirOfClamp;   ;;direction of point equals the direction of its adjacent clamp point
PorOfPoint- property of point;  DirOfPoint- direction of point;  DirOfBre-direction of break-out
DirOfClamp- direction of clamp
```

The radius/diameter of a bundle depends on the gauge and number of wires that the bundle includes. In principle it should be given as an input. However, the available input of the 3D routing only includes the number and gauge of wires. The method introduced in Sub-section 2.3.2.1 is adopted to calculate the bundle radius according to the wire bundle and gauge.

The actual bundle radius calculation is more complex than this method. However, no matter how complex the method is, the output of this calculation is always the radius of the bundle. The advanced calculation method for complex situations (e.g. different wires are included in one bundle) can be developed independently afterwards and this will not influence the 3D-routing method presented here.

After these calculations, the input parameters of a bundle are fully available. They will be passed down to the *Bundle* HLP for the geometric instantiation of the bundle.

5.3.2.2 Parameter definition for other HLPs

After the generation of a bundle, the existence of covering on the branch will be checked. A branch may contain more than one covering. If one or more coverings exist, the covering start and end parameters (i.e. $u-1$ and $u-2$) and the thickness th_{cover} will be extracted from the input data. These parameters together with the bundle radius r_{bundle} will be passed down to the *Covering Assembly* HLP.

Then, the connector will be generated. The branch ends will be analysed first. If the connector(s) exists, the position P_{Con} and direction $\overrightarrow{d_{con}}$ of the connector(s) will be extracted from the data file. These parameters together with bundle radius r_{bundle} will be sent to the *Connector Assembly* HLP as the inputs to support the generation of the geometric model of the connector.

Finally, the parameters for the *Clamp Set* HLP will be generated. The bundle waypoints are checked first to determine how many clamps exist. If clamps exist, five parameters in Table 5-2 need to be prepared for the *Clamp Set* HLP. Four of the parameters can be extracted from the input data file directly, but not their inner radius. The inner radius equals the outer radius of the harness branch. It might be the outer radius of either the bundle or covering. Therefore, the logical equation listed below is used to calculate the clamp inner radius r_1^{clamp}.

$$r_1^{clamp} = \begin{cases} r_{bundle} & \text{- if clamp holds the bare harness bundle} \\ r_2^{cover} & \text{- if clamp holds one of coverings} \end{cases} \tag{5.1}$$

With the previously defined parameters, the branch geometric model consisting of the geometric model of the components can be generated automatically and an example of the model is shown in Figure 5-25.

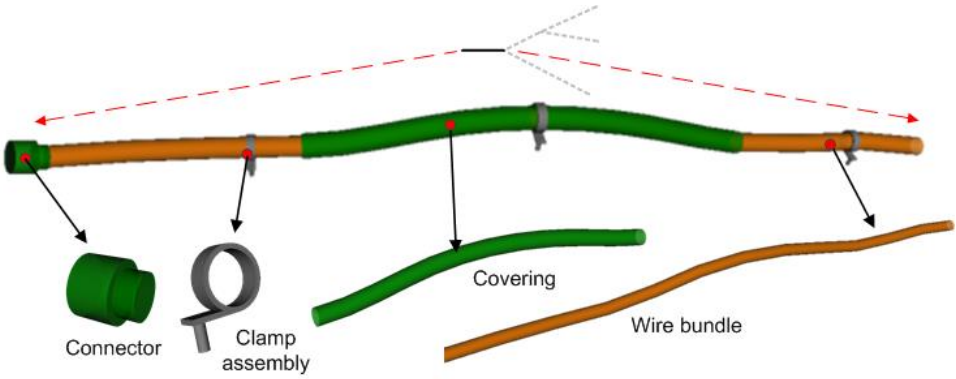


Figure 5-25: Decomposition of the DMU of a harness branch

5.3.3 Development of the harness parametric modelling module

The harness parametric modelling module includes the aggregation of parametric modelling modules of branches and the mathematic calculation that transfers the input of the harness module into the required input parameters of the branch module.

As shown in Figure 5-26, the input parameters for the branch definition can be divided into three: 1) the parameters with the same name and value as the harness input parameters, 2) the

parameters with only the same name as the harness input parameters, and 3) the parameters only defined in the branch parametric modelling module. The three types of parameters are represented by the round, square, and rhombus shapes respectively in the transfer example shown in Figure 5-26.

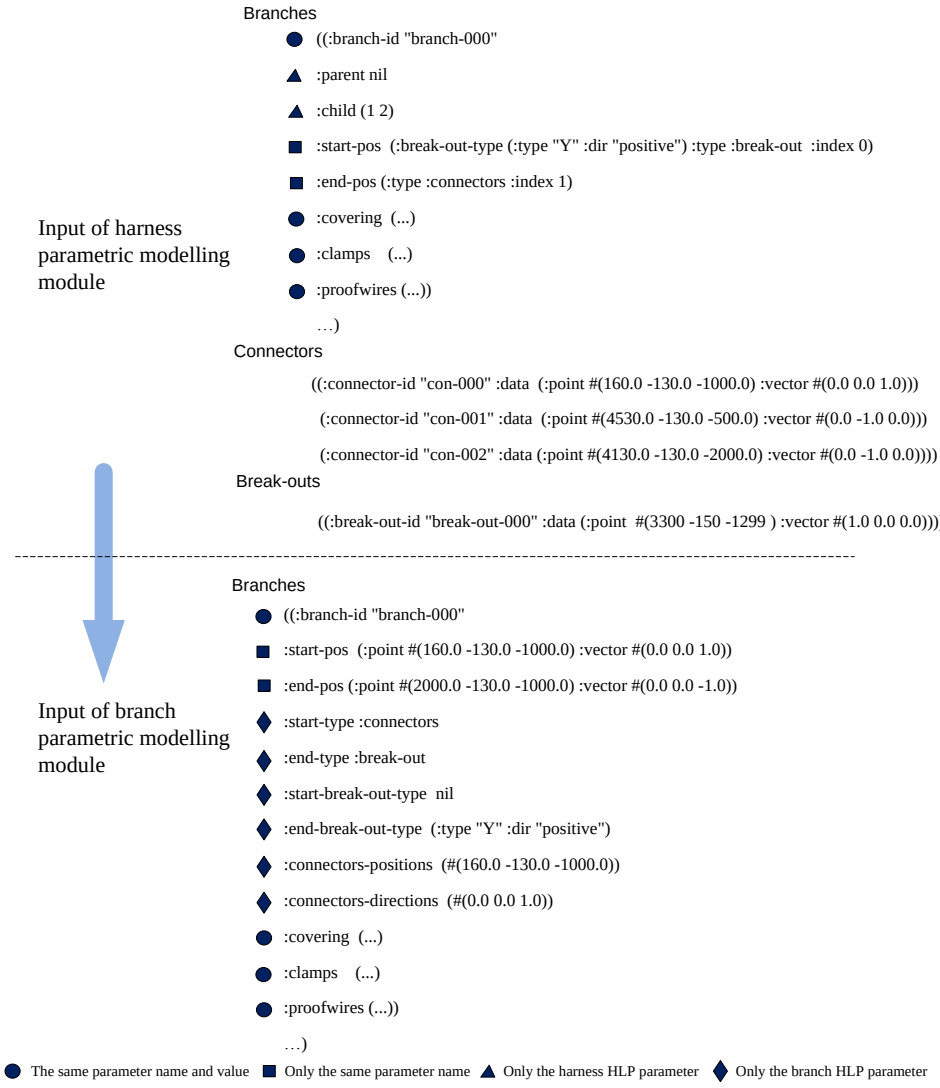


Figure 5-26: Transfer from the harness definition to the branch definition

The branch ID, covering definition, clamp definition, and property of the wires are the intrinsic features of a branch. They do not need any modification and will be passed down directly from the harness definition to the branch definition.

The start and end of a branch are defined by parametric relationships in the harness definition. The type and index of the end component, such as ***(:type :connectors :index 0)*** and ***(:type :breakout :index 0)*** are assigned to ***:start-poi*** and ***:end-poi*** of the branches. This

method guarantees that any modification of the connector or breakout will not influence the logical relationship among the branches. In the branch definition, each branch is defined independently from the others. Therefore, the parametric relationship in the harness definition needs to be transferred to the actual values of the parameters. For instance, the definition of the branch start point ***:start-poi (:type :connectors :index 0)*** in harness definition needs to be transferred to ***:start-poi (:point #(160.0 -130.0 -1000.0) :vector #(0.0 0.0 1.0))*** for the generation of the actual branch geometric model.

Besides the shared parameters, the input of dedicated parameters of a branch definition is also defined here. The ***:start-type*** and ***:end-type*** indicates whether the branch ends are connectors or breakouts. If the end is a breakout, the ***:breakout-type*** will be assigned by the ***:start-breakout-type*** and/or ***:end-breakout-*****type**. If the end is a connector, the position and direction of the end will be assigned to the ***:connectors-position*** and ***:connector-direction*** and the ***:breakout-type*** will be **Nil**.

After these processes, the required input parameters to define a branch are fully defined. They will be passed down to the branch parametric modelling module to enable the generation of branch models which finally constitute the entire harness model. Although each branch is generated independently, the geometric connectivity among different branches is guaranteed since the breakout interface between branches has been considered. An example of an entire wire harness that is the implementation of the harness parametric modelling module is shown in Figure 5-27.

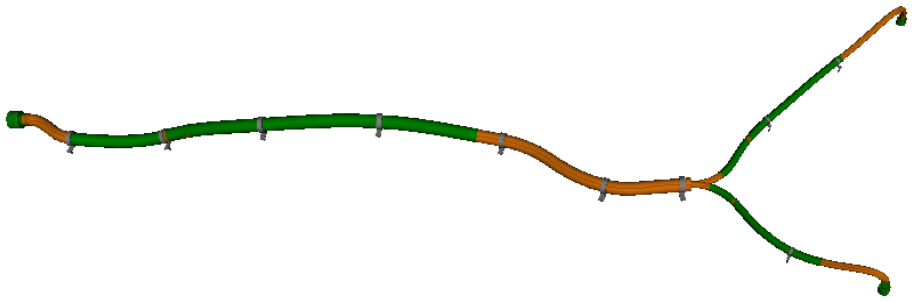


Figure 5-27: Geometric model of an entire wire harness

5.4 Introduction of harness analysis tools and the preparation of their input data

In the harness Refinement step, the previously generated harness model will be analysed by six analysis tools, namely the:

- **Harness Cost Analysis Tool (HCAT),**
- **Geometric Collision Analysis Tool (GCAT),**
- **Harness Bend Radius Analysis Tool (HBRAT),**
- **Clamp Distance Analysis Tool (CDAT),**
- **Harness CLearance Analysis Tool (HCLAT), and the**
- **Harness Separation Analysis Tool (HSAT).**

These tools are developed according to the methods of harness cost calculation and design constraint analyses of the harness Refinement. The mathematical principles of the analyses

can be found in the harness optimization problem definition presented in Section 6.2.

These analyses cannot be carried out with the harness geometric model immediately since each analysis tool needs some dedicated geometric data of the harness model. The preparation of the dedicated data for each analysis tool is carried out by the Capability Module (CM) of the tool. According to the input requirement of each tool, its Capability Module will extract the data, such as bundle length and bundle diameter, from the harness geometric model and generate the formalized data. Details of the CMs of the analysis tools are introduced in Appendix E.

Chapter 6 Development of the 3D-routing Refinement

During the Initialization step detailed in Chapter 4, a harness is simplified into a set of polylines. Some of the design rules, such as bend-radius violation, are also simplified. Some harness features that scarcely influence the initial results are not even considered during this phase. With this simplified method, there is no guarantee that the result generated in this phase is an optimum solution, nor even feasible for the actual routing problem.

Hence the second part of the two-step optimization method, namely the harness Refinement, is adopted to handle this problem in order to generate a design result that satisfies the actual design constraints. In the Refinement step, the harness geometric model is as accurate as the real harness DMU, and the actual design rules are adopted. The Refinement step is a conventional optimization process. The optimizer moves the harness model generated in the Initialization step to feasible areas by adjusting the design variables. Meanwhile, the cost of the harness is also minimized.

The entire Refinement workflow is shown in Figure 6-1 in the HDEE context. The development of the Harness MMG used to generate the harness model and the analysis tools used to evaluate the harness performance are already presented in Chapter 5. This chapter will detail the harness optimization problem definition and the solution to the problem. It will also briefly describe the technical implementation of the Refinement software tool. The use of the MMG and the analysis tools to support the optimization process will also be mentioned.

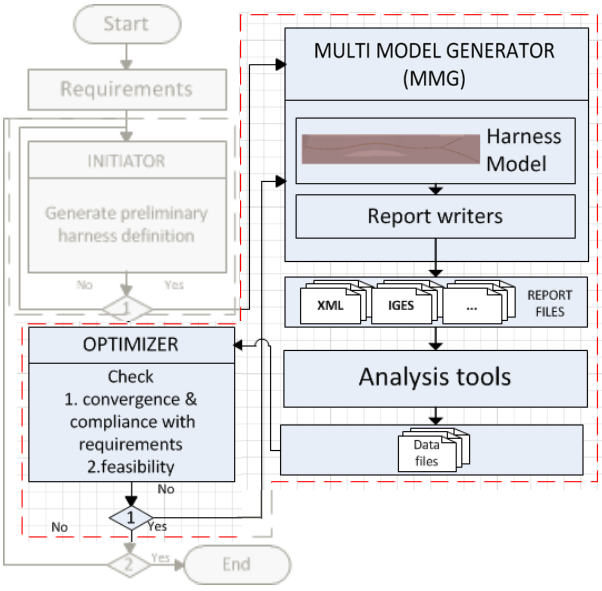


Figure 6-1: The optimization loop in the Harness DEE

In Section 6.1, the limitations of the Initialization, and the necessity for the Refinement, are discussed. Then the harness optimization problem definition is detailed in Section 6.2. An

approach to solving this optimization problem is presented in Section 6.3, followed by the introduction of the technical implementation in Section 6.4.

6.1 Limitations of the Initialization

As shown in Chapter 4, the Initialization is able to generate a preliminary harness definition including the number and initial positions of the clamps (i.e. the design variables). However, it has some limitations when considering the actual harness design problem. These limitations are listed below:

6.1.1 Inter-harness and inner-harness geometric collision

In the Initialization, the road map-based pathfinding is implemented per harness in order to quickly generate a preliminary harness definition. The pathfinding algorithm only handles the current harness but does not consider other harnesses. Hence, different harnesses will intersect with each other when they share the same edges and/or vertexes of the road map. The 3D routing of harnesses needs to take care of all the harnesses in one wiring zone and a geometric collision between harnesses is not allowed.

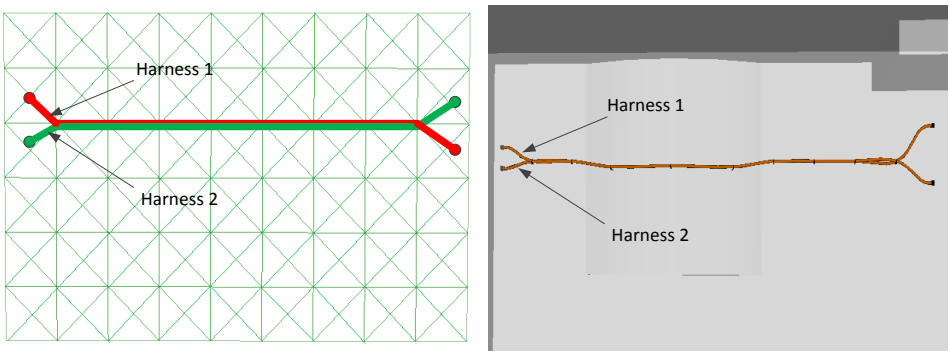


Figure 6-2: Illustration of inter-geometric collision using simplified (left) and actual (right) harnesses

A collision example is shown in Figure 6-2. The pathfinding program in Initialization always finds the minimum-cost path, in this example the shortest path. As a result, a large part of the two harnesses in this example shares the same map edges.

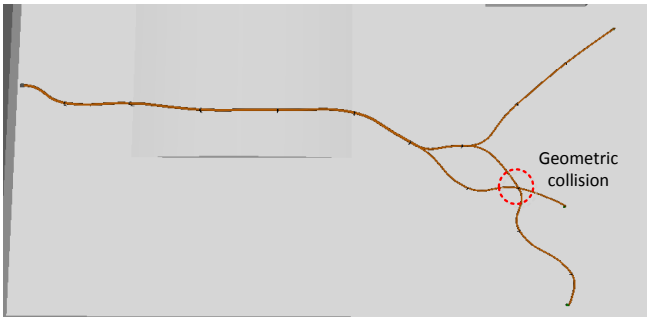


Figure 6-3: Inner-geometric collision between branches of a harness

Similar to the geometric collision between harnesses, different branches of one harness may also have a geometric collision due to the use of the same road map. An example is shown in Figure 6-3. Both the inter-harness and inner-harness geometric collisions need to be solved in

the Refinement step.

6.1.2 Geometric collision between harnesses and geometric components

The path of a wire harness is determined by its waypoints (i.e. clamps, breakouts or connectors). In the Initialization step, when using the road map-based pathfinding method, a set of line segments located between the waypoints is used to represent the harness path. The geometric collision check there is based on these line segments.

The actual harness path, however, is represented by a swept object whose central curve goes through these waypoints. In order to get a smooth curve, the central curve may deviate from these line segments between the waypoints. As shown on the left side of Figure 6-4, the deviated curve may intersect with other geometric components. Even if the curve itself is geometric-collision-free, the bundle may overlap with other geometric components, as shown on the right-hand side of Figure 6-4.

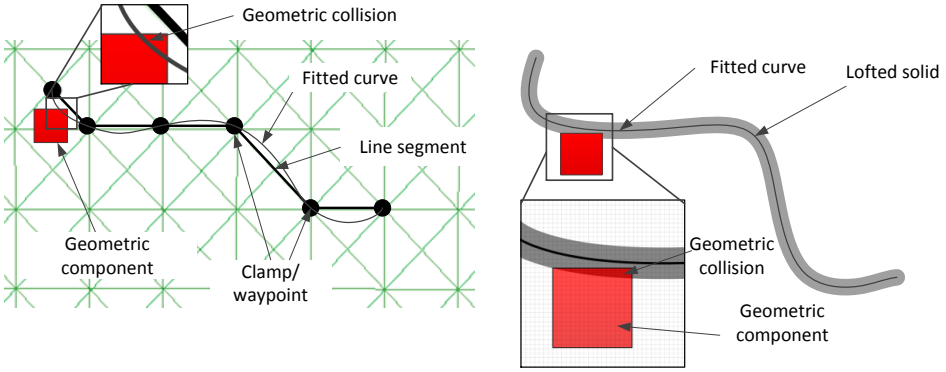


Figure 6-4: A geometric collision at the harness central curve (left) and harness bundle (right)

6.1.3 Bend-radius violation

Bend-radius violation is another problem of the line segment representation. In the Initialization step, the CSBPC method (see Sub-section 4.3.1.4) is used to avoid extreme bend-radius violation. However, due to the simplification, this method cannot guarantee the preliminary harness is bend-radius violation-free when considering the actual design rule. In order to get a feasible final routing result, the bend-radius violation needs to be handled in the Refinement step, using the actual central curve of the harness.

6.1.4 Non-optimum results

The main function of the Initialization is to find a preliminary harness definition to support the subsequent Refinement. Due to the fact that the harness has to be routed along the road map edges, the preliminary routing results are mostly not the optimum result in terms of the actual problem definition. Therefore they need to be optimized further to get a better result.

Due to the previously mentioned limitations, the 3D harness routing Refinement is adopted to generate the actual optimum result considering all the selected design rules. In the following three sections, the definition of the harness optimization problem, the development of the optimization method, and the technical implementation of the method are presented respectively.

6.2 Problem definition of the harness Refinement

The harness Refinement is defined and solved as an optimization problem. The selection and

further development of the optimization method is based on the brief optimization problem definition presented in Chapter 3.

The mathematical definition in Chapter 3 is recapped here:

$$\begin{aligned} &\min_{x} f(x) \\ &f(x)=\sum_{j=1}^{M} L_j(x)Co_j(x) \\ &\text{with respect to } x \\ &\text{subject to } C(x)\le 0 \end{aligned}$$

$\vec{x}$ is the vector of design variables and represents the position of the clamp and breakout points. The objective function $f(x)$ is the cost of the entire harness. This is represented by the summation of each branch cost $f_j(x)$, where j is the index of each branch. $f_j(x)$ is computed as the product of L_j and $Co_j(x)$. L_j is the length of branch j and depends on the design variables x. $Co_j(x)$ is the cost coefficient of branch j and depends on its bundle diameter, the amount of clamping elements, and required protective layers. $C(x)$ are the constraint functions to account for each design rule that has to be satisfied. The design variables, cost function, and design constraints are detailed below.

6.2.1 Definition of the design variables

In practice, design engineers modify the position and direction of the clamps and breakouts to control the shape and position of harness geometric models. However, in order to decrease the optimization problem scale, i.e. the number of design variables, only the positions of these waypoints are adopted as the design variables. The directions attached to these waypoints are calculated according to these positions and the routing environment. Details can be found in Section 4.4. $\vec{x_0}$; the initial values of the design variables, is defined by the preliminary harness definition generated in the Initialization step.

6.2.2 Definition of the objective function

The objective/cost function $f(x)$ represents the wire harness cost and it quantifies the harness performance. This quantification enables the optimizer to carry out a trade-off study to find a better harness during the optimization process. In practice, different stakeholders have different cost evaluation methods according to their own interests. For instance, the harness manufacturing companies prefer to calculate the cost based on the money they spend. However, the airlines want to include the harness weight since it affects the operating cost. The accurate cost calculation method should be given as an input to the 3D-routing tool; however it is not available yet for this proof of concept research. Therefore, the cost function defined here is based on some reasonable assumptions and the calculation result is in Euros.

$f(x)$ is computed as the summation of all the branch costs. The factors that influence the branch cost are classified into three contributors, namely: the bundle, clamp, and covering. They together determine the total cost of a branch. The cost of two harness branches that have the same length might be different if any of the three components is different. In order to address the influence of the three components on branch cost, a so-called cost coefficient $Co(x)$ is introduced. The cost of a branch is defined as the product of the branch length and the cost coefficient. The branch length is the length of the branch central curve and the cost coefficient is defined as the linear cost density of the branch. $Co(x)$ may not be uniform along the branch length since different clamping and covering methods may be used when the

branch goes through different environmental zones, as illustrated in Figure 6-5.

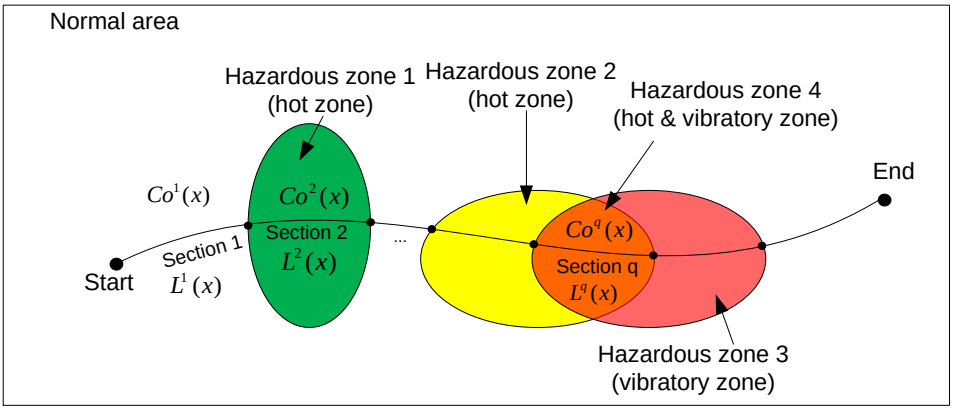


Figure 6-5: The cost coefficient of a harness branch in different zones

In order to calculate a non-uniform cost coefficient, the piecewise calculation method and sub-cost coefficients are introduced. With the piecewise method, the entire branch is divided into various sections according to the position of the hazardous zones along the branch, and the entire branch cost is the summation of all the sections' costs. The cost of section k, $f^k(x)$, is the product of the section length $L^k(x)$ and the section cost coefficient $Co^k(x)$. $Co^k(x)$ is calculated on the basis of three sub-cost coefficients, namely the sub-cost coefficient of bundle $Cob(x)$, clamp $Coc(x)$, and covering (protection) $Cop(x)$. These sub-cost coefficients, whose definitions will be detailed later, address the influence of the bundle, clamp, and covering respectively on the section cost $f^k(x)$. The $Co^k(x)$ is the summation of all the sub-cost coefficients, i.e. $Co^k(x) = Cob^k(x) + Coc^k(x) + Cop^k(x)$. Since different environmental areas (e.g. temperature and vibration level) need different protection and fixing methods, the value of each sub-coefficient may be different in different sections.

For instance, the branch in Figure 6-5 goes through a normal area and four hazardous zones. In section 1, $Co^1(x) = Cob^1(x) + Coc^1(x) + 0$ since a bundle and clamps exist in a branch but a covering is not used here. For section 2, because a covering needs to be used in the hot zone, $Co^2(x) = Cob^2(x) + Coc^2(x) + Cop^2(x)$. Section q is a combination of the hot zone and vibratory zone, hence $Co^q(x) = Cob^q(x) + Coc^q(x) + Cop^q(x)$. In this section, more clamps need to be used due to the vibration. Therefore, $Coc^q(x)$ will be bigger than in non-vibratory areas.

The final harness cost is defined in Equation (6.1), where S is the total number of harness sections of the branch j, and $L_j^k(x)$ and $Co_j^k(x)$ are the length and cost coefficient of branch section k respectively.

$$f(x) = \sum_{j=1}^{M} f_j(x) = \sum_{j=1}^{M} \sum_{k=1}^{S} L_j^k(x) Co_j^k(x) \tag{6.1}$$

In the following sub-sections, the definitions of the three sub-cost coefficients are detailed:

6.2.2.1 Bundle sub-cost coefficient $-Cob_j^k(x)$

$Cob_j^k(x)$ is the bundle sub-cost coefficient of section k on branch j. It depends on the bundle material and diameter of this bundle section. We assume that all the bundles have the same material. The bundle diameter/radius is determined by its wire gauge and number. The wire gauge and number is the same for all the sections of a branch. Using the calculation method mentioned in Sub-section 2.3.2.1, the bundle radius r_{bundle} can be calculated. The total cost of a branch section is shown in Equation (6.2),

$$f^{bk}_{\ j} = p_u e_{bundle} V_j^k = p_u e_{bundle} (\pi r_{bundle}{}^2 L_j^k) = L_j^k \times \pi r_{bundle}{}^2 p_u e_{bundle} \tag{6.2}$$

where p_u is the bundle unit weight price, e_{bundle} is bundle density, and V_j^k is the volume of section k. The volume depends on the section length L_j^k and radius r_{bundle}, and it together with the density e_{bundle} determines the bundle weight. According to the weight and the unit weight price, the total cost of a harness is calculated. From Equation (6.2) we conclude that $Cob_j^k = \pi r_{bundle}{}^2 p_u e_{bundle}$, and this represents the cost of the unit length bundle.

6.2.2.2 Clamp sub-cost coefficient $-Coc_j^k(x)$

The sub-cost coefficient of clamps Coc_j^k equals to $f^{ck}_{\ j} / L_j^k$. L_j^k is the branch length and $f^{ck}_{\ j}$, defined in Equation (6.3), is the cost summation of all the clamps in section k, branch j.

$$f^{ck}_{\ j} = nC_{\cos t\, j}^{\ \ k} = n(C_{m\, j}^{\ k} + C_i) \tag{6.3}$$

In this equation, n is the number of clamps in section k, branch j. $C_{\cos t\, j}^{\ \ k}$ is the cost of one clamp in section k. The clamp cost consists of the installation cost C_i and the material cost $C_{m\, j}^{\ k}$. C_i is assumed to be the same for all the clamps and given as an input. $C_{m\, j}^{\ k}$ is determined by the clamp unit weight price p_u, clamp density e_{clamp}, and clamp volume V, as shown in Equation (6.4).

$$C_{m\, j}^{\ k} = p_u e_{clamp} V = p_u e_{clamp} w_{clamp} \pi((r_2^{clamp})^2 - (r_1^{clamp})^2) \tag{6.4}$$

, where r_1^{clamp}, r_2^{clamp}, and w_{clamp} are the geometric parameters of the clamp and are already defined in Figure 5-8.

From the above equations we can conclude that $Coc_j^k = \dfrac{n(p_u e_{clamp} w_{clamp} \pi((r_2^{clamp})^2 - (r_1^{clamp})^2) + C_i)}{L_j^k}$.

In order to minimize the number of clamps, the adjacent clamps are always placed at the clamping distance approaching the maximum allowed clamping distance $D_{\max\ j}^{clampk}$, which depends on the design environment and design specification.

Therefore the number of clamps approximate to $L_j^k / D_{\max\ j}^{clampk}$. The sub-cost coefficient is then transferred to

$$Coc_j^k \approx \frac{L_j^k}{D_{\max\ j}^{clampk}} \frac{(p_u e_{clamp} w_{clamp} \pi((r_2^{clamp})^2 - (r_1^{clamp})^2) + C_i)}{L_j^k} = \frac{(p_u e_{clamp} w_{clamp} \pi((r_2^{clamp})^2 - (r_1^{clamp})^2) + C_i)}{D_{\max\ j}^{clampk}}.$$

6.2.2.3 Covering the sub-cost coefficient $-Cop_j^k(x)$

The sub-cost coefficient of covering $Cop_j^k = f_j^{pk} / L^{cover\,k}{}_j$. $L^{cover\,k}{}_j$ is the branch section length and f_j^{pk}, shown in Equation (6.5), is the cost of the covering outside the branch section k. The covering cost is determined by the unit weight price p_u, covering density e_{cover}, and volume V. The covering density and volume determine the covering weight. Based on the weight, and the unit weight price p_u, the total cost is calculated.

$$\begin{aligned} f_j^{pk} &= p_u e_{cover} V = p_u e_{cover}(2\pi r_1^{cover} L^{cover\,k}{}_j th_{cover} + \pi L^{cover\,k}{}_j th_{cover}{}^2) \\ &= L^{cover\,k}{}_j \pi p_u e_{cover}(2r_1^{cover} th_{cover} + th_{cover}{}^2) \end{aligned} \tag{6.5}$$

In Equation (6.5), r_1^{cover} is the inner radius of the covering and equals the outer radius of bundle. th_{cover} is the covering thickness. These geometric parameters are already defined in Figure 5-7. From Equation (6.5) we can work out that $Cop_j^k = \pi p_u e_{cover}(2r_1^{cover} th_{cover} + th_{cover}{}^2)$.

With the cost coefficient method, not only the harness cost is calculated, but also the hot zones and vibratory zones that influence the use of coverings and clamps respectively are handled. Both the covering and the clamp have dedicated sub-cost coefficients Cop and Coc. These sub-cost coefficients will be low in a low temperature or vibration zone and high in a high temperature or vibration zone. If the same type (i.e. either hot or vibratory) zones overlap with each other, the worst-case scenario (e.g. the highest temperature) caused by these zones will be considered when calculating the sub-cost coefficient. By using the cost-coefficient approach, the influences of the grey zones are quantified and accordingly the optimizer can find the best design from a set of competing ones, such as a short bundle with expensive cover to cross a hot area or a long bundle going around the hot area without any special thermal protection covering.

6.2.3 Definition of the design constraints

The physical design of harnesses needs to satisfy many design rules. During the 3D-routing Refinement, these rules are handled as constraint functions. The following part of this sub-section presents the constraint functions used to address the selected rules, namely 1) geometric collision-free, 2) bend-radius violation-free, 3) satisfaction of the clamp distance, including clamping distance and fixing distance, 4) satisfaction of the segregation and separation requirements, and 5) satisfaction of the clearance between the harness and geometric components.

6.2.3.1 Geometric collision constraint function $-C(x)_{collision}$

As presented in Sub-section 2.3.2.3, a harness can have three geometric collision types. Each type is represented by a sub-collision constraint function $C(x)^i_{collision}$, shown in Figure 6-6. $C(x)^i_{collision}$ is the total number of $type-i$ geometric collisions and will be detailed below. The constraint function of geometric collision $C(x)_{collision}$, defined in Equation (6.6), is the total number of all these sub-functions. $C(x)_{collision}$ is a non-negative integer. When $C(x)_{collision} > 0$, the given harness has geometric collisions; when $C(x)_{collision} = 0$, the harness is geometric collision-free.

$$C(x)_{collision} = C(x)^1_{collision} + C(x)^2_{collision} + C(x)^3_{collision} \tag{6.6}$$

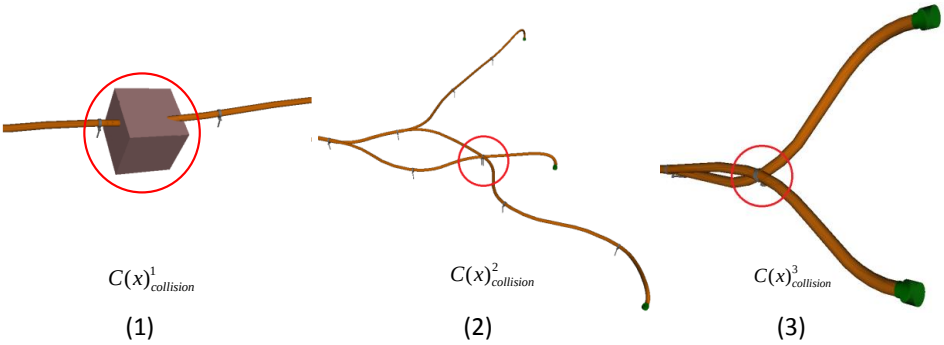


Figure 6-6: Three geometric collision types: (1) collision between the harness and other geometric components, (2) collision between different branches, (3) collision between two harnesses

- **Constraint function of the Type-1 collision**

The collision between harness branches and other geometric components, such as an airframe or aircraft system, is addressed as Type-1 collision. Its constraint function $C(x)^1_{collision}$ is defined with Table 6-1. This table lists all the branches and all the geometric components. 1 in the intersection cell means a geometric collision between the branch and the component; 0 means it is collision-free. $C(x)^i_{collision}$ equals the summation of all the 1s in this table.

Table 6-1: Calculation of the Type-1 geometric collision

Components \ Branches	1	2	…	M
1	1	0	…	1
2	0	0	…	0
…	…	…	…	1
Z	0	1	…	1

- **Constraint function of the Type-2 collision**

The Type-2 collision is the collision between two branches of the same harness. As shown in Figure 6-7, there are four different collision scenarios, the second of which, namely *collision at breakout,* is allowed and is not an actual geometric collision.

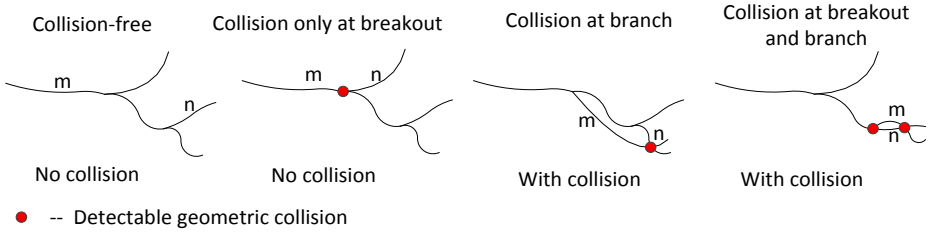


Figure 6-7: Harness branches geometric collision scenarios

The constraint function of the Type-2 collision $C(x)^2_{collision}$ is defined with Table 6-2. This table lists the combination of all the branches. If two branches have a geometric collision, the intersection cell of these branches is 1 and otherwise it is 0. $C(x)^2_{collision}$ equals the summation of all the number 1s in this table.

Table 6-2: Calculation of the Type-2 geometric collision (- : no need to be checked)

Branches / Branches	1	2	…	M
1	-	-	-	-
2	1	-	-	-
…	…	…	…	-
M	0	1	…	-

- **Constraint function of the Type-3 collision**

In the harness automatic 3D-routing method, all the wire harnesses in a wiring zone are routed one by one and the previously routed harnesses will be kept in the routing environment. Hence, the harness currently being routed may have geometric collisions with the previously routed ones (i.e. the Type-3 collision).

The constraint function $C(x)^3_{collision}$ is defined with Table 6-3. The geometric collisions between the current harness i and all the previously routed harnesses are checked. If a collision occurs, then the intersection cell between harness i and the previous harness is 1 and otherwise it is 0. $C(x)^3_{collision}$ of the current harness equals the summation of all the 1s in this table.

Table 6-3: Calculation of the Type-3 geometric collision

Previous harnesses / Current harness	1	2	3	…	i-1
1	1	0	0	…	1

6.2.3.2 Bend-radius constraint function $-C(x)_{bendradius}$

According to the design specifications presented in Sub-section 2.3.2.2, a wire harness is not allowed to take a sharp turn and the minimum bend radius should be several times bigger than the diameter of the harness itself. The bend-radius constraint $C(x)_{bendradius}$ function is used to represent the violation (or satisfaction) of this design rule of a harness and $C(x)^j_{bendradius}$ represents the violation of a harness branch j.

This design rules only ask for a Boolean result, namely whether $r^{bending}_{\min} < r^{bending}_{allowed}$, where $r^{bending}_{\min}$ is the minimum bend radius and $r^{bending}_{allowed}$ is the allowed bend radius. True (1) means the design rule is violated and False (0) means a feasible solution.

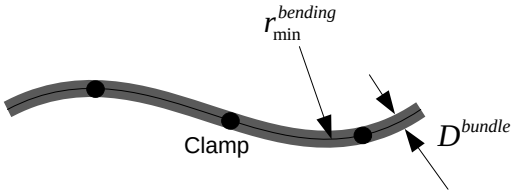


Figure 6-8: The bend radius and diameter of a bundle

Since the Boolean value cannot provide the trend information to facilitate the convergence of the optimization process, Equation (6.7) is used to define the $C(x)^j_{bendradius}$.

$$C(x)^j_{bendradius} = r^{bending}_{allowed} - r^{bending}_{\min} \tag{6.7}$$

$C(x)^j_{bendradius}$ is a rational number. When $C(x)^j_{bendradius} > 0$ the rule is violated and when $C(x)^j_{bendradius} \le 0$, the branch j is violation-free.

In practice, $C(x)^j_{bendradius} = -1000$ is not better than $C(x)^j_{bendradius} = -1$ since both of them satisfy the design rule. Therefore, we specify $C(x)^j_{bendradius} = 0$ if $\varsigma D^{bundle_j} - r^{bendradius_j}_{\min} \le 0$. Equation (6.7) is then updated to

$$C(x)^j_{bendradius} = max(0,\ r^{bending}_{allowed} - r^{bending}_{\min}) \tag{6.8}$$

The logical calculation $max(x, y)$ finds the larger number from two variables. When the branch j violates the rule, $C(x)^j_{bendradius} > 0$. Therefore, it is the actual violation value. When the branch j is violation-free, $C(x)^j_{bendradius} = 0$.

$C(x)_{bendradius}$ is the summation of all the branches' constraint function $C(x)^j_{bendradius}$ and defined in Equation (6.9).

$$C(x)_{bendradius} = \sum_{j=1}^{M} Max(0, C(x)^j_{bendradius}) \tag{6.9}$$

When all the branches are bend-radius violation-free, $C(x)_{bendradius} = 0$, which means the design constraint is satisfied.

6.2.3.3 Clamp-distance constraint function $-C(x)_{clamping}$ and $C(x)_{fixing}$

The clamp distance includes the clamping distance and fixing distance. As described in the design rules presented in Sub-section 2.3.2.5, the clamping distance needs to be smaller than the maximum allowed clamping distance and the fixing distance needs to be bigger than the minimum allowed fixing distance. These two rules are addressed by the two constraint functions $C(x)_{clamping}$ and $C(x)_{fixing}$ in the optimization.

- **Clamping-distance constraint function** $-C(x)_{clamping}$

The clamping distance check is performed per harness branch, between each pair of adjacent clamping points. These clamping points include the actual clamps as well as breakouts and connectors. The constraint function of a pair of clamping points i and $i+1$ is defined in Equation(6.10).

$$C_{clamping}^{i,i+1} = \begin{cases} 0 & if \quad D_{i,i+1}^{clamp} = \left\| x_{i+1} - x_i \right\| \le D_{\max}^{clamp} \\ 1 & if \quad D_{i,i+1}^{clamp} = \left\| x_{i+1} - x_i \right\| > D_{\max}^{clamp} \end{cases} \tag{6.10}$$

In Equation (6.10), x_{i+1} and x_i are design variables representing the position of adjacent clamping points, and $D_{i,i+1}^{clamp}$ is the clamping distance between them. $D_{\max}^{clamp}$ is the maximum allowed clamping distance to be respected. This value depends on the environment that the harness goes through (e.g. vibratory area). If $C_{clamping}^{i,i+1} = 1$, these two clamps violate the clamping distance rule; otherwise they are violation-free.

The clamping distance constraint function $C(x)_{clamping}$ is defined as the summation of all $C_{clamping}^{i,i+1}$ on each branch, as shown in Equation (6.11).

$$C_{clamping} = \sum_{j=1}^{M} C_{clamping}^{j} = \sum_{j=1}^{M} \sum_{i=1}^{C-1} C_{clamping}^{i,i+1} \tag{6.11}$$

In this equation, M is the total number of branches and C is the total number of clamping points on a branch. If $C_{clamping} = 0$, the entire harness is violation-free on the clamping distance.

- **Fixing-distance constraint function** $-C(x)_{fixing}$

The fixing-distance check of a complete harness is carried out per design variable, namely at each clamping point except the connectors since the position of the connectors is the input of the 3D harness routing. The fixing-distance constraint function of a clamping point is defined in Equation (6.12).

$$C_{fixing}^{i} = \begin{cases} 0 & if \quad d_{fix}^{clamp_i} = \left\| x_i - ps_i \right\| \ge d_{fix-min}^{clamp} & ps_i \ exists \ on \ fixable \ structure \\ 1 & if \quad ps_i = nil & no \ ps_i \ exists \ on \ fixable \ structure \\ 1 & if \quad d_{fix}^{clamp_i} = \left\| x_i - ps_i \right\| < d_{fix-min}^{clamp} & ps_i \ exists \ on \ fixable \ structure \end{cases} \tag{6.12}$$

In this equation, $d_{fix\text{-}min}^{clamp}$ is the minimum allowed fixing distance defined by Equation (2.2). As shown in Figure 6-9, x_i is the clamping point and ps_i is the fixing point on the fixable surface. $\left\| x_i - ps_i \right\|$ is the actual fixing distance. If ps_i cannot be found (i.e. $ps_i = nil$), namely the clamp i does not locate in front of any fixable structure, the fixing distance rule is considered to be violated. When $\left\| x_i - ps_i \right\| < d_{fix-min}^{clamp}$, the fixing distance of clamp i also violates the design rule; when $\left\| x_i - ps_i \right\| \ge d_{fix-min}^{clamp}$, the design rule is satisfied.

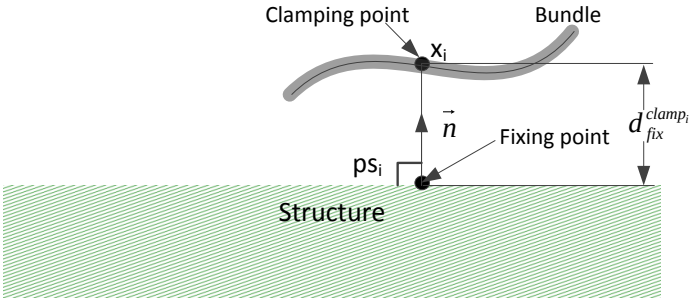


Figure 6-9: Definition of the harness fixing distance

The constraint function $C(x)_{fixing}$ is defined as the summation of the constraint functions of all the clamping points, as shown in Equation (6.13).

$$C_{fixing} = \sum_{j=1}^{M} C_{fixing}^{j} \tag{6.13}$$

In the equation, M is the total of the clamping points except the connectors. If all the clamping points are fixing-distance violation-free, $C_{fixing} = 0$ and the design constraint is satisfied.

6.2.3.4 Harness-separation constraint function- $C(x)_{sep}$

As described in Sub-section 2.3.2.8, each harness branch needs to be some distance away from the other harnesses. The allowed separation distance depends on the EGS code of the two branches.

If the harness currently being routed has M branches and previous harnesses have N branches in total, a separation requirement check needs to be carried out between these M and N branches, per branch pair. The separation constraint function of the current branch i and the previously routed branch j is defined as $C(x)_{i,j}^{sep} = \max(0, D_{i,j}^{allowed\_sep} - (D_{i,j}^{actual\_sep} - r_i - r_j))$.

As shown in Figure 6-10, $D_{i,j}^{actual\_sep}$ is the actual minimum 3D distance between branches i and j, measured from the central curves. $D^{allowed\_sep}$ is the minimum allowed separation distance between these branches, and it depends on the EGS code of the two branches. r_i and r_j are the radius of the two branches. If the separation requirement is satisfied, the constraint function equals 0. Otherwise, it is bigger than 0.

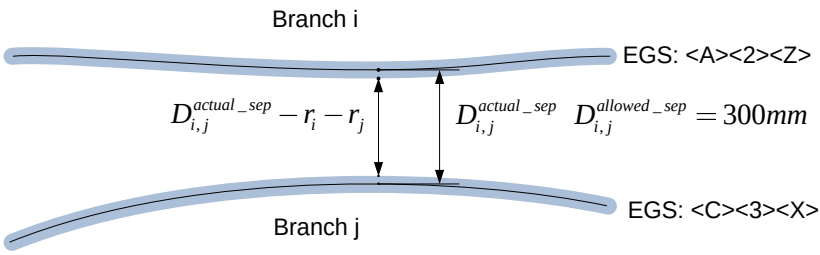


Figure 6-10: The separation requirement between two branches

As shown in Equation (6.14), $C(x)_{sep}$, the separation constraint function of the entire harness, is defined as the summation of the check results of all the branch pairs. If $C(x)_{sep} = 0$, the separation requirement of the current harness is satisfied.

$$C(x)_{sep} = \sum_{i=1}^{M}\sum_{j=1}^{N} C(x)_{i,j}^{sep} = \sum_{i=1}^{M}\sum_{j=1}^{N} \max(0, D_{i,j}^{allowed\_sep} - (D_{i,j}^{actual\_sep} - r_i - r_j)) \tag{6.14}$$

6.2.3.5 Harness-clearance constraint function- $C(x)_{clearance}$

As described in Sub-section 2.3.2.4, each harness needs to be $N\,mm$ away from dynamic envelopes that cover all the possible positions of moving parts and systems. The geometric models of the envelopes are defined by engineers and given as inputs to the 3D routing.

The harness-clearance constraint function is defined in a Boolean format, as shown in

Equation (6.15).

$$C(x)_{i,j}^{clearance} = \begin{cases} 0 & geometrycollision(env, thickerbundle) = 0 \\ 1 & geometrycollision(env, thickerbundle) = 1 \end{cases} \tag{6.15}$$

$geometrycollision(A,B)$ is a function that checks whether geometry A and B have geometric collisions. env is the geometric model of the dynamic envelop. $thickerbundle$ is the geometric model of an auxiliary bundle. The auxiliary bundle has the same central curve of the bundle that is about to be checked and its diameter equals the $diameter\ of\ the\ original\ bundle + 2N\ mm$. As shown in Figure 6-11, if the auxiliary bundle is geometric collision-free with the dynamic envelop, the actual bundle will be at least $N\,mm$ away from the dynamic envelope, namely $C(x)_{i,j}^{clearance} = 0$.

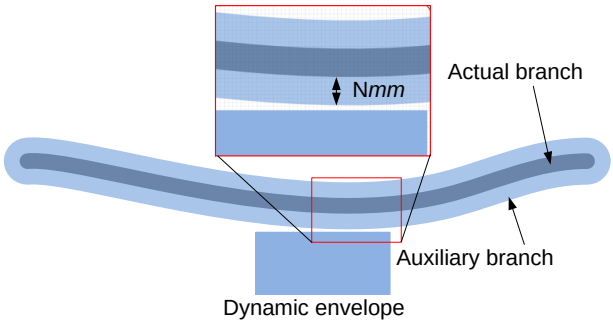


Figure 6-11: Illustration of the harness-clearance check method

$C(x)_{clearance}$, the clearance constraint function of an entire harness is defined as the summation of the check results between each branch and every dynamic envelop, as shown in Equation (6.16). M and N are the total number of harness branches and dynamic envelops.

$$C(x)_{clearance} = \sum_{i=1}^{M}\sum_{j=1}^{N} C(x)_{i,j}^{clearance} \tag{6.16}$$

6.3 The harness Refinement method

The optimization loop involving geometric models is enabled by three modules, namely the optimizer, harness geometric modelling module (Harness MMG), and the analysis tools. These three modules constitute the detailed optimization framework shown in Figure 6-12. This framework is based on the unified description of MDO framework (architecture) proposed by J.R.R.A. Martins[30].

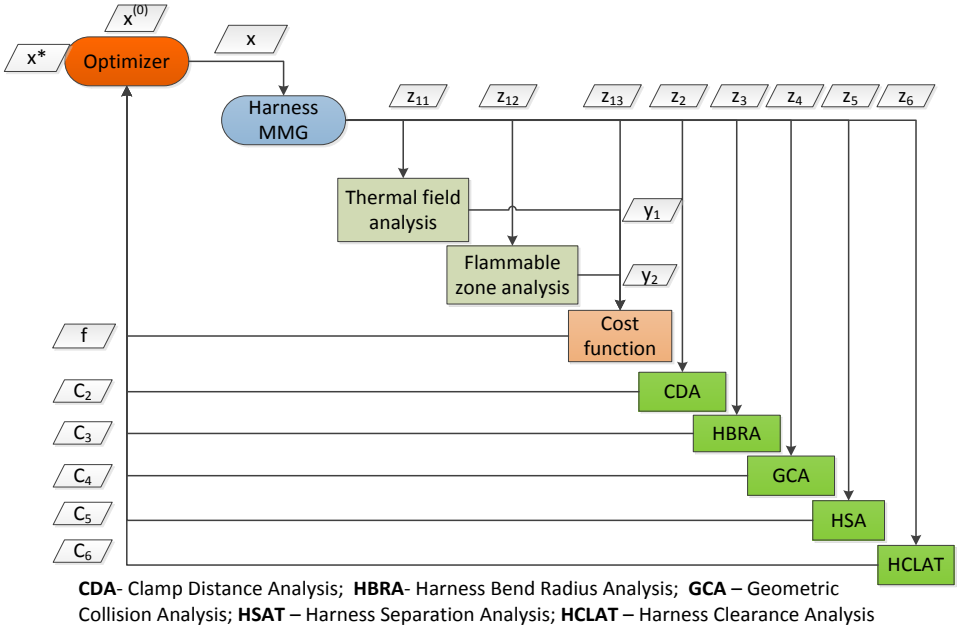


Figure 6-12: The optimization framework of wire harnesses

This monolithic optimization framework includes only one optimizer, which takes care of the entire optimization loop. The Harness MMG generates a harness geometric model using the design variable x prepared by the optimizer and also prepares the dedicated report data z for the subsequent analysis tools according to this geometric model. The analysis tools calculate the harness cost f (e.g. in Euros) and evaluate the constraint functions C. The results of f and C are sent back to the optimizer to support the decision making. If necessary, the optimizer generates new design variables to start a new loop until one of the stop criteria described in Sub-section 6.3.3.5 is met. In the following part of this section, the development of the Refinement is presented.

6.3.1 Harness geometric modelling

The automatic geometric modelling in the optimization loop is enabled by the Harness Multi Model Generator (HMMG) detailed in Chapter 5. With the given input harness configuration data (see the example in Figure G-3 in Appendix G), the HMMG generates a harness geometric model. The Capability Modules (CMs) will extract the required information from this geometric model and prepare dedicated report data (see Appendix E) for the subsequent analyses.

6.3.2 Analyses of a harness geometric model

The analyses of a harness geometric model include the calculation of the harness cost function $f(x)$ and the evaluation of various constraint functions $C_i(x)$.

6.3.2.1 Calculation of the harness cost

As shown in Equation(6.1), the cost of a wire harness $f(x)$ is the summation of all the branch's costs and therefore the harness cost calculation is implemented per branch. The cost of a branch depends on the branch length and the cost coefficient which is determined by the

bundle, the coverings, and the clamps.

- **Calculation of the bundle length**

The branch length is represented by the length of the bundle central curve. This curve is NURBS curve (see the bundle definition in Chapter 5) and its geometric model is generated by the KBE system, GDL. A built-in function of GDL can tell $L_j(x)$, the curve length of branch j, directly from the geometric model of the bundle.

- **Calculation of the bundle sub-cost coefficient**

Bundle sub-cost coefficient $Cob_j = \pi r_{bundle}{}^2 p_u e_{bundle}$ depends on the bundle uniform density e_{bundle}, unit weight price p_u, and radius r_{bundle}. The uniform density and unit weight price are simplified into the values shown in Table 6-4. The bundle radius is a property of a bundle. It depends on the number and gauge of bundle wires and is already calculated while generating the harness geometric model. During the analysis, the radius can be extracted directly from the bundle geometric model prepared by the CM of this analysis tool.

Table 6-4: Parameters for the bundle sub-cost coefficient calculation

e_{bundle} -uniform density, including the conductor and insulator	$0.00896 g/mm^3$ (copper density)
p_u - unit weight price	$1\ euro/g$

- **Calculation of the clamp sub-cost coefficient**

Clamp sub-cost coefficient $Coc_j^k = \dfrac{(p_u e_{clamp} w_{clamp} \pi((r_2^{clamp})^2 - (r_1^{clamp})^2) + C_i)}{D_{\max\ j}^{clamp\ k}}$ depends on installation cost C_i, clamp geometric dimensions r_1^{clamp}, r_2^{clamp}, and w_{clamp}, unit weight price p_u, maximum allowed clamping distance $D_{\max}^{clamp}$, and the material density of clamp e_{clamp}. C_i is assumed to be the same for all the clamps and given as an input (e.g. 50 Euros). r_1^{clamp} is the same as the bundle's outer diameter. The remaining clamp geometric dimensions are in proportion to r_1^{clamp} and their calculation details can be found in Chapter 5. The p_u of the clamp and e_{clamp} is assumed to be the same as the p_u of the bundle and e_{bundle} respectively. $D_{\max}^{clamp}$ depends on the routing environment. It is 24 inches in normal areas and will decrease in the flammable zone. The definition of the flammable zone and how the flammable zone influences the clamping distance can be found in Appendix D.

- **Calculation of the covering sub-cost coefficient**

Covering sub-cost coefficient $Cop_j^k = \pi p_u e_{cover}(2r_1^{cover} th_{cover} + th_{cover}{}^2)$ depends on the unit weight price p_u, covering density e_{cover}, covering inner radius r_1, and covering thickness th_{cover}. It is assumed that the p_u of the covering is the same as the p_u of the bundle. The covering inner radius r_1 is the same as the bundle radius, and the covering thickness th_{cover} is a fixed value and given as an input. The covering density e_{cover} depends on the temperature of the routing zone. It is the same as e_{bundle} in the normal temperature area and will increase in a hot zone. The description of hot zones and how the hot zones influence the e_{bundle} and the harness cost is presented in Appendix C.

The above-mentioned harness cost calculation method has been developed into a software tool called the Harness Cost Analysis Tool (HCAT) to implement the calculation automatically. The calculation is carried out per branch and the total harness cost is the summation of all the branches. The workflow of this tool is illustrated in Figure 6-13.

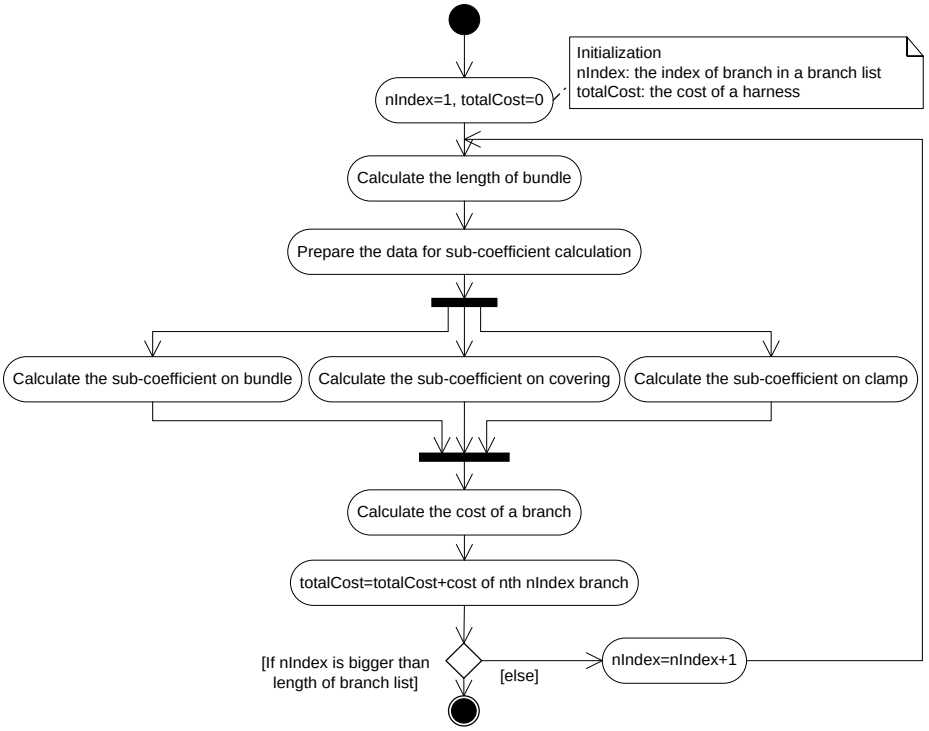


Figure 6-13: Workflow of the Hazardous Cost Analysis Tool (HCAT)

6.3.2.2 Calculation of the geometric collision constraint function

The geometric collision constraint function consists of three parts.

The Type-1 collision happens between harness branches and the geometric components of the routing environment. In the KBE system used to take care of all the geometric operations, both the geometric components and harness branches are represented by the Boundary Representation (BREP)[63]. A built-in function of the KBE system can tell whether two given geometries have a collision. As shown in Table 6-1, the check result is 1 if there is a collision and otherwise the result will be 0. All the branches and geometric component combinations are checked and the check result $C(x)^1_{collision}$ is the summation of all the 1s in Table 6-1.

The Type-2 collision happens between harness branches. The check starts from branch 0 (the index of the branch is illustrated in Figure 5-16). The geometric collision between this branch and the following branches will be checked using the BREP collision check function of the KBE system. In order to handle the scenario called *collision only at breakout* (see Figure 6-7), a function is developed to check whether the two branches share one breakout according to the parent-child relationship defined in the harness electrical definition (see the example in Figure G-1 in Appendix G). If so, as shown in Figure 6-14, two new branches which start/end

at $\frac{1}{10}$ of the branch length away from the breakout will be built. The collision check will be carried out with the new branches. The check result is 1 if there is an actual collision and otherwise the result will be 0, and this result will be recorded in Table 6-2. Then the check moves to the next branch until all the branches are checked. The result $C(x)^2_{collision}$ is the summation of all the number 1s in Table 6-2.

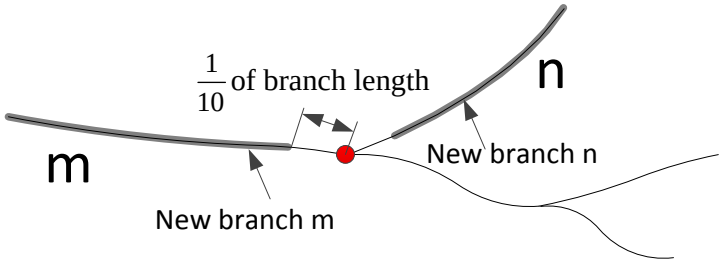


Figure 6-14: Generation of new harness branches

The Type-3 collision happens between a harness currently being routed and a previously routed harness. Both the harnesses are represented by BREP. The built-in function of the KBE system can tell whether two harnesses have a collision. As shown in Table 6-3, the check result is 1 if there is a collision and otherwise the result will be 0. The result $C(x)^3_{collision}$ is the summation of all the number 1s in this table.

The above mentioned geometric collision check has been developed into a software tool called the Geometric Collision Analysis Tool (GCAT) to implement the check automatically. The collision check is carried out per collision type and the total collision cost is the summation of all three collision types.

6.3.2.3 Calculation of the bend-radius constraint function

The bend-radius violation analysis is carried out per bundle. The bend-radius violation analysis of a bundle consists of 1) calculating the minimum allowed bend radius and 2) calculating the actual minimum bundle bend radius $r^{bending}_{\min}$, and 3) comparing the two values.

The minimum allowed bend radius is calculated with the formula $\max(\varsigma^{bundle} D^{bundle}, \varsigma^{wire} D^{wire}) + D^{bundle}/2$. D^{bundle} and D^{wire} are the diameter of the bundle and the thickest wire respectively. ς^{wire} and ς^{bundle} are the allowed bend-radius ratio of the wire and bundle, and they depend on the material of the wire and bundle, as shown in Table 2-2. $D^{bundle}/2$ is the bundle radius. The bundle diameter D^{bundle_j} and the diameter of the thickest wire D^{wire} are prepared by CM of this analysis tool immediately after generating the harness geometric model (see Appendix E.3); ς^{bundle} and ς^{wire} are extracted from Table 2-2 and given as inputs.

A built-in function of the KBE system can calculate $r^{bending}_{\min}$ of the bundle central curve automatically. Then the violation result of a bundle can be calculated according to Equation (6.8), and consequently, the violation result of a harness can be calculated according to Equation (6.9).

The previously mentioned analysis process is implemented into a software tool called Harness Bend Radius Analysis (HBRA), which can carry out the analysis automatically.

6.3.2.4 Calculation of the clamp-distance constraint function

The clamp-distance constraint function calculation consists of the calculation of the clamping-distance constraint function $C(x)_{clamping}$ and fixing-distance constraint function $C(x)_{fixing}$.

- **Calculation of the clamping-distance constraint function**

The clamping-distance violation of an entire harness is checked per branch, between each pair of adjacent clamping points. Calculating $C(x)^{i,i+1}_{clamping}$, the constraint function of a pair of clamping points, consists of three parts: 1) calculating the actual clamping distance between the two clamping points x_i and x_{i+1}, 2) evaluating the maximum allowed clamping distance $D^{clamp}_{\max}$, and 3) comparing the two values.

The 3D coordinate of the clamping points of each branch is extracted from the harness geometric model by the CM and given as input (see Figure E-5 in Appendix E). With the coordinate of the adjacent clamping points, the actual distance is calculated.

The maximum allowed clamping distance depends on the routing environment. In the normal area, the allowed clamping distance is 24 inches and this will decrease in the flammable zone due to the pipe containing flammable fluid. In this zone, the closer to the pipe, the smaller the allowed clamping distance. The maximum allowed clamping distance can be calculated with the given clamping points and the position of the pipe. The calculation process is detailed in Appendix D and the allowed clamping distance is the output of this calculation.

After the actual and allowed clamping distances are known, $C(x)^{i,i+1}_{clamping}$ can be calculated by comparing the two values. The clamping distance constraint function of an entire harness is the summation of all the pairs of the clamping point.

- **Calculation of the fixing-distance constraint function**

The fixing distance check is implemented per clamping point on each branch. As shown in Equation (6.12), the calculation of a fixing-distance constraint function consists of three parts: 1) evaluating the minimum allowed fixing distance $d^{clamp}_{fix\text{-}min}$, 2) calculating the actual fixing distance between the clamping points x_i and the fixable structure, and 3) comparing the two values.

$d^{clamp}_{fix\text{-}min}$ depends on the bundle diameter, bundle sagging, and the clearance requirement between the harness and geometric components (see the definition in Equation (2.2)). In this research, in order to simplify the calculation process, $d^{clamp}_{fix\text{-}min}$ is calculated only once before the 3D-routing phase. The maximum clamping distance that determines the maximum sagging distance and the maximum recommended bundle radius (i.e. 1 inch) are used to calculate the $d^{clamp}_{fix\text{-}min}$. If this allowed fixing distance is satisfied, a harness which is thinner and has a smaller clamping distance (i.e. a smaller sagging distance) will always have enough clearance to the attached structure.

The actual fixing distance between the clamping points x_i and the fixable structure is calculated by a built-in function of the KBE system. This function can find the shortest point ps_i on each fixable surface to x_i and get the fixing distance set $\{\|x_i - ps_{i,1}\|, \|x_i - ps_{i,2}\|, ... \|x_i - ps_{i,M}\|\}$, where M is the total number of fixable structures. If x_i does not locate in front of any fixable surface, namely the fixing distance set is $\{nil\}$, the

design rule is considered to be violated and $C(x)^{j}_{fixing}$ is set to 1. Otherwise a further check is needed. All the values in the set are compared with the allowed fixing distance. The smallest feasible fixing distance is selected as the fixing distance of the clamping point and the fixable surface will be used to fix the clamp. In this case, $C(x)^{j}_{fixing}=0$. If none of the values in the set are feasible, $C(x)^{j}_{fixing}=1$. $C(x)_{fixing}$ is the summation of the check results of all the clamping points.

The previously mentioned clamping and fixing distance check is implemented into a software tool called the Clamp Distance Analysis Tool (CDAT), which can carry out the analysis automatically.

6.3.2.5 Calculation of the harness-separation constraint function

As shown in Equation (6.14), the constraint function of an entire harness is calculated per branch pair. The check between the currently being routed branch i and the previously routed branch j consists of three parts: 1) calculating the allowed distance between the two branches, 2) calculating the actual 3D distance between the two branches, and 3) comparing the two values.

The allowed distance is calculated with Table 2-4. As described in Sub-section 2.3.2.8, every branch has a segregation code EGS. This code for a branch/harness currently being routed is included in the harness geometric model and can be extracted directly from the model. The code for the previously routed harness branches is prepared by the CM of this analysis tool (see Appendix E.5). With these two codes, the allowed distance is calculated.

The actual 3D distance between the central curves of the two branches is calculated by a built-in function of the KBE system. With the two given geometric models of the central curves, which are provided by the CM and extracted from the current geometric model respectively, this function will tell the actual minimum 3D distance. This distance minus the radius of the two bundles is the actual distance (measured from the inside) between the two branches. Then the $C(x)^{sep}_{i,j}$ can be calculated. The summation of all $C(x)^{sep}_{i,j}$ equals $C(x)^{sep}$.

The previously mentioned calculation process is implemented into a software tool called the Harness Separation Analysis Tool (HSAT). This tool takes care of all the analyses in the optimization loop, automatically.

6.3.2.6 Calculation of the harness-clearance constraint function

As shown in Sub-section 6.2.3.5, the harness-clearance constraint function is calculated by the geometric collision check between the auxiliary geometric model of each branch and the dynamic envelop. The auxiliary branch models of a harness are prepared by the CM of this tool (see Appendix E.6). These models and the dynamic envelop are both given as inputs to this check. The built-in function of the KBE system can detect a collision between the two geometric models. If these two geometric models are geometric collision-free, then $C(x)^{clearance}_{i,j}=0$, and otherwise $C(x)^{clearance}_{i,j}=1$. Then $C(x)_{clearance}$ of the entire harness can be calculated.

The previous check is implemented into a software tool called the Harness CLearance Analysis Tool (HCLAT) to calculate the clearance constraint function.

6.3.3 Development of the optimizer

6.3.3.1 Selection of the optimization algorithm

Harness 3D-routing optimization has two features:

- **Gradient-free:**

The algorithm of the harness 3D optimization needs to be gradient-free. This is because some constraint functions of the optimization, such as the geometric-collision constraint function, do not contain any gradient information.

- **Slight adjustment of the design variables:**

The harness Initialization step already generates a promising harness preliminary definition. The followed optimization only needs to slightly adjust the design variables (i.e. the clamps and breakouts) to achieve a feasible and optimum result.

These two features support the selection of the Generalized Pattern Search (GPS)[64-66] optimization algorithm for the 3D-routing optimization. The Generalized Pattern Search is a subclass of the Pattern Search algorithm and the word Generalized comes from the method to refine the mesh[64]. The Pattern Search belongs to the Direct Search, which is an optimization method that does not require the gradient information in order to carry out the optimization[67]. In contrast to other Direct Search algorithms, such as the Genetic Algorithm, which moves the design variables randomly in the searching space, the GPS carefully adjusts the positions of clamps and breakouts to nearby places. This feature of GPS is very suitable to refine the harness preliminary definition that has already been defined in the harness Initialization step. The pseudocode of the GPS algorithm is presented in Figure 6-15.

```
;;;*******************************************************
;;;Pseudocode of Generalized Pattern Search algorithm
;;;*******************************************************
Begin
;;; Initialization
  Set x_b = x_0  ;;  x_0 : initial vector in R^N  ; x_b :the benchmark design variables
      Δ_0        ;; initial mesh size
      Δ_Tol      ;; Tolerance on mesh size; if Δ ≤ Δ_Tol  the optimization stops
;;; Polling
   For i = 0,1,... ;
      F_t = {f(t) : t ∈ {x_i ± Δ_i e_j : j ∈ N}} ;;e_j is a vector in R^N ; its element j is 1 and other elements are 0
      If min(F_t) ≤ f(x_b)
         set x_{i+1} = t_min  ;t_min belongs to R^N ; is the vector in of min(F_t)
             x_k = t_min
             Δ_{i+1} = αΔ_i  ; α ≥ 1
      Else
         set x_{i+1} = x_i ;
             Δ_{i+1} = βΔ_i  ; 0 < β <1  ;; refine the mesh
      End
      If Δ_{i+1} < Δ_Tol
         Break;
      End
   End
End
```

Figure 6-15: Pseudocode of the Generalized Pattern Search algorithm

The 3D position of the clamps and breakouts are transformed into a $1\times N$ vector, where N is equal to three times the total number of breakouts and clamps. For the harness illustrated in Figure 6-16, the transformation method is presented in Table 6-5.

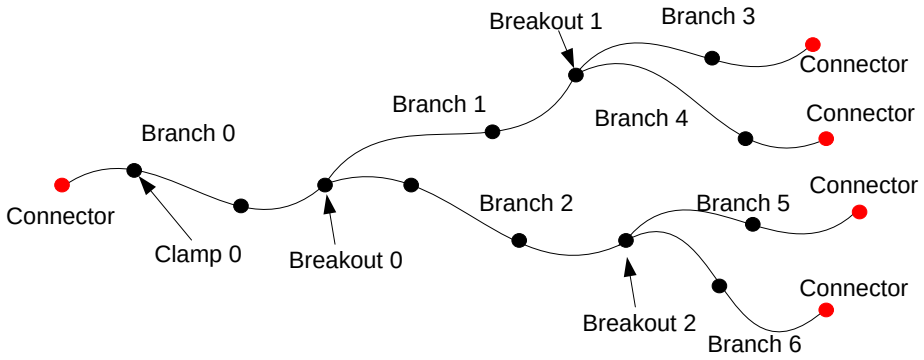


Figure 6-16: Index of the branches and breakouts on a harness

The transformation is implemented first on the clamps and then on the breakouts. A clamp point (x, y, z) will be transformed into a 1×3 vector $[x\,y\,z]$ and grouped with other clamp vectors. Then the breakouts are transformed and appended.

Table 6-5: Standard for design variable transformation

Branch 0									Branch …	Branch N				Breakout 0			...	Breakout N		
C_0			.....			C_i			…	...	C_j			B_0			...	B_N		
x_1	x_2	x_3	x_4	x_5	x_6	…	…	…	…	…	x_{3j+1}	x_{3j+2}	x_{3j+3}	x_{3j+4}	…	…	…	…	…	x_M

This transformation is bi-directional. 3D points need to be transformed into a vector required by the optimizer and the vector also needs to be transformed back to 3D points to support the geometric modelling. The standard in Table 6-5 supports both the transformations.

In Figure 6-15, x_0 is the initial values of the design variables and is assigned to x_b. $f(x_b)$ will be calculated as the initial benchmark. Then each element in the design variable vector will be modified by a certain step Δ_i to generate a new vector. Each time only one element will be changed. Since the elements of the design variable vector are the x, y, and z coordinates of the clamp or breakout, the modification is to move each clamp or breakout in one of the six directions (i.e. up/down/left/right/front/back).

Moving all the elements in the vector will generate a set of new vectors and consequently a set of values of the objective function F_t. The minimum value of the objective function $\min(F_t)$ in the current loop will be taken out to compare with the benchmark value of $f(x_k)$. If $\min(F_t) < f(x_k)$, the benchmark will be set to $\min(F_t)$ and the vector of design variable x_{i+1} will be set to $t_{\min}$. The step size Δ_i of the next move (i.e. the mesh size) will be increased in order to increase the convergence speed. This is called a successful iteration, or a successful polling in GPS terms. If $\min(F_t) \geq f(x_k)$, this iteration is unsuccessful. The step Δ_i will be decreased to start searching for a local optimum value. Moving all the vector elements in one optimization loop is called the *Exploratory Move,* and updating the design variables after one loop is known as the *Pattern Move*. $\min(F_t) < f(x_k)$ is called a *successful pattern*. Otherwise it is called a *failed pattern*.

If the pathfinding has already reached an optimum position (no matter global or local), moving the design variables will no longer provide a successful pattern. Therefore the mesh size will be decreased continuously until it is smaller than the mesh size tolerance Δ_{Tol}. Then

the optimization stops.

6.3.3.2 Discussion of bounds of the design variables

Bounds, i.e. $l \le x \le u$, specify the potential positions of the design variables. The design variables of the harness optimization (i.e. the position of the clamps and breakouts) need to be placed inside an aircraft or a wiring zone. However, this requirement cannot be represented by the bounds of the design variables due to the irregular shape of the aircraft (zone).

Actually, the bounds of the design variables is not necessary. The pathfinding in the Refinement step always starts from the preliminary definition generated in the Initialization step. Moving the clamps and breakouts to the distinct positions will significantly increase the cost of the harness. The GPS is a heuristic algorithm and therefore it always chooses the best result as a benchmark for the next iteration. This means that starting from the preliminary definition, the algorithm will not move the harness to a remote area, which would make the cost function surge. Therefore, no bounds need to be given to each design variable. In other words, the bounds of the design variables are $\pm\infty$. A case study given in Figure 6-17 proves the above argument.

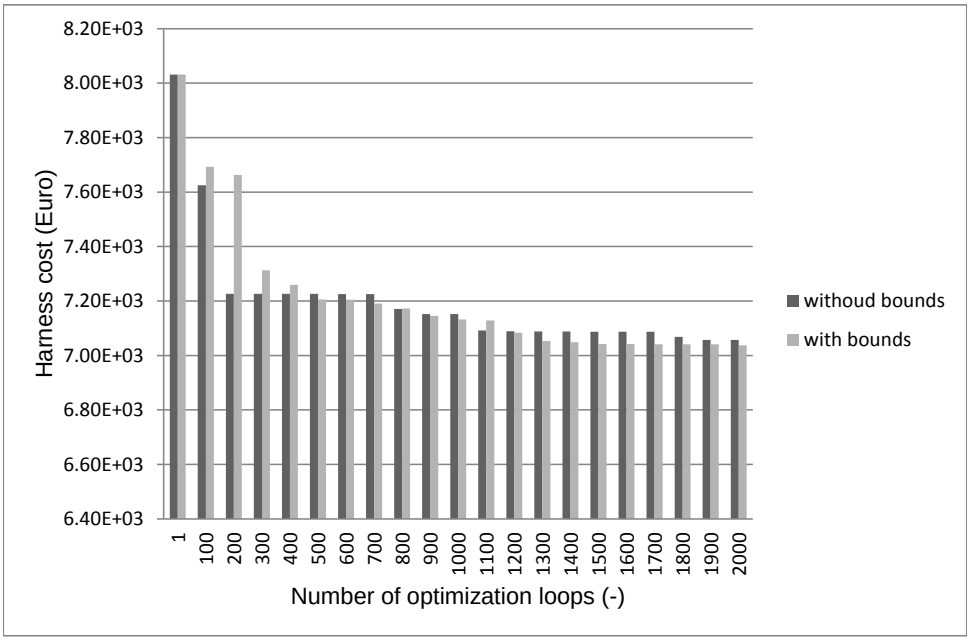


Figure 6-17: Comparison between optimization with and without bounds

In this case study, the bounds of the two optimization implementations are set to $x_0 - 500mm \le x \le x_0 + 500mm$ and $x_0 - \infty mm \le x \le x_0 + \infty mm$ (i.e. without a bound) respectively. x_0 is the initial value of x. The first bound means that each clamp/breakout point can be moved in a $1m \times 1m \times 1m$ cube whose centre point is a clamp or breakout defined in the harness preliminary definition. The second bound means that the clamps and breakout can be moved anywhere. This case study compares the routing performance of the two scenarios. The results show that these two implementations perform similarly in terms of the capability to find the optimal result and the consumed time. Therefore, the harness

optimization is set to be bounds-free.

The requirement that the design variables need to be routed inside an aircraft will be satisfied by the harness geometric collision check. If the harness which connects with the equipment located inside the aircraft is geometric collision-free (with the airframe), then all the clamps and breakouts will be inside the aircraft.

6.3.3.3 Satisfaction of the non-linear constraints

The non-linear constraint functions represent the design rules that have to be satisfied. During this harness Refinement, the constraints are considered as a part of the objective function. The actual objective function $f(x)$ is upgraded to a new function $\theta(x)$, defined in Equation (6.17).

$$\theta(x) = f(x) + \sum w_i C_i(x) \tag{6.17}$$

Here, $C_i(x)$ is one of the constraint functions, namely $C(x)_{collision}$, $C(x)_{bendradius}$, $C(x)_{clamping}$, $C(x)_{fixing}$, $C(x)_{clearance}$, and $C(x)_{sep}$. $C_i(x)$ will be 0 if the rule i is satisfied. Consequently $\sum w_i C_i(x)$ will be zero if all these functions are satisfied and $\theta(x)$ will turn to the actual objective function $f(x)$. w_i is the weight of the constraint function i. The reason to include the weight is detailed below.

In general, the value of the actual objective function $f(x)$ varies from hundreds to thousands of Euros according to the assumed data. The constraint function $C_i(x)$ is only, for example, 1 or 2 if there are 1 or 2 geometric collisions of a harness. This constraint function value is almost negligible while comparing with the objective function and therefore the constraint violation is almost impossible to be handled by the optimizer.

Satisfaction of the design constraints has a higher priority than getting a cheaper harness. Therefore, the weight of the constraint functions is introduced to ensure that the penalty of the constraint functions and original cost function have a reasonable proportion in $\theta(x)$ and consequently the design rule violations can be handled by the GPS optimizer. The principles to set the weight are given below:

- **The optimizer first moves the design variables to a feasible area if the preliminary harness definition is infeasible;**
- **The optimizer pursues a better result in terms of $f(x)$ in both the feasible and infeasible areas;**
- **The design variables will not be moved back to the infeasible area from a feasible area.**

The weight w_i is calculated according to the harness initial cost $f(x_0)$. The $f(x_0)$ is represented by the scientific notation $a.bcde \times 10^n$. We specify $w_i = w \times 10^n$. w is the round up integer of $a.bcde$ and therefore is never smaller than $f(x_0)$. For instance, if $f(x_0)$ is 3.456×10^n, then the weight w_i will be 4.000×10^n.

The constraint function whose output is the summation of the Boolean value, such as $C(x)_{collision}$ and $C(x)_{clamping}$, can use the weight directly to constitute the objective function $\theta(x)$. However this weight is not suitable to be directly used by continuous constraint functions, such as the bend radius constraint function $C(x)_{bendradius}$ and the harness separation

constraint function $C(x)_{sep}$. These continuous constraint functions provide the trend to the optimizer and the proportion of $w_i C(x)_i$ will be too big (when the harness configuration is far away from the feasible area) or too small (when the harness configuration is close to the feasible area) in $\theta(x)$. Therefore an extra process is needed in order to include these constraint functions in the objective function $\theta(x)$. Details of the process are presented in Appendix H by using the example of the bend-radius constraint function.

The performance to handle the constraint function as a part of the objective function is demonstrated by a 3D-routing test case and the routing result is illustrated in Figure 6-18.

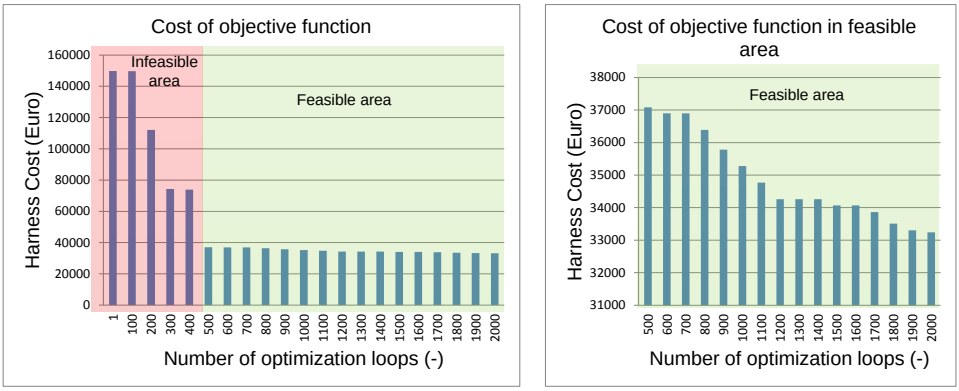


Figure 6-18: The cost function decreasing trend in the entire optimization process (left) and the feasible area (right)

In this example the preliminary harness definition generated by the Initialization has three constraint violations: 1) a bend-radius violation, 2) a geometric collision between two branches, and 3) a geometric collision between the harness and the routing environment. During the optimization, the optimizer first moves the design variables to eliminate the constraints gradually until it reaches the feasible area, where $\sum w_i C_i(x) = 0$ and $\theta(x)$ becomes the actual objective function $f(x)$. In the feasible area, the cost of the objective function $f(x)$ (or $\theta(x)$) continuously decreases until one of the stop criteria is reached. Because the penalty to get the design back to the infeasible area is very high, as shown in the picture, the harness will stay in the feasible area until the end of the optimization.

6.3.3.4 Consideration of the reserved space

It is preferred that harnesses are routed in the reserved space and this preference is considered in the Refinement step. As shown in Sub-section 2.3.2.7, the coefficient ς_{res} which is smaller than 1 is applied to decrease the cost of the harness routed in the reserved space. Since routing in the reserved space is only a preference but the constraint functions have to be satisfied, ς_{res} is applied to the actual cost function $f(x)$ only to calculate the auxiliary cost $f_{auxcost}$. The constraint functions in the new objective function $\theta(x)$ are not influenced.

The coefficient ς_{res} is only used in the Refinement step to support the decision making of the optimizers and this does not influence the actual harness cost.

6.3.3.5 Discussion of the stop criteria of the optimization

The aim of the harness Refinement is to find a better (compared with the preliminary harness definition) and more feasible harness definition in limited optimization loops. The

optimization will stop if the design is converged or the maximum allowed optimization loop is reached. Then the feasibility of the routing result is checked according to the output of the constraint functions. If the result is feasible, then the 3D routing stops. The infeasible result is mainly caused by two reasons: 1) the optimizer is not able to find an existing feasible solution and/or 2) a feasible solution does not exist in terms of the design specifications. The first reason is caused by the limitations of this optimization method and the second reason is caused by the improper problem definition, namely the harness routing problem does not have a feasible solution. In these two cases, the design engineer needs to take over the automatic routing process and a manual design needs to be implemented.

6.4 Implementation of the 3D-harness Refinement tool

The harness Refinement tool consists of three components, namely: 1) the optimizer, 2) the harness geometric modelling module, and 3) the harness analysis module. These three components are integrated into a software platform that enables an entire optimization workflow. The commercial GPS optimizer is available in Matlab[67]. The harness geometric modelling module (Harness MMG) and analysis tools have been developed and detailed in Chapter 5. The development of the platform shown in Figure 6-19 is the main task here.

The software platform is a part of the harness design automation software tool and contains the Refinement itself and also the so-called *Preparation* and *Closure* process. The *Preparation* takes over the workflow from Initialization and makes everything ready to start the Refinement. The *Closure* handles the visualization and report generation of the Refinement result.

The entire harness design automation tool is developed with two software applications, the optimization Toolbox of Matlab and GDL, which is the KBE system used in this research. Matlab provides the optimization solvers and GDL takes care of the geometric manipulations and has a powerful programming language ANSI Common Lisp[42] to control the entire design workflow.

In order to achieve the GDL and Matlab collaboration, two interfaces that enable the exchange of the workflow and data between the two applications are built. The first one is a Matlab server. GDL calls the server and sends the preliminary harness definition to this server to start the optimization process. This is considered as the outer loop of the harness Refinement. The second one is a GDL server. This server is independent from the other part of the GDL workflow and is only responsible for the geometric modelling (MMG) and analysis (analysis tools). This part is considered the inner loop of the harness Refinement.

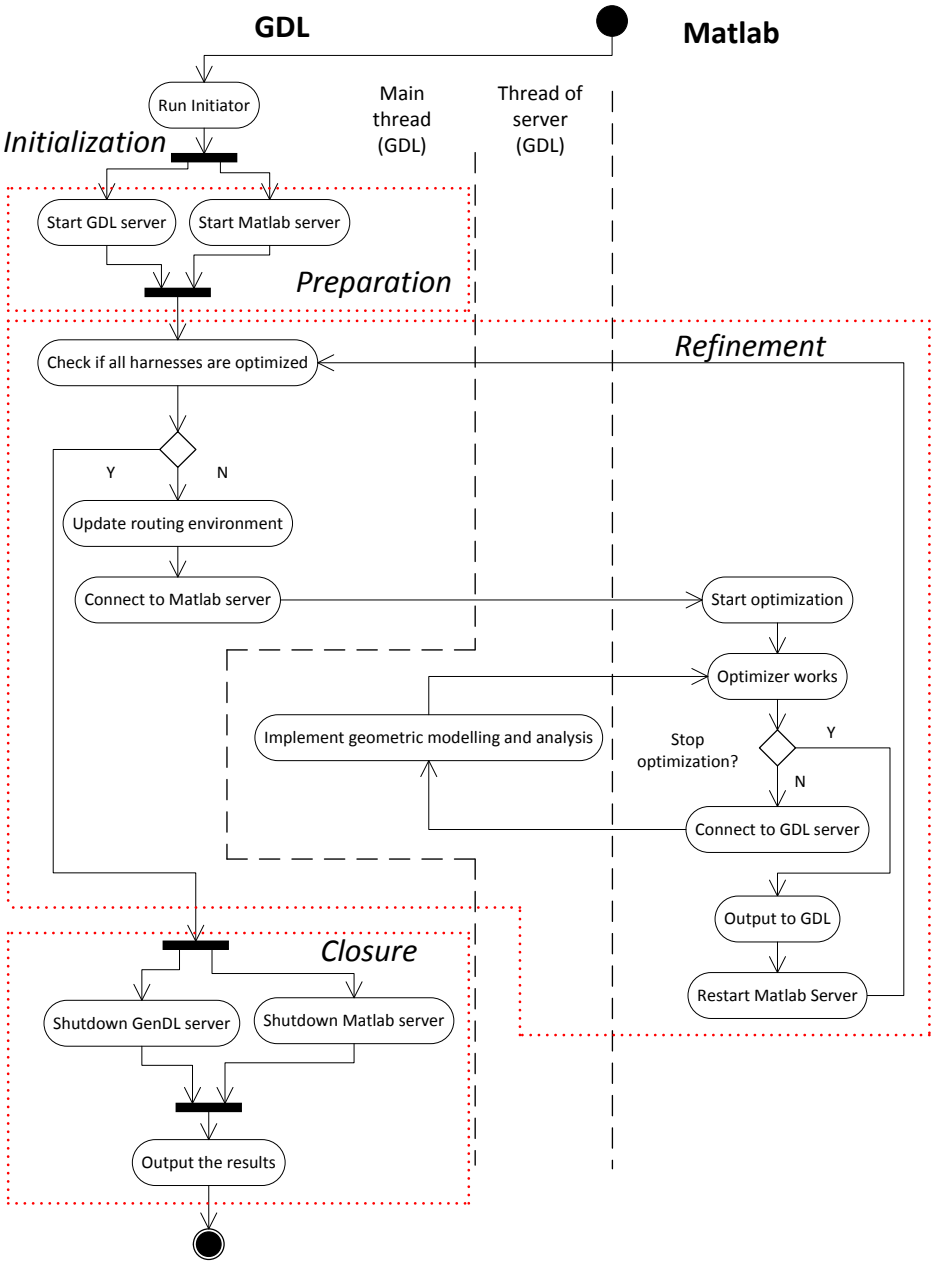
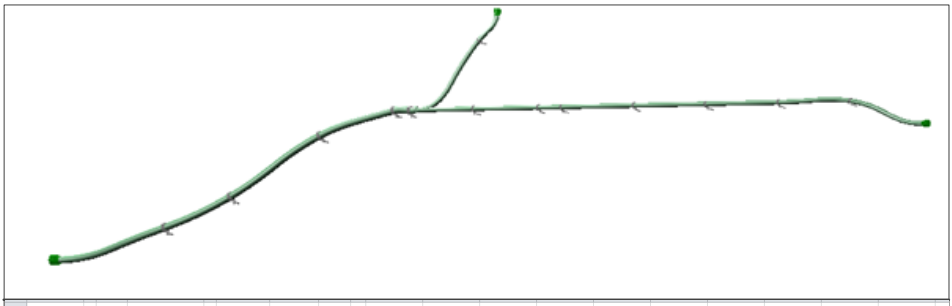


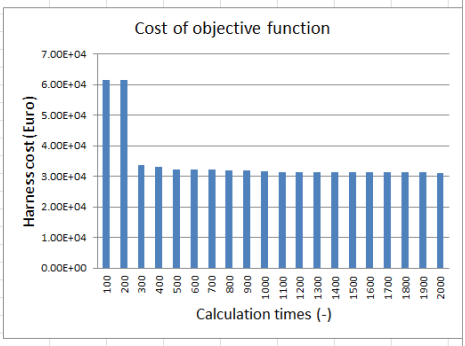
Figure 6-19: Workflow of the harness Optimization software tool in the context of the harness design automation workflow

As shown in Figure 6-19, the Refinement tool starts immediately after the Initialization. It refines all the harnesses in a wiring zone and outputs the routing results when the routing is finished. The output results include the harness geometric model and some data sheets that include the harness information and the log file of the 3D-routing tool. An example output

result of the harness 3D optimization is given below.



1	ncount	=	100	fun_cost_glo	=	6.17E+04	Elapsed	time	is	28.68238	seconds.
2	ncount	=	200	fun_cost_glo	=	6.16E+04	Elapsed	time	is	57.80314	seconds.
3	ncount	=	300	fun_cost_glo	=	3.37E+04	Elapsed	time	is	84.83056	seconds.
4	ncount	=	400	fun_cost_glo	=	3.32E+04	Elapsed	time	is	111.9245	seconds.
5	ncount	=	500	fun_cost_glo	=	3.22E+04	Elapsed	time	is	138.8311	seconds.
6	ncount	=	600	fun_cost_glo	=	3.22E+04	Elapsed	time	is	165.5326	seconds.
7	ncount	=	700	fun_cost_glo	=	3.22E+04	Elapsed	time	is	191.9878	seconds.
8	ncount	=	800	fun_cost_glo	=	3.19E+04	Elapsed	time	is	218.4362	seconds.
9	ncount	=	900	fun_cost_glo	=	3.18E+04	Elapsed	time	is	245.0278	seconds.
10	ncount	=	1000	fun_cost_glo	=	3.15E+04	Elapsed	time	is	271.2302	seconds.
11	ncount	=	1100	fun_cost_glo	=	3.14E+04	Elapsed	time	is	298.0667	seconds.
12	ncount	=	1200	fun_cost_glo	=	3.14E+04	Elapsed	time	is	324.4163	seconds.
13	ncount	=	1300	fun_cost_glo	=	3.13E+04	Elapsed	time	is	350.5481	seconds.
14	ncount	=	1400	fun_cost_glo	=	3.12E+04	Elapsed	time	is	376.9012	seconds.
15	ncount	=	1500	fun_cost_glo	=	3.12E+04	Elapsed	time	is	403.0295	seconds.
16	ncount	=	1600	fun_cost_glo	=	3.12E+04	Elapsed	time	is	429.4518	seconds.
17	ncount	=	1700	fun_cost_glo	=	3.12E+04	Elapsed	time	is	456.1041	seconds.
18	ncount	=	1800	fun_cost_glo	=	3.12E+04	Elapsed	time	is	481.9657	seconds.
19	ncount	=	1900	fun_cost_glo	=	3.12E+04	Elapsed	time	is	507.56	seconds.
20	ncount	=	2000	fun_cost_glo	=	3.10E+04	Elapsed	time	is	536.8802	seconds.



Harness cost = 30954 (euro) , collision = 0 , bend radius violation = 0 , clamp violation= 0

Figure 6-20: A geometric model (top) and calculation log (bottom) of a harness Refinement

Chapter 7 3D-routing case studies

Aiming to validate the functionality of the 3D-routing tool and to further prove the feasibility of the proposed design approach, two routing cases are presented in this chapter.

In Section 7.1 two routing environments are introduced. Then, the 3D routing is carried out and the routing results are presented in Section 7.2.

7.1 Definition of the routing environment

In order to implement the case studies, first the routing environments are defined. The two routing environments studied here are 1) the fuselage sections and 2) the test boxes. The routing environments include not only geometric models but also environmental information, such as hot zones and flammable zones. The reserved space which the harnesses are preferred to go through is also included. In both cases, the geometric models are generated manually by means of commercial CAD software and then saved as STEP files, so as to emulate the actual application scenario of this HDEE.

7.1.1 Introduction of the fuselage sections

In practice, 3D routing is implemented per wiring zone to generate zone EWISs. The zone EWISs connect with each other to form a connected aircraft EWIS.

The first routing environment consists of two adjacent fuselage sections. The two fuselage sections, called fuselage-front and fuselage-rear, are illustrated in Figure 7-1. The cross section of both parts is a circle whose diameter is 5020 mm. The length of both sections is 6000 mm.

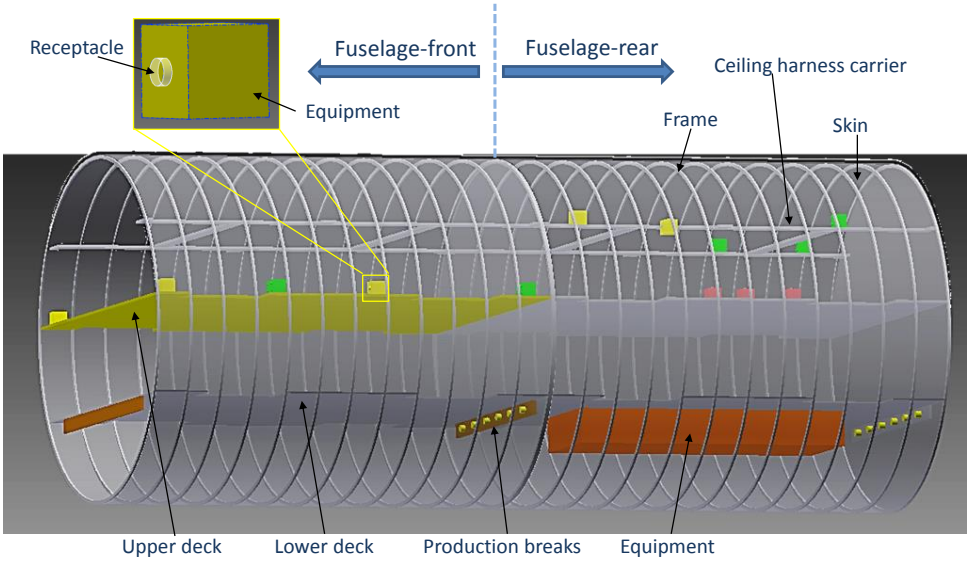


Figure 7-1: Illustration of the two fuselage sections

These sections are considered as two wiring zones. The reason to include two sections is to demonstrate that the 3D-routing tool is able to route per zone and is able to combine two zone EWISs.

Multiple harnesses will be routed in each fuselage section respectively. The fuselage sections are considered as a representative complex routing environment since they include 1) the geometric obstacles that harnesses should avoid; 2) the attachable structures harnesses should be fixed on; 3) the free space where harnesses can go; 4) the reserved space where harnesses are preferred to go through; and 5) the electronic equipment (including receptacles) harnesses need to connect with.

Both sections are assembly models consisting of different geometric components, such as the skin, upper deck, lower deck, frame, ceiling harness carrier, and some pieces of equipment. Two reserved spaces are located underneath the lower deck of the fuselage-rear. They are defined during the harness design process. Details of the spaces will be presented when routing harnesses in them. In these sections, harnesses are allowed to be fixed on the frames, the ceiling wire harness carrier, and the lower deck. Harnesses are not allowed to be fixed on the skin, upper deck, or the equipment.

During the 3D routing, the design rules presented in Sub-section 2.3.2 need to be satisfied. The parameters of the related design rules and their values used in this test case are listed in Table 7-1.

Table 7-1: Routing parameters for the fuselage sections

Minimum allowed fixing distance	50 mm
Maximum allowed clamping distance	609.6 mm
Bend-radius ratio to bundle diameter	6
Bend-radius ratio to wire diameter	10
Reserved space coefficient	0.5
Clearance between harness and moving parts	N mm[1]

7.1.2 Introduction of the test boxes

The second routing environment is composed of two boxes, shown in Figure 7-2. The first box includes 1) an outer shell where the harness can be fixed, 2) a curved surface to demonstrate that this routing tool can route a harness not only on planes but also on curved surfaces, 3) a small box to simulate geometric obstacles, and 4) a hot zone and a flammable zone that influence the covering cost and the clamp cost respectively. The second box includes 1) an outer shell where the harness can be fixed and 2) a geometric model of a dynamic envelope representing all the possible positions of a moving part.

[1] Company confidential data

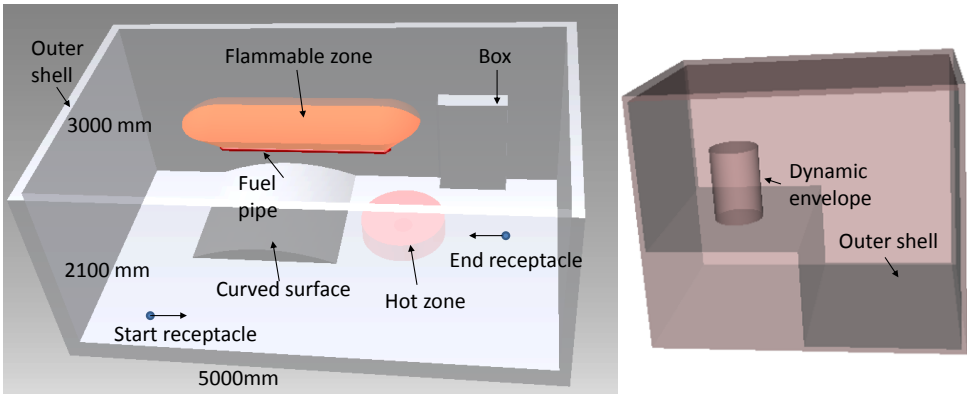


Figure 7-2: Illustration of the test boxes

The hot zone in the first box caused by a heating source is represented by a cylinder. The parameters of this heating source are given in Figure 7-3. With the given parameters, the geometric shape of the hot zone shown in Figure 7-2, and the coefficient e_{cover} determining the covering cost of the harnesses are calculated with the methods presented in Appendix C.

```
(:centre-point #(3240.0 -150.0 -1000.0)
 :source-radius 100 mm
 :source-height 20 mm
 :temperature 300 C°
 :normal-direction #(0.0 0.0 1.0)
 )
```

Figure 7-3: Parameters of the hot zone

The flammable zone is the area above the fuel pipe. The parameters defining the flammable zone in this test box are given in Figure 7-4. With these parameters, the geometric shape of the flammable zone presented in Figure 7-2 is calculated using the method presented in Appendix D.

```
((:radius 609.6 mm
 :gravity-dir #(0.0 1.0 0.0)
 :line (#(1000.0 -500.0 -100.0) #(3000.0 -500.0 -100.0))
 :pipe-radius 5.08 mm
))
```

Figure 7-4: Parameters of the flammable zone

In these test boxes, multiple independent routing tasks will be carried out. The equipment and receptacles that are the start/end points of harnesses are not modelled in these boxes. Instead, these points and the direction vectors attached to these points are given directly to each routing task. Examples of a start and an end receptacle are presented in Figure 7-2. The routing parameters used in this test case are the same as for the fuselage sections (Table 7-1).

7.1.3 Responsibilities of the two routing environments

The 3D routing will be implemented in the two routing environments. However, not all the capabilities of the tool will be demonstrated in both. For instance, the capability to handle hazardous zones is not presented in test case (1) since no hazardous zones are included here. Instead, the capability that the tool is able to route multiple harnesses in a representative complex routing environment is demonstrated. The capability to handle various design constraints is highlighted with test case (2), because this routing environment is concise and the satisfaction of the design rules is easier to present. The detailed functionality check matrix of the two routing cases is presented in Table 7-2.

Table 7-2: Functionality check matrix of the two routing cases

		(1) Fuselage sections		(2) Boxes	
		Included	Detailed	Included	Detailed
Design constraints	Bend radius	Y	N	Y	Y
	Geometric collision	Y	Y	Y	Y
	Clamping distance	Y	N	Y	Y
	Fixing distance	Y	N	Y	Y
	(Non-)fixable structure	Y	Y	Y	N
	Harness separation (EMI)	N	N	Y	Y
	Clearance between harnesses and moving parts	N	N	Y	Y
Hazardous areas (soft constraints)	Flammable zone	N	N	Y	Y
	Hot zone	N	N	Y	Y
Automation	Automatic routing	Y	Y	Y	Y
	Automatic updating	N	N	Y	Y
Space reservation		Y	Y	N	N
Integration of different zone EWISs		Y	Y	N	N

- Included: Y/N- problem will/will not be handled in this case
- Detailed: Y/N –solution of the problem will/will not be discussed

7.2 Implementation of the case studies

In this section, the 3D-routing of the two routing cases will be presented.

7.2.1 The case study in the fuselage sections

In order to emulate the current manual design process, routing in two fuselage sections will be implemented respectively and the routing results of both sections will be assembled.

7.2.1.1 Harness routing in the fuselage-front section

Seven harnesses will be routed inside the fuselage-front to connect between the receptacles of the equipment and production breaks, which are the interface between two zones and is defined as position and direction vectors. The electrical definitions of these harnesses are given in Appendix I and their topology structures are shown in Figure 7-5.

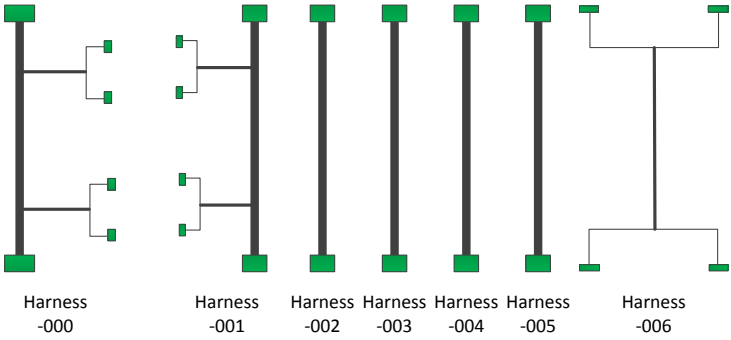


Figure 7-5: Topology structure of the wire harnesses routed in the fuselage-front section

In the Initialization step carried by the HDEE, the harnesses are routed one by one. Since the routing environment, the harness definition, and the design specifications are simplified (see Chapter 4), the feasibility of the routing results is not guaranteed. The Initialization result shown in Figure 7-6 clearly denotes the violation of some design rules, such as the bend radius (indicated by the red circles). These violations will be addressed by the subsequent Refinement step.

The Refinement of the 7 harnesses is also implemented one by one. The simplification of the routing problem is no longer used. Instead, the actual fuselage model will be adopted and the optimizer can move wire harnesses freely in it to explore different harness configurations. The violation check of the design rules and the calculation of the harness costs are carried out by the actual geometric model involved analysis tools. This guarantees that the feasible results generated by this tool are also feasible in terms of the actual routing problem.

Although the harness Refinement is implemented one by one, the interferences between the different harnesses, such as the geometric collision discussed here and the separation requirement discussed in Sub-section 7.2.2, are handled. The previously-routed harness will be treated as a geometric obstacle for the harness currently being routed. The geometric collision will be detected by the Geometric Collision Analysis Tool (GCAT) and solved by the optimizer. When the Refinement is finished, the feasible harness will not have interferences with others.

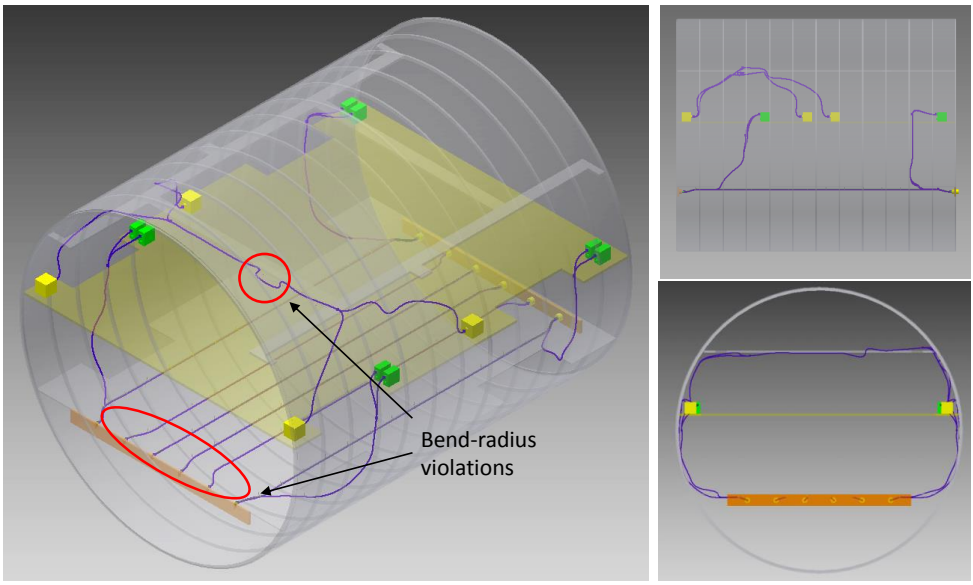


Figure 7-6: Initialization result of the wire harnesses inside the fuselage-front section

The routing results can be visualized immediately by the routing tool and/or exported as a STEP file. The harness models defined in the STEP file can be integrated into the routing environment by commercial CAD tools, as shown in Figure 7-7.

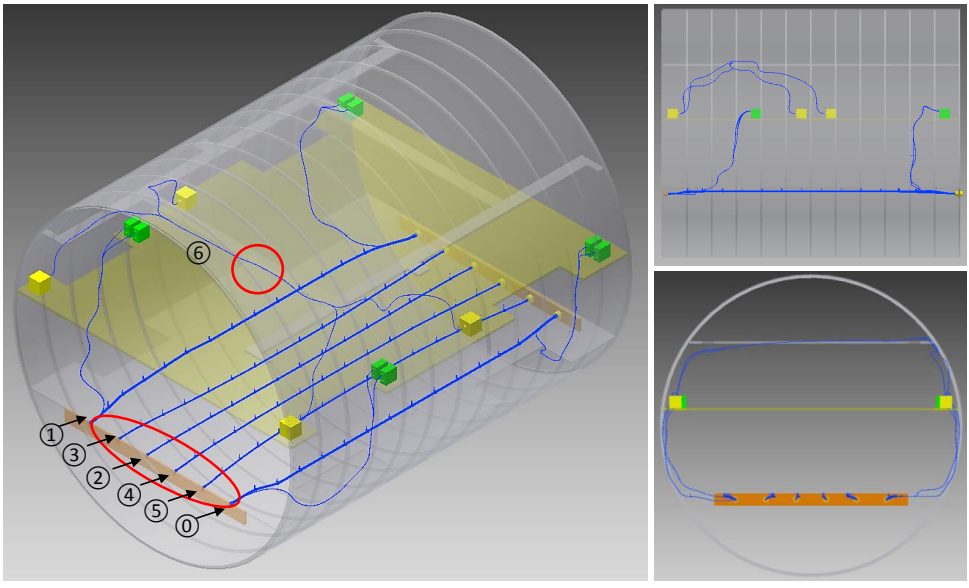


Figure 7-7: Optimization result of the wire harnesses inside the fuselage-front section

When comparing the optimized harnesses with the Initialization results, some improvements can be found. The harness sections inside the red circles in Figure 7-6 have sharp turns due to the limitations of the road map. In the same sections (circled in Figure 7-7), the bend-radius

violations are eliminated. In addition to the bend-radius violation, the optimizer actually also eliminates the violations of other design rules. This process is illustrated by the decrease of the objective function $\theta(x)$ with iterations of the optimization, in Figure 7-8.

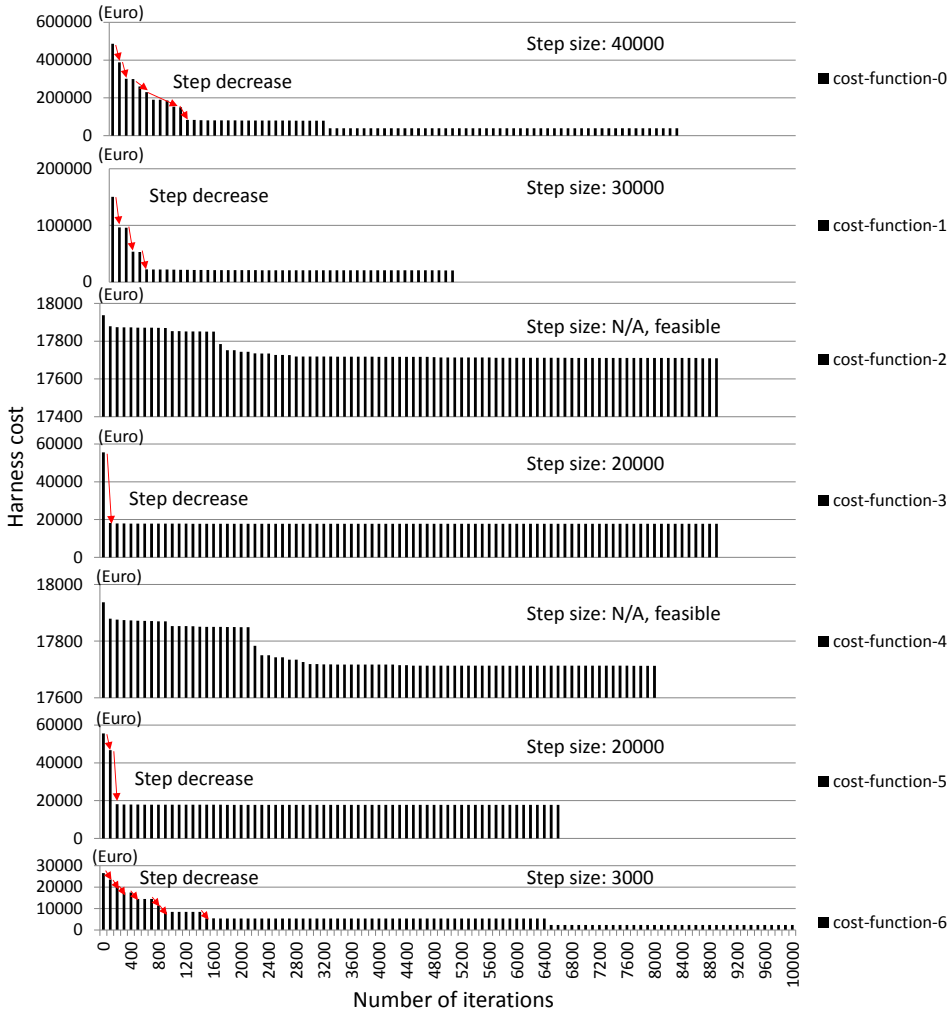


Figure 7-8: Decrease of the objective function $\theta(x)$ with the number of optimization iterations

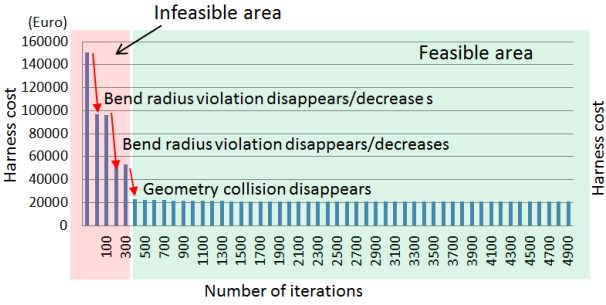
As mentioned in Chapter 6, the new objective function $\theta(x)$ consists of two parts: the actual objective function $f(x)$ and the penalty function to account for constraint violations $\sum w_i c_i(x)$. In order to force the optimizer to find a feasible solution first, the value of each penalty function $w_i c_i(x)$ is very high. If one of the violation is eliminated, the value of $\theta(x)$ will have a step decrease, illustrated by the red arrows.

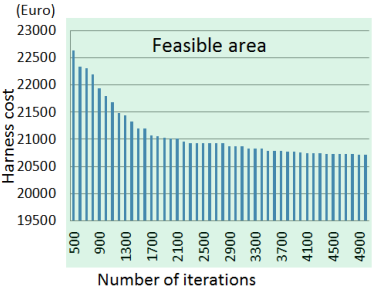
The step decrease can be found in harnesses 0, 1, 3, 5, and 6, where violations of design rule(s) exist at the start of the optimization. The size of the step actually is the weight of the constraint functions (see Sub-section 6.3.3). During the optimization process, the optimizer

adjusts the design variables to eliminate the violations gradually. Each step decrease means the elimination of a constraint violation.

When all the constraint violations are eliminated, the objective function $\theta(x)$ will turn to the original objective function $f(x)$. The optimizer continues to modify the position of the clamps and breakouts for a better result until one of the stop criteria is met. Because the penalty of the constraint violation is very high, the harness in the feasible area will not be moved back to the infeasible area anymore.

In Figure 7-8, the overview of the harness cost decreases is presented. However, details, such as the change of the original objective function $f(x)$ (i.e. the cost function in the feasible area) and the meaning of each step decrease, are not presented yet. In order to conveniently show these details, the result of harness-1 (the second harness) is extracted from the previous figure. Its objective function in both the infeasible and feasible areas is given in Figure 7-9.





Times	θ(x)	f(x)	Geo Collision	Bend Radius Violation	Clamp Violations
0	150543	20891.76	30000	96143.16	3508.23
100	96627.8	20801.47	30000	42226.03	3600.28
200	96013.3	20769.05	30000	42226.03	3018.19
300	53599.5	20651.76	30000	0	2947.69
400	53241.8	20692.62	30000	0	2549.17
500	22627.3	20660.2	0	0	1967.08
600	22330.6	20657.46	0	0	1673.12
...	...	...	...	...	...
4900	20721.1	20535.53	0	0	185.56
5000	20721.1	20535.53	0	0	185.56

Figure 7-9: Decrease of the objective function $\theta(x)$ (top-left) and $f(x)$ (top-right) of harness-1 and their log data (bottom)

In this figure, the left-hand side shows the cost of the objective function $\theta(x)$ in both the infeasible and feasible areas. At the start of the optimization, the actual harness cost $f(x)$ is $20891.76 Euro$ [1]. The weight of the constraint functions (i.e. the step size) is then set to 30000, according to the constraint weight calculation method presented in Sub-section 6.3.3. At the start of the optimization, the geometric collision plenty $wc(x)_{collision} = 30000$ and bend radius violation plenty $wc(x)_{bending} = 96143$, as shown in the bottom table in Figure 7-9. According to these numbers, we can tell that the harness-1 has 1 ($=30000/30000$) geometric collision and 3 ($\approx 96143/30000$) bend-radius violations. The small number (3508) of clamp violations is caused by the requirement to place clamps at equal distances. This is just a preference and not

[1] The number means the cost of the harness. It is calculated based on the assumed data. Its unit can be any currency. The consistency of the unit is kept for the routing of all the harnesses.

a violation of any design rule.

According to the constraint definitions presented in Sub-section 6.3.2, the function value of some constraints (e.g. the geometric collision) is N[1] times the weight in $\theta(x)$. Decreasing by the step size that equals one or more times the weight value means the elimination of one or more constraint violation, such as one or more geometric collision. The function value of other constraints (e.g. bend-radius violation) that changes continuously does not always happen to be N times in $\theta(x)$ (e.g. $wc(x)_{bending} = 96143 = 3 \times 30000 + 6143$). A decrease whose step size is smaller than one times the weight in these constraint functions means some improvements rather than elimination of a constraint violation. By the end if $wc(x) = 0$, the violation of this constraint is considered to be eliminated.

During the optimization of harness-1, after maximum 100 iterations, 1 bend-radius violation is eliminated and another bend-radius violation is improving, since the $\theta(x)$ (or the Bend-Radius Violation function) decreases by around 54000 (i.e. 30000 elimination + 24000 improvement). After 200 iterations, the 3 bend-radius violations are all eliminated. After 400 iterations, the geometric collision is also eliminated. From this moment, the harness becomes feasible. The optimizer still adjusts the clamps aiming for a better (cheaper) harness until the optimization is converged. This process is clearly illustrated in the bottom log file in Figure 7-9.

7.2.1.2 Harness routing in the fuselage-rear section

Six harnesses will be routed in this fuselage section. The detailed definitions of the harnesses are given in Appendix I and their topology structures are shown in Figure 7-10. Routing harnesses in this fuselage section is similar to the previous section except three differences that 1) a geometric obstacle is added to block the shortest path of some harnesses; 2) one of the harnesses routed here is very complex; and 3) reserved spaces are considered.

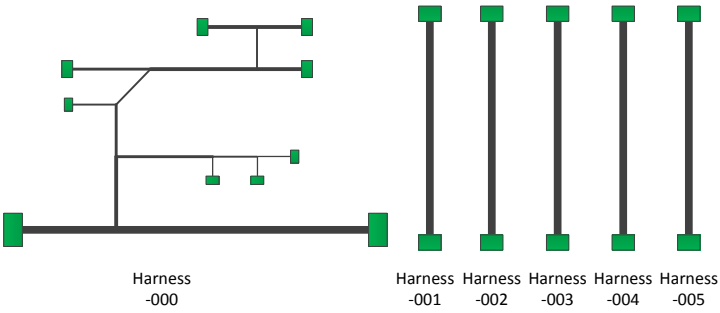


Figure 7-10: Topology structure of the wire harnesses routed in the fuselage-rear section

In the fuselage-rear section, a piece of equipment is placed under the deck to force four harnesses in the middle of the fuselage to take a detour. In the Initialization, all the harnesses are independently routed in the same road map and interference with other harnesses is not taken into account. Therefore, as shown in Figure 7-11, harnesses 2 and 3 share some road map edges, and the same is true for harnesses 4 and 5. During the Refinement step, this geometric collision will be solved. The optimizer can detect a collision between different harnesses and between harnesses and the geometric obstacle and try to eliminate both of

[1] N is an integer and equal to number of violations/collisions.

them. Meanwhile it also achieves the optimum path. The Refinement results are illustrated in Figure 7-12, where harnesses 2, 3, 4, and 5 are routed just next to the geometric obstacle to avoid the geometric collision and also to achieve the minimum length.

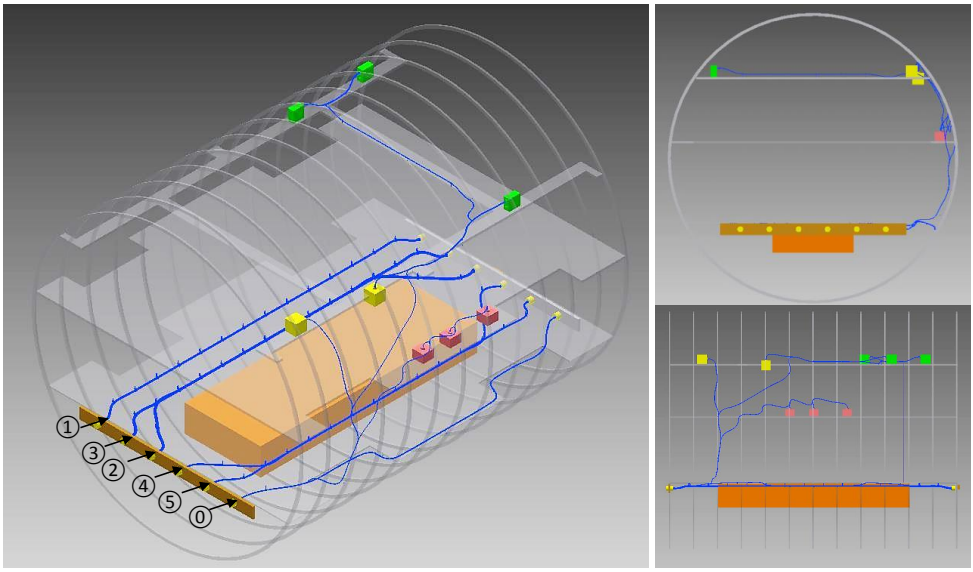


Figure 7-11: Initialization results of wire harnesses inside the fuselage-rear section

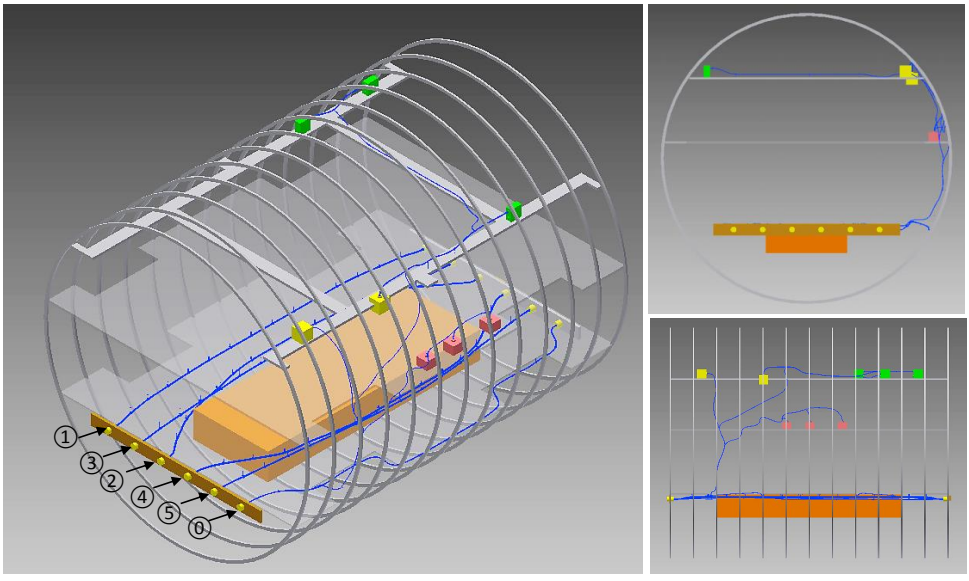


Figure 7-12: Refinement results of wire harnesses inside in the fuselage-rear section

The harness-000 is the most complex harness in the case studies. It has 10 connectors, 17 branches, and 8 breakouts. The equipment which has the receptacles that this harness is going to connect with is widely distributed in this routing environment (see Figure 7-11). This increases the difficulty of the routing. The complexity of the harness influences both the

Initialization and the Refinement. The Initialization result of a simple harness such as harness-001 is very likely to be violation-free of constraints. However, this complex harness has 5 geometric collisions, 6 bend-radius violations, and 1 clamp-distance violation. The Refinement needs more iteration to eliminate these design constraint violations. Although more iteration is needed, this tool is still able to find a feasible path for such a complex harness. The elimination of the design constraints is illustrated in Figure 7-13.

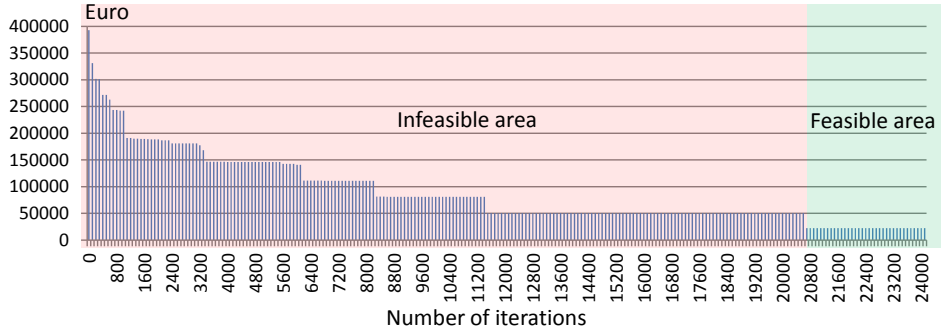


Figure 7-13: Decrease of the objective function of harness-000

In this fuselage section, the reserved space is also considered. In principle, the reserved space definition should be given as inputs. However, it is not available yet. Therefore a simple reserved space is added to this section and the different harness paths generated with and without considering this reserved space are presented, as shown in Figure 7-14.

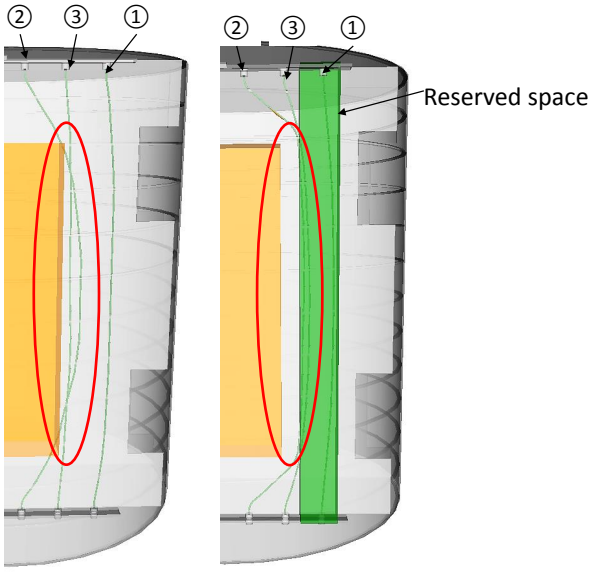


Figure 7-14: Harness routing results without (left) and with (right) considering the reserved space

It is preferable to route harnesses in the reserved space but they are allowed to be placed outside the space if it is not possible to completely place them inside. As shown in Figure 7-14, the middle parts of harness 2 and harness 3 are placed in the reserved space since the

costs of these harnesses are lower in these cases. At the two ends, the harnesses are routed outside the reserved space because it is not possible to route the harnesses inside.

7.2.1.3 Assembly of harnesses located in different wiring zones

The large routing task in an aircraft is broken down into several wiring zone-based sub-tasks. When the sub-tasks are finished, the harnesses located in different wiring zones, such as the fuselage-front and fuselage-rear, need to connect with each other. By using pre-defined production breaks (i.e. the interface between two zones and is defined as position and direction vectors), the harnesses generated independently in different zones will connect with each other seamlessly, as shown in Figure 7-15.

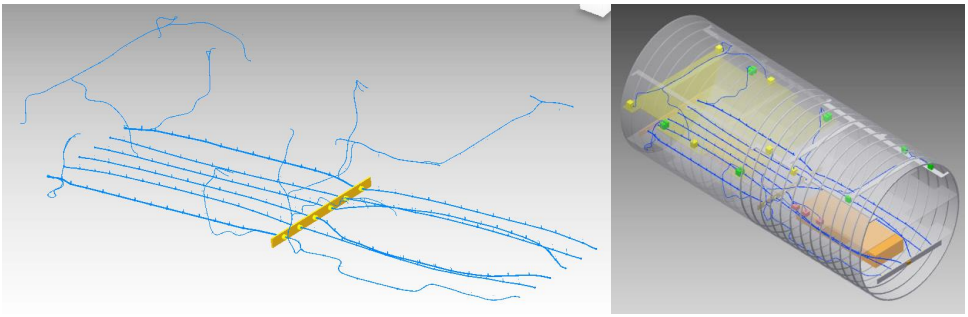


Figure 7-15: View of the assembled wire harnesses in two wiring zones, independently (left) and in the fuselage routing environment (right)

7.2.2 The case study in the test boxes

The capabilities of the routing tool demonstrated in this test case include handling the design constraints and grey zones, and updating the harness geometric models when the routing environment or the harness electrical definition is updated.

7.2.2.1 Satisfaction of the design constraints

In this part the capability of the 3D-routing tool to handle some typical design rules presented in Sub-section 2.3.2 is demonstrated.

- **Satisfaction of the bend-radius requirement**

The shortest path between the start and end points is the line segment. During the harness routing process, the optimizer tries to attain the shortest path. In practice, the shortest path is not always a feasible solution considering design rules such as the bend-radius violation.

As shown in the top of Figure 7-16, due to the direction at the harness start point, the path generated without considering the bend-radius violation obviously violates the design rule. The turn at the start point is so sharp that even the lofted surface used to represent the bundle cannot be generated correctly.

This problem is handled by the routing tool when the bend-radius violation is considered, in both the Initialization and Refinement step. Considering the bend-radius violation, the Initialization step will generate a smoother preliminary harness definition. Then, the Refinement step starts. The Harness Bend Radius Analysis Tool will check whether the bend radius is violated accurately and any bend-radius violation will lead to a very high penalty in the objective function $\theta(x)$. Therefore, the optimizer will adjust the clamps to new positions to avoid this violation. As a consequence, the smooth harness shown in the bottom of Figure 7-16 is achieved.

Some design constraints such as geometric collision between harnesses does not need to be considered in the Initialization step. However, the Initialization is necessary for achieving a bend-radius violation-free result because of two reasons. Firstly, the Refinement step is not able to handle the preliminary harness definitions that have sharp turns such as the one shown in the top of Figure 7-16. Secondly, a smoother path is generally longer than the path that has a sharp turn and therefore needs more clamps for support. The number of clamps (i.e. design variables) cannot be changed in the Refinement step. Therefore a preliminary harness definition which is smooth and has enough clamps needs to be generated as an input for the Refinement.

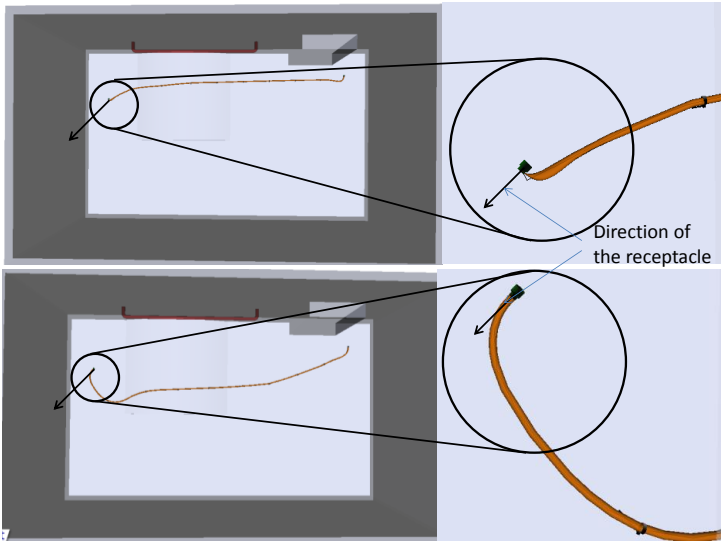


Figure 7-16: The routing result without (top) and with (bottom) considering the bend-radius violation

- **Satisfaction of the geometric collision between branches**

In the Initialization step, due to sharing the road map, different branches of the same harness may have geometric collisions. These collisions need to be solved in the Refinement step. The example of the Initialization result having a geometric collision is shown in the top of Figure 7-17. The penalty of this geometric collision is very high in the objective function $\theta(x)$. Therefore the optimizer will try to eliminate this collision by moving the clamps and breakouts first and then search for the minimum of $f(x)$. The optimization result that is geometric-collision-free is presented in the bottom of Figure 7-17. Clearly, the geometric collision between the two branches is eliminated by moving some clamps of a branch up to make this branch fly over another one.

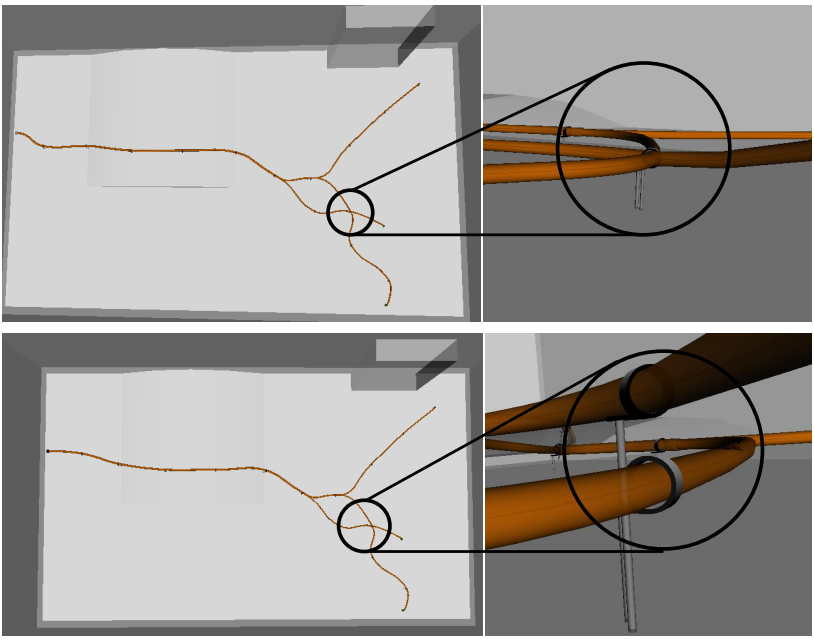


Figure 7-17: Solution of the geometric collision between branches; top: result of the Initialization; bottom: result of the Refinement

The elimination of the constraint violations can also be seen from the log file of $\theta(x)$, as shown in Figure 7-18. The preliminary configuration of this harness actually has a geometric collision between the branches and a bend-radius violation. During the routing process, the cost of $\theta(x)$ drops significantly twice within the first 200 loops. These two big steps are the elimination of the geometric collision and the bend radius violation. From this moment, the harness becomes feasible (i.e. at point A). The optimizer continues to adjust the clamps and breakouts to decrease the harness cost until the design is converged (at point B).

Actually the 3D routing can stop at point A if the 3D routing only wants to find a feasible harness route. The optimization can also be carried out further to achieve a lower cost harness such as point B but with extra calculation time. Since the design time is also considered as the cost to a company, it is a trade-off process to select from 1) a lower cost (price) harness but longer calculation time or 2) a higher cost (price) harness but shorter calculation time. Provided the calculation time can be measured accurately and monetized, this trade-off can be solved as a multi-objective optimization which considers both the calculation time and harness cost. Of course, both the trade-off and the multi-objective optimization are out of the boundary of the research presented in this dissertation.

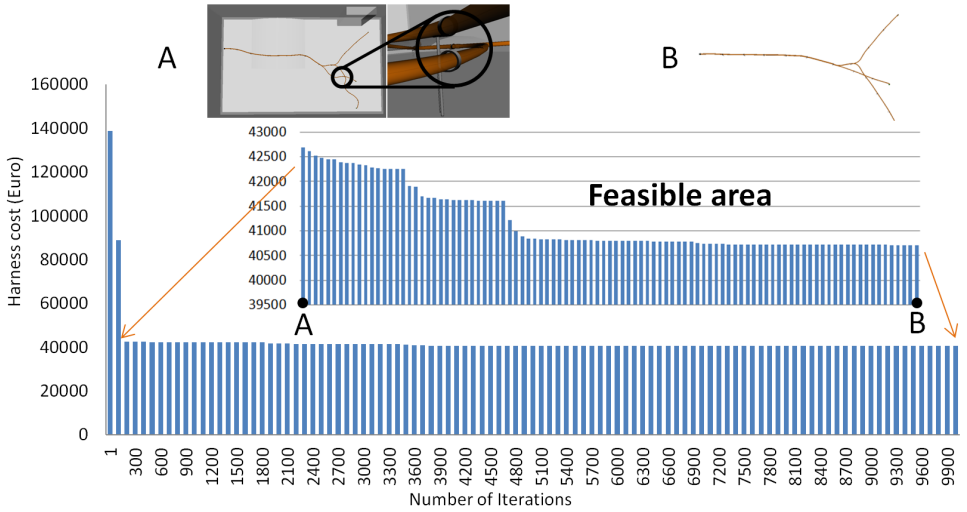


Figure 7-18: Illustration of the log file of the objective function $\theta(x)$

- **Satisfaction of the clamping-distance and fixing-distance constraints**

Wire harnesses need to be supported with a suitable clamping distance and fixing distance.

The actual fixing distance should be bigger than the allowed fixing distance and the clamping distance should be smaller than a certain value. In the Initialization, the fixing distance is handled by keeping a certain offset distance when generating the offset surfaces for placing the harness. The clamping distance is handled when generating the road map. In most cases, both rules are already satisfied in the preliminary harness definition.

During the Refinement, the fixing distance and clamping distance are also checked since the clamps and breakouts will be moved at each optimization loop. Since the violation of both constraints will lead to a high penalty function in $\theta(x)$, the optimizer will always try to avoid their violation.

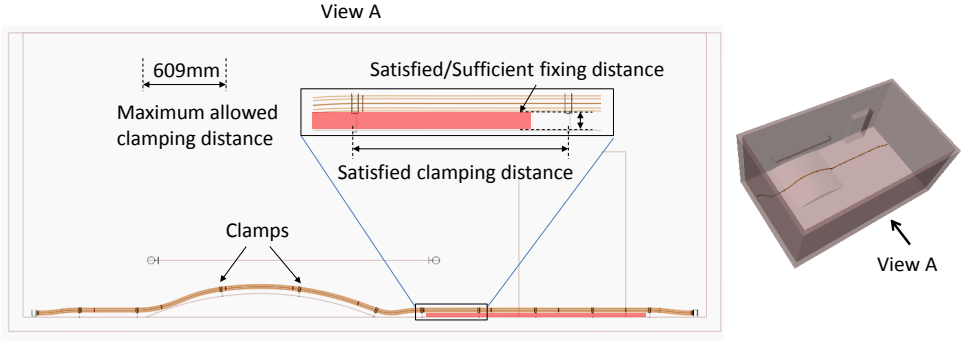


Figure 7-19: Satisfaction of the clamping distance and fixing distance

A routing result that satisfies both the fixing and clamping distance is presented in Figure 7-19, where none of the clamping distances are bigger than 609 mm (i.e. the maximum allowed clamping distance). In addition, the sufficient fixing distance is highlighted by the red block located between the harness and the geometric structure.

- **Satisfaction of the clearance rule between harnesses and moving parts**

Harnesses need to be $N\,mm$ away from the dynamic envelopes of moving parts. As shown in Figure 7-2, a dynamic envelope is a predefined geometric model. This model and the required distance $N\,mm$ are given as inputs to the 3D routing.

In the Initialization step, the dynamic envelope is considered as a geometric component and therefore it is avoided by the harness, as shown in Figure 7-20. In the Refinement step, the optimizer tries to move the preliminarily defined harness approaching the dynamic envelope for a shorter path. Meanwhile, the clearance rule is checked in every optimization loop. The violation of this rule will cause a high penalty in the objective function $\theta(x)$ and therefore it is avoided by the optimizer. As shown in Figure 7-20, the refined harness is placed just next to the dynamic envelope for a shorter path, but still with enough clearance to satisfy the design rule.

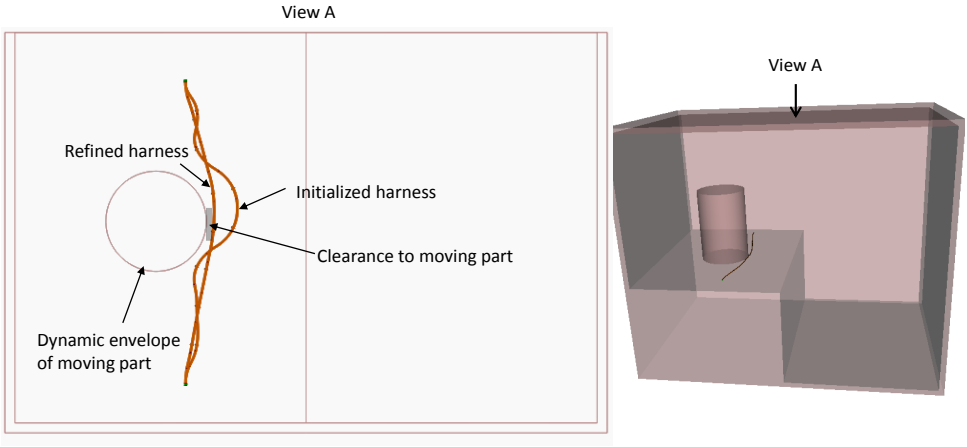


Figure 7-20: Harness Initialization and Refinement results considering the clearance rule

- **Satisfaction of the separation requirements between harnesses**

A harness needs to be routed a certain distance away from other harnesses to satisfy the separation requirements. The required distance depends on their EGS codes, which are given as inputs to the 3D routing.

The harness separation requirement is only considered in the Refinement step. Since the requirement is not handled in the Initialization step, the preliminarily defined harness may violate this rule, such as harness-2 shown in Figure 7-21 (left-hand side).

In this example, harness-1 is the previously routed harness used to demonstrate the satisfaction of the clearance rule, and harness-2 is the harness currently being routed. According to the EGS codes of the two harnesses and the design requirement presented in Table 2-4, the allowed distance between these harnesses is 300 mm. However, the actual minimum distance is just 195 mm. During the Refinement step, the optimizer tries to eliminate this violation by moving harness-2 to the right-hand side, although this increases the total length/cost of the harness. Finally, the optimum result is found. As shown in Figure 7-21 (right-hand side), the optimizer places harness-2 just 300 mm away from harness-1 to satisfy the design rule and also to keep the harness as short as possible.

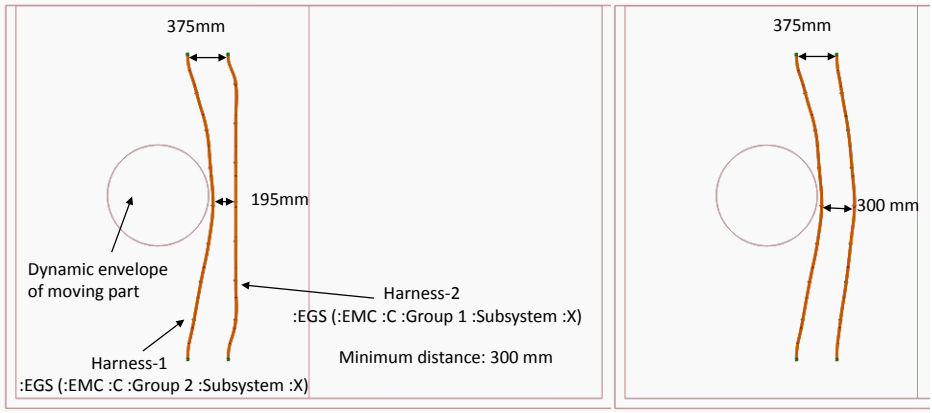


Figure 7-21: Initialization (left) and Refinement (right) results of harness-2

7.2.2.2 Harness routing in the grey areas

Besides the hard constraints, other design rules such as the clamping rule in the flammable zone or the protection rule in the hot zone also need to be satisfied. These rules are referred to as soft constraints since they relate to trade-off studies, i.e. 1) going through flammable zones or hot zones with extra clamps or coverings to satisfy the applicable rules, or 2) taking a detour with a longer bundle without using these components. This trade-off is handled in the objective function of the optimization.

- **Trade-off for a hot zone**

A single start and end harness is routed here. The routing tool considers the hot zone in both the Initialization and Refinement steps. It first finds the optimum path, shown in the left-hand picture of Figure 7-22, in the Initialization step on the basis of the road map definition. Then, according to the Initialization result, the optimizer carries out the optimization for the actual optimum path in the Refinement step. The right-hand picture of Figure 7-22 shows that the optimum result has a detour avoiding the hot zone. Obviously, the tool thinks taking a detour to avoid the use of protective covering is better. In order to validate the correctness of the routing tool, the same harness is routed through the hot zone on purpose. The cost of these two paths, which are 33,499.84 and 49,181.12 Euros, shows that taking a detour to avoid this hot zone is a better solution.

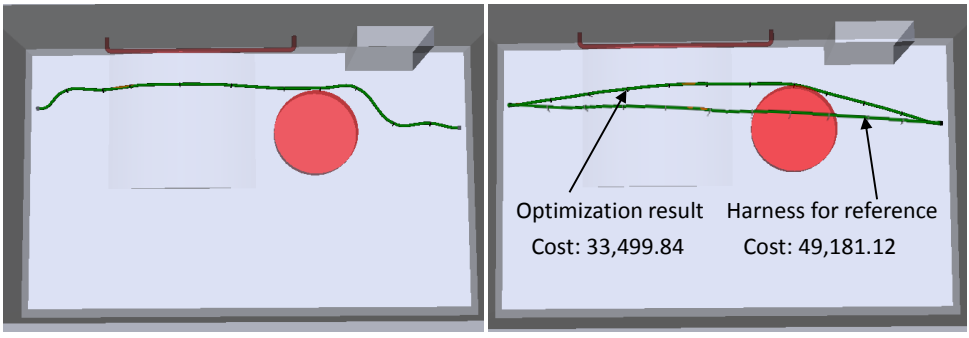


Figure 7-22: Initialization (left) and Refinement (right) results of routing a harness in a hot zone

- **Harness routing in a flammable zone**

A harness will be routed around a flammable zone. A road map which has a variable edge length (i.e. a shorter edge in the flammable zone and a normal edge length outside the zone) will be generated first in the Initialization step. The edge length is determined by the distance between the edge and the flammable pipe, as presented in Appendix D. Then the pathfinding will be implemented to generate a preliminary harness definition. In this definition, the clamping distance inside the flammable zone is satisfied.

Then the Refinement will be implemented. In this phase, the clamping distance of the harness in the zone needs to be carefully checked since the allowed clamping distance in the flammable zone is smaller. The method used to calculate the allowed clamping distance according to the position of the harness and the flammable pipe is detailed in Appendix D. The violation of the clamping distance will lead to a high penalty in the objective function $\theta(x)$; therefore the optimizer always tries to exclude this. Meanwhile, the optimizer also tries to decrease the harness length as much as possible by adjusting the design variables. The results of the Initialization and the Refinement are presented in Figure 7-23.

In this entire routing process, the trade-off study only happens in the Initialization step since the harness can either be routed in the flammable zone with extra clamps but a shorter length or routed outside the zone with extra length but still less clamps. In the Refinement step, since the number of clamps no longer changes, the optimizer tries to decrease the harness length only.

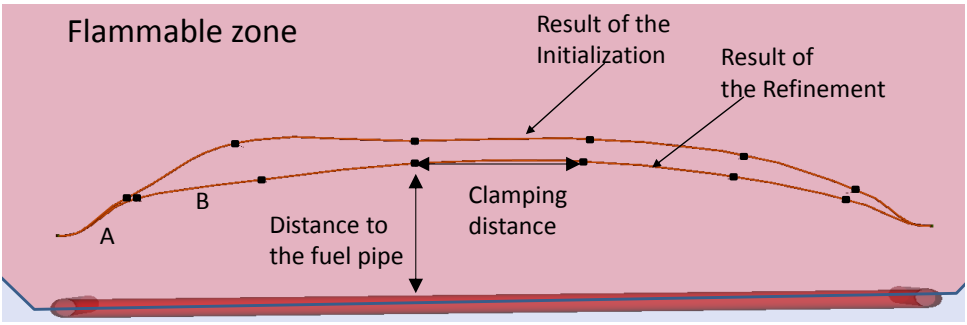


Figure 7-23: Routing result of a harness in the flammable zone

7.2.2.3 Handling the update of routing environments and harness electrical configurations

During the entire harness design process, the harness configuration is changed very often due to the frequent changes of the routing environment and/or the harness electrical definition. The automatic updating of harnesses is addressed by this tool. Two examples are given here to demonstrate that the routing tool is able to generate a new harness when the routing environment and the harness electrical definition are changed.

- **Handling the routing environment update**

The routing environment update is represented by adding a geometric obstacle, as illustrated in Figure 7-24. In both routing environments, the same harness needs to be routed between the same start point and end point.

In the original routing environment, as shown in diagram A of Figure 7-25, the short cut is taken since this harness is geometric collision-free. When the geometric obstacle is added, the entire routing process (i.e. the Initialization and Refinement) needs to be re-run. In the

Initialization step, the road map is updated to prevent the map edges from contacting with the obstacle. Then the preliminary path will be found on the basis of the new road map, as shown in diagram B in Figure 7-25.

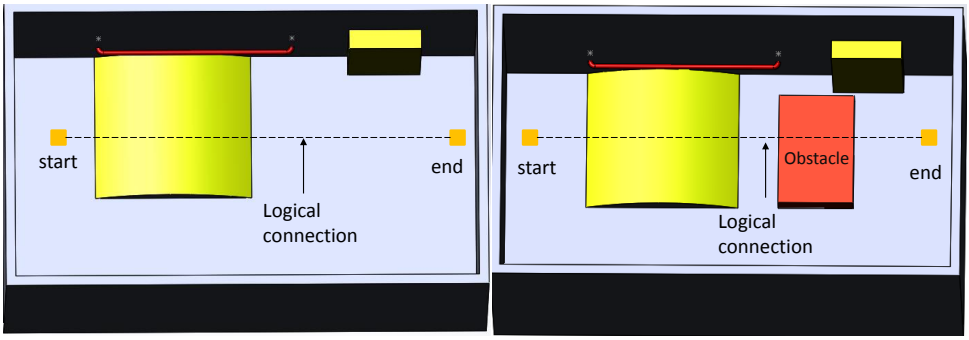


Figure 7-24: Illustration of the original routing environment (left) and the updated one (right)

The Refinement step starts with the preliminary harness definition. The new obstacle will be included in the geometric collision check. The optimizer on the one hand will try to decrease the harness length but on the other hand it will keep the harness away from that obstacle, as shown in diagram C in Figure 7-25.

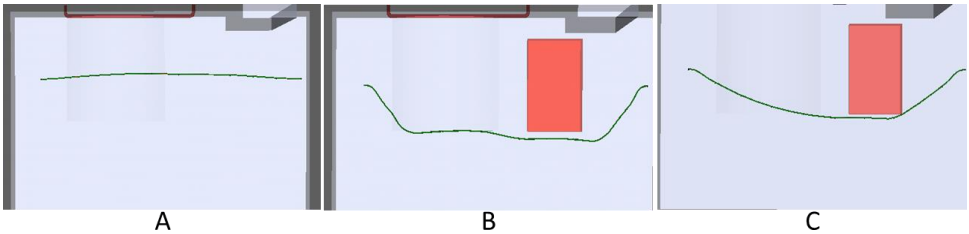


Figure 7-25: Updating process of a wire harness when the routing environment has been changed, (A: original harness; B: updated result after the Initialization; C: updated result after the Optimization)

- **Handling the electrical definition update**

During the harness design process, the harness electrical definition will sometimes be changed. In this test scenario, it is assumed that a connection, as illustrated by the dashed line in Figure 7-26, needs to be added between the two pieces of electrical equipment that connector A and connector B connect with respectively. Therefore, a cable is added to build the connection between them and the two harnesses become one big harness.

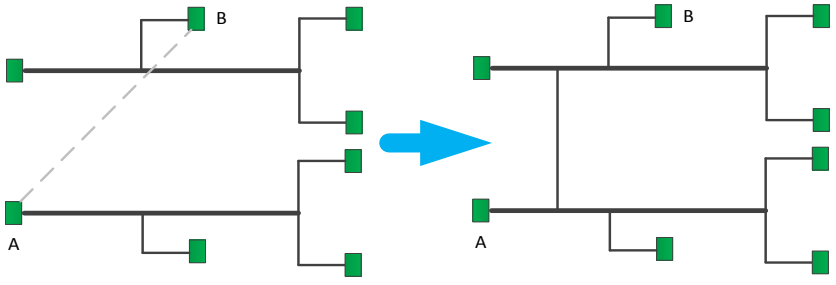


Figure 7-26: Updating the harnesses electrical definition

Due to the update of the electrical definition, the harness geometric model also needs to be updated. The Initialization will generate a preliminary harness definition with the bi-level optimization approach which was proposed dedicatedly to handle the multi-origin/destination problem such as this example. Then the Refinement will be carried out to get a better design. The evolvement of the harness geometric models caused by the evolvement of the electrical definition is shown in Figure 7-27.

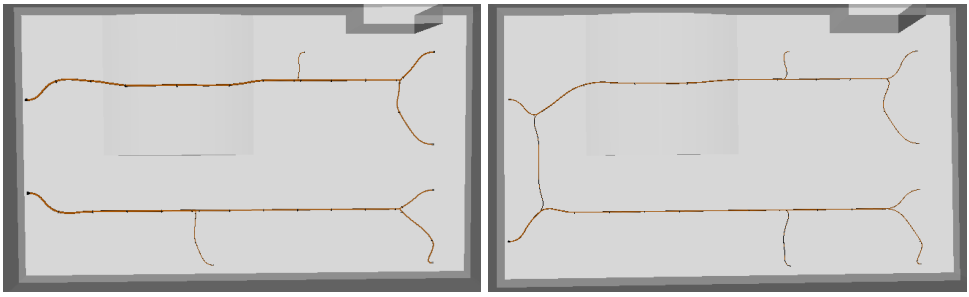
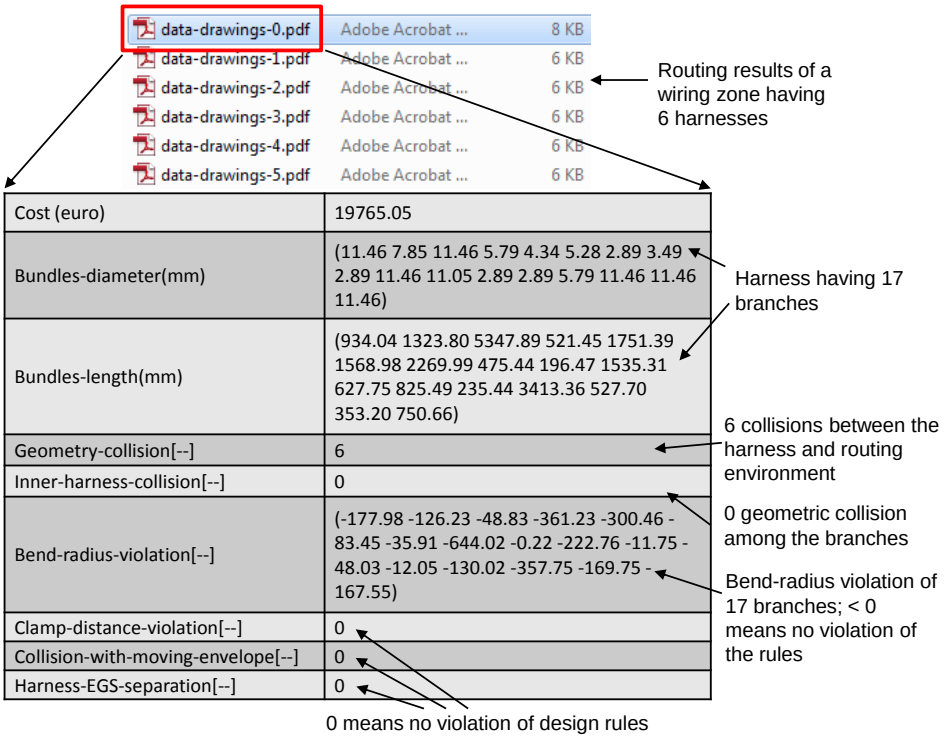


Figure 7-27: Updating the harness geometric model when the electrical definition has been changed; left: the original two harnesses; right: the new harness

7.2.3 Output of the routing results

When the harness routing is finished, the routing results are exported. The results include the previously presented harness geometric models and some report datasheets. The geometric models are exported in a neutral CAD format, such as STEP and IGES files. These files can be read back by commercial CAD tools to support the subsequent manufacturing design and harness installation. The datasheets describe the routing results per wiring zone. The harness cost, length of each bundle, and violation/satisfaction of the design rules are included. An output example is illustrated in Figure 7-28.



Cost (euro)	19765.05
Bundles-diameter(mm)	(11.46 7.85 11.46 5.79 4.34 5.28 2.89 3.49 2.89 11.46 11.05 2.89 2.89 5.79 11.46 11.46 11.46)
Bundles-length(mm)	(934.04 1323.80 5347.89 521.45 1751.39 1568.98 2269.99 475.44 196.47 1535.31 627.75 825.49 235.44 3413.36 527.70 353.20 750.66)
Geometry-collision[--]	6
Inner-harness-collision[--]	0
Bend-radius-violation[--]	(-177.98 -126.23 -48.83 -361.23 -300.46 -83.45 -35.91 -644.02 -0.22 -222.76 -11.75 -48.03 -12.05 -130.02 -357.75 -169.75 -167.55)
Clamp-distance-violation[--]	0
Collision-with-moving-envelope[--]	0
Harness-EGS-separation[--]	0

Figure 7-28: Illustration of a harness pathfinding result

Chapter 8 Conclusions and Recommendations

8.1 Conclusions

The EWIS physical design of the Aircraft Electrical Wiring Interconnection System (EWIS), especially the 3D routing which is considered the most critical and time-consuming phase in the whole physical design, is a challenging process. The design needs to 1) handle the complex EWIS system that needs to interconnect with the increasing number of electronic systems caused by More-Electrical-Aircraft (MEA) and Full-Electrical-Aircraft (FEA); 2) respect many design specifications issued by authorities; and 3) tackle the design changes in the airframe and other systems in a collaborative aircraft design environment. These challenges mean that designers work under high pressure and their design results are error-prone.

Design automation is a solution to these challenges considering that the EWIS physical design is a repetitive process and the EWIS components are mainly selected from catalogues. Automating this design process will release the engineers from hard work, enable them to focus on the creative work, and increase the quality of the design results. However, no or very limited mature solutions which can automate the physical design, especially the 3D routing, have been found in either academic or industrial domains.

The work presented in this dissertation has offered a solution to automate 3D harness routing. The proposed solution has solved the Multi-Origin/Multi-Destination feature, which makes the harness pathfinding unique and different from other pathfinding problems (e.g. car navigation); some essential design rules which have to be satisfied, such as geometric collision-free, bend-radius violation-free, clamping distance, and fixing distance (see Sub-section 2.3.2); and environmental information, including grey areas and the reserved space.

The successful 3D-routing solution has been enabled by a two-step, hybrid optimization strategy.

3D harness routing has been modelled and solved as an optimization problem. The positions of clamps and breakouts are represented by the design variables of the optimization. The harness cost which design engineers want to minimize is represented by the optimization objective function. The various design rules are represented by the constraint functions.

However, this optimization cannot be solved immediately since the number and initial values of the design variables, namely the number and position of clamps, are not included in the inputs of the 3D routing. The electrical definition defines the connectivity among the multiple receptacles. This definition clearly presents the number and gauge of wires among the receptacles and the number of breakouts. However, it does not include the number and position of the clamps or the position of the breakouts. Actually, the number and position of the clamps, as well as the position of breakouts are the output of the harness-routing process. In order to handle this problem, a two-step, hybrid optimization strategy has been devised.

The first step, called Initialization, generates preliminary harness definitions that include the number and initial position of the clamps and breakouts. The second step, called Refinement, is in charge of the refinement of these preliminary harness definitions.

- **Initialization**

During the Initialization step, a road map-based, bi-level optimization method has been used. The routing environment is discretized to generate a road map, whose vertexes are the potential positions for the placement of clamps. Bi-level optimization architecture has been

proposed to handle the multi-origin/destination issue. This architecture decomposes a complex harness into branches and uses the breakouts to maintain the coordination between global-level and local-level optimizations. The global/harness-level optimization is only responsible for moving breakout points to get different breakout configurations. According to the configurations, A* pathfinding algorithms finds the best path going through the road map between the start and the end point without knowing the number or initial value of the clamps, at the local/branch level. Actually, the positions of the road-map vertexes on the found path are the positions of the clamps. Then, the costs of all the branches are sent back to the global optimizer to support the decision making. On the basis of the local calculation results, the global optimizer determines whether the optimization is converged and if necessary it will move the breakouts to generate new configurations for the next iteration until a satisfactory result is reached.

During the generation of the road map and the local pathfinding, the design rules that have to be satisfied in the Refinement step are also considered to generate more promising results for the Refinement.

When the pathfinding is finished, a post-process is carried out to transfer the road map-based result, which consists of some vertexes and edges of the map, to a preliminary harness definition. This definition contains the number and initial values of the clamps and breakouts and is the input for the second optimization step.

- **Refinement**

In the Initialization, the 3D routing is simplified into a road map-based pathfinding. Due to the simplification, there is no guarantee that the optimum result of the Initialization is also the optimum result of the actual pathfinding problem. Actually, even the feasibility cannot be guaranteed.

In the Refinement step, the preliminary harness definition is instantiated into the actual harness geometric model automatically by the parametric geometric modelling module, i.e. Harness Multi-Model Generator. The geometric model is then analysed by the analysis tools involving geometric models to calculate the harness cost and check for violation of the design rules. According to the analysis results, new harness definitions are generated for the next optimization loop until the optimum result is reached.

- **Supporting technologies**

The proposed 3D-routing solution is supported by Multidisciplinary Design Optimization (MDO) and Knowledge-Based Engineering (KBE). MDO technology is responsible for the systematic exploration of the routing space to achieve the feasible and optimum harness design in both the Initialization and Refinement steps. KBE technology takes care of all the operations involving geometry. These operations include the parametric geometric modelling and all the geometric model-involved analyses in the Refinement step. Besides, the generation of the road map and the post-process in the Initialization step is also enabled by the KBE technology.

- **The modular design approach**

The automatic 3D-routing method has been developed into a computer software tool called 3DAR. The development of this tool has followed the modular design approach. Different functions of the tool, such as Initialization, harness cost calculation, and geometric violation analysis, have been carried out by different software modules and these modules loosely connect with each other. Due to the modularity, modifying, including, or excluding a module does not influence the rest of the tool.

In 3D-routing cases, such as those presented in Chapter 7, the 3DAR reads in the 3D-routing inputs, namely harness electrical definitions, design specifications, and routing environments and produces geometric models and routing reports for the harnesses as outputs, as shown in Figure 8-1. These processes are implemented in full automation. When the routing environment or the harness electrical definitions have changed, the 3D-routing tool updates the design automatically.

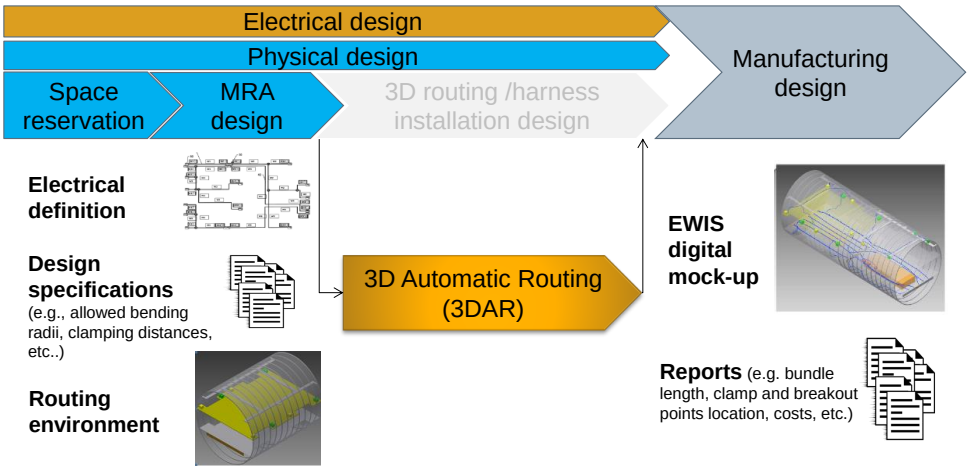


Figure 8-1: Location of the 3D automatic routing tool in the harness design workflow

When compared with previously published harness 3D-routing research, the tool developed in this research is able to handle more harness features/design rules (see Table 8-1), and even more design rules can be included in the future.

Table 8-1: Functionality of this and previous published research

Harness features & design rules	This research	Other research A [14]	Other research B[13]
Multi-destination/multi-origin	Y	Y	-
Clamping distance	Y	-	-
Fixing distance	Y	-	-
Bend radius	Y	-	-
Geometry collision	Y	Y	Y
(Non-)fixable structure	Y	-	Y
Harness separation(EMI)	Y	-	-
Clearance between harnesses and moving parts	Y	-	-
Flammable zone	Y	-	-
Hot zone	Y	-	-
Space reservation	Y	-	-

'Y'— Handled; '-' —Not handled/Not mentioned

8.2 Recommendations

The previously proposed method has proved that the 3D harness routing can be automated. However, the gap between the functionality of the 3D-routing tool and actual industrial needs still provides some improvement opportunities, such as those listed below:

- **The actual clamping approach**

Primary supports of the wire harness include clamps, brackets and/or stand-offs. In this 3D-routing tool, only the clamps and stand-offs have been considered since the bracket's stiffness needs to be analysed when it is used. This analysis is too complex to be included in this research at the moment.

However, brackets are widely used in practice. Clamps and/or stand-offs can only be attached to fixable surfaces perpendicularly and therefore the tangential direction of harnesses at the clamp point is always parallel to the fixed surface. This limits the number of feasible paths. Brackets have different shapes, such as L or Z shapes. When a clamp is fixed by an L-shape bracket, for example, the tangential direction of the clamp is changed by 90 degrees. Consequently, harnesses have more fixing options when brackets are used.

In order to enable the generation of a 3D-routing result that is the same as the actual design result, brackets need to be included in future work.

- **Selection of components from catalogues**

In the current 3D-routing tool, some harness components, such as clamps and covering, are defined according to the diameter of the related harness bundle, and the costs (e.g. in Euros) of these components are calculated according to these definitions.

In practice, most components are selected from product catalogues. The available component types and their catalogue price (the cost) are already defined before the 3D routing. 3D models of the components are also defined in the geometry libraries of products. During the 3D routing, the components are selected from libraries rather than defined by design engineers.

In future work, the product catalogues and geometry libraries need to be connected with the current routing tool. The actual catalogue price and weight will be used to support the cost analysis of harnesses in the 3D-routing phase. The geometry libraries will be used to support the generation of harness geometric models.

- **Parallel and/or distributed computing**

In addition to enhancing the functionality of 3DAR, the calculation efficiency of this tool can also be improved by using parallel or distributed computing.

In the Refinement step, Generalized Pattern Search (GPS) is used to refine the preliminary harness definition. In each optimization loop, the optimizer generates $6\times N$ harness configurations to explore the routing space. N is the number of clamps and/or breakouts. These harness configurations are independent of each other; therefore, the geometric modelling and various analyses of each harness can be implemented in parallel. Furthermore, different analyses of one harness are carried out by different analysis tools. These analyses are also independent of each other and therefore can also be implemented in parallel. Currently, 3DAR has 6 analysis tools. If we assume the calculation time of all the analysis tools is the same, the total analysis time of one harness can be 6 times faster if the 6 analyses are carried out in parallel. This means that if the parallel calculation is applied, the calculation can be maximum $6\times 6\times N$ times faster without considering the communication and coordination time.

In the routing case of fuselage sections presented in Chapter 7, the optimization of a single harness that has 14 clamps takes about 2 hours on a normal desktop. With parallel computing, the calculation time will be just 15 seconds.

References

1. Airliners, *A380 Electrical Harnesses*. 2012 [cited March 06 2012]; available from: http://www.airliners.net/aviation-forums/tech_ops/read.main/157173/#menu77
2. A.D. Helfrick, *Principles of Avionics*. Seventh ed. 2012: Avionics Communications Inc.
3. Safran Group, *More Electric Aircraft*. 2014 [cited September 25 2014]; available from: http://www.safran-group.com/site-safran-en/innovation-429/areas-of-expertise/more-electric-aircraft/.
4. K. Wong, *What Grounded the Airbus A380?* 2006 [cited March 06 2012]; available from: http://www.cadalyst.com/management/what-grounded-airbus-a380-5955.
5. English Russia, *How the Sukhoi Superjets Are Being Built*. 2013 [cited November 23 2014]; available from: URL:http://englishrussia.com/2013/03/27/how-the-sukhoi-superjets-are-being-built/2/.
6. Boeing, *Boeing 737 Facts*. 2015.
7. D. Dinh-Khanh, A. Mifdaoui, and T. Gayraud, *Fly-By-Wireless for next generation aircraft: Challenges and potential solutions*. In *Wireless Days (WD), 2012 IFIP*. 2012.
8. Wikipedia, *TWA Flight 800*. [cited October 08 2015]; available from: https://en.wikipedia.org/wiki/TWA_Flight_800.
9. Wikipedia, *Swissair Flight 111*. [cited October 08 2015]; available from: https://en.wikipedia.org/wiki/Swissair_Flight_111.
10. Federal Aviation Administration, *Development of an Electrical Wire Interconnect System Risk Assessment Tool*. 2006.
11. M. Kingsley-Jones, *The race to rewire the A380*. Flight International 2006 [cited May 07 2012]; available from: http://www.flightglobal.com/news/articles/farnborough-first-news-the-race-to-rewire-the-airbus-207894.
12. A. Angrand, *A Brand-New Wiring System for the Dreamliner.* Safran Magazine, 2007.
13. C. van der Velden, C. Bil, X. Yu, and A. Smith, *An intelligent system for automatic layout routing in aerospace design.* Innovations in System and Software Engineering, 2007. 3(2): pp. 117-128.
14. A.B. Conru, *A genetic approach to the cable harness routing problem*, in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. 1994. pp. 200-205.
15. N.S. Ong, *Activity-based cost tables to support wire harness design.* International Journal of Production Economics, 1993. 29(3): pp. 271-289.
16. R. Billsdon and K. Wallington, *Wiring harness design- can a computer help.* Computing & Control Engineering Journal, 1998. 9(4): pp. 163-168.
17. J.M. Ritchie, G. Robinson, P.N. Day, R.G. Dewar, R.C.W. Sung, and J.E.L. Simmons, *Cable harness design, assembly and installation planning using immersive virtual reality.* Virtual Reality, 2007. 11(4): pp. 261-273.
18. J. Ashour, *Four Steps to Procure Harnesses Without HADs*. 2014 [cited October 10 2014]; available from: http://www.interconnect-wiring.com/news/bid/70954/Four-Steps-for-No-HAD-of-an-F-16-Electrical-Wiring-Harness.
19. Federal Aviation Administration, *43.13-1B - Acceptable Methods, Techniques, and Practices - Aircraft Inspection and Repair*. 1998.
20. Avionikits, *RV-10 with G-900X*. [cited August 23 2016]; available from: http://www.avionikits.com/G900_rv10.html.
21. Federal Aviation Administration, *Aircraft Electrical Wiring Interconnect System (EWIS) Best Practices, Aircraft EWIS Practices Job Aid 2.0*.
22. G. La Rocca, *Knowledge Based Engineering Techniques to Support Aircraft Design and Optimization*, Ph.D. Dissertation, 2011.

23. T. van den Berg, *Harnessing the potential of Knowledge Based Engineering in manufacturing design*, Ph.D. Dissertation, 2013.

24. S.W.G. van der Elst and M.J.L. van Tooren, *Application of a Knowledge Engineering Process to Support Engineering Design Application Development.* Collaborative Productive and Service Life Cycle Management for a Sustainable World, 2008: pp. 417-431.

25. U.S. Department of Defense, *MIL-W-5088L. Wiring, Aerospace Vehicle*. 1991.

26. The National Aerospace Laboratory of the Netherlands (NLR), *Facilitating the design of future cable bundles, Annual Report 2012*. pp. 24.

27. U.S. Department of Defense, *MIL-C-27500G. Cable, power, electrical and cable special purpose, electrical shielded and unshielded, general specification for*. 1988.

28. Wikipedia, *American wire gauge*. [cited September 13 2016]; available from: https://en.wikipedia.org/wiki/American_wire_gauge.

29. A. Rahden, *Autodesk Inventor - Electrical Routing*. 2009 [cited March 03 2016]; available from: https://youtu.be/6R0b9_BEZ-I.

30. J.R.R.A. Martins and A.B. Lambe, *Multidisciplinary Design Optimization: A Survey of Architectures.* AIAA Journal, 2013. 51(9): pp. 2049-2075.

31. G. La Rocca, *Knowledge based engineering: Between AI and CAD. Review of a language based technology to support engineering design.* Advanced Engineering Informatics, 2012. 6(2): pp. 159-179.

32. P. Bartholomew, *The role of MDO within aerospace design and progress towards an MDO capability*, in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 1998, American Institute of Aeronautics and Astronautics.

33. G. Schuhmacher, I. Murra, L. Wang, A. Laxander, O. O'Leary, and M. Herold, *Multidisciplinary Design Optimization of a Regional Aircraft Wing Box*, in *9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 2002, American Institute of Aeronautics and Astronautics.

34. W.L. Neu, W.H. Mason, S. Ni, Z. Lin, A. Dasgupta, and Y. Chen, *A multidisciplinary design optimization scheme for container ships*, in *8th Symposium on Multidisciplinary Analysis and Optimization*. 2000, American Institute of Aeronautics and Astronautics.

35. M.H. Love, *Multidisciplinary design practices from the F-16 Agile Falcon*, in *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. 1998, American Institute of Aeronautics and Astronautics.

36. F. Flager, B. Welle, P. Bansal, G. Soremekun, and J. Haymaker, *Multidisciplinary process integration and design optimization of a classroom building.* Journal of Information Technology in Construction, 2009. 14: pp. 595-612.

37. J. Vandenbrande, T. Grandine, and T. Hogan, *The search for the perfect body: Shape Control for multidisciplinary design optimization*, in *44th AIAA Aerospace Sciences Meeting and Exhibit*. 2006, American Institute of Aeronautics and Astronautics.

38. G. La Rocca and M.J.L. van Tooren, *Knowledge-Based Engineering Approach to Support Aircraft Multidisciplinary Design and Optimization.* Journal of Aircraft, 2009. 46(6): pp. 1875-1885.

39. A.H. van der Laan, *Knowledge based engineering support for aircraft component design* Ph.D. Dissertation, 2008.

40. J. Sobieszczanski-Sobieski, A. Morris, and M. van Tooren, *Multidisciplinary Design Optimization Supported by Knowledge Based Engineering*. 2015: Wiley.

41. Genworks International, *GenDL*. 2015 [cited June 27 2015]; available from: http://www.genworks.com.

42. P. Graham, *ANSI Common Lisp*. 1996: Prentice Hall.

43. G. La Rocca and M.J.L. van Tooren, *Development of Design and Engineering Engines to Support Multidisciplinary Design and Analysis of Aircraft*, in *Delft Science in Design–A Congress on Interdisciplinary Design*. 2005. pp. 107-124.

44. G. La Rocca and M.J.L. van Tooren, *Enabling distributed multi-disciplinary design of complex*

products: a knowledge based engineering approach. Journal of Design Research, 2007. 5(3): pp. 333-352.

45. M.J.L. van Tooren, M. Nawijn, J. Berends, and J. Schut, *Aircraft Design Support using Knowledge Engineering and Optimisation Techniques*, in *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. 2005, American Institute of Aeronautics and Astronautics.

46. M.J.L. van Tooren, *Sustainable Knowledge Growth*, Inaugural Speech, 2003.

47. H. Wang, *Global-local Knowledge Coupling Approach to Support Airframe Structural Design*, Ph.D. Dissertation, 2014.

48. L.R. Lewis, *Rapid motion planning and autonomous obstacle avoidance for Unmanned Vehicles*, Monterey, California. Naval Postgraduate School, Master Thesis, 2006.

49. M.G. Earl and R. D'Andrea, *Iterative MILP methods for vehicle-control problems.* Robotics, IEEE Transactions on, 2005. 21(6): pp. 1158-1167.

50. Q. Gong, R. Lewis, and M. Ross, *Pseudospectral Motion Planning for Autonomous Vehicles.* Journal of Guidance, Control, and Dynamics, 2009. 32(3): pp. 1039-1045.

51. A. Richards and J.P. How, *Aircraft trajectory planning with collision avoidance using mixed integer linear programming*. In *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*. 2002. IEEE.

52. C.Y. Lee, *An Algorithm for Path Connections and Its Applications.* IRE Transactions on Electronic Computers, 1961. EC-10(3): pp. 346-365.

53. T.A. Yang, S. Shekhar, B. Hamidzadeh, and P.A. Hancock, *Path planning and evaluation in IVHS databases*. In *Vehicle Navigation and Information Systems Conference*. 1991.

54. T.P. Hartley and Q.H. Mehdi, *In-Game Adaptation of a Navigation Mesh Cell Path*. In *Computer Games (CGAMES), 2012 17th International Conference on*. 2012.

55. W. van Toll, A.F. Cook, and R. Geraerts, *Navigation meshes for realistic multi-layered environments*. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. 2011.

56. X. Cui and H. Shi, *An Overview of Pathfinding in Navigation Mesh.* International Journal of Computer Science and Network Security, 2012. 12(12): pp. 48-51.

57. M. Grenier, *Pathfinding concept, the basics* [cited June 12 2013]; available from: URL: http://mgrenier.me/2011/06/pathfinding-concept-the-basics.

58. S. Even, *Graph Algorithms*. 2011: Cambridge University Press.

59. N.P. Padhy, *Artificial Intelligence and Intelligent Systems*. 2005: Oxford University Press.

60. C. Wilt, J. Thayer, and W. Ruml, *A Comparison of Greedy Search Algorithms*. In *the Third Annual Symposium on Combinatorial Search*. 2010. Stone Mountain, Atlanta, Georgia, USA.

61. M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, *Smart hill climbing for agile dynamic mapping in many-core systems*. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*. 2013.

62. Shi-Kuo Chang, *Data Structures and Algorithms*. 2003: World Scientific.

63. I. Stroud, *Boundary Representation Modelling Techniques*. 2006: Springer-Verlag London.

64. C. Audet and J.E. Dennis, Jr., *Analysis of Generalized Pattern Searches.* SIAM Journal on Optimization, 2002. 13(3): pp. 889-903.

65. R.M. Lewis and V. Torczon, *Pattern Search Methods for Linearly Constrained Minimization.* SIAM J. on Optimization, 1999. 10(3): pp. 917-941.

66. C. Audet and J.E. Dennis, Jr., *A Pattern Search Filter Method for Nonlinear Programming without Derivatives.* SIAM Journal on Optimization, 2004. 14(4): pp. 980-1010.

67. R. Hooke and T.A. Jeeves, `` *Direct Search'' Solution of Numerical and Statistical Problems.* J. ACM, 1961. 8(2): pp. 212-229.

68. ASTM International, *Standard Specification for Standard Nominal Diameters and Cross-Sectional Areas of AWG Sizes of Solid Round Wires Used as Electrical Conductors*. 2008.

69. H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, *On the shape of a set of points in the plane.* IEEE Transactions on Information Theory, 1983. 29(4): pp. 551-559.

70. Z. Zhu, M.J.L. van Tooren, and G. La Rocca, *A KBE Application for Automatic Aircraft Wire Harness Routing*, in *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*. 2012: Honolulu, Hawaii, U.S.

71. M. Bahrami, *Steady Heat Conduction*. [cited December 12 2015]; available from: http://www.sfu.ca/~mbahrami/ENSC%20388/Notes/Staedy%20Conduction%20Heat%20Transfer.pdf.

Appendix A. American wire gauge

Table A-1: Standard nominal diameters and cross-Sectional areas of AWG sizes of solid round wires at 20°C[68]

Size	Diameter		Cross-Sectional Area		Size	Diameter		Cross-Sectional Area	
AWG	mils	mm	cmils	mm^2	AWG	mils	mm	cmils	mm^2
4/0	460.0	11.684	211 600	107.2	29	11.3	0.287	128	0.0647
3/0	409.6	10.404	167 800	85.0	30	10.0	0.254	100	0.0507
2/0	364.8	9.26	133 100	67.4	31	8.9	0.226	79.2	0.0401
1/0	324.9	8.25	105 600	53.5	32	8.0	0.203	64.0	0.0324
1	289.3	7.35	83 690	42.4	33	7.1	0.180	50.4	0.0255
2	257.6	6.54	66 360	33.6	34	6.3	0.160	39.7	0.0201
3	229.4	5.82	52 620	26.7	35	5.6	0.142	31.4	0.0159
4	204.3	5.19	41 740	21.1	36	5.0	0.127	25.0	0.0127
5	181.9	4.62	33 090	16.8	37	4.5	0.114	20.2	0.0103
6	162.0	4.11	26 240	13.3	38	4.0	0.102	16.0	0.00811
7	144.3	3.67	20 820	10.6	39	3.5	0.0890	12.2	0.00621
8	128.5	3.26	16 510	8.37	40	3.1	0.0787	9.61	0.00487
9	114.4	2.91	13 090	6.63	41	2.8	0.0711	7.84	0.00397
10	101.9	2.59	10 380	5.26	42	2.5	0.0635	6.25	0.00317
11	90.7	2.30	8 230	4.17	43	2.2	0.0559	4.84	0.00245
12	80.8	2.05	6 530	3.31	44	2.0	0.0508	4.00	0.00203
13	72.0	1.83	5 180	2.63	45	1.76	0.0447	3.10	0.00157
14	64.1	1.63	4 110	2.08	46	1.57	0.0399	2.46	0.00125
15	57.1	1.45	3 260	1.65	47	1.40	0.0356	1.96	0.000993
16	50.8	1.29	2 580	1.31	48	1.24	0.0315	1.54	0.000779
17	45.3	1.15	2 050	1.04	49	1.11	0.0282	1.23	0.000624
18	40.3	1.02	1 620	0.823	50	0.99	0.0252	0.980	0.000497
19	35.9	0.904	1 290	0.653	51	0.88	0.0224	0.774	0.000392
20	32.0	0.813	1 020	0.519	52	0.78	0.0198	0.608	0.000308
21	28.5	0.724	812	0.412	53	0.70	0.0178	0.490	0.000248
22	25.3	0.643	640	0.324	54	0.62	0.0158	0.384	0.000195
23	22.6	0.574	511	0.259	55	0.55	0.0140	0.302	0.000153
24	20.1	0.511	404	0.205	56	0.49	0.0125	0.240	0.000122
25	17.9	0.455	320	0.162					
26	15.9	0.404	253	0.128					
27	14.2	0.361	202	0.102					
28	12.6	0.320	159	0.0804					

Appendix B. Geometric simplification of detailed routing environments

B.1 Background

An aircraft DMU (i.e. harnesses routing environment) includes many geometric features, such as chamfers and rivet holes. These features are essential for aircraft manufacture and assembly; however, they are too detailed and unnecessary for 3D harness routing. During an automatic 3D-routing phase, these unnecessary features significantly slow down the routing tool. During a manual design process, they increase the waiting time of engineers and decrease the smoothness of interactive operations.

B.2 Current solutions and their problems

Actually, design engineers have already realized the issue that routing environments contain many unnecessary details, and they have some solutions to handle this issue, as presented in Table B-1.

Table B-1: Current solutions to the overly detailed geometry and the problems caused

Current solutions to the overly detailed geometry	Problems caused
Switch off display of irrelevant geometric parts	May have geometric collision and other violations when display everything
Use CGR (Catia Graphical Representation) file	Useful only for visualization
Ignore the details of some parts	Details are still there

However, these three solutions still have some limitations:

1) During the design process, design engineers switch off the load and/or display of irrelevant geometric parts. This method helps to release the loading and rending workloads of the computer. However, as the geometric components are not visible, harnesses may be placed in positions where the components are already placed. Therefore, geometric component-related violations, such as geometric collision, may occur when these geometric parts are loaded and displayed again.

2) CGR is the triangulated format used by Catia V5. It uses many faceted surfaces to represent a geometric model. The problem is that CGR can only represent the outer shape of geometric models. None of the geometric features, such as holes or extruded geometric, can be recognized any more in this representation, although the outer shapes of the features are still there. However, due to the simplification or actually losing geometric features, loading and rending the CGR files are faster. This format is mainly used for visualization only. It is not suitable for the 3D-routing tool since the tool needs the lost information to evaluate which surfaces harnesses can be fixed to, what is the distance between a harness and a component, and so forth.

3) During a manual design process, design engineers sometimes mentally ignore some irrelevant geometric parts. These parts are not checked or measured during the entire design process. However, these geometric parts are still loaded and rendered and they will still

influence the 3D-routing efficiency and the smoothness of the interactive operations.

B.3 Simplification methods

In order to simplify the actual routing environment to increase the loading, rending and 3D-routing efficiency but still keep the required geometric information in the routing environment, two simplification methods have been studied. The first is called the *alpha-shape-based geometric simplification method* and it is mainly used to simplify equipment, such as actuators. The second is called the *feature recognition-based geometric simplification method* and it is mainly used to simplify the airframe, such as ribs and spars.

B.3.1. Alpha-shape-based geometric simplification

Alpha-Shape[69] is a set of piecewise line segments representing the intuitive notional shape of a set of points. The point set comes from and represents the shape of geometric models that are about to be simplified. Alpha is a variable. By selecting different Alpha values, different approximations can be obtained, as shown in Figure B-1.

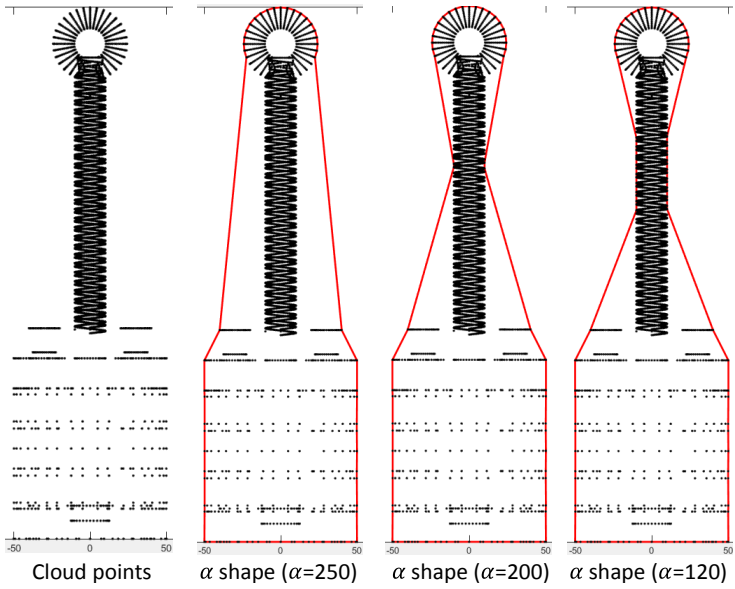


Figure B-1: Alpha-shapes of cloud points, at different alpha values

Generating alpha-shape representation is only one step in this simplification method, which includes 5 steps in total. These 5 steps are illustrated in Figure B-2.

Step 1: Tessellation. The geometric model is first tessellated, generating a set of 3D cloud points. The tessellation size depends on the alpha value, to avoid the generation of too many points (i.e. the size is too small) and also to guarantee that the alpha shape can be generated (i.e. the size is too large).

Step 2: Projection. These 3D points are projected to X-O-Y, X-O-Z, and Y-O-Z planes to get three sets of 2D points.

Step 3: Alpha-shape generation. Three alpha-shape profiles are generated respectively based on the three sets of 2D points.

Step 4: Extrusion: Three alpha-shape profiles are extruded to generate three solid models.

Step 5: Intersection: The three solid models are intersected to get the geometric approximation of the actual model.

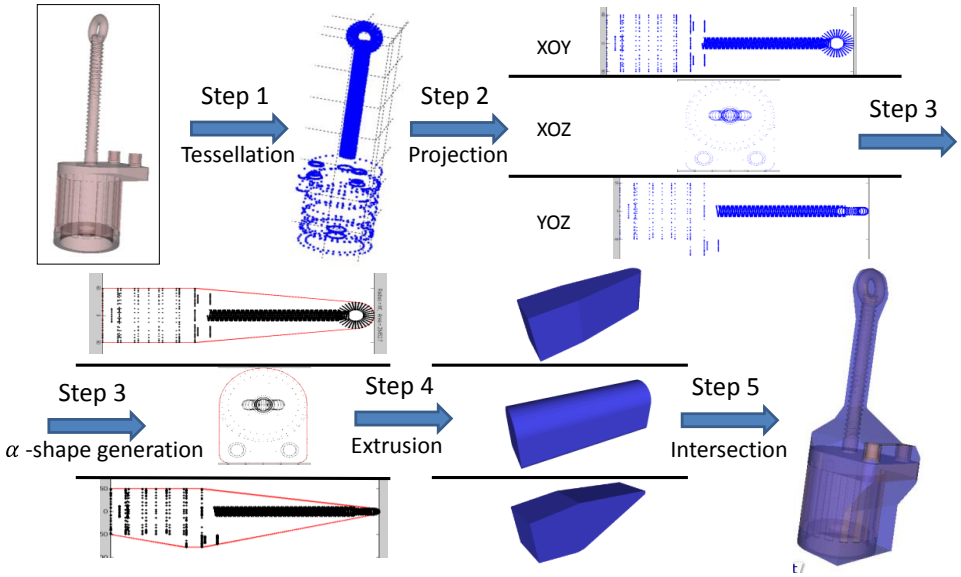


Figure B-2: Five steps of the simplification method

After the simplification, the position and outer profiles of the geometric components are still kept. Many geometric features which are unnecessary for 3D routing, such as threads, bolts and nuts, and screws of the geometric components, are eliminated.

The size of STEP file of a geometric model can represent the number of features of the geometric model. The STEP file of an actual aircraft actuator and the simplified actuator are 8012 KB and 504 KB respectively. The file size of the simplified model is only 6% of the actual model. Therefore, the computer workload to load and render the geometric components is reduced significantly after elimination of these features.

The simplification of an entire routing environment, which generally is an assembly model, is carried out per geometric part. The placement of each actual geometric model, including the 3D position and 3D rotation, is still maintained in the simplified model. When all the components are simplified, the simplified geometric models will be placed at the same place of the actual geometric models. The assembly constraints and the product tree are not changed.

B.3.2. The feature recognition-based geometric simplification

Harness-routing environments are given in the format of a STEP file to the 3D-routing tool. After the transfer via the STEP file, the information of the geometric features, such as the extruding, loft and holes, is lost in the product tree although the geometric shape of these features exists in the geometric model.

Feature recognition here means recognizing the shape and parametric information of the lost geometric features. After the features are recognized, they will be rebuilt in the product tree. Then the features that are unnecessary for 3D routing can be deleted to reduce the complexity of the routing environment.

The routing environment, such as an aircraft tail, is an assembly model that includes geometric parts. The feature recognition is implemented automatically per geometric part by a commercial CAD tool, Autodesk Inventor. The assembly relationship of the geometric parts in the assembly model is not changed during the process. When the feature recognition of a part is completed, all the recognized features will be added to the product tree of this part. Then the unnecessary features for 3D routing are deleted. After all the geometric parts are simplified, the simplified routing environment is immediately used for the 3D routing.

The simplification process of an aircraft tail is illustrated in Figure B-3. In this example, the file size of the simplified model is around a quarter of the actual geometric part.

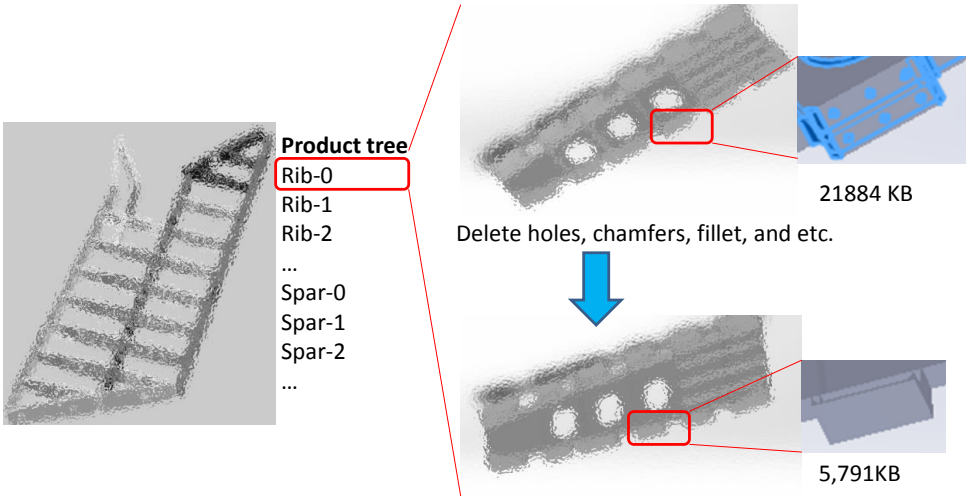


Figure B-3: The feature recognition-based geometry simplification method (company confidential pictures are processed)

Appendix C. Hot zones and their influence on harness routing

C.1 Description and modelling of hot zones

Heat is a common phenomenon inside an aircraft. It is caused by many reasons, such as heat emission from the avionic system or combustion inside the engine. It is a very complex process to build an accurate temperature field inside an aircraft because of the complex physical situations, such as different heat sources, the irregular shape of geometric components, and various heat transfer mediums. However, temperature fields, which are defined in Equation (C.1), do exist. Harnesses routed in these fields need extra protection to keep the harnesses from damage.

$$T = T(x, y, z, t) \tag{C.1}$$

In order to demonstrate that this tool is able to route harnesses in a routing environment with hot zones, a simplified but reasonable hot-zone modelling approach is adopted. This approach is introduced in a published paper [70] and the following section is adapted from this paper.

This approach assumes that a hot zone is caused by a high temperature device which is accommodated on an aluminium alloy panel. To simplify the problem, the heating source is modelled as a cylinder defined by its radius and height. The lower surface of the cylinder is in close contact with the upper surface of the panel. The temperature of this cylinder is uniform. Its centre coordinate is (x_0, y_0, z_0). We assume that there is no air flow in the surrounding areas and the normal environment temperature is $20^{o}\mathrm{C}$. The heat dissipation power of this device is 1000 W. The thickness of the aluminium alloy panel is 10 mm and its thermal conductivity is $121 W/m\cdot{}^{o}C$. Since the device emits the heat stably, there is a stable temperature field around this device. Hence, the time t in Equation (C.1) is ignored.

According to heat transfer principles, the influence of radiation and convection can be ignored because of the low temperature and no air flow. The heat conduction via air can also be ignored because of the relatively low thermal conductivity of air. The main heat transfer method is conduction via the panel. The conduction process can be solved as the cylinder heat conduction problem and the boundary of the panel do not need to be considered since the cylinder is much smaller than the panel.

$$Q = 2\pi\lambda th_{panel}(T_1 - T_2) / \ln\frac{r_2^{hot}}{r_1^{hot}} \tag{C.2}$$

Equation (C.2)[71] is used to calculate the temperature distribution. Q is the heat diffusion power; th_{panel} is the thickness of the panel; λ is the thermal conductivity; r_1^{hot} and r_2^{hot} are the radii from the centre of the heating source to the position on the rib; and T_1 and T_2 are the centigrade temperatures at the position with the two radii respectively. The top view of a temperature field example calculated with the equation on the panel surface is presented in Figure C-1.

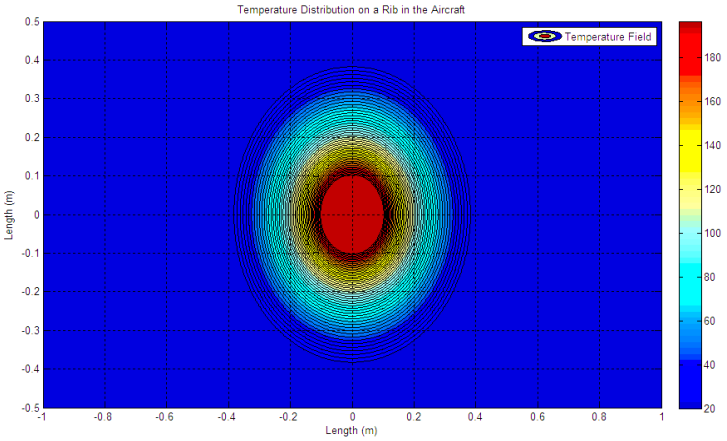


Figure C-1: Temperature distribution of the heating source

This pie-shape temperature field is used in this research to represent all the hot zones. The parameters listed in Table C-1 can be changed to obtain different zones in the routing environment.

Table C-1: Parameters to define a hot zone

Heating source parameters	Example values
:centre-point	#(3240.0 -150.0 -1000.0)
:source-radius	100mm
:source-height	20mm
:temperature	300mm
:normal-direction	#(0.0 0.0 1.0)

C.2 Influence of hot zones on 3D routing of harnesses

Heat influences the use of coverings. A high-density covering needs to be used if the temperature increases. It is assumed that the covering does not need to be used if the temperature is lower than $100^{o}\mathrm{C}$ and therefore covering density $e_{cover}=0$. If a zone is hotter than $100^{o}\mathrm{C}$, the e_{cover} is calculated with Equation (C.3), where e_{bundle} is the density of a bundle (see Table 6-4) and T_{source} is the highest temperature (the temperature of the centre) in the hot zone.

$$e_{cover}=\max(0,(T_{source}-100)/50)\times e_{bundle} \tag{C.3}$$

As shown in Figure C-2, a hot zone is represented by a Boundary Representation (B-rep) geometric model. The radius of a hot zone is calculated with Equation (C.2) and its height is set to 3 times the source height. The outer boundary of the geometry is the $100^{o}\mathrm{C}$ contour.

If a harness branch has contact with this geometric model, the e_{cover} calculated by Equation (C.3) will apply to the entire branch to support the cost calculation for the covering. Details of the cost calculation method are presented in Chapter 6.

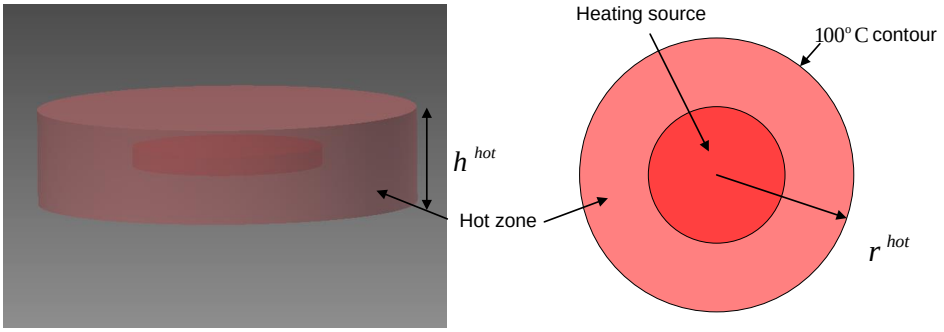


Figure C-2: Illustration of the hot zone caused by the heating source

Appendix D. Flammable zones and their influence on harness routing

D.1 Description and modelling of flammable zones

A flammable zone is a zone located above a pipe used to transport flammable liquid. Harnesses need intensive clamping when routing in this zone (see Sub-section 2.3.2.6).

The rule presented in Sub-section 2.3.2.6 can also be interpreted as "*the maximum allowed clamping distance of a harness which is routed above fluid lines* is equal to *the distance between the bundle centre line and centre line of fluid pipe* (i.e. D^{bf}) minus *the radius of the pipe* r_{pipe}", as shown in Equation (D.1). The radius of a harness is relatively small so therefore it is not considered here.

$$D_{\max}^{clamp} = D^{bf} - r_{pipe}$$
$$, where\ 2\ inches + r_{pipe} \le D_{bf} \le 24\ inches + r_{pipe} \quad \text{(D.1)}$$

Routing within the 2 inch-area around the pipe is forbidden and routing 24 inches away from the pipe does not need to consider this rule. Therefore the 2 and 24 inches plus the pipe radius indicate the inner and outer boundaries of the flammable zone.

This rule only applies when the harness is routed "above" the pipe. However, the "above" is not clearly defined in the design specification discussed in Sub-section 2.3.2.6. Actually it can be different from aircraft to aircraft. For example, in a military aircraft which could fly upside-down, the "above" can be 360°. Since the definition of "above", which should be given as an input, is not available yet, it is defined as a $\pm 45^\circ$ sector from the reverse direction of gravity and its centre is on the pipe centre line, as shown on the left-hand side of Figure D-1. On the basis on this definition, the flammable zone of a pipe is defined and illustrated on the right-hand side of Figure D-1.

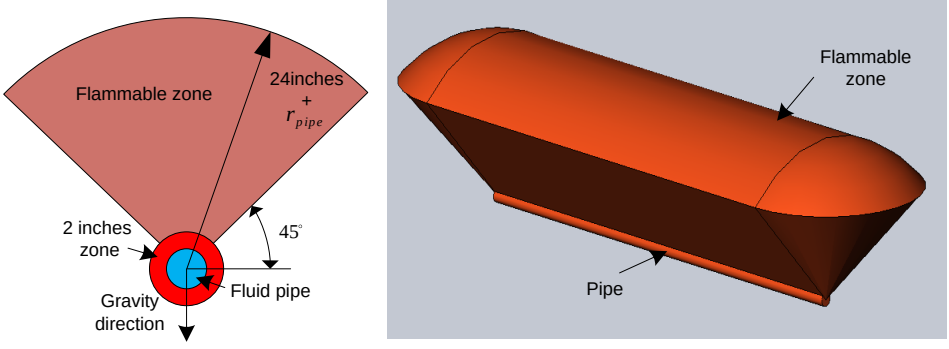


Figure D-1: Illustration of a flammable zone (left: cross-sectional view, right: trimetric view)

The position and shape of a flammable zone depends on the parameters of the pipe and the direction of gravity. These parameters are given below.

Table D-1: Parameters to define flammable zones

Parameters to define flammable zone	Example values
:radius ;;distance from outer boundary to the pipe	609.6mm (24 inches)
:gravity-dir	#(0.0 1.0 0.0)
:line ;;the start and end point of a pipe	(#(1000.0 -500.0 -100.0) #(3000.0 -500.0 -100.0))
:pipe-radius	5.08 mm

D.2 Influence of flammable zones on the harness clamping distance

The maximum allowed clamping distance $D_{\max}^{clamp}$ is 24 inches in normal areas. It will be smaller in a flammable zone and the actual $D_{\max}^{clamp}$ value depends on the locations of the bundle segment located between clamping points i and j, as shown in Figure D-2.

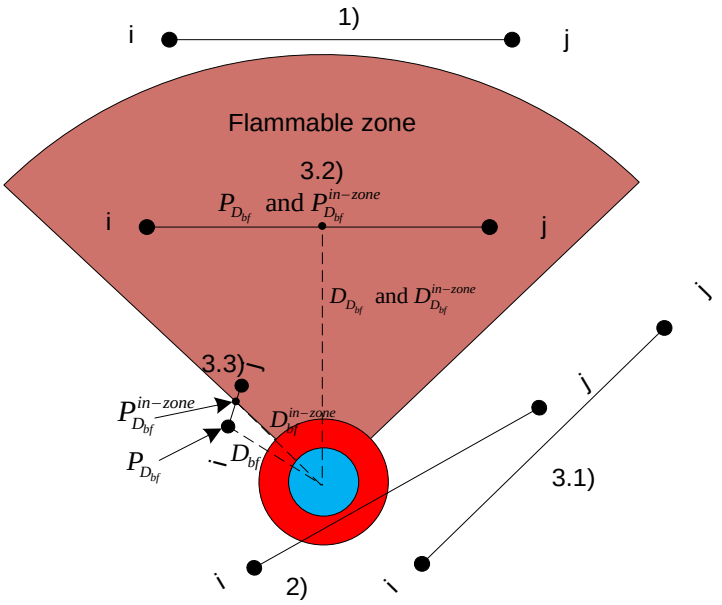


Figure D-2: Possible locations of bundle segments in the area surrounding the flammable zone

Calculation of the actual $D_{\max}^{clamp}$ of these location scenarios is detailed below:

1) if $D_{bf} > 24\ inches + r_{pipe}$, $D_{\max}^{clamp} = 24 inches$.

2) if $D_{bf} < 2\ inches + r_{pipe}$, there is no need to check $D_{\max}^{clamp}$ since routing in a 2-inch zone is not allowed.

3) if $2\ inches + r_{pipe} \le D_{bf} \le 24\ inches + r_{pipe}$, extra calculations are needed:

In order to carry out the calculations, four definitions are introduced:

ij: line segment between clamp i and j;

$P_{D_{bf}}$: point on ij and having the shortest distance to the pipe;

$P_{D_{bf}}^{in-zone}$: point on ij, having the shortest distance to the pipe, and in the flammable zone;

$D_{bf}^{in-zone}$: distance from $P_{D_{bf}}^{in-zone}$ to the pipe centre line.

3.1) if the point $P_{D_{bf}}^{in-zone}$ does not exist, namely ij does not contact with the flammable zone, $D_{\max}^{clamp} = 24 inches$.

3.2) if $P_{D_{bf}}^{in-zone}$ exists and is the same as $P_{D_{bf}}$, $D_{\max}^{clamp} = D_{bf} - r_{pipe}$

3.3) if $P_{D_{bf}}^{in-zone}$ exists but is not the same as $P_{D_{bf}}$, $D_{\max}^{clamp} = D_{bf}^{in-zone} - r_{pipe}$

According to the above rules, a software function has been developed to check whether two adjacent clamping points, which are given as inputs, satisfy the smaller $D_{\max}^{clamp}$ caused by the flammable zone. The pseudocode of this check function is given in Figure D-3.

```
;;;**********************************
;;;Pseudocode of the function vertexes-connected?()
;;;**********************************
1 Begin
2  Set return-value = false
3  If distance(V_i ,V_j ) <=24 inches
4   If (D_bf > 24 inches + r_pipe) ;;the edge does not contact with the zone
5     Set return-value = true
6   Else if (D_bf < 2 inches + r_pipe) ;;too close to the pipe
7     Set return-value = false
8   Else if (2 inches + r_pipe ≤ D_bf ≤ 24 inches + r_pipe)
9     If (P_D_w^in-zone = nil) ;;the edge does not contact with the zone
10        Setf retrun-value = true
11    Else
12        If (D_i,P_D_w^clamp < D_bf^in-zone and D_i,P_D_w^clamp < D_bf^in-zone)
13            Setf retrun-value = true
14        Else
15            Setf retrun-value = false
16        End
17    End
18  End
19 End
19 Return return-value
20 End
```

Figure D-3: Pseudocode of the clamping distance check function in a flammable zone

Appendix E. Introduction of Capability Modules

A Capability Module (CM) of an analysis tool is responsible for the preparation of dedicated data for the tool. The harness 3D-routing tool includes 6 analysis tools. The CMs of these 6 tools are introduced here.

E.1 Capability Module of the Harness Cost Analysis Tool (HCAT)

A harness cost is the summation of the branch costs and a branch cost consists of a bundle cost, a covering cost, and a clamping cost. In order to calculate a harness cost using the Harness Cost Calculation Tool (HCAT), information on the bundles, coverings, and clamps needs to be extracted by the CM from the harness geometric model.

- **Geometric model of all the bundles**

The geometric object ***bundle*** contains two child-objects: a bundle central curve and a bundle model itself. The bundle central curve determines the length of the bundle. The bundle model will be used to calculate the bundle cost coefficient.

- **Geometric information of covering**

A covering cost depends on the length and thickness of the covering. The length is determined by the start and end positions of the covering on the bundle it covers.

```
(:bundle
 (#<harness-model::bundle @ #x2509eee2>  #<harness-model::bundle @ #x2509f87a>
#<harness-model::bundle @ #x2509fb9a>)
  :covering
 ((:covering-pos ((0 1)) :covering-thickness (1 cm))
 (:covering-pos ((0 1)) :covering-thickness (1 cm))
 (:covering-pos ((0 1))  :covering-thickness (1 cm))))
:clamp-cost
(#<harness-model::clamp @ #x107c65222> #<harness-model::clamp @ #x107c65202>
 #<harness-model::clamp @ #x107c65ac2> #<harness-model::clamp @ #x107e05ca2>
 #<harness-model::clamp @ #x107c66652> #<harness-model::clamp @ #x107fc4972>)
(#<harness-model::clamp @ #x107fc4992> #<harness-model::clamp @ #x107fc49b2>
 #<harness-model::clamp @ #x107fc49d2>)
(#<harness-model::clamp @ #x1080ebec2>)
)
```

Figure E-1: Example output of the Capability Module of HCAT

- **Geometric information on clamps**

The cost of clamps on a branch is the summation of all the clamps' costs. A clamp cost includes the material cost and installation cost. The installation cost is given as an input and

the material cost is calculated with the geometric information of the clamp model.

The example CM output of a harness that has 3 branches is shown in Figure E-1.

E.2 Capability Module of the Geometric Collision Analysis Tool (GCAT)

The three geometric collision types shown in Figure 6-6 need to be checked by GCAT.

The geometric model of bundles is needed for the Type-1 (i.e. a collision between the current harness and the routing environment) and the Type-2 (i.e. a collision between the current harness and previously routed harnesses) geometric collision check. The Type-3 collision check needs not only the geometric model but also the parent-child relationship to determine whether the detected collision is an actual geometric collision between different branches or an allowed intersection at breakout points (see Figure 6-7).

An example CM output data for GCAT is shown in Figure E-2.

```
Wire bundle list:
(#<surf::brep-from-surface @ #x2581cb22> #<surf::brep-from-surface @ #x2581d0c2>
#<surf::brep-from-surface @ #x2581d64a>  #<surf::brep-from-surface @ #x2581dbd2>
#<surf::brep-from-surface @ #x2581e15a>  #<surf::brep-from-surface @ #x2581e6e2>
#<surf::brep-from-surface @ #x2581ec6a>)
Bundle relation:
(:branches
 ((:branch-id "branch-000" :parent nil :child (1 2))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child (3 4))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child (5 6)))
  (:branch-id "branch-003" :parent (:type :branches :index 1) :child nil)
  (:branch-id "branch-004" :parent (:type :branches :index 1) :child nil)
  (:branch-id "branch-005" :parent (:type :branches :index 2) :child nil)
  (:branch-id "branch-006" :parent (:type :branches :index 2) :child nil)
  ))
```

Figure E-2: An example output of the Capability Module of GCAT

E.3 Capability Module of the Harness Bend Radius Analysis Tool (HBRAT)

The harness-bend radius violation check needs to know the actual and allowed bend radius of each bundle. The actual bend radius is calculated from the bundle central curves, and the allowed bend radius is calculated with the diameter of both the bundle and the thickest wire (see Sub-section 2.3.2.2).

The CM of HBRAT extracts the central curves and the diameters of the bundles and the thickest wire from each bundle geometric model and formalizes them into an output list. An example of the output data is shown in Figure E-3.

```
((#<surf::my-fitted-curve @ #x257c74f2> 11.0 mm 3.2 mm)
(#<surf::my-fitted-curve @ #x257cf95a> 8.9 mm 3.2 mm)
(#<surf::my-fitted-curve @ #x257d07aa> 7.3 mm 3.2 mm)
(#<surf::my-fitted-curve @ #x257d15e2> 7.3 mm 3.2 mm)
(#<surf::my-fitted-curve @ #x257d2482> 7.3 mm 3.2 mm)
(#<surf::my-fitted-curve @ #x257d333a> 3.7 mm 3.2 mm)
(#<surf::my-fitted-curve @ #x257d41f2> 3.7 mm 3.2 mm))
```

Figure E-3: An example output of the Capability Module of HBRAT

E.4 Capability Module of the Clamp Distance Analysis Tool (CDAT)

The clamp distance analysis (including clamping distance and fixing distance) of a whole harness is carried out per branch. It needs to know the branch types and waypoints of each branch.

Branches have different types in terms of their position in harness topology. As illustrated in Figure E-4, the types are "*single-harness*", "*start-branch*", "*middle-branch*" and "*end-branch*".

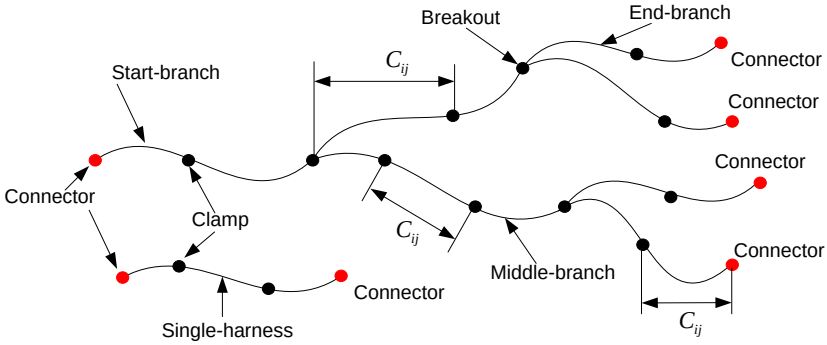


Figure E-4: Illustration of the harness branch types

The waypoints used to generate a branch central curve include connectors, clamps and breakouts. The clamping distance violation check is implemented between each adjacent waypoint pair. The fixing distance check is implemented per clamping point except for the connectors (i.e. the start point of the "*start-branch*" and the end point of the "*end-branch*") since connectors are inputs and their position will not be changed during the 3D-routing phase.

In the output data sheet, the waypoints and type of each branch are included, as shown in Figure E-5.

```
((:branch-type "start-branch" :way-points  (#(160.0 -130.0 -1000.0) #(469.22992307692306 -150.0
-799.9994999999999) #(942.1281361718976 -146.16076069404244 -474.99999999999983)
#(1489.2403040074637 -297.2280205730821 -474.9999999999993) #(2042.4467995212879 -
297.23635789488736 -474.9999999999998)   #(2589.978624671271 -149.99999999999994 -
449.9992499999998) #(3053.846384615384 -150.0 -449.99924999999985)
#(3386.154215384615 -150.0 -449.99924999999985) #(3423.0773076923074 -150.0 -
449.99924999999985)))

 (:branch-type "middle-branch" :way-points  (#(3423.0773076923074 -150.0 -
449.99924999999985) #(3792.3082307692302 -150.0 -799.9994999999999)
#(4124.616061538461 -150.0 -799.9994999999999)   #(4161.539153846153 -150.0 -
799.9994999999999)))

 (:branch-type "middle-branch" :way-points  (#(3423.0773076923074 -150.0 -
449.99924999999985) #(3521.3767899982877 -453.95021013840665 -314.9999250000005)
#(3532.298954698952 -487.7224557093407 -300.00000000000057)))

 (:branch-type "end-branch" :way-points (#(4161.539153846153 -150.0 -799.9994999999999)
#(4530.0 -130.0 -500.0)))

 (:branch-type "end-branch" :way-points  (#(4161.539153846153 -150.0 -799.9994999999999)
#(4161.539153846153 -150.0 -1149.99975) #(4161.539153846153 -150.0 -1500.0) #(4130.0 -
130.0 -2000.0)))

 (:branch-type "end-branch" :way-points (#(3532.298954698952 -487.7224557093407 -
300.00000000000057) #(3400.0 -900.0 -130.0)))

(:branch-type "end-branch" :way-points  (#(3532.298954698952 -487.7224557093407 -
300.00000000000057) #(3957.2989546989525 -681.5841835640113 -350.00000000000006)
#(4382.298954698952 -875.4459114186815 -300.0000000000001) #(4500.0 -1000.0 -130.0))))
```

Figure E-5: An example output of the Capability Module of CDAT

E.5 Capability Module of the Harness Separation Analysis Tool (HSAT)

Two harnesses are not always allowed to be routed next to each other. The (minimum) actual distance between two harnesses needs to be greater than the allowed distance. The actual distance is calculated with the harness geometric models. The allowed distance depends on the segregation code of both harnesses. The segregation code (also called the EGS code) is given as input for the 3D routing and it is a property of every harness.

The separation check is implemented per branch pair. The CM of HSAT is responsible for extracting the branch geometric model and the EGS code and formalizing them into a list, as shown in Figure E-6.

```
((#<harness-model::bundle @ #x277beee2> :EGS (:EMC :A :Group 1 :Subsystem :X)) (#<harness-
model::bundle @ #x277bf87a> :EGS (:EMC :A :Group 1 :Subsystem :X)) (#<harness-model::bundle
@ #x277bfb9a>) :EGS (:EMC :A :Group 1 :Subsystem :X)))
```

Figure E-6: An example output of the Capability Module of HSAT

E.6 Capability Module of the Harness CLearance Analysis Tool (HCLAT)

Harnesses need to have some clearance from moving parts. The clearance analysis is carried out per branch and it is transferred to the geometric collision check between the auxiliary geometric model of each branch and the moving parts.

The CM of HCLAT is in charge of the preparation of the auxiliary geometric models of

harness branches. Each auxiliary geometric model is generated by thickening the actual bundle by N mm. Then these auxiliary geometric models will be formalized to a list and sent to the HCLAT. The CM output of a harness that has 5 branches is presented in Figure E-7.

```
(#<surf::brep-from-surface @ #x10a53df52> #<surf::brep-from-surface @ #x107908a32>
#<surf::brep-from-surface @ #x10b3cfae2>> #<surf::brep-from-surface @ #x10b415522>
#<surf::brep-from-surface @ #x10b50b302>)
```

Figure E-7: An example output of the Capability Module of HCLAT

Appendix F. Pseudocode

```
;;;**********************************
;;;Pseudocode of A* searching algorithm
;;;**********************************
1 Begin
2   Give start, goal;
3   Set Openlist =(), Closedlist =();
4   Set g(start) =0, h(start) =heuristic-diatance(start, goal);
    f(start) = g(start) + h(start);
5   Move start to Openlist;
6   While(Openlist != nil)
7       Get node n which is first node in Openlist;
8       If(n == goal)
9           Break;
10      Remove(n) from Openlist;
11      Add (n) to Closedlist;
12      For each y in neighbor-nodes(n)
13        If ((y in closedlist) or (y is unwalkable))
14            Continue;
15        Eles if (y not in Openlist)
16          Add(y) to Openlist;
17          Set father-node(y) =n;
18          g(y) = g(n) + dist_between(n,y);
19          h(y) =heuristic-diatance(y,goal);
20          f(y) = g(y) + h(y);
21        Eles            ;;;(y in Openlist)
22          If((g(n) + dist_between(n,y)) < g(y))
23            Set father-node(y) =n;
24            g(y) = g(n) + dist_between(n,y);
25            f(y) = g(y) + h(y);
26          End
27        End
28      End
29    Sort Openlist from small to big in terms of f value;
30  End
    ;;;track back along the father-nodes to get optimized path
31  If(previous path finding success)
32     Set n = goal, Pathlist = (goal);
33     While(n != start)
34         n= Get-father-node(n);
35         Add(n) to Pathlist;
36       End
37  End
38 End
```

Figure F-1: Pseudocode of the A* pathfinding algorithm

```
;;;*********************************************************
;;;Pseudocode of Comprehensive Hill Climping searching algorithm
;;;*********************************************************
;;; breakouts-com-cost-lst;; list storing the combination of breakouts sets and the according cost
;;; breakouts-com-lst ;; list storing one breakouts set
1  Begin
2   Give Origin, Destinations;
3   Set breakouts-com-cost-lst =(); ;;set the cost list of break outs combination  to null
4   Fcost-benchmak = infinite
;;set the break outs combination list to a null list, the length of it equals to number of breakouts
5   Set breskouts-com-lst = (nil,nil,...nil)
6   Loop for i =0 to numofbrekaouts -1
7       (nth i breakouts-com-lst) = Origin;   ;;make all break outs locate on Origin node
8   End
;;Calculate the harness cost according to the break outs position
9   Fcost = harness-cost(breakouts-com-lst)
    ;; combine the breakouts list and harness cost of this list
    ;; and add them into the breakouts-com-cost-lst
    ;; meanwhile sort the breakouts-com-cost-lst from small to big in terms of the value of Fcost.
10  Insert-to-cost-lst(Fcost, breakouts-com-list, breakouts-com-cost-lst)
11  While(t)
12      Fcost-temp =Fcost of first item of breakouts-com-cost-lst
13      If Fcost-benchmak < Fcost-temp   ;; if no further improvements can be found, the search stops
14          Return
15      End
16      Fcost-benchmak = Fcost-temp
17      Loop for nBreakout = 0 to numofbrekaouts -1
18          Get breakouts-com-lst from first item of breakouts-com-cost-lst
19          Set adjacent-node = get-adjacent-node((nth nBreakout breakouts-com-lst))
20          Loop for nAdjacent = 0 to length(adjaecnt-node) -1
                ;;replace a break-out with its adjacent nodes
21              (nth nBreakout breakouts-com-lst) = (nth nAdjacent adjaecnt-node)
23              If breakouts-com-lst does not exist in breakouts-com-lst
22                  Fcost = harness-cost(breakouts-com-lst)
                    ;; combine the breakouts list and harness cost of this list
                    ;; and add them into the breakouts-com-cost-lst
                    ;; sort the breakouts-com-cost-lst from small to big in terms of the value of Fcost.
23                  Insert-to-cost-lst(Fcost, breakouts-com-lst, breakouts-com-cost-lst)
24              End
25          End
26      End
27 End
```

Figure F-2: Pseudocode of the Hill Clamping algorithm

Appendix G. Harness definitions at different phases during a 3D routing

```
;;input data of harness
(
(:branches
 ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index 0) :end-poi (:type :break-out :index 0) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 8))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child nil :start-poi (:type :break-out :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 8))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child nil :start-poi (:type :break-out :index 0) :end-poi (:type :connectors :index 2) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 8))
 )
 :connectors ;; the connectors here means the receptacles
 ((:connector-id "con-000" :data (:point #(160.0 -130.0 -1000.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(4530.0 -130.0 -500.0) :vector #(-0.0 -1.0 0.0)))
  (:connector-id "con-002" :data (:point #(4130.0 -130.0 -2000.0) :vector #(-0.0 -1.0 0.0)))
 )
 :break-out
 ((:break-out-id "break-out-000" :data (:point nil :vector nil))
 ) ) )
```

Figure G-1: Harness electrical definition

```
(:branches
 ((:branch-id "branch-000" :parent nil :child (1 2):start-poi (:type :connectors :index 0) :end-poi (:type :break-out :index 0) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point (#(160.0 -130.0 -1000.0) #(499.9991666643941 -150.00000000000009 -1299.999857142857) #(961.3423020772439 -150.00000000000009 -1299.999857142857) #(1489.240322716952 -297.2278503682244 -1225.0) #(2042.446802788276 -297.2361876518922 -1225.0) #(2580.287219043727 -150.00000000000009 -1299.999857142857) #(2900.000166654767 -150.00000000000009 -1299.999857142857) #(3300.0003333198288 -150.00000000000009 -1299.999857142857)) :vector nil) :proofwires (:number 2 :AWG 8))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child nil :start-poi (:type :break-out :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point (#(3300.0003333198288 -150.00000000000009 -1299.999857142857) #(3700.0004999848907 -150.00000000000009 -899.9995714285714) #(4100.000666649953 -150.00000000000009 -499.9992857142856) #(4530.0 -130.0 -500.0)) :vector nil) :proofwires (:number 1 :AWG 8))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child nil :start-poi (:type :break-out :index 0) :end-poi (:type :connectors :index 2) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1)) :clamps (:point (#(3300.0003333198288 -150.00000000000009 -1299.999857142857) #(3700.0004999848907 -150.00000000000009 -1700.000142857143) #(4130.0 -130.0 -2000.0)) :vector nil) :proofwires (:number 1 :AWG 8))
 )
 :connectors
```

```
((:connector-id "con-000" :data (:point #(160.0 -130.0 -1000.0) :vector #(0.0 0.0 1.0)))
 (:connector-id "con-001" :data (:point #(4530.0 -130.0 -500.0) :vector #(-0.0 -1.0 0.0)))
 (:connector-id "con-002" :data (:point #(4130.0 -130.0 -2000.0) :vector #(-0.0 -1.0 0.0)))
 )
:break-out
((:break-out-id "break-out-000" :data (:point #(3300.0003333198288 -150.00000000000009 -1299.999857142857) :vector nil)) )))
```

Figure G-2: Detailed harness definition after pathfinding in the Initialization

```
((:branches
 ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index 0) :end-poi
  (:break-out-type (:type "Y" :dir "positive") :type :break-out :index 0) :covering (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1))
  :clamps
  (:point
   (#(499.9991666643941 -150.00000000000009 -1299.999857142857)
    #(961.3423020772439 -150.00000000000009 -1299.999857142857)
    #(1489.240322716952 -297.2278503682244 -1225.0)
    #(2042.446802788276 -297.2361876518922 -1225.0)
    #(2580.287219043727 -150.00000000000009 -1299.999857142857)
    #(2900.000166654767 -150.00000000000009 -1299.999857142857)
    #(3260.0003166533224 -150.00000000000009 -1299.999857142857))
   :vector
   (#(0.936522496385804 1.5943790431618966e-15 -0.35060749245174716)
    #(0.9234669552752419 -0.3758820446738917 0.07695109490090922)
    #(0.9883503571696088 -0.13587038035754398 0.06857704589877504)
    #(0.9885581243014584 0.1346603931855437 -0.06796626652021638)
    #(0.9259666462851162 0.36726845664745733 -0.08774765363979373)
    #(0.9999999999999999 0.0 0.0) #(1.0 0.0 0.0))
   :normal
   (#(0.0 -1.0 -4.547473508864641e-15)
    #(-0.37699989895189906 -0.9262132995105705
      -9.835283238815315e-15)
    #(-0.13619099802995377 -0.9906825990475481
      -4.547473539590252e-15)
    #(0.1349725009301162 -0.990849344750588 0.0)
    #(0.36869059382512664 -0.9295521749879753 -4.899099368711319e-15)
    #(0.0 -1.0 0.0) #(0.0 -1.0 0.0))
   :stand-height
   (50.0 46.236324856592496 49.999999662168335 49.99999966351788
    46.41132141458103 50.0 50.0))
```

```
  :proofwires (:number 2 :AWG 8))
 (:branch-id "branch-001" :parent (:type :branches :index 0) :child nil :start-poi (:break-out-type (:type "Y" :dir "positive") :type :break-out :index 0) :end-poi (:type :connectors :index 1) :covering
  (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1))
 :clamps
  (:point
   (#(3700.0004999848907 -150.00000000000009 -899.9995714285714)
    #(4100.000666649953 -150.00000000000009 -499.9992857142856))
   :vector
   (#(0.7071066759609576 4.823322836800134e-15 0.7071068864121218)
    #(0.9008445676673157 -4.935625632301182e-16 0.4341417566927731))
   :normal
   (#(9.09494701772928e-15 -1.0 -2.27373675443232e-15)
    #(0.0 -1.0 -1.13686837721616e-15))
   :stand-height (50.000000000000014 50.000000000000014))
  :proofwires (:number 1 :AWG 8))
 (:branch-id "branch-002" :parent (:type :branches :index 0) :child nil :start-poi (:break-out-type (:type "Y" :dir "positive") :type :break-out :index 0) :end-poi (:type :connectors :index 2) :covering
  (:covering-pos ((0.1 0.24) (0.25 0.7)) :covering-thickness (1 1))
  :clamps
  (:point
   (#(3700.0004999848907 -150.00000000000009 -1700.000142857143))
   :vector
   (#(0.7644331271567392 9.88490533679983e-15 -0.6447030278394611))
   :normal (#(9.09494701772928e-15 -1.0 -4.54747350886464e-15))
   :stand-height (50.000000000000014))
  :proofwires (:number 1 :AWG 8)))
 :connectors
((:connector-id "con-000" :data
  (:point #(160.0 -130.0 -1000.0) :vector #(0.0 0.0 1.0)))
 (:connector-id "con-001" :data
  (:point #(4530.0 -130.0 -500.0) :vector #(0.0 -1.0 0.0)))
 (:connector-id "con-002" :data
  (:point #(4130.0 -130.0 -2000.0) :vector #(0.0 -1.0 0.0))))
 :break-out
((:break-out-id "break-out-000" :data
  (:point #(3300.0003333198288 -150.00000000000009 -1299.999857142857)
   :vector #(1.0 0.0 0.0))))))
```

Figure G-3: The full harness definition after the post-process in the Initialization

Appendix H. Method to include the bend radius constraint function in the cost function

As mentioned in Sub-section 6.3.3.3, direct use of the weight $w_{bending}$ together with bend radius constraint function $C(x)_{bendradius}$ in cost function $\theta(x)$ will make $w_{bendradius}C(x)_{bendradius}$ (or $w_{bendradius}C(x)^j_{bendradius}$ of branch j) too large or too small. As shown on the left-hand side of Figure H-1, when $C(x)^j_{bendradius}$ is distant from the origin point (e.g. 4 or 5), $w_{bendradius}C(x)^j_{bendradius}$ will be very large and therefore the penalty of other constraints $w_iC(x)_i$ and the original cost function $f(x)$ are almost negligible. When $C(x)^j_{bendradius}$ is close to the origin point (e.g. 0.3), $w_{bendradius}C(x)^j_{bendradius}$ will be too small and its penalty in $\theta(x)$ is almost negligible and $C(x)^j_{bendradius}$ is very difficult to be satisfied (i.e. $C(x)^j_{bendradius} \le 0$) although this is almost satisfied.

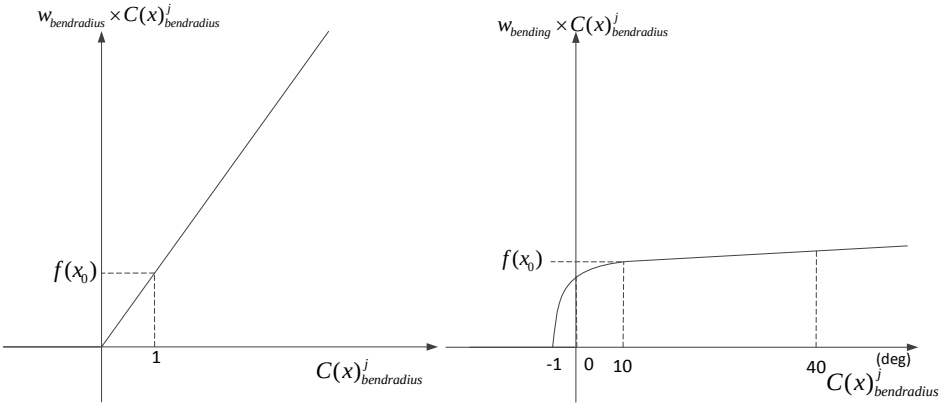


Figure H-1: Product of the bend radius constraint function (left: original; right: piecewise) and weight

In order to still provide the trend to the optimizer but make $w_{bendradius}C(x)_{bendradius}$ in proportion to other $w_iC(x)_i$ and $f(x)$ (i.e. the right-hand side of Figure H-1), a piecewise function shown in Equation (H.1) is introduced.

$$C^j_{bendradius} = \begin{cases} 0 & \text{if } C^j_{bendradius} \le -1 \\ atan(C^j_{bendradius} + 1) & \text{if } -1 < C^j_{bendradius} \le N \\ atan(N) + (C^j_{bendradius} + 1 - N)/(1 + N^2) & \text{if } N < C^j_{bendradius} \end{cases} \quad \text{(H.1)}$$

When branch j does not violate the minimum allowed bend radius (i.e. $C^j_{bendradius} \le 0$), $C^j_{bendradius} = 0$.

When $-1 < C^j_{bendradius} \le N$, $C^j_{bendradius}$ is an arctangent function. $atan(C^j_{bendradius})$ is very sensitive to $C^j_{bendradius}$ when $C^j_{bendradius}$ is a small number. This guarantees that any small violation is not allowed and consequently will be eliminated. When the design variables are in feasible areas (i.e. $C^j_{bendradius} \le 0$), the penalty to move them back to an infeasible area is high since $atan(C^j_{bendradius} + 1)$ around $C^j_{bendradius} = 0$ is high.

The arctangent function approach works well when $C^j_{bendradius}$ is small (e.g. 5 or 10). When $C^j_{bendradius}$ increases (e.g. to 50 or 100), $C^j_{bendradius}$ approaches $\pi/2$ and $atan(\Delta C^j_{bendradius}) \approx 0$. Hence, the bend radius analysis tool cannot provide enough heuristic information to the optimizer and consequently the design gets stuck in the plateau of the infeasible area. In order to handle this problem, the third function piece is proposed. This is the extension of $atan'(N)$ from point N. This definition makes the piecewise function smooth at point N and still sensitive to the change of $C^j_{bendradius}$ at the large violation area but also not too large. Choosing a suitable number for N is the key point to the success of this method. Too large N will make $atan(C^j_{bendradius})$ less sensitive to $C^j_{bendradius}$ and vice versa. In this research N is set to 10 after various routing tests.

The bend radius constraint function $C(x)_{bendradius}$ is the summation of the constraint function of all branches, namely $C(x)_{bendradius} = \sum_{j=1}^{M} C^j_{bendradius}$; and it together with weight $w_{bending}$ constitute the bend radius constraint part in objective function $\theta(x)$.

Appendix I. Input data for the case studies

```
#|
;;; --------------------------------
;;; Last update: 01-03-2014
;;; Affiliation: TU/Delft
;;; Author(s): Zaoxu Zhu
;;; --------------------------------
|#

;; the wire harness input data for fuselage section front
(
;;;harness-0;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(:branches
 ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index 0) :end-poi (:type :break-out :index 0) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 1))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child (3 4) :start-poi (:type :break-out :index 0) :end-poi (:type :break-out :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 10))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child (5 6) :start-poi (:type :break-out :index 0) :end-poi (:type :break-out :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 8 :AWG 1))
  (:branch-id "branch-003" :parent (:type :branches :index 1) :child nil :start-poi (:type :break-out :index 1) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  (:branch-id "branch-004" :parent (:type :branches :index 1) :child nil :start-poi (:type :break-out :index 1) :end-poi (:type :connectors :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  (:branch-id "branch-005" :parent (:type :branches :index 2) :child (7 8) :start-poi (:type :break-out :index 2) :end-poi (:type :break-out :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 10))
  (:branch-id "branch-006" :parent (:type :branches :index 2) :child nil :start-poi (:type :break-out :index 2) :end-poi (:type :connectors :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 1))
  (:branch-id "branch-007" :parent (:type :branches :index 5) :child nil :start-poi (:type :break-out :index 3) :end-poi (:type :connectors :index 4) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  (:branch-id "branch-008" :parent (:type :branches :index 5) :child nil :start-poi (:type :break-out :index 3) :end-poi (:type :connectors :index 5) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  )
  :connectors
 ((:connector-id "con-000" :data (:point #(-1250.0 -1220.0 -5850.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(-2278.0 445.0 -4240.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-002" :data (:point #(-2118.0 445.0 -4240.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-003" :data (:point #(-1250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
```

```
  (:connector-id "con-004" :data (:point #(-2278.0 445.0 -440.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-005" :data (:point #(-2118.0 445.0 -440.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 ((:break-out-id "break-out-000" :data (:point nil :vector nil))
  (:break-out-id "break-out-001" :data (:point nil :vector nil))
  (:break-out-id "break-out-002" :data (:point nil :vector nil))
  (:break-out-id "break-out-003" :data (:point nil :vector nil))
  )
  )


;;;harness-1;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:branches
 ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index 0) :end-poi (:type :break-out :index 0) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 2))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child (3 4) :start-poi (:type :break-out :index 0) :end-poi (:type :break-out :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 10))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child (5 6) :start-poi (:type :break-out :index 0) :end-poi (:type :break-out :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 8 :AWG 2))
  (:branch-id "branch-003" :parent (:type :branches :index 1) :child nil :start-poi (:type :break-out :index 1) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  (:branch-id "branch-004" :parent (:type :branches :index 1) :child nil :start-poi (:type :break-out :index 1) :end-poi (:type :connectors :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  (:branch-id "branch-005" :parent (:type :branches :index 2) :child (7 8) :start-poi (:type :break-out :index 2) :end-poi (:type :break-out :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 10))
  (:branch-id "branch-006" :parent (:type :branches :index 2) :child nil :start-poi (:type :break-out :index 2) :end-poi (:type :connectors :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 2))
  (:branch-id "branch-007" :parent (:type :branches :index 5) :child nil :start-poi (:type :break-out :index 3) :end-poi (:type :connectors :index 4) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
  (:branch-id "branch-008" :parent (:type :branches :index 5) :child nil :start-poi (:type :break-out :index 3) :end-poi (:type :connectors :index 5) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 10))
   )
  :connectors
 ((:connector-id "con-000" :data (:point #(1250.0 -1220.0 -5850.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(2278.0 445.0 -4240.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-002" :data (:point #(2118.0 445.0 -4240.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-003" :data (:point #(1250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-004" :data (:point #(2278.0 445.0 -440.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-005" :data (:point #(2118.0 445.0 -440.0) :vector #(0.0 0.0 -1.0)))
```

```
 )
 :break-out
 ((:break-out-id "break-out-000" :data (:point nil :vector nil))
  (:break-out-id "break-out-001" :data (:point nil :vector nil))
  (:break-out-id "break-out-002" :data (:point nil :vector nil))
  (:break-out-id "break-out-003" :data (:point nil :vector nil))
  )
  )
;;;;harness-2;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:branches
 ((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 5))  )
 :connectors
 ((:connector-id "con-000" :data (:point #(-250.0 -1220.0 -5850.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(-250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 (nil)
  )
;;;harness-3;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:branches
 ((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 5))  )
 :connectors
 ((:connector-id "con-000" :data (:point #(-750.0 -1220.0 -5850.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(-750.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 (nil)
  )
;;;harness-4;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:branches
 ((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 5))  )
 :connectors
 ((:connector-id "con-000" :data (:point #(250.0 -1220.0 -5850.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 (nil)
  )
;;;harness-5;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

```
 (:branches
 ((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 5))  )
 :connectors
 ((:connector-id "con-000" :data (:point #(750.0 -1220.0 -5850.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(750.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 (nil)
  )
;;;;harness-6;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:branches
 ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index 0) :end-poi (:type :break-out :index 0) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 15))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child (3 4) :start-poi (:type :break-out :index 0) :end-poi (:type :break-out :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 15))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child nil :start-poi (:type :break-out :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 15))
  (:branch-id "branch-003" :parent (:type :branches :index 1) :child nil :start-poi (:type :break-out :index 1) :end-poi (:type :connectors :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 15))
  (:branch-id "branch-004" :parent (:type :branches :index 1) :child nil :start-poi (:type :break-out :index 1) :end-poi (:type :connectors :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 1 :AWG 15))
   )
  :connectors
 ((:connector-id "con-000" :data (:point #(-2228.0 395.0 -5650.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(-2228.0 395.0 -2750.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-002" :data (:point #(2241.0 395.0 -5650.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-003" :data (:point #(2241.0 395.0 -3350.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 ((:break-out-id "break-out-000" :data (:point nil :vector nil))
  (:break-out-id "break-out-001" :data (:point nil :vector nil))
  )
  )


  )
```

Figure I-1: Electrical definition of a wire harness routed in the fuselage-front section

```
#|
;;; --------------------------------
;;; Last update: 02-03-2014
;;; Affiliation: TU/Delft
;;; Author(s): Zaoxu Zhu
;;; --------------------------------
|#

;; the wire harness input data for fuselage section back
(
;;;harness-0;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(:branches
 ((:branch-id "branch-000" :parent nil :child (1 2) :start-poi (:type :connectors :index 0) :end-poi (:type :break-out :index 0) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 30 :AWG 15))
  (:branch-id "branch-001" :parent (:type :branches :index 0) :child (3 4) :start-poi (:type :break-out :index 0) :end-poi (:type :break-out :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 16 :AWG 15))
  (:branch-id "branch-002" :parent (:type :branches :index 0) :child nil :start-poi (:type :break-out :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 30 :AWG 15))
  (:branch-id "branch-003" :parent (:type :branches :index 1) :child (5 6) :start-poi (:type :break-out :index 1) :end-poi (:type :break-out :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 15))
  (:branch-id "branch-004" :parent (:type :branches :index 1) :child (7 8) :start-poi (:type :break-out :index 1) :end-poi (:type :break-out :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 6 :AWG 15))
  (:branch-id "branch-005" :parent (:type :branches :index 3) :child nil :start-poi (:type :break-out :index 2) :end-poi (:type :connectors :index 2) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 8 :AWG 15))
  (:branch-id "branch-006" :parent (:type :branches :index 3) :child (9 10) :start-poi (:type :break-out :index 2) :end-poi (:type :break-out :index 4) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 15))
  (:branch-id "branch-007" :parent (:type :branches :index 4) :child (11 12) :start-poi (:type :break-out :index 3) :end-poi (:type :break-out :index 5) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 4 :AWG 15))
  (:branch-id "branch-008" :parent (:type :branches :index 4) :child nil :start-poi (:type :break-out :index 3) :end-poi (:type :connectors :index 3) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 15))
  (:branch-id "branch-009" :parent (:type :branches :index 6) :child (13 14) :start-poi (:type :break-out :index 4) :end-poi (:type :break-out :index 6) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 30 :AWG 15))
  (:branch-id "branch-010" :parent (:type :branches :index 6) :child nil :start-poi (:type :break-out :index 4) :end-poi (:type :connectors :index 4) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 28 :AWG 15))
  (:branch-id "branch-011" :parent (:type :branches :index 7) :child nil :start-poi (:type :break-out :index 5) :end-poi (:type :connectors :index 5) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 2 :AWG 15))
  (:branch-id "branch-012" :parent (:type :branches :index 7) :child nil :start-poi (:type :break-out :index 5) :end-poi (:type :connectors :index 6) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires
```

```
(:number 2 :AWG 15))
  (:branch-id "branch-013" :parent (:type :branches :index 9) :child (15 16) :start-poi (:type :break-out :index 6) :end-poi (:type :break-out :index 7) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 15))
  (:branch-id "branch-014" :parent (:type :branches :index 9) :child nil :start-poi (:type :break-out :index 6) :end-poi (:type :connectors :index 7) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 30 :AWG 15))
  (:branch-id "branch-015" :parent (:type :branches :index 13) :child nil :start-poi (:type :break-out :index 7) :end-poi (:type :connectors :index 8) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 30 :AWG 15))
  (:branch-id "branch-016" :parent (:type :branches :index 13) :child nil :start-poi (:type :break-out :index 7) :end-poi (:type :connectors :index 9) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 30 :AWG 15))
   )
  :connectors
 ((:connector-id "con-000" :data (:point #(-1250.0 -1220.0 -5930.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-001" :data (:point #(-1250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-002" :data (:point #(-1692.0 1496.0 -5210.0) :vector #(0.0 0.0 1.0)))
  (:connector-id "con-003" :data (:point #(-2198.0 475.0 -3500.0) :vector #(0.0 1.0 0.0)))
  (:connector-id "con-004" :data (:point #(-1807.0 1496.0 -4000.0) :vector #(0.0 1.0 0.0)))
  (:connector-id "con-005" :data (:point #(-2188.0 475.0 -2300.0) :vector #(0.0 1.0 0.0)))
  (:connector-id "con-006" :data (:point #(-2198.0 475.0 -3000.0) :vector #(0.0 1.0 0.0)))
  (:connector-id "con-007" :data (:point #(-1707.0 1546.0 -1500.0) :vector #(0.0 0.0 -1.0)))
  (:connector-id "con-008" :data (:point #(1697.0 1546.0 -1820.0):vector #(0.0 0.0 1.0)))
  (:connector-id "con-009" :data (:point #(1707.0 1546.0 -788.0) :vector #(0.0 0.0 -1.0)))
  )
 :break-out
 ((:break-out-id "break-out-000" :data (:point nil :vector nil))
  (:break-out-id "break-out-001" :data (:point nil :vector nil))
  (:break-out-id "break-out-002" :data (:point nil :vector nil))
  (:break-out-id "break-out-003" :data (:point nil :vector nil))
  (:break-out-id "break-out-004" :data (:point nil :vector nil))
  (:break-out-id "break-out-005" :data (:point nil :vector nil))
  (:break-out-id "break-out-006" :data (:point nil :vector nil))
  (:break-out-id "break-out-007" :data (:point nil :vector nil))
  )
  )
;;;harness-1;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  (:branches
 ((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1) :covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 2))
  )
  :connectors
 ((:connector-id "con-000" :data (:point #(1250.0 -1220.0 -5920.0) :vector #(0.0 0.0 1.0)))
```

```
 (:connector-id "con-001" :data (:point #(1250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
 )
:break-out
(nil)
 )
;;;harness-2;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 (:branches
((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1)
:covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 5))  )
:connectors
((:connector-id "con-000" :data (:point #(-750.0 -1220.0 -5920.0) :vector #(0.0 0.0 1.0)))
 (:connector-id "con-001" :data (:point #(-750.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
 )
:break-out
(nil)
 )
;;;harness-3;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 (:branches
((:branch-id "branch-000" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1)
:covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 5))  )
:connectors
((:connector-id "con-000" :data (:point #(-250.0 -1220.0 -5920.0) :vector #(0.0 0.0 1.0)))
 (:connector-id "con-001" :data (:point #(-250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
 )
:break-out
(nil)
 )
;;;harness-4;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 (:branches
((:branch-id "branch-005" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1)
:covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 0))  )
:connectors
((:connector-id "con-000" :data (:point #(250.0 -1220.0 -5920.0) :vector #(0.0 0.0 1.0)))
 (:connector-id "con-001" :data (:point #(250.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
 )
:break-out
(nil)
 )
;;;harness-5;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
 (:branches
((:branch-id "branch-006" :parent nil :child nil :start-poi (:type :connectors :index 0) :end-poi (:type :connectors :index 1)
:covering (:covering-pos ((0 1)) :covering-thickness (1)) :clamps (:point nil :vector nil) :proofwires (:number 10 :AWG 0))  )
```

```
:connectors
((:connector-id "con-000" :data (:point #(750.0 -1220.0 -5920.0) :vector #(0.0 0.0 1.0)))
 (:connector-id "con-001" :data (:point #(750.0 -1220.0 -130.0) :vector #(0.0 0.0 -1.0)))
 )
:break-out
(nil)
 )
 )
```

Figure I-2: Electrical definition of a wire harness routed in the fuselage-rear section

Acknowledgements

Time flies. It's almost 7 years since the start of this Ph.D. research. Looking back on this period, I first would like to acknowledge my promoter, Michel van Tooren, who enables me to work on this topic. Your full trust means that I am allowed to explore different directions and try different methods for the research problem, and your feedback helps me to find the best solution from those methods.

I also would like to thank the China Scholarship Council (CSC) and the great country at the back of it for the study-abroad opportunity it provides and the fund for the first 4 years of this Ph.D. research. Cees Timmers, the coordinator of the CSC-TU Delft collaboration, provides me enough support when I was applying for this position. I still remember the interview I had with you at a hotel in Beijing. Thank you for your work and the positive feedback to Michel.

This research was also supported by Fokker Elmo, the Dutch wire harness company. The 2-years collaboration not only means that I can consolidate the Ph.D. work and finalize this dissertation but also provides me a chance to learn from the experienced engineers of the company and to tackle the new challenges in this research domain. I would like to give my appreciation to the people who work for this collaboration, Simon Taylor, Kees Nuyten, Charly Heuts, Dew Ponit, and especially to John Van Lugtenburg – the project coordinator.

Special thanks I want to give to Gianfranco La Rocca, my daily supervisor. You have helped me a lot on both the technical work and reporting skills. I did learn a lot during this period although making progress was painful sometimes for both of us.

During my Ph.D. period, our group has been reorganized. I would like to thank Leo Veldhuis, our new group leader, for your support on this research and your feedback from the perspective of a typical engineering problem rather than of this specific technical domain. I also would like to thank Lin, Bettie, Nana, and Michiel – the secretaries and IT technician of the groups – for your warm support.

In the aerospace domain, few people's work is related to wire harnesses. Tobie, my ex-colleague, is one of these. I enjoyed the discussion with you on this domain and the inspirations got from you are very helpful. Here I also want to thank all my (ex-) colleagues, for the every moment we had together, from the coffee corner to the canteen; from a normal working day to the team-building activities. Extra appreciation is given to all my Chinese colleagues. You mean that I am not far away from my homeland.

KE-works is a spin-off of TU Delft. Because of the similar research interesting and the hospitality of its co-founders Joost Schut, Jochem Berends, and Stefan van der Elst, I stayed there for about three years. I really had a good time to sit together with all the KE-workers. I got useful suggestions and feedback on my research; I learned how to play table football; and I very enjoyed the team-building activities – KE-onions. In addition, the most important thing is that I understood the typical Dutch lunch culture there.

Claire Taylor has read the draft of this dissertation to check the grammar and spelling errors. Thank you for the corrections and suggestions. I learned a lot from our back and forth email communication.

Chenkai, my elder son, you come means I am a father. When you were born, it was a difficult period of my research work. I got stuck when I was trying to find a suitable approach to handle the research problem. With your accompanying, I not only solved the problems in my work, but also learned how to take care a new-born baby, a toddler, and a child, from

changing a diaper to sending you to the day-care and primary school on time. Yuxi, my younger son, you come means I can validate the knowledge and experience I learned when taking care your brother. As you may not know yet, validation is very important in your papa's work. I really love you both cute little men.

I really appreciate my parents, Zhu Shuyang and Wu Huailing. You do not only bring me up but also take care of your grandsons now. I know they are endearing and you love them. But I have to say that your babysitting is very helpful.

Last but not least, my deepest gratitude and love go to my wife, Cui Fang. Thank you for the contribution to our family, the many years' accompanying, and the unconditional love. You are the one in my life.

About the author

Zaoxu Zhu was born in September 1984, in Shouxian, Anhui province, P.R. China.

He finished his high-school education in Shouxian No.1 High School, in 2002. In the same year, he started his bachelor study in China Agriculture University, in Beijing, China and obtained the Bachelor of Science (BSc) degree in 2006.

In September 2006, he started his master study in Beijing University of Aeronautics and Astronautics (also known as Beihang University) in the major of Mechatronic Engineering and obtained the Master of Science (MSc) degree in 2009.

Since December 2009, he started his Ph.D. research in the Faculty of Aerospace Engineering, Delft University of Technology, the Netherlands. The aim of his research work is to automate the geometric design of aircraft Electrical Wiring Interconnection System by using the advanced technologies, such as Knowledge Based Engineering and Multidisciplinary Design and Optimization.