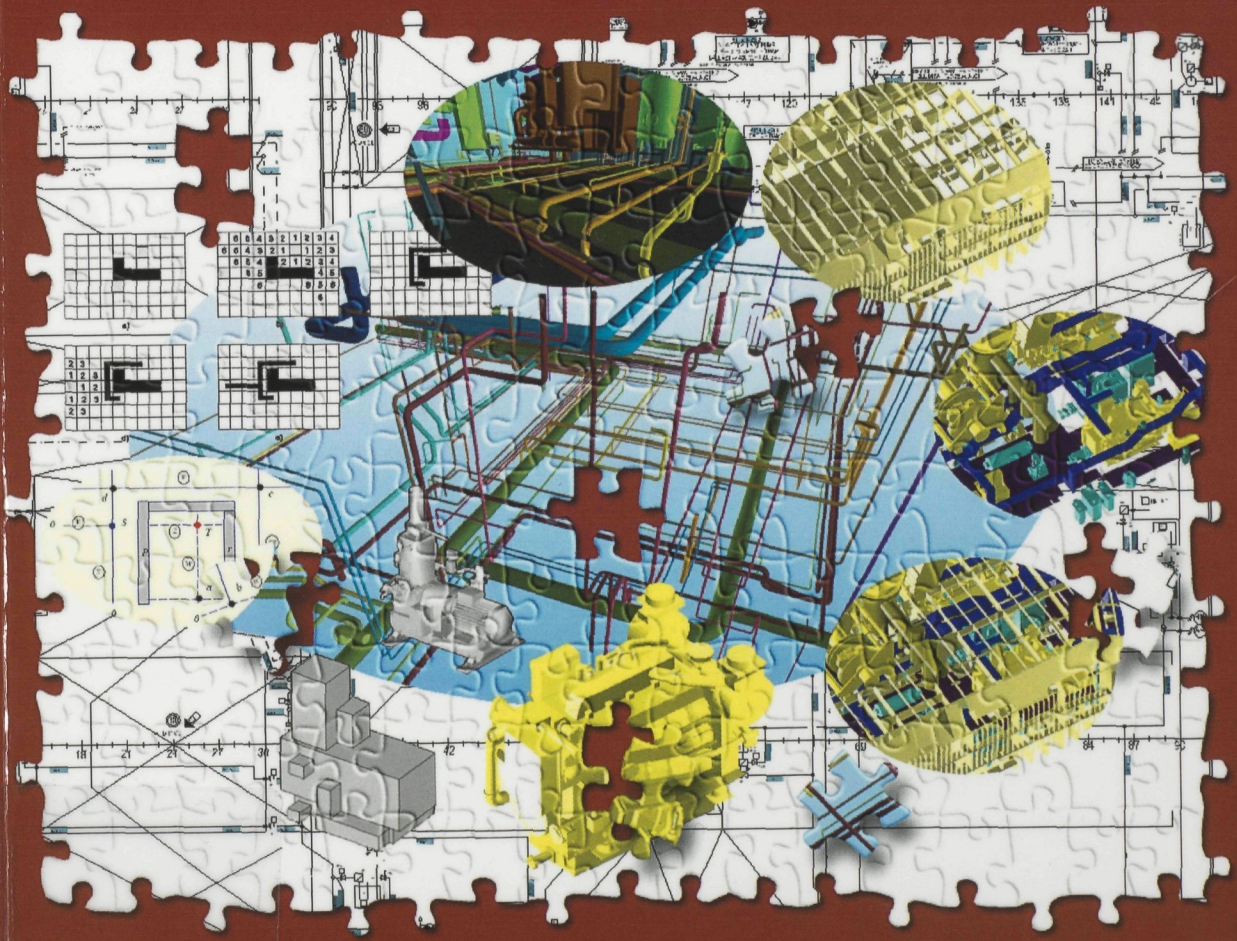
PIPE ROUTING FRAMEWORK FOR DETAILED SHIP DESIGN



ANDI ASMARA

924370/5106142

Pipe Routing Framework for Detailed Ship Design

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 woensdag 10 juli 2013 om 15.00 uur

door

Andi ASMARA

master of science in automation and robotics, Universität
Dortmund, Duitsland

geboren te Jakarta, Indonesië

Dit proefschrift is goedgekeurd door de promotor:

Prof.dr.ir. U. Nienhuis MBA

Samenstelling promotiecommissie:

Rector Magnificus, voorzitter
Prof.dr.ir. U. Nienhuis MBA, Technische Universiteit Delft, promotor
Prof.Dr.-Ing. S. Krueger, TU Hamburg-Harburg
Prof.ir. J.J. Hopman, Technische Universiteit Delft
Prof.ir. D. Stapersma, Technische Universiteit Delft
Prof.dr. I. Horvath, Technische Universiteit Delft
Dr.ir. J.M.G. Coenen, Technische Universiteit Delft

Published by

VSSD, Delft, The Netherlands
internet: http://www.vssd.nl/hlf
e-mail: hlf@vssd.nl

ISBN 97890-6562-326-3

Copyright © 2013 by Andi Asmara

All rights reserved. No part of the material protected by this copyright notice may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the prior permission of the author.

Contents

Chapter 1

Introduction

1.1 Process Innovations in Shipbuilding

The shipbuilding industry is a mature market. True to its nature, the dominance within the shipbuilding industry has shifted from the European markets, penalized by their high cost of production, to the (at least sometime) low-cost Asian destinations such as South Korea, Japan and China. This happened mostly for relatively non-complex ships like oil tankers, dry bulk carriers, and container vessels. To maintain their market competitiveness, the European shipyards and co-makers need to dig into niche markets by concentrating on building specialized vessels, because the cost of production for this kind of ships is relatively less important. To be more competitive, European shipyards must also compensate their disadvantage of higher production cost with their skill to design and build complex ships. However, the "low-cost" Asian shipyards are continuously expanding their knowledge to build complex ships. Therefore the "more-expensive" European shipyards are forced to reduce cost.

This situation forces the European shipyards to be more efficient, in design and production processes. Product and process innovation have become main issues. Projects such as LeaderSHIP 2015 at the European Union level, Concurrent Engineering, Planning, Pricing, and Production (CE3P) and Integraal Samenwerken in The Netherlands, are aimed at improving the current level of design and production processes in shipbuilding.

One of the main subjects in those projects is to conduct research and development to utilize information and communication technology to improve the design and engineering processes, and to implement that in the real process. Improving process control; simulating the shop-floor and on-board production processes; developing a multi-party communication framework; and innovative ways to route pipes and other conduits are some of the research subjects.

1.2 Innovative Ways to Route Pipes in Ships

A ship has many systems and subsystems that consist of a large number of pipes. Even a small vessel might have more than one thousand pipe and in a larger vessel this number can reach three to four thousands pipes and more, and all pipes must be routed. Pipe routing is one of the most important activities during detailed design because many other detailed design activities depend on it. Also it is important because this activity consumes a significant part of the detailed design man-hours as the pipe systems in a ship typically consist of thousands of pipe elements. Park and Storch [2002] mention that the time that is needed can reach 50% of the total design process time. That number is significant, because according to American Bureau of Shipping, the labor cost is around 60% of total cost of ship. Based on our interview with the engineering department of a shipyard, the process of pipe routing for a middle size complex ship can consume 30-40 thousand man hours.

There are basically four main characteristics of ship building that encourage people to utilize computers to route pipes in the design process. The first characteristic is that many of the middle to large size special-purpose ships normally have a unique specification which leads to the necessity to redo the design process for each vessel. The high sensitivity of pipe design to changes in the specification is the second one. The third one is the time needed to route the pipes is substantial and sometimes can be the bottle neck in the design and production process. Last but not least is the fact that the pipe routing is largely done manually by pipe engineers who need many years of experience to do this properly and efficiently.

The design of the piping systems consumes a large part of the engineering effort for a modern ship. The fact that makes it so significant is that the nature of ship production is different from other vehicles e.g. cars. Since the 1920s, nearly all cars have been mass-produced; they were designed once and produced in a large quantity. In ship production, especially for the building of specialized ships like dredgers, offshore and naval vessels, every ship that is built has a different specification that requires the design process to be done for each ship. Even ships with the same functionality often have a different specification.

Pipes are needed for distribution of fluids and gases between pieces of equipment in the ship. Due to the complexity of the systems that are needed to support the operability of the ship, the total amount of pipes needed is large. Also the variability is large; only few pipe pieces are the same in different ships. The complexity of pipe systems is compounded by the fact that the space available for pipe systems in a ship is very limited. Normally, in the end, most of the available space will be occupied by the pipes. If a small modification is subsequently needed, e.g. if there is a specification change for the size of a component and/or the component needs to be moved to a different place, all the pipes that are connected to that component need to be rerouted or at least need to be modified. Moreover and more awkwardly, when the compartment is already crowded with pipes, the changes might affect other pipes that are not immediately connected

to that component, but lie in the vicinity.

In the day to day practice of ship design, the iterative nature of the entire design process is such that structural changes, or changes of the equipment specifications or relocation of elements of the ship are prone to occur, requiring time-consuming rework for any pipe affected.

In order to be more cost effective in the production stage, most of the pipes should be installed during the pre-outfitting stage. It means that the pipes that belong to one section of the ship must be installed immediately after the steel construction of that section is ready, so when that section is assembled to the hull on the slipway, all pipes are already in their place.

Nowadays in practice, the pipe routing process is largely done manually by a pipe designer using CAD software which assists in checking for collisions and defining the details of pipes e.g. the pipe bend radius and its specifications. Therefore the experience of the designers is the most important ingredient for this process. The assessment of the design results is a subjective matter due to heterogeneous design preferences and conventions of designers, Kang et al. [1996].

Much research has been done to develop innovative ways to route pipes and most are aiming to have a capability to route pipes automatically. Automatic pipe-routing has been a research topic for a long time resulting in various approaches, not only in the ship production application, Wangdahl, Pollock, and Woodward [1974], but also in process plant design, Matsui et al. [1979] and Guirardello and Swaney [2005]. The research started with 2D workspace and simple obstacles, Wangdahl et al. [1974], and gradually extended to the stage of 3D workspace with multiple constraints and multiple objectives, Park and Storch [2002]. In terms of the optimization techniques, either deterministic, Newell [1972], non-deterministic, Fan et al. [2006] or a combination of both methods have been used to improve the results, Asmara and Nienhuis [2008].

Unfortunately most of the research has focused on how to route the pipe itself, without having due regard to the data that is needed to be prepared beforehand, nor what to do with the result afterward, e.g. Newell [1972], Sandurkar and Chen [1998] and Zhu and Latombe [1991]. Furthermore, most of the research on the automatic pipe-routing problem is based on a simple environment that consists of a small number of pipes, obstacles and conditions, such as Sandurkar and Chen [1998] and Zuurmond [2004]. Only a few of them consider the practical aspects of pipe routing in ships, Park and Storch [2002].

Until the date of this thesis (2013), it was questionable whether it is possible to develop the complete methodology to perform automatic pipe-routing in the real ship design process and satisfy all the rules and standards in ship design, engineering and production.

In our attempt to improve the pipe routing process, we made a thorough investigation of the current pipe routing process in practice. This has been done by carrying out extensive interviews and discussions with experienced pipe engineers, and also investigating their work results. Eventually, the quality of the manual routing by experienced pipe engineers is excellent and does not require many

improvements. What is more important is to shorten the time needed for pipe routing processes while maintaining the same quality compared to the current process. This can be achieved without fundamental changes to the commonly used manual pipe routing method but takes advantage of computer power by translating manual processes into computer procedures.

Those facts motivate the main research objectives of this thesis:

1. to what extent can the expertise of a pipe engineer be identified and translated into procedures that lend themselves to be computerized?
2. how should we put together a pipe routing methodology based on the results of the first question and combine it with advanced optimization techniques in a practically applicable method?

In the next sections, we describe several issues related to the development of the pipe routing methodology. These consist of the current situation of the pipe routing process, pipe routing know-how and design requirements. For each issue, we introduce a corresponding research question.

1.3 Pipe Routing Process in Ships

1.3.1 Piping Design Phases

In the design of a ship, there are two main skill areas involved, the naval architect and the marine engineer. The naval architect is concerned with the hull, its construction, form, habitability and ability to endure its environment. The marine engineer is responsible for the various systems inside the ship necessary to support the ship's functionality. More specifically, this refers to the machinery required for propulsion, steering, anchoring and ship securing, cargo handling or other mission-relevant functions, air conditioning, power generation and its distribution, see Klein Woud and Stapersma [2003] ,Harrington [1992], Taylor [1996].

If we look in more detail to the various main systems that support the functionality of the ship, each of these needs some separate sub-systems to support its function. For example the propulsion system needs some other systems to support its main components. The sub-systems that are needed by the main engine normally consist of a fuel oil system, an oil lubrication system, a starting air system, cooling water system (sea and fresh water) and an exhaust gas system.

Every system in a ship, regardless if it is a main or auxiliary system, needs pipes or ducts to transfer liquid or gas between its component parts (including tanks), and/or needs cables to deliver electrical power to the equipment and receive electronic signals. Therefore the piping, HVAC, and cabling is very important in a ship and can be said to be as important as the blood vessel system in a human body. Indeed, also the walkway and stairs for a human being may be seen as a distribution channel. The complexity of arranging the distribution channels is high.

Although our methodology is aimed to be also suited for distribution of electric

power, air and even people, we, for now, concentrate on the piping system. The design of the piping system is done in four different phases; conceptual design, preliminary design, contract design, and detailed design, Harrington [1992]. During concept design, a tentative list of requirements is developed based on the available ship characteristics. If sufficient detail arrangement is available, a preliminary check can be performed to make sure that the major pipe systems can be accommodated. However, normally in this phase the available data is insufficient to develop independent cost estimates for each system. In practice, the cost estimation is usually extrapolated from data for existing ships of similar design.

The major piping system components are selected and arranged in the ship during the preliminary design. Preliminary estimates of system flows, pressure and temperature are made to support component selection. Piping system component selection needs to meet the piping system performance requirements with due consideration of the weight, cost, and reliability. The performance requirements of the piping systems are determined on the basis of the ship mission, size, operating profile, main machinery, and other factors.

In this design phase, the schematics that depict the interconnection between the components in piping systems, called piping system diagrams, are started to be developed with a preliminary level of detail. The approximate locations of major components and the largest pieces of piping are determined and the general arrangement is prepared and reviewed to ensure that enough space is available for piping and other distributive systems.

In the contract design, the additional details and specifications of each system are developed based on the outlines that are defined in the preliminary design phase. Contract guidance drawings are developed to illustrate relationships and interconnections between systems that may not be understood easily from a written specification. These drawings together with specifications define the system sufficiently to ensure that the owner's requirements for performance and quality are mutually understood and agreed, and to permit the shipbuilder to prepare a bid.

The last phase of the piping system design is the detailed design, and it produces a full definition of every pipe system element in a drawing format that is used to manufacture all parts of the systems, and also to install them in the ship. While the first three design phases are primarily focused on system performance, this phase is focused on construction.

The detailed design is begun with a completion of the piping system diagrams as more detailed and final data become available. The piping system diagrams are used to ensure that the systems will meet the specification requirements. They also help to ensure compatibility of all elements in the systems with each other and also with other elements like machinery interfaces. Normally, the information of the system arrangement is included in the piping system diagram with varying level of detail. The piping system diagrams contain the foundation of every piping system e.g. the component symbols, pipe size and specification, valve description, flow direction, and other useful information. The quality and clarity of the piping

diagrams are important because they serve as the baseline for all processes in the detailed design phase.

After the piping system diagrams have a sufficient level of detail, the pipe routing process can be started. The basic approach to pipe routing is developing a collision free route of a pipe between two or more connection points in a 3D, obstacle-scattered environment, according to the rules and standards. Pipe routing is difficult because, among other reasons, the pipe is generally subject to multiple design constraints. The most important constraint is that the pipe routing solution must comply with the marine classification and regulations. In addition, the space available in a ship and allowed to be used for pipes is usually limited. This condition is especially true for certain areas of the ship, such as the engine room of a ship. One more unavoidable aspect that makes this activity even harder is the fact that the specifications of the ship are changing quite frequently during the design process often requiring re-work.

The piping system diagrams are not the only information that is needed to perform a pipe routing process. It also needs the 3D model of each system component to know the exact location of the connection point of the pipe and to prevent the collision between pipes and the system components. The 3D model of the hull and superstructure is also needed to ensure that pipes can be routed optimally without collision or unnecessary penetration of the hull construction. Besides that, additional information of the system components might be needed, such as the information about pipe systems.

When a pipe has been routed, it is divided into several parts, called pipe spools and the pipe spool drawing is generated to be used in production and installation of the pipe in the ship.

If we look at those four design phases, the pipes are actually routed during the detail design phase and as mentioned in the previous sections, the pipe routing process requires many working hours. Therefore, it is logical to focus on this stage.

However, we also draw attention to the benefits to be able to route pipes automatically in the pre-contractual phase. During this phase, the cost estimation of pipes is needed. Currently, it is merely estimated using statistical methods based on data from previous ships. The implementation of automatic pipe routing in this phase will give more confidence on the estimated cost. Therefore we need to investigate to answer the first research question below:

Research question #1: *On which design phases should we focus and what is the reason for that?*

1.3.2 Pipe Routing Knowhow

As we mentioned above, the pipe routing process is largely done manually by a pipe designer using CAD software which assists in checking for collisions and

defining the details of pipes. The assistance that can be provided by current versions of CAD software is limited to the specification that immediately relates to the pipe and the standard components that are attached to that pipe. It does not take into account the type of the system to which the pipe belongs and/or the category of area where the pipe is routed. It will not give a warning if a pipe is routed in violation of the marine classification, for example if a pipe from a fuel oil system was routed above the combustion engine.

As described in Subsection 1.3.1, a pipe engineer needs complete information before he can start to route a pipe. Since our goal is to incorporate the manual routing process into the automatic one, we need to identify what data is available and needed.

Routing is a difficult task for pipe designers and the expertise of the pipe designer is very important to assure that all pipes are routed in a proper way. Currently, formal guidance on how to route a pipe in a ship does not exist. We only can find specification requirements of pipes and systems in the standardization books such as in a marine classification guide, or in scattered parts of a few books that describe piping systems. Nevertheless, there are some informal rules that have evolved into a common body of knowledge among pipe designers on how to route a pipe that belongs to a certain system, inside a particular area in the ship. Most of that common knowledge is based on the logical way of routing that is acquired by experience. An example is that pipes that run in the same direction and lie close together should be routed in parallel. Also pipes should be routed in a certain order based on the diameter of each pipe and the system that particular pipe belongs to.

In this thesis, we investigate the common knowledge of how to route pipe, and the adoption of that knowledge to be implemented in the methodology. Thus, we seek answers to the second, third and fourth research questions below:

Research question #2: What information is needed to perform the pipe routing process, who is responsible to provide it and how can one get it?

Research question #3: What is the common knowledge to route pipes that is used as guidance by a pipe engineer?

Research question #4: Which knowledge from Research Question #3 should be adopted in our proposed methodology and to what extent can the current practical knowledge be absorbed into a programmable methodology?

1.3.3 Quality of the Pipe Routed

In the current practice, pipes that already routed are claimed to have a good quality if the following conditions are fulfilled:

1. The functional requirements are satisfied; pipes are connecting pieces of equipment perfectly without having excessive length, unnecessary bends and without collisions,
2. It complies with maritime rules and regulations as imposed by classification societies,
3. It fulfills the subjective values of the pipe designer who has routed those pipes; this judgment is very subjective but still in line with the common knowledge of how to route a pipe.

Judging if the above conditions are met, can only be performed by manually examining the routed pipes. In addition to the subjectivity involved, analyzing the quality of a result can be very exhausting. Moreover, due to the one-off nature of many ships makers, it is almost impossible to make an objective comparison between ships. To overcome this inconvenience, it is important to transform the qualitative subjective judgment to quantitative objective analysis.

The quantitative way of scoring the quality of a result set can be useful, not merely to figure out the quality of the final result, but also to assist the pipe engineer and the pipe router algorithm to find the best solution, or at least a very good solution.

For that reason, we strive to answer our fifth research question:

Research question #5: How to evaluate the quality of a set of routed pipes quantitatively?

1.3.4 Design Requirements

Before we actually design the automatic routing system, we need to follow some practical requirements. The first requirement is to make sure that the system should be efficient and effective and allow the user to find the pipe route with less effort than by manual routing using existing CAD software. This includes the ease with which the methodology can be prepared and all the needed data gathered. In addition to that, the solution that is found should be optimal and comply with the marine classification rules and regulations.

When pipe routing is done manually by pipe designers, the quality of the solution highly depends on the expertise of the pipe designers. One of the goals of the first requirement is to reduce the differences in the solution quality between pipe designers with different expertise.

The methodology needs some data before it can be used to perform the automatic pipe routing process. As described in the procedure of the detailed design process, this methodology needs the following data as its input: the piping system diagram, 3D model of the component arrangement, 3D model of the hull and superstructure, and additional data of the system components such as the equipment's connection points and the type of a component.

In the current pipe design process all information that is needed is available, either already in the CAD software or in some other format. Normally, during the design process the ship is modeled using CAD software, including the 3D volume of the hull and components, and all the underlying data. Because of this, the automatic routing system that is developed should be able to exchange data with the existing CAD software. By this connectivity, the user can use the existing CAD software and the automatic routing system simultaneously.

Currently there are several CAD software packages that are widely used by shipyards as design tools, and in some shipyards more than one kind of CAD software package is used. This requires the automatic routing methodology to be developed as a non-proprietary set of instruments. The automatic routing system should be able to be used together with any kind of CAD software.

The marine classifications and the common rules for pipes are constantly improving. New rules and knowledge emerge that must be adopted by the automatic routing system. As one of the requirements, the automatic routing must be expandable to allow for such changes.

One of the requirements is to reduce the time needed during the detailed design. The proposed methodology should be implemented carefully, require smaller amounts of operator time and its computation time needs to be optimized.

In short, there are four main requirements that have to be fulfilled by a proper pipe routing methodology: to minimize user input and reduce user routine tasks; to be integrated easily with other systems, such as existing CAD software; non proprietary and expandable; and able to produce good results within an acceptable time frame.

This research mainly focuses on methodology development. Its implementation in the form of an automatic routing system package is mainly targeted as a laboratory to investigate and prove the effectiveness of the methodology itself. The development of the tools ready to be used in production is of course not within the scope of this thesis.

These requirements are tightly related to the development of the automatic routing methodology and to the implementation of this concept into actual tools. While pursuing these tasks, we seek answers to the sixth research question below:

Research question #6: How can a routing algorithm be efficiently implemented that satisfies the four key requirements and works in any given 3D environment?

1.4 Research Approach and Laboratory

This thesis focuses on the creation of the methodology and its application for the pipe routing process in ship design. It starts with the development of the standard procedures for pipe routing. These are based on the common sense and experience of the pipe engineers when they route pipes manually in a ship. In practice, this

can be divided in two categories. The first category is related to the functionality of the pipes. For example, pipes that run together in the same direction should be aligned in parallel, or the hot water pipe and cold water pipe should be routed together for ease of maintenance. Such standard procedures should be possible in the proposed methodology. The second category is the common sense of pipe engineers that can be categorized as their personal preferences, and this category can be neglected.

The behavior of how the pipe engineers do the pipe routing in different parts of a ship is also investigated, because in practice pipe engineers might use a different approach for different pipe locations.

The rules and standards of marine classifications must also be adopted to ensure that the results of the proposed methodology are allowed to be used in practice. However in this thesis not all of the marine classifications are included, but the method for including and implementing those classifications is investigated.

Within this context, suitable routing algorithms are investigated and developed. Since much research on automatic routing has been done for many years, there are many different algorithms that have been proposed. Those algorithms are developed not only for routing of pipes, but often for other subjects such as microchip design; to route the cables in airplanes; ground traffic navigation systems; and finding the route for character movement in computer games.

Those existing algorithms are analyzed, tested and compared. Based on those test results a combination of more than one algorithm is developed, and included in the proposed methodology.

After defining the suitable routing algorithm for the methodology, the data that is needed by the algorithm can be identified. There are two main categories of data identification; identification of the type of data and the source of data. Because there are many types of data and also many sources of data, the generic type of data is defined.

In the pipe routing process we are dealing with a 3D environment. All computations in the automatic routing algorithms included in the methodology are done in the 3D environment. For that reason, to allow building our laboratory, we had to develop the specific 3D library.

In order to prove that the proposed methodology can fulfill the research objectives stated in section 1.2, a laboratory is developed in the form of a computer application package that contains the test implementation of the proposed methodology. The laboratory consists of three applications; the interface module for data exchange with the outside world, the router module that contains the automatic pipe routing algorithm and the knowledge-based module that contains the standard rules and routing criteria.

Using that laboratory, the proposed methodology has been thoroughly validated and verified through experimental work in a realistic pipe routing design process in a complex area of a ship. The experiments presented were carried out to find the solution in the detail design phase of the ship.

1.5 Organization of the Work

This book consists of 7 chapters, starting with this chapter, the Introduction. The next chapter discusses the current practical pipe routing process in a ship. We start this chapter by describing the stakeholders and their functions in the pipe routing process. Then, we explain the tools and information data that are needed by the pipe engineer. We also describe the essential steps of the pipe routing process. In this chapter, the common rules for pipe routing in a ship are discussed and the common mistakes that are made by an inexperienced pipe engineer are shown. Based on that, we conclude this chapter with the list of rules that must be included in the proposed methodology.

Chapter 3 starts with the description of the outline of the functional framework of the proposed methodology. It translates the common pipe routing rules found in practice that in principle lends itself to practical, real-life application, to the functional requirements. Since we intend to create a methodology, the procedure of data retrieval and user interfaces are included in the functional requirements as well. Then we review the state of the art of the automatic pipe routing method. In this chapter, several well-known automatic pipe routing researches are reviewed and discussed.

We start Chapter 4 by describing in detail shortest path algorithms. It starts with the deterministic method and is followed by the heuristic method to solve the shortest path problem. Then, the most popular algorithms are compared. Based on this result the suitable algorithms are selected. In this chapter we also discuss the other research subjects that are mentioned in Chapter 3.

Chapter 5 extends the outline of the functional framework that have been discussed at Chapter 2 and 3. In this chapter, we investigate and implement our proposed methodology into a pipe routing tools to perform the automatic pipe routing process.

Chapter 6 tests the performance of our methodology. It starts by showing the capability of the methodology to route pipes in the machinery room. Then, we continue to discuss the quality of the result. The sensitivity analysis of the proposed methodology is tested and described.

Chapter 7 contains the conclusion of this research and some suggestions for useful future research.

Chapter 2

Pipe Routing in Practice

2.1 Introduction

Chapter 1 explained the need to have an innovative approach to route pipes in a vessel to remain competitive in the global ship manufacturing market and indicated the considerable efforts that have been made in this thesis to achieve significant improvements to the pipe routing process. In Section 1.2 we described our main goal to research a methodology for an innovative approach to route pipes in a ship in such a way that tools that will be created based on that methodology can be used in a real ship design process.

The proposed methodology must be built based on the proven pipe routing process i.e. the actual pipe routing process in practice. This also helps to make sure that future users of this methodology feel at home. This is essential because it will increase the level of acceptance of automatic (pipe) routing.

Section 1.3.2 provided an overview of the current situation in the practical design process. In this chapter, the current practical process will be described in more detail and in a structured way and the common knowledge of pipe routing will be summarized. This chapter is concluded with the list of aspects that must be met to realize the targeted methodology.

The starting point of this chapter is by looking at the pipe routing as a project that should be managed, executed, and monitored efficiently. There are three core areas relevant for our purposes: people (or stakeholders), tools (including information), and process, as can be seen in Fig. 2.1. The next sections of this chapter explain those three core areas for the pipe routing process.

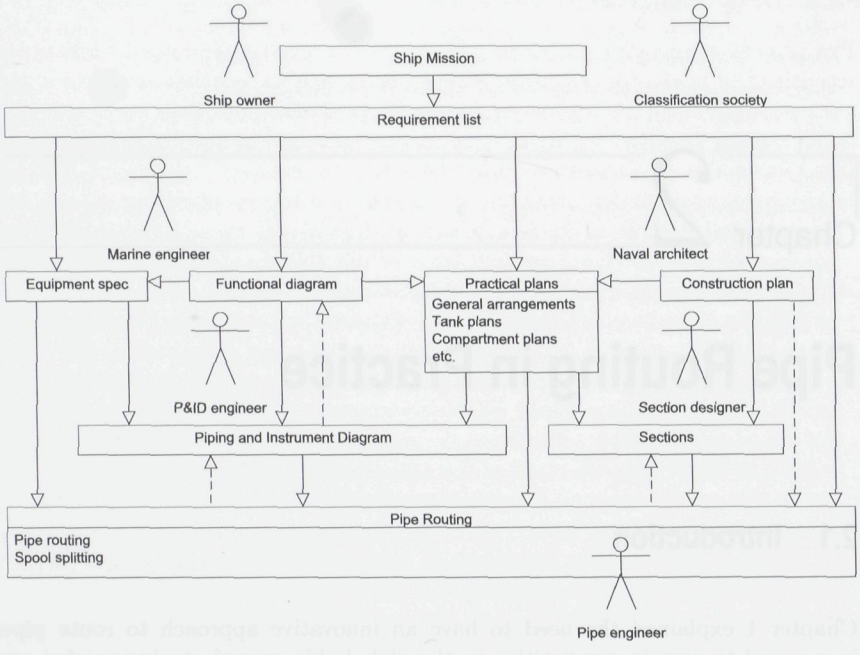


Figure 2.1: *Pipe Routing Processes*

2.2 Stakeholders

By a definition, a stakeholder is a party that affects or is affected by the actions of the business as a whole. In other words, a stakeholder can be defined as "those groups without whose support the organization would cease to exist.", Freeman and Reed [1983].

As described in Section 1.2, pipe routing is one of the most significant activities during the detailed design stage because all other activities in that stage depend on it. Therefore, many departments of the shipyard and its co-makers have a connection with the pipe routing process. The client, the procurement department, the basic design group, right through to the production and service departments and also the classification society; all of them can be considered as stakeholders of the pipe routing process. However, in this chapter we limit ourselves to the particular stakeholders that have a direct effect on the task of individual pipe engineers.

Project Engineering Manager

The project engineering manager[1] is the person responsible for all engineering activities and all disciplines in the project: process, naval architecture, structural, mechanical, electrical and instrumentation. In the pipe routing scope, the main task is to assemble and supervise pipe routing groups, and together with the pipe group leaders determine the following items:

1. Inventory of the required specifications, such as piping classes, equipment spacing requirement, the dimension of the walkways
2. Manpower planning
3. Defining what CAD software package will be used in the project
4. Dividing the ship into working areas and assign each area to (a group) of pipe engineers
5. Align the work with production planning

Pipe Group Leader

The pipe group leader has a responsibility to lead a group of pipe engineers that route pipes in one or more working areas. The group leader must have a hands-on experience to route pipes and an excellent capability to judge routing results both in terms of technical expertise and management expertise.

Pipe Engineer

The pipe engineer is the person who performs the task to route pipes in his working area. Every pipe engineer must have an excellent knowledge of piping systems in a ship and must be familiar with using the 3D software package. The quality of the routed pipes depends on the expertise of the pipe engineer. Most of the time, a pipe group leader also acts as one of the pipe engineers.

CAD Software Administrator

Currently, pipe engineers create piping systems using CAD software as their tools. In the last decade, the 3D CAD software packages have evolved into complete suites that allow many users to work together to perform multiple tasks using the same model. The complexity of the software package requires it to be maintained by an administrator.

In the beginning of the ship design project, the administrator configures the CAD software according to the standard that is required by the contract. An example is to define the minimum bending radius of pipes.

[1]Note that this may be organized differently in companies but the essence of the process remains the same. The same applies to the other functions in this section; they express the various roles that need to be fulfilled in the process.

Marine Engineer

As described in Subsection 1.3.1, to support the functionality of a ship, various systems such as propulsion and steering are needed. In its operation, those systems need some other subsystems in order to run properly. For example, a diesel engine needs fuel to run, needs oil for its moving parts, its temperature must be maintained in its operation range, and so on. For those purposes it needs to have a fuel system, lubrication system, cooling water system, and many other systems.

The marine engineer is the person responsible to translate the specification and the systems previously mentioned into the functional requirements and diagrams. The marine engineer must have an extensive knowledge of the equipment, instrumentation and functionality of the process.

Pipe and Instrument Diagram Engineer

Before the functional diagrams are used by pipe engineers, they are translated into Piping and Instrument Diagrams (P&ID). P&ID is basically a functional diagram that is enriched with topological data of the equipment, piping and instrument.

The Piping and Instrument Diagram engineer is the person responsible to translate the functional diagrams previously mentioned into the P&ID. Just as the marine engineer, the P&ID engineer must also have an extensive knowledge of the equipment, instrumentation and functionality of the process.

Naval Architect

The naval architect is responsible for five elements, Lewis [1988]; Hydrostatics, Hydrodynamics, Structure, Arrangement and Constructions. In short, a naval architect engineer is responsible to make sure that the ship supports its main functions efficiently and safely. This includes floating, moving and carrying.

Section Designer

The Section[2] designer is responsible to render the construction plan into a 3D section model while maintaining the requirements imposed by the naval architect. It includes defining all brackets, profiles, plates and holes in a section.

During the detail design phase, if it was needed, a pipe engineer makes a request to the section designer to make small changes in the section in order to be able to route a pipe optimally, such as moving a location of a hole in a plate. In this case, before the section designer makes any changes, he must get approval from the naval architect to ensure that the changes do not have a negative effect on the ship's structural integrity.

[2] A section is a production specific part of a complete ship's structure. The subdivision into section is made e.g. to take account of crane capacity and to increase production efficiency.

3D Model Builder

3D CAD software uses extensive libraries including the 3D model library, This consists of the common components and normally can be re-used for other projects.

However, since there are many customized components, it is not possible to cover everything from the library. The 3D model builder is the person responsible to create new models of components and add them to the library for future use.

The quality of the 3D model is important in the pipe routing process, a complete 3D model of one component will help a pipe engineer to get all the necessary data of that component.

Ship Owner

The ship owner is the (future) owner of the ship that is currently being built in the course of a project. This is the stakeholder who determines the type and the capabilities of that ship.

During the whole phase of ship building, progress must be reported to and approved by the ship owner. What makes things more complicated for the shipyard is the fact that during that process some specifications might be changed by the ship owner. One of the parts that is very sensitive to changes is the piping system. A very small ship specification change can cause to a lot of pipes to be re-routed.

Marine Classification Society

A classification society is a non-governmental organization that establishes and maintains technical standards for the construction and operation of ships and offshore structures. Classification societies set technical rules, confirm that designs and calculations meet these rules, survey ships and structures during the process of construction and commissioning, and periodically survey vessels to ensure that they continue to meet the rules.

Today there are a number of classification societies, the largest of which are Det Norske Veritas, Lloyd's Register, Bureau Veritas and the American Bureau of Shipping. In particular, classification societies may be authorized to inspect ships, oil rigs, submarines, and other marine structures and issue certificates on behalf of the state under whose flag the ships are registered.

Every classification society defines its own rules and standards. However many rules are similar from one society to another. Normally each releases a set of books of Rules for Classification of Ships twice a year.

2.3 Tools and Information

Tools have two main functions, to support or amplify a person's efforts allowing them to be more efficient doing their tasks, or to replace a human operator when

a tool can do the job more effectively. Some tools are focusing on one of these things, but most tools combine elements of both.

2.3.1 Tools

3D CAD Software - Outfitting Tools

Nowadays, the outfitting tools in 3D CAD software is the most essential piece of equipment to help pipe engineers to route pipes. To route pipes in a ship there are many aspects that need to be handled correctly by a pipe engineer; an example is to choose the pipe correctly according to the specification in the P&I Diagram, or to make sure that there is no collision between pipes and/or with other components in the ship. With the help of 3D CAD software, the pipe engineer can solve those tasks using the function in the software.

3D CAD Software - Diagram Tools

The diagram tools are needed by the pipe and instrument engineer to create the piping and instrument diagram. By using advanced diagram tools, the P&I diagram can be easily created with the help of the system library. Also, the P&I diagrams that are created have a connection with the 3D model that will be used by a pipe engineer. In that way, the consistency between the diagram and the 3D model is ensured.

2.3.2 Information

Functional and P&I Diagram

The functional diagram consists of graphical symbols and lines which illustrate the process and its flow. It identifies the functions of its instruments such as sensors, valves, indicators and instrument interconnections. The P&I diagram basically almost the same with the functional diagram. The main difference is that P&I diagram also contains the location of the pipes and instruments. It acts as the primary guidance for a pipe engineer to accomplish his task.

Section and Construction Plan

The construction plan is a drawing that was generated during the basic design phase. It is the basis for the section drawing that will be made in the detailed design phase. A section is a 3D steel construction that represents a construction plan in detail. The construction plan is not the only guidance to create a section. The section designer must always ensure that the section complies with the specification that was issued by the naval architect.

Beside the P&ID, the section is also necessary before pipe engineers route the pipes. In some cases, when section details are not available, pipe engineers may have to use the construction plan.

Piping Specification

The required specifications can be separated into two categories, based on their sources; the marine classification society and the demand from the ship owner. The specification from the marine classification is absolute; it means that no matter how hard it is to be implemented, it must be followed. Otherwise, the ship will not comply with the marine class and it will not be certified by the class organization.

The request from the ship owner, however, can be negotiated. It means that some conditions can be violated as long as the ship owner agrees with it. The ship owner may be compensated by other benefits, for example, if the changes will cause the pipes to be shorter and thus cheaper to maintain.

Basic Design Information

Pipe engineers also need other information like the general arrangement and the specification of the equipment, the compartment plan and the tank plan. All of this information is available from the basic design phase.

Beside this information, the complete 3D models of components such as pumps, engines and other equipments are also needed.

Component 3D Model

The 3D model of a standard component usually is available in the CAD software. For customized components, a 3D model must be created. Ideally, the component 3D model should come from the component supplier. For example, if the shipyard buys a main engine from a supplier, next to the main engine the supplier must also provide its 3D model.

However, in practice the component 3D models are often created by the shipyard. There are two main reasons for this. The first is the fact that the model from the supplier has too much detail. Most of the time all the bolts and small holes are included in the model, which is not feasible to be used by the shipyard because it reduces the performance of the CAD software. The second reason is that sometimes the component supplier charges additionally to the shipyard for supplying the 3D model.

2.4 Pipe Routing Process

2.4.1 Organizational Process

In order to execute a project efficiently, before the process of pipe routing in ship can be executed, it is necessary to plan the project. The first thing that must be done is to determine the manpower that is available. The manpower planning is extremely important. In practice, more than one ship is build at the same time

in a shipyard. This means that the available manpower is not always the same, and if needed, more pipe engineers will be hired temporarily for a certain project.

The expertise and skill of the available pipe engineers also need to be known beforehand. In a shipyard that uses more than one type of CAD software, the choice of which CAD software will be used depends on the available manpower. In some projects, the pipe system design process of a single ship is done in two different CAD software packages. The reason is that some of the pipe engineers are only able to use a certain type of CAD software and the others are only able to use the other type of CAD software.

The allocation of working area and the pipe engineer group assignment also depends on the available manpower at that time. Normally there are 3 to 7 pipe engineers in one group, depending on the size and the complexity of the working area.

Inside each pipe group, the pipe group leader divides the working area into smaller areas and assigns each member of the group to route pipes in those areas. However, there can also be the case where more pipe engineers work together on the same area, and they each have to route pipes for a different system.

In order to be more cost effective in the production stage, most of the pipes should be installed during the pre-outfitting stage. This means that the pipes that belong to one section of the ship must be installed immediately after the steel construction of that section is completed, so that when that section is assembled to the hull on the slipway, all pipes that belong to that section are already installed. This situation only can be reached if the pipe spool drawings of those pipes are available a few weeks before section assembly starts. This allows that the pipe spools can be manufactured in time. For that reason, it is especially beneficial for the pipe routing process to be synchronized with the production plan.

Every pipe group leader then makes a comprehensive planning for their group. Pipes that require more time to manufacture compared to the average get a higher priority. This task demands a high level of expertise from the pipe group leader.

2.4.2 Routing Process

After a pipe engineer received an assignment from the group leader, he starts with collecting all the data needed. He collects the information with respect to the section, P&ID, piping specification and other drawings from the basic design phase.

Then the pipe engineer examines the section to evaluate the spaces that are usable for pipes. In most cases, the section 3D model is available, but not always. If this is the case, the pipe engineer must resort to the construction plan. It adds difficulty to routing pipes in that area.

At the same time, the pipe engineer also needs to examine the P&I Diagram, and in combination with the usable space in the section, he can start to determine how the pipes should be routed.

Most of the time, the components of the ship such as main engines, pumps, or other equipment are not placed in the 3D environment yet. In that case, the pipe engineer must place those components based on the general arrangement drawing. However, the main components such as the main engine and the other system that related with the mechanical drive system are already defined during the basic design process, and the information is available for the pipe engineer.

In the ideal situation, starting from that point, the pipe engineer can begin to route the pipes according to the criteria of pipe routing. This is discussed in detail in Section 2.5.

Then the pipe engineer routes pipes one by one. After one or more pipes in the area are routed, all valves will be placed. Then those pipes are split into pipe spools for the purpose of production, handling and assembly. Thanks to the current CAD software functionality, the process to place the valve and to split a pipe into pipe spools have been improved. However, even though most of the latest CAD software package already include the automatic routing functionality, it still only focus of finding a path between two nozzles[3], and basically the total layout of the pipes is still defined highly depends on the pipe engineer.

As we described in Subsection 1.3.1, a ship has many systems and subsystems that consist of a large number of pipes. Even a small vessel might have more than one thousand pipe and in a larger vessel this number can reach three to four thousands pipes and more. In short, many pipes must be routed, while the space that is available is limited. In order to accomplished the task to route all pipes efficiently and comply to the rules, pipe engineers must employ a smart routing strategy. The summary of this strategy is explained in Section 2.6.

2.4.3 Working Area

During the routing process, a pipe engineer needs to make an assessment of his working area. It is needed because the way pipes are routed depends on some aspects. First, it depends on the pipe specification itself; to which system that pipe belongs, the pipe diameter, the pipe material, the surface treatment of the pipe, and specific requirements for the pipe such as if it must be installed sloped down or not. The second aspect is the number of pipes that must be routed in a certain area. The last one is the technical aspect of the system itself. Things like the placement of pipes and the sprinklers for the system that intend to be used to wash the cargo area is part of this.

Based on those aspects, the areas of a ship can be categorized into three different types:

1. Machinery type area
2. Accommodation type area
3. Technical type area

[3]Nozzle is the term that widely used in the 3D CAD application to represent the pipe connector or pipe end. In this thesis, we adopt this term and use it in all chapters.

The compartments in the ship that are categorized as the machinery type area are the main engine room, auxiliary engine room and pump room. This type of area can be considered to have the highest density in terms of the number of pipes per cubic meter. Also because there are many pipes there, more than one pipe engineer will be assigned to route pipes in that area. As a consequence, they must have a good coordination to make sure that the pipes are routed in the same manner and do not violate each other.

The greatest difficulty of routing pipes in this type of area beside the large number of pipes is the fact that one pipe can follow many different paths. Therefore, a pipe engineer must find the optimal combination of pipe path to ensure that all pipes can be routed properly.

The accommodation type area consists of the accommodation spaces and the control room. In this type of area, the number of pipes is not as large as in the machinery type area, but the space that is available is extremely limited. In the accommodation area, all pipes must be hidden, either below the floor or above the ceiling. Beside the limited space, the task to route pipes in this area is further complicated by the rule that requires the black and gray water system pipes to have a slope.

In accommodation areas, normally a pipe engineer is already able to figure out the rough path of each pipe. However, since the space is very limited, he must be able to make a good arrangement of pipes, and avoid conflicts.

The technical type area is the area of the ship where other systems are placed. For example in a dredging ship, a technical area is where the large dredge pipes are located.

In terms of finding the path of the pipe, this area normally has the lowest difficulty level compared to other area types. However, routing pipes in this type of area is still difficult since one needs a deep knowledge about the technical system itself.

2.4.4 Object Constraints

During the routing process, a pipe engineer considers every object (e.g. a component, another pipe or a piece of the steel structure) and area in a ship according to the possibility to be penetrated or used by pipes. Thus, an object or an area is no longer seen as its functionality but translated into an object constraint that can be categorized into four types:

1. Absolute Constraint
2. Soft Constraint
3. Negotiable Constraint
4. Rules Constraint

The absolute constraint refers to an object or an area that is absolutely not allowed to be passed by any pipe. An example of this type of constraint is a piece of equipment, or an area outside the ship.

The soft constraint is an object or an area that is preferably not passed by a pipe. Basically this constraint should be satisfied, but if there is no other way, it is allowed to be violated. In another word, pipes are allowed to be routed through this area, but there will be a penalty for that. An example is the edge part of the walking corridor in the machinery room.

The negotiable constraint is an absolute constraint that in a certain situation can be negotiated to be passed through by a pipe, by changing or moving the object or by modifying the pipe itself. An example is a stiffener of the steel construction. In a normal situation, a pipe engineer must avoid routing a pipe through a stiffener. However, if there is no other way, or if there is a big advantage to route a pipe through that area, it is possible to negotiate with the naval architect to change or move that particular stiffener.

The rules constraint is an object or an area that becomes an absolute or a negotiable constraint for a certain pipe due to the pipe routing rules. For example, the area above the combustion engine becomes a negotiable constraint for an oil pipe.

Normally, a modern CAD software package can detect a pipe violating an absolute constraint and some of the negotiable constraints. It then can generate a warning to the pipe engineer. However, the soft and rules constraints are not actually available in a CAD package, thus they can only be detected by the expertise of the pipe engineer.

2.4.5 Collaboration between Pipe Engineers

Every pipe engineer is responsible for his own task to route pipes in his or her own working area and/or for the system assigned to him. They can work on their own for some pipes that connect two nozzles in their own area. However, most pipes are crossing other people's areas.

For example, in the bilge and ballast system pipes are running from aft ship to fore ship. In practice, even for those pipes, every pipe engineer is only responsible to route pipes in his own area. However, he must think about the whole route of those pipes, so that when he sets the location where they leave his working area, it will not cause a problem for the responsible pipe engineer of that neighboring area. In this situation, pipe engineers should communicate with each other.

2.4.6 Modification Possibility

In an ideal situation, the available detailed section structure is suitable for all pipes to be routed well. In practice, it happens quite often that the existing section structure must be changed to enable the pipe engineer to route pipes there. In that situation, the pipe engineer may ask to the section designer to change the existing section design. Then, after consulting with the naval architect, if the changes do not affect the structural integrity of the ship or other functional requirements, the existing section can be modified.

In the other way around, there are also cases where the pipe engineer had finished routing pipes in a certain location of a section, but due to the changes in the ship specification, the naval architect requests the section structure designer to change the section structure. Those changes are then communicated to the pipe engineer, and he must make the corresponding modifications.

There is also a possibility that the pipe engineer makes a request to the P&ID engineer to change the way some pipes should be logically arranged.

The expertise of the pipe engineer plays a big role in all of this, because the more knowledge and experience the pipe engineer has, the more creative he can be. He can then make a prediction that the changes in the section structure or in the P&ID are feasible without affecting the integrity and/or the performance of the process system in the ship.

2.5 Criteria for Pipe Routing in Ships

In Subsection 2.4.2 it is mentioned that when he routes pipes in a ship, every pipe engineer must consider the criteria of pipe routing to ensure that the pipes are routed according to the rules and standard. In practice, pipe engineers follow two main criteria that can be categorized as follows:

1. Pipes must comply with the rules and standard from the Marine Classification Society involved in that ship
2. Pipes must be routed in such a way that the production, installation, and maintenance cost of those pipes are as low as possible

2.5.1 Rules of Pipe Routing

For safety reasons, pipe routing must follow the rules from a marine classification society. There is a special chapter regarding the rules and standards for piping systems in ships that consists of many rules. Those rules can be categorized into two kinds. The first one is the rules that are related with the specification of the pipe including the piping components, such as flanges, valves, supports.

Piping system for	Class I		Class II		Class III	
	p(bar)	t(C)	p(bar)	t(C)	p(bar)	t(C)
Steam, thermal oil	> 16	or > 300	≤ 16	and ≤ 300	≤ 7	and ≤ 170
Fuel, lubricating oil	> 16	or > 150	≤ 16	and ≤ 150	≤ 7	and ≤ 60
Other media	> 40	or > 300	≤ 40	and ≤ 300	≤ 16	and ≤ 200

Table 2.1: *Classes of piping system (excerpt from DNV)*

This kind of rules is applied in the P&I Diagram by the P&ID engineer, and later on the pipe engineer must use it as a guidance. As an example, one of these rules relates to the specification of the material, shown in Table 2.1.

In this section, the rules of pipe routing as described by a marine classification society will be described. The following list of rules only contains selective rules that have a direct influence on the way a pipe engineer performs his task. The complete list of rules can be seen in the classification books.

1. **The number of detachable pipe connections shall be limited to those which are necessary for mounting and dismantling**
   While dividing a pipe into several pipe spools, the pipe engineer must choose to use flanges or a welded pipe. In practice, a pipe engineer must make a clear choice if a pipe must be welded in or must be removable. For example in technical spaces or on a connection to a ship component, every pipe must be removable. In those cases flanges must be used. However, in electrical spaces every pipe must be welded.
2. **The support of the piping system shall be such that detrimental vibrations will not arise in the system**
   To fulfill this rule, every pipe must be routed close to the steel construction for ease of the installation of the pipe support. One of the reasons that pipes that run together should be routed in parallel is because the pipe support installation is easier.
3. - **Installation of pipes for water, steam or oil behind or above electric switchboards shall be avoided as far as possible. If this is impracticable, all detachable pipe joints & valves shall be at a safe distance from the switchboard or well shielded**
   - **All detachable pipe connections and valves in oil fuel pressure piping shall be at a safe distance from boilers, exhaust pipes or other heated surfaces and electrical appliances**
   - **Detachable pipe connections and valves in hydraulic pressure piping shall be at a safe distance from electrical appliances, boilers, exhaust pipes and other sources of ignition**

   Those three rules above have the same goal: to prevent the possibility of fire. Before a pipe engineer begins to route those pipes, he must make a good evaluation of the space management.
4. - **Centrifugal bilge pump shall be located as low as possible**
   - **Centrifugal sea-water cooling pumps shall be installed as low as possible in the ship**

   Those two rules above are suggesting that the bilge and the sea-water cooling system pipes should be routed as low as possible too.
5. **The overflow system shall be so arranged that water from the sea cannot enter through the overflow main line into other tanks in case of any tanks being damaged**
   This means that the overflow pipes should be routed in a slope. However, in practice it is also allowed to have a horizontal pipe.
6. **Piping conveying flammable liquids under pressure in the engine and boiler room shall be laid in well lit places, in order that the piping may be kept under observation**

It means that for those kind of pipes, a pipe engineer must try to route it in a space that can be seen.

7. • **Bilge suction pipes are, as far as practicable, not to be carried through double bottom tanks**
   • **Tank air pipes shall be placed at the highest part of the tank and as far away as possible from the filling**
   • **Water pipes and air and sounding pipes through freezing chambers shall be avoided**
   • **Fuel oil pipes shall not be led through fresh water tanks**
   • **The arrangement of piping and valves shall be such that oil cannot enter tanks not intended for this purpose**

   Those five rules above are about avoiding to route a certain type of pipe in a forbidden location.

Basically, the list of rules above apply to all class society, however the details may vary.

2.5.2 Minimization of Pipe Cost

In the pipe routing process, routed pipes that merely comply with the maritime rules will not be good enough. Every pipe engineer must also consider to minimize the pipe cost. There are three elements of the total cost of a pipe; pipe material, production and installation cost. To calculate the pipe cost is not trivial and most inexperience pipe engineers only consider minimizing the length of the pipe. Meanwhile, the more experienced pipe engineer can easily estimate the total cost based on the last two elements above.

To minimize the production cost, it is not enough only to minimize the length of the pipes, but the diameter and the thickness must be considered as well. A pipe engineer should be able to identify that a pipe with a small diameter can be much more expensive compared to a pipe with a larger diameter if the wall thickness of the smaller pipe is larger than normal.

The knowledge of the piping system is also important to be mastered. For example, a pipe engineer must know by head that the piping system that transfers sea water must have a special treatment to prevent corrosion. Table 2.2 shows systems that normally need a special pipe treatment and Table 2.3 shows some common knowledge of pipe routing based on the pipe system type. Table 2.3 shows that for a certain system, pipes are preferably routed below the floor rather than through the top. Also according to the classification rules, pipes for some system must be routed as low as possible.

The minimization of the installation cost is helped by having the pipe spool sketch ready for production in time, so that the pipe can be installed during the pre-outfitting stage. Beside that, a pipe must be divided into pipe spools in such a way that the pipe spools can be handled easily. The installation cost would also be reduced by a clever arrangement of supports and this can be achieved by having pipes to be routed in parallel.

	System Name	Pipe Treatment
Seawater	Bilge & Ballast	Galvanized
	Fire Fighting	Galvanized
	Cooling Water	RILSAN/ABCITE/Galvanized
Sanitary	Deck Drains	Galvanized
	Sanitary	Galvanized
Sounding	Water Tank	Galvanized
	Oil Tank	Galvanized

Table 2.2: *List of Systems that need a special treatment*

System Name	Below	Lowest	Remarks
Degassing			Large diameter
Jetwater			Large diameter and heavy thickness
Draught Measuring			Slope down
Lubrication Oil Bowthruster			Slope down
Starting Air			Expensive pipes so keep it short
Fuel Oil Transfer	Y	N	For overflow slope down
Fuel Oil Service	Y	N	
Lubrication Oil Transfer	Y	N	
Lubrication Oil Service	Y	N	
Dirty Oil and Sludge	Y	N	Slope down
Sea Cooling Water	Y	Y	Large diameter, expensive treatment
Fresh Cooling Water	Y	N	
Bilge	Y	Y	
Ballast	Y	Y	
Firefighting and deckwash			High pressure
Air, Filling, and Sounding			Keep it straight, bending is ≤ 30

Table 2.3: *Piping system knowledge*

2.6 Pipe Routing Common Knowledge

The common knowledge to route pipes is gained by every pipe engineer by experience, following the rules that are supplied by a marine classification society. However, this knowledge is not normally properly documented. We performed many interviews with experts in pipe routing to absorb their knowledge. During the interview period, the pipe routing guideline was compiled by one of the very experienced pipe engineer team leaders v.d. Berg [2009] who summarized the most important parts of the pipe routing common knowledge.

2.6.1 General Guidance of Pipe Routing

1. For pipe routing in crowded areas it is recommended to maintain layers of pipes with the same direction at the same elevation. (longitudinal direction, cross direction)
2. Think ahead during routing pipelines. Large diameters first. Be aware that the pipe you are currently working on is not the last one to put in. Especially in machinery area the available space for piping is limited. Begin to route pipes from Bilge, Ballast, Sea Cooling Water, or Sounding systems.
3. Pipes that are required to have a slope must be kept high as long as possible.
4. Think about the possibility of supporting the pipelines. Not too far away from deck, bulkhead or tank top.
5. Make a clear choice whether a pipe must be welded in or must be removable.
6. For several systems we use specific construction details. These details will be available on the corresponding pipe diagram.
7. Keep in mind that pipes should comply with the rules from classification society. (See subsection 2.5.1).

2.6.2 Dividing a Pipe to Pipe Spools

1. In technical spaces or on a unit every pipe must be removable. Therefore flanges must be used. Pay attention to use the right type of flange. (Welding neck, PN10, SAE-flange, O-ring etc.)
2. In other parts of the ship pipes may be welded in. In electrical spaces this is a must. Never use flanges above electrical equipment. (See subsection 2.5.1)
3. Make a clear choice whether a pipe must be welded in or must be removable.
4. Maximum length off pipe spools depends on the situation. A pipe that will be put in during pre-outfit can often be longer than pipes put in during outfit. In case of any doubt a shorter pipe will be the best option. Maximum length will be 6 meter. (metric pipe 5 meter)
5. After dividing a pipeline the parts must be easy to handle. No square spools with legs more than 2 x 2 meter.

6. Flanges in a sloped part of a pipe are difficult to assemble in the pipe shop. This will affect the accuracy of the pipe spool. Always try to place a flange set in a straight part of the pipe.
7. Valves, fittings, gaskets, etc. must be replaceable. Therefore it is necessary to create an easily loosening pipe spool which is directly connected to it.

2.6.3 Bended Pipes

1. In bended pipes we will divide the pipe on rounded dimensions. Preferable, include sufficient straight for the clamp length so flanges (or sleeves) can be welded prior to bending the pipe in the bending machine.
2. One pipe spool with more than one bend needs to be placed on the bending machine only once. Remember this while dividing a pipeline. So it is better to have one pipe with two bends instead of two pipes with one bend each.
3. Route pipes in such a way that no welding is necessary between two bends. So the minimum length must be including sufficient straight for the clamp length of the bending machine.

2.6.4 Welding Elbow

1. Use of 1.5 diameter (LR) welding elbows must be restricted. Bending is always preferred. Think about internal grinding after welding elbows in pipes that must be coated afterward. The bending radius of a modern bending machine is only a fraction larger than the radius of a 1.5 diameter welding elbow.
2. Use of 1 diameter (SR) welding elbows must be avoided. These are far more expensive than 1.5 diameter elbows. Often specifications require a minimum radius of 1.5 diameter.
3. Place flanges preferably directly to a 1.5 diameter (LR) welding elbow.

2.7 Some Mistakes in Practice

In the previous section, the common knowledge of pipe routing has been described. However, since up to now official guidance does often not exist, not every pipe engineer is aware of it. For pipe engineers that have a lot of experience, that common knowledge is something that they have learned by routing pipes in previous projects and by getting feedback from production and outfitting employees. On the other hand, inexperienced pipe engineers most likely will not consider the common knowledge sufficiently.

Fig. 2.2, 2.3, and 2.4 show basic mistakes in pipe routing; routing a pipe without considering that the pipe must be close enough to a steel construction to be well supported.

Figure 2.2: *Pipe needs to be moved near the steel construction*

Figure 2.3: *Pipe should be moved to the right*

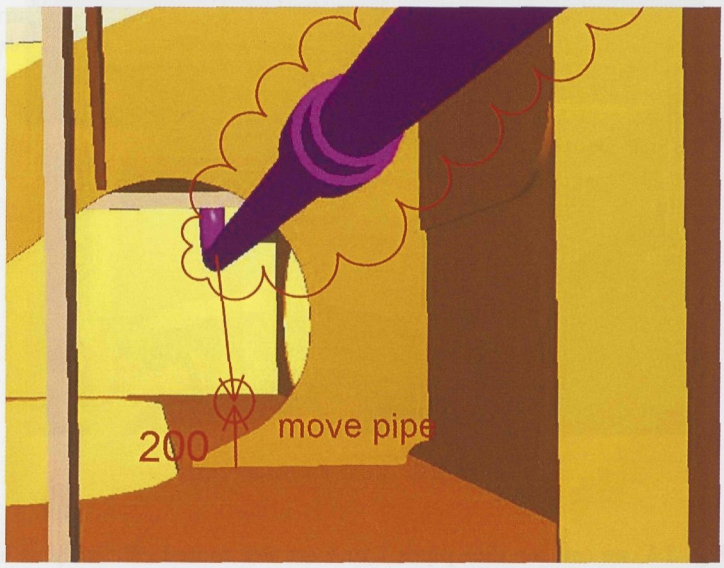


Figure 2.4: *Pipe is not close enough to be supported*

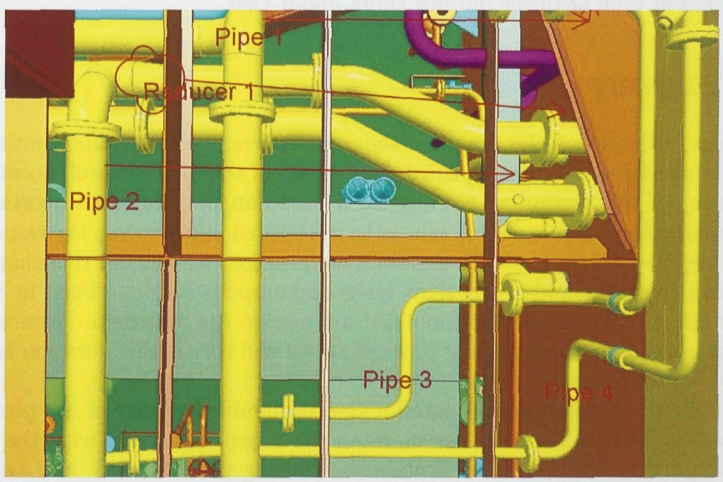


Figure 2.5: *Combination of pipes*

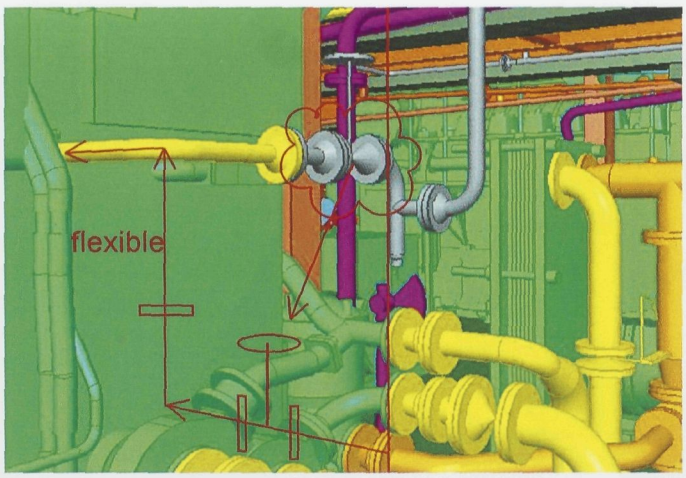


Figure 2.6: *Valve misplacement*

In Fig. 2.5, we can see that pipes 1 to 4 are not routed efficiently as a group of pipes. What happened in this example is that the pipe engineer routes pipe 1 and 2 without considering that pipes 3 and 4 must be connected as well.

When a valve needs to be placed, a pipe engineer needs to consider that the valve is properly accessible to be used. Fig. 2.6 shows an example where an inexperienced pipe engineer did not think about that.

2.8 Summary

This chapter described the practical aspects of current manual pipe routing processes in answering the second, third, and a part of the fifth research questions of this thesis. In this section we summarize it based on the addressed questions.

The second research question relates to the needed information, the responsible person and how to get that. To answer this question, we started this chapter by explaining the stakeholder and their tasks and responsibilities. Then in Section 2.3, the list of tools and information that are needed are described. However, this list relates to the manual routing process, so we will revisit this question again in the next few chapters.

The third research question asks for the common knowledge of the pipe routing process. Section 2.4 describes in detail the process; starting with the organizational part, we continued to explain the routing process itself, and how pipe engineers collaborate with another. Then, Section 2.5 and 2.6 described in more detail the criteria and the common knowledge of pipe routing process.

The fifth research question relates to measuring the quality of the routed pipes.

In Section 2.5 the pipe routing criteria were explained, and even though they are not translated into quantitative measures yet, this section has answered part of this question.

Chapter 3

Pipe Routing Framework

3.1 Introduction

As discussed in Chapter 1, the main goal of this thesis is to research a new methodology to improve the current pipe routing process, and to validate the tools based on the proposed methodology. Next to the main requirement to find a "good" solution, there are four functional requirements of the proposed methodology that are explained in Section 1.3.4: to minimize user input and user decision; to be integrated easily with other systems, such as existing CAD software; non proprietary and expandable; and able to produce results within an acceptable time frame.

In Chapter 2, we have discussed the practice of the pipe routing process in a ship. It highlighted the important aspects that must be followed by the pipe engineer to route pipes in a ship according to the marine classification regulations. Pipe engineers also need to consider to lower the cost of pipes, including material, production and installation cost. In Section 2.6 the most important common knowledge of pipe routing in a ship was described.

In this chapter the outline of the functional framework is decided and the elements that need to be further investigated will be highlighted. The second part of this chapter will present a brief review of literature with a focus on the highlighted functionality of the proposed methodology.

This chapter will describe at some length the practicalities of the functional framework of our methodology. While this may seem uninteresting from the point of view or research, we point out what our research aims at developing an integrated pipe routing methodology. To validate it, it is necessary to dwell also on the context of the pipe routing process.

The basic outline of the functionality framework is shown in Fig. 3.1. The detail of the proposed methodology, including the architecture and the imple-

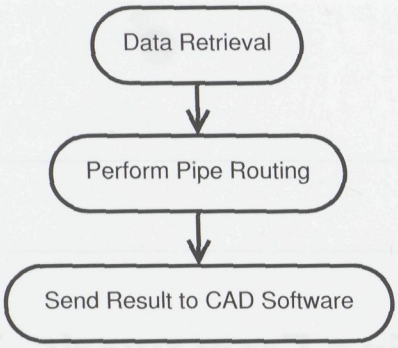


Figure 3.1: *The Outline of the Functional Framework*

mentation will be explained in Chapter 5.

As one of the goals, the proposed methodology must be able to be used in the real design process. It means that the methodology must be able to be used together with any existing 3D CAD software package. As a proof of concept, in our case it must be compatible with both Nupas-Cadmatic and Tribon M3 software packages.

There are three main steps in the functional framework; it begins with the process to retrieve the data that is needed for the routing process. Then the routing process is performed according to the common knowledge that is described in Subsection 2.6. After this is finished, the result must be exported to the CAD software for further processing.

3.2 Required Data

In order to define the part of data retrieval, we need to investigate three things; what data we need, where those data reside, and what the type of those data is. In Section 2.3 we have discussed the tools and the information that are currently needed by a pipe engineer to route pipes in a ship. In our methodology, almost the same data are needed by the pipe router module to route pipes in ship.

3.2.1 Piping and Instrument Drawing

In the same manner as for the manual routing process, the pipe router module of our methodology needs to have access to the P&I diagram as the primary guidance to route pipes. A P&I diagram (see an example in Fig. 3.2) shows to which equipment nozzles, pipes should be connected. This can be conceptualized by the analogy of a road trip. To travel from location A to location B, we need to use a road map to find the direction. In a P&I diagram, both location A and B

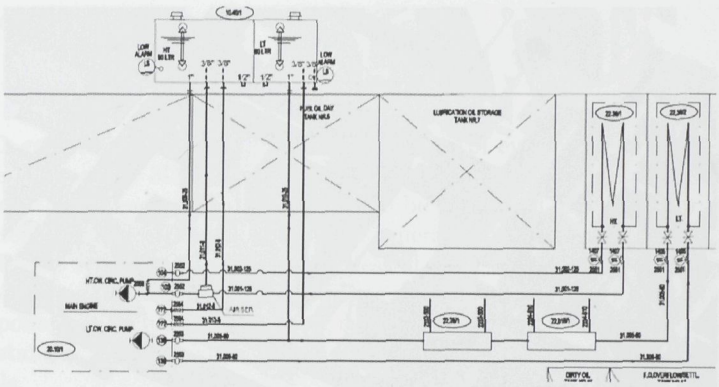


Figure 3.2: *Process and Instrument Diagram*

are shown as a certain nozzle of equipment A and B, and the pipe acts as a road in a road map between A and B.

Beside that, a P&I diagram also contains some specification of pipes, such as the pipe diameter and pipe specification. It also includes measuring instruments, valves, and other pipe components, such as a pipe reducer.

In a P&I diagram, every pipe and other component has a unique name. In practice, together with the P&I diagram, there is a document that contains the detailed specification of every pipe, valve and instrument.

Very often, a P&I diagram is merely a plain drawing that does not have any relation with a database in a CAD software package. This is problematic for process efficiency. For example, as it does not have any connection with the 3D model database, a pipe engineer needs to manually look for the unique name in the P&I diagram and find the component with the same name in the CAD software.

Starting a few years back, the so called Smart P&I diagram concept is introduced. A Smart P&I diagram is a P&I diagram that has a live connection between each object in it with the same object in the databases of the CAD software package. For example, every pipe in the P&I diagram has a link with the pipe specification database. With this feature, the consistency between the P&I diagram and the routed pipes can be easily maintained.

In manual routing, a pipe engineer usually uses a hard copy of the P&I diagram by making a print of it. In this case, the normal plain drawing of a P&I diagram can be used, because the pipe engineer can mentally translate all lines and symbol in the P&I diagram. However, the pipe router module in our proposed methodology needs to have the Smart P&I diagram.

Looking at the development trend of the major CAD software companies, in the future this requirement will become standard. Only a few CAD software

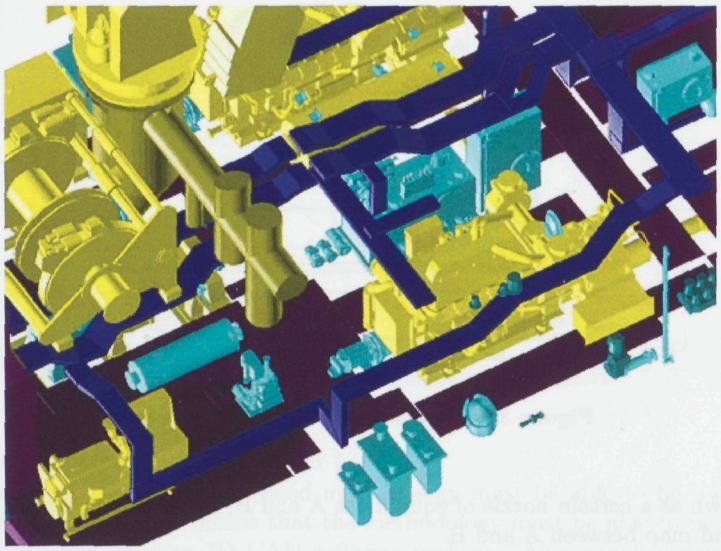


Figure 3.3: *3D General Arrangement*

companies have already implemented the Smart P&I diagram concept. In our laboratory case, both Nupas-Cadmatic and Tribon M3 only implement the link between pipes and valves in the P&I diagram with the pipe specification part in the CAD data. On the other hand, both software packages are able to generate the old fashioned P&I diagram. Thus, to bridge the gap, a simple Smart P&I diagram tool is included in the proposed methodology.

3.2.2 General Arrangement

Using the same road map analogy, a P&I diagram shows the road direction and the start and end location. However, it only shows the relative topological direction and location. The absolute start and end locations are represented in a general arrangement.

In an early design process, a general arrangement is often merely a 2-dimensional drawing that shows the location of every major component in a ship. Normally, there is one general arrangement drawing for each deck. Later on, the 3D model of those components are placed in a 3D space based on those general arrangement drawings, resulting in what we call a 3D general arrangement.

Both Nupas-Cadmatic and Tribon M3 have a 3D modeling package, but they use a different kind of implementation to import and export data. Thus, to fulfill the non proprietary requirement, the proposed methodology must adopt a common way that can serve both software packages, and indeed also more general

packages.

3.2.3 Component Details

In a 3D general arrangement, the exact location of every component is known. However, using the 3D general arrangement alone is not enough to know the exact location of each nozzle that needs to be connected. For this purpose, the completeness of the 3D models in a 3D general arrangement is needed. There are at least 3 requirements of completeness of a 3D model; it has the correct 3D volume[4], it has the exact location and type of every nozzle, and for a certain type of component, it includes the working space area. Fortunately for our laboratory circumstances, in both Nupas-Cadmatic and Tribon M3 this kind of detail is available.

As we discussed before, a Smart P&I diagram has a live connection between it and another database, thus there is a connection between every component in a Smart P&I diagram and the 3D general arrangement. Using both the Smart P&I diagram and the 3D general arrangement, the pipe router module in the proposed methodology can start the routing process in a free space. However, since our objective is to route pipes in a ship, the steel construction of the ship needs to be known. It provides both constraints to a feasible solution and supports for suspending the pipes.

3.2.4 3D Steel Construction

The first thing that is needed to know about a steel structural element of a ship is the type of construction, whether it is a normal plate, watertight bulkhead, bracket, pillar or stiffener. This is important because it has a direct effect on how pipes should be routed. For example, if there is no other way, pipes can be routed through a plate (if it is not affecting the strength and stiffness of the structure), but not through a stiffener or a pillar. However, if it is needed, a pipe engineer can make a request to the naval architect to make a modification in the steel structure, even though this is not recommended.

Just as other kinds of data mentioned above, the 3D steel construction is available in both Nupas-Cadmatic and Tribon M3.

3.2.5 Tank Plan Drawing

This plan is one of many "practical plans" based on the general arrangement. Other examples are the Watertight compartment and Watertight door plan, the maintenance routes plan, the overview plan of the fire fighting and detection system. They show certain aspects of the design that are related to the overall

[4]3D volume is the term that widely used in the 3D CAD application to represent the physical dimension of a 3D model. In this thesis, we adopt this term and use it in all chapters.

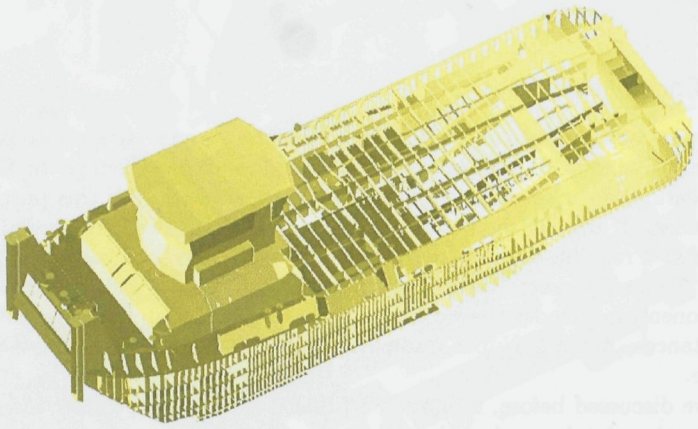


Figure 3.4: *3D Steel Construction*

functioning of the ship and therefore, like the P&I diagram, are of prime interest to the user.

Some of the pipe routing rules are related to a certain type of tank in a ship. For example, it is not allowed to route an oil pipe through a freshwater tank. To accommodate this kind of rules, the tank type and location must be known. Unfortunately, in practice the 3D steel construction model does not contain that information currently. The complete information is available only as a two dimensional drawing called a tank plan drawing.

In the current situation, in most cases the tank plan drawing is created during the basic design process using a two dimensional software package.

3.2.6 NoGo Area

In Subsection 2.4.4, the four types of an object constraint were explained. It was also mentioned that an object constraint can be a real object or an area. A NoGo area is an area that is defined as an absolute constraint or as a soft constraint or as a combination of both.

For example, in a walking corridor inside the machinery room, a pipe should not be routed in the middle of that corridor but it can be routed along the edge of it. In the proposed methodology, the possibility to create that kind of NoGo Area must be accommodated.

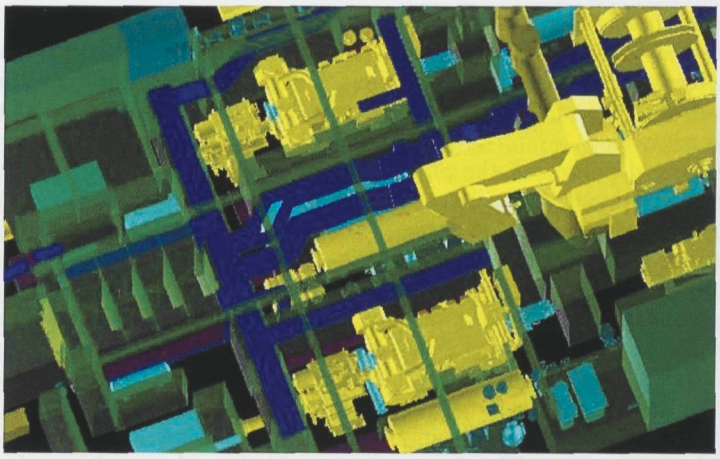


Figure 3.5: *General Arrangement with NoGo Area*

3.3 Pipe Routing

In the previous section, the data needed by the pipe router module in the proposed methodology are discussed. The next step is to conclude the requirements of the data retrieval part in the proposed methodology.

1. Smart P&I diagram information
2. Common interface to retrieve 3D general arrangement volume data
3. Common interface to retrieve component's detail data
4. Common interface to retrieve 3D steel construction data
5. 3D tank plan information
6. 3D NoGo area information

In case this information is not present in the current modeling tools, simple additional tools must be provided for our methodology to work properly.

While it is a time consuming task to arrange for all the interfaces and thereby facilitate the data retrieval, it is not of interest for our research and is not described here.

In Chapter 2, it was mentioned that there are two main steps in the pipe routing process. The first step is finding the actual pipe path to connect nozzles, and after that the routed pipe is divided into one or more pipe spools. In this thesis, we only focus on the first step and skip the creation of the pipe spool. Since we know that the pipe spool creation process can be done quite easily in the CAD software package, this process should be performed there. It is decided to follow this approach because this research is intended to fill the functional gap in existing CAD software, instead of replacing them.

The result of the routing process must be as good as possible, and it implements the common knowledge that is described in Section 2.6 to ensure the quality. However, since we decided to pursue only the pipe path finding and skip the creation of the pipe spool, the common knowledge that is related to pipe spools is neglected.

The basic definition of the pipe routing process is to find an optimal path between two equipment's nozzles, more if a branch exists, while avoiding collisions. For its implementation in ship design, that pipe routing process must also follow the common facts for a piping system in a ship. Most of the pipes in a ship are rigid and normally those pipes are routed orthogonally to make the installation and maintenance easier. The pipes are divided into several different functional systems that require a different pipe specification. In a ship, the number of pipes is very large. Therefore, to lower the cost, it is preferred to have pipes follow routes that are as short as possible, and also to make sure that those pipes can be produced and installed as cheaply as possible. It means that we need to use a shortest path algorithm. The shortest path problem has been subject of research for years, and we will look in more detail at this in Chapter 4.

However, most of the times, the shortest path alone is not sufficient, because that path might not comply with the rules and regulations, nor indeed might it lead to lowest cost. For example, the shortest path algorithm that only optimize the length of a pipe might generate the pipe path that is too far from the steel construction, thus it is not possible to install a pipe support for that pipe. This leads us to define the first requirement to choose the shortest path algorithm; instead of merely trying to find the shortest distance, the shortest path algorithm must optimize the path in a weighted manner. The shortest path algorithm must rather be the lowest cost algorithm.

In some cases, there is also the possibility that a user wants to alter the pipe path manually. To accommodate this requirement, the methodology must have a functionality to allow a user to add/remove some conditions pertaining to the environment. For example, if the best path of a certain pipe is blocked by a plate, the user should be able to override that collision and let the pipe router module route the pipe through it.

Most of the shortest path algorithms aim to find one optimum path at a time, thus it optimizes only the pipe in hand without considering any other pipes in the group. Therefore the globally optimum set of pipe routes is difficult to be reached. In most cases, by choosing a good sequence of pipes to be routed, the global optimum solution can be approached to a fair measure. Whether the optimum found is the global optimum cannot be mathematically ascertained. The problem of choosing the right order of pipe routing is known as the combinatorial optimization problem.

Therefore to get (or at least approach) the globally optimum solution, we need to combine the shortest path algorithm and combinatorial optimization. So for the proposed methodology, we need to find the optimization method that is able to solve the combinatorial problem and can be used together with the shortest path

algorithm that is suitable for our purpose. Ultimately this hybrid optimization must be constructed.

In the previous section, we remarked that a user of the methodology may add an object constraint, for example a NoGo area. Since this is a manual process, there is always a possibility that the additional constraint might block some pipes. If this is the case, it might add more complexity to the hybrid optimization within the methodology. This might cause the hybrid optimization to keep trying to find a solution that is actually blocked by the additional NoGo area. To prevent this, the proposed methodology must be able to detect it before the hybrid optimization process is started. For this purpose, we need to utilize a fast path finding algorithm to be included in the methodology to act as the fast detector if a certain NoGo area blocks one or more pipes.

Section 2.5.1 and 2.6 discussed the common knowledge and strategy to route pipes in a ship. From the pipe router point of view, those rules must be translated into requirements for the pipe router algorithm. One of the design requirements stated in Section 1.3.4 mentions that a goal of the proposed methodology is to minimize user input and user decision. To fulfill this requirement, the predefined rules must be built as a knowledge base. It means that it should be easy to add and modify rules by the pipe routing experts.

Section 2.4.3 discussed that the way in which pipes are routed depends on the location of those pipes in a ship. In the manual routing process, by experience, the pipe engineer varies his or her routing behavior based on that. Consequently, the routing behavior of the pipe router module in the proposed methodology also should depend on the area in which the pipes are placed. Since each type of area has unique characteristics, the methodology has to include a different strategy of the routing process for a different type of area.

There is one other important matter that needs to be addressed. A pipe routing process is all about optimizing the path of pipes. The first thing that should be known before any optimization is performed is to define the objective of the optimization. Also, the validation criteria must be decided.

As we intend to use the 3D model built by means of the CAD software, we have to prepared to get the 3D model with a very high level of detail (as mentioned in Subsection 2.3.2). The number of objects in a ship can be more than one hundred thousand. The high level of detail in the 3D model may thus reduce the performance of the methodology with regard to the computation time to find the routing path and also for the performance of the user interface part. Obviously, we need to find a way to simplify the model while maintaining accuracy and routing validity. A brief introduction of model simplification will be given in Chapter 4.

The last part of the functional framework is to send back the result to the CAD software package, so the pipe engineer can continue to the second step of the routing process, splitting pipes into one or more pipe spools. This part of the methodology has been implemented, but since it is purely a computer programming problem, it will not be discussed in this thesis.

3.4 Literature Review of Automatic Pipe Routing

In this section, we survey the state of the art in automatic pipe routing. Automatic pipe routing has been a research topic for a long time resulting in various approaches, not only in the shipbuilding area, but also in process plant design. The research started with a 2-D workspace and simple obstacles, and gradually extended to the stage of a 3D workspace with multiple constraints and multiple objectives. In terms of the optimization technique, deterministic, heuristic, or a combination of both methods have been used to improve the results.

We filter out the studies that have small relevance for the pipe routing implementation in shipbuilding. This is important since the details of the routing process highly depend on the area of implementation.

However, it does not mean that the research that concentrates on other fields than shipbuilding can immediately be ignored. The similarity of the environment and the type of the paths (in our case pipes) are the most important things. As already explained in the previous section, in a ship the majority of the pipes are rigid and orthogonal. Thus, we still include in our survey the pipe routing studies which are implemented in industrial plants, even though in a ship the pipes are not routed in dedicated pipe racks which is common practice in an industrial plant.

Since the whole process concerns many fields of research, we focus on two main problem domains;

1. *The path finding strategy.* The in-depth review of the shortest path algorithms will be done in Chapter 4. In this section, we only survey the methods that were used in the reviewed literature, and consider if these also take into account practical aspects, like branches.
2. *The objective of the optimization.* Since our main goal is to use the proposed methodology in a real ship design process, the optimization objectives used in previous studies is an interesting aspect to be reviewed.

Unfortunately, to date most of the algorithms are demonstrated to solve the pipe routing problem only in academic situations, such as a system with only a few pipes and in a simplified environment. They have paid little or no attention to the scalability of the algorithm to be applied in the real ship design process which consists of a much larger number of pipes. Some of the researches also neglect the existence of pipe branches. Moreover, the algorithms are also mainly used for the space problem - to find a pipe route without collision - without considering the vitally important aspect that the solution must comply with the marine classification rules and regulations.

3.4.1 Early Years

The automatic pipe routing research started in the 1970's when Newell [1972] presented his work to route pipes automatically in chemical plants. He adopted Nicholson [1966] method to find the shortest path, and in his implementation,

Newell partly considered pipe branching. In 1974 by using the algorithm from Dijkstra [1959], Wangdahl, Pollock, and Woodward [1974] attempted to solve the pipe routing problem in a ship, but they only considered a two dimensional environment. The main drawback of their research is that since the pipes are routed one at a time, the globally optimum set of pipe routes might not be found since they do not include a mechanism to properly order the pipes. What makes it worse is that in some cases, by solving the shortest path problem one pipe at a time, a situation might result where a pipe can not be routed because it was blocked by the previously routed pipes. In his attempt to solve that problem, Rourke [1975] reviewed several algorithms but failed to find the solution.

3.4.2 Zhu and Latombe

In the year 1991, Zhu and Latombe [1991] described a system for automatically performing the pipe routing using robot path planning techniques. They regard each pipe as the trace left behind by a rigid object (a robot) moving in the pipe workspace, and a pipe routing problem as a multi-robot path planning problem. In their work, the approximate cell decomposition approach was chosen. Then to find the shortest path, the A* algorithm from Hart, Nilsson, and Raphael [1968] is utilized.

They also attempt to solve the condition where some pipes are blocking each other, using a strategy called backtracking. If a pipe was blocked by another pipe, the backtracking strategy will be triggered. That is, it must change the routes of some of the previously routed pipes to make room for the current pipe. Zhu and Latombe developed a sophisticated backtracking strategy that only considers the pipes that are actually blocking the current pipe.

The most interesting part in the work of Zhu and Latombe is that they also consider some practical rules by introducing the expert design constraints, which consists of process constraints, structural constraints and accessibility constraints. Process constraints relate to the process that is carried out in the pipes, e.g. a high temperature pipe should have an expansion loop to ensure thermal flexibility, a drainage pipe should be non-ascending and a heat sensitive pipe should be kept sufficiently far away from high temperature equipment. Structural constraints relate to the mechanical properties of the pipes, more precisely their capacity to remain in their position without falling down, e.g. pipes should be near enough to a major support structure such as a wall or a beam. Accessibility constraints relate to the constructibility of the pipe layout and its ease of operation and maintenance, e.g. there must be an access path for removing all major equipments for off-site repair and the frequently used valves should be accessible. Those three types of constraints then were formulated into two kinds of geometric constraints; Location constraints, and Shape constraints.

The location constraints specify the forbidden, undesirable, or preferable regions for a pipe route to go through. They are conceptualized as *hard virtual obstacles*, *soft virtual obstacle*, and *virtual sinks*. During the routing process, a

hard virtual obstacle acts as a real obstacle, while a soft virtual obstacle can be traversed by pipes with some additional cost. A virtual sink acts in the same manner as a soft virtual obstacle, but instead of getting a penalty, a bonus may apply to the pipes that are routed through this region. The location constraints are used during the cell generation step. The shape constraints apply to the shape of the pipe routes. For example it is to ensure that a drainage pipe should be non-ascending. The shape constraints are applied during the cell generation level and at the path generation level.

Their approach, however, was focused on the study case environment with a relatively small number of pipes and obstacles. The type of the obstacles are also only a basic shape, which might not be sufficient for the real environment. One other important thing that was left out is the aspect of pipe branching.

Unfortunately, they did not continue their research in the subject of pipe routing but were more interested in the subject of robotics.

3.4.3 Kang's Expert System

In 1996, Kang, Myung, and Han proposed a method for generating the optimal route for pipes using a knowledge-based expert system called NEXPERT, Kang et al. [1996] and Kang et al. [1999]. The knowledge-base is constructed on the basis of documented design knowledge and the empirical knowledge of human experts on the piping design of a ship. The system is modeled with the following objectives: to minimize user input and user decision; to structure the knowledge-base for easy addition of knowledge; to make the system easy to use; and to be used in the real shipyard design process.

In the expert system, there are three different objects; pipe-path, pipe-element, and space-element. Space-elements represent spaces and obstacles where pipe elements should or should not be placed. Constraints are implemented as rules, and algorithms are implemented as sub-routines. The knowledge-base in the system consists of three parts; the meta-control knowledge that makes main decisions, the global designer that finds the optimal arrangement of main pipes in a two dimensional section plan, and the detail designer which will expand the 2D section plan along the transverse coordinate into 3D space along the ship length.

In their research, Kang et al. constructed three different knowledge-bases that store 167 rules and 106 supporting methods. To verify their method, the piping design expert system had been tested to route pipes in the deck of a bulk carrier. Using the proposed system they can reduce the working time from 4 hours to 1 hour.

However, even though the way the knowledge-base was constructed makes it easy to be expanded with a new rule, their method is practically hard to be used since it is difficult to define all design knowledge quantitatively. Also the knowledge base is difficult to maintain in case some predefined rules are changed.

3.4.4 Sandurkar and Ito

In 1998, Sandurkar and Chen [1998] utilized a heuristic optimization approach based on the genetic algorithm of Goldberg [1989] to automatically perform the pipe routing. The interesting part of their research is that they are one of the first to use the real tessellated model as an obstacle rather than using only the boundary box of the model. Tessellation is the process of creating a two-dimensional plane using the repetition of a geometric shape with no overlaps and no gaps. Tessellation techniques are often used to manage data sets of polygons and divide them into suitable structures for rendering. Normally, at least for real-time rendering, the data is tessellated into triangles, which is sometimes referred to as triangulation.

By using the real tessellated model, the working environment is a closer representation of to the real environment when compared to only using the boundary of the real object. However, since the number of triangles in one tessellated complex object is large, the computation time to perform the collision detection is also larger. To tackle this problem, Sandurkar and Chen utilized the RAPID method from Gottschalk et al. [1996].

In their research, besides optimizing the pipe length, they also define the desired number of bends and the angle of the bends. This approach proved to be able to find a solution. However, it was applied only to one model with less than 10 obstacles and a single pipe and still took 18-19 hours of computations before the best layout of the pipes was found.

Using a different approach, Ito [1999] also utilized a genetic algorithm approach to find the best path. The workspace is defined by using the cell decomposition approach. Then, each cell is given a potential value, according to its location and characteristics. The potential value of the obstacle cells are high and the cells located next to the wall have the lowest potential value because that path is more favorable. The objective function is defined not only considering the pipe length, but to minimize the total cell value of the selected path.

This method was only tested on a two dimensional space and using primitive shapes for the obstacles. In practice, as described in previous chapters, the number of pipes that need to be routed is large. Also the complexity of the problem rises exponentially with the number of pipes to be routed. Therefore, even allowing for the large performance gains of computers, the method still seems unsuited for practical application.

3.4.5 Zuurmond

In 2004, Zuurmond [2004] proposed his approach to solve the automatic pipe routing problem, by utilizing the algorithm from Dijkstra [1959] to minimize the pipe length. Beside that, his method also restricts the drainage pipes to be non-ascending. The workspace is defined by using the cell decomposition approach.

For the obstacles Zuurmond simplified the real model into some cuboids. This

approach has an important advantage; to have a much lower computation time compared to the real tessellated model while still allowing a good representation of the real model.

The most important contribution of his work is not in the routing algorithm itself, but more in his description of the routing process in practice at a shipyard. Also, the importance to have the Smart P&I diagram tools was mentioned.

Since the aim of his research is global pipe routing, Zuurmond used a large cell size and always defined the pipe path in the center of axis of a cell. Therefore, even though the density of each cell was calculated to ensure that the total number of pipes that are routed in a certain cell always fit, the result still shows that there are collisions between pipes.

Another drawback is that before pipes can be routed automatically, a lot of manual setup must be done. He mentioned the importance of using the Smart P&I diagram, but still manually defined each connection point. Also, the simplification of the real model into cuboids is done manually.

3.4.6 Park and Storch

One of the most comprehensive research attempts in terms of considering more practical aspects is the research of Park and Storch [2002]. They considered many practical aspects, like branches, and practical constraints which will be translated into total pipe cost functions. Thus, their objective value is to minimize the total cost of the pipes.

The total cost of the pipes includes the material cost, installation cost and operability cost. The material cost of pipes and elbows depends on pipe size and length. The bending cost follows a step function because cold bending is used for small diameter pipes and high frequency bending is used for large pipes. As for installation cost, it is directly related to pipe-support cost. They also consider the distance between pipes and major structural object for pipe-support installation. The location of the valves are considered as part of the operability cost. They follow the guidance from ABS [1998] to measure the degree of comfort to operate a valve, then translate it into cost.

In terms of the routing method, they proposed the cell generation method. Using their method, the globally optimum set of pipe routes problem that was mentioned by Rourke [1975] can be solved.

However, even though they used a part of an engine room in a real ship, the research was intended as an early study. Until now it was not continued for more elaborate cases. Therefore, it was tested only for a small portion of the engine room. Also for reasons of simplicity, the boundary boxes of the real obstacles are used.

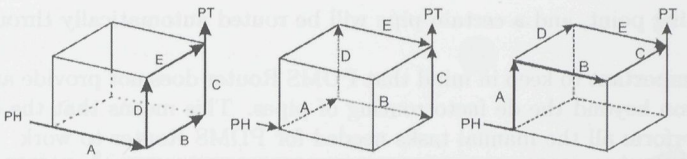


Figure 3.6: *Level 1 mode (excerpt from AVE [2007])*

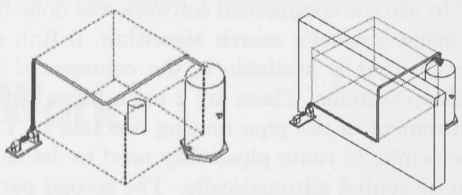


Figure 3.7: *Level 2 and Level 3 mode (excerpt from AVE [2007])*

3.4.7 Commercial Automatic Pipe Router

In the past few years, commercial pipe router software has been developed and deployed by ASD, Alias, and AVEVA. Judging from their commercial leaflets, they have some basic functional similarity. However, since the software is commercial, they did not make any technical publication on the algorithm that they use. The most detailed documentation available is the user guide of the Pipework Design User Guide from AVEVA AVE [2007].

The automatic router from AVEVA which is known as the PDMS Router routes the pipes orthogonally, and uses the graph theory principles from Gibbons [1985], and a shortest-path algorithm from Wang and Crowcroft [1992] to get the minimum pipe length. The objective of the PDMS Router is to minimize material cost while avoiding collision with the obstacles. To enhance its performance, the routing algorithm used by PDMS Router has three levels of operational modes; called Level 1, Level 2, and Level 3. Initially, Level 1 mode was used to find the path, and if it fails, a search is conducted using Level 2. Similarly, if no free route is found after Level 2, then Level 3 is used.

Fig. 3.6 shows how the Level 1 mode searches for an orthogonal route between the head point PH and tail point PT of a pipe, using the minimum number of bends and elbows. Fig. 3.7 shows the example of Level 2 and Level 3 mode of the PDMS Router. In the left figure we can see that the bending points can be dynamically moved along the boundary box of the path. The Level 3 is used if both Level 1 and 2 failed. In Level 3, the boundary box of the path is automatically extended to allow the pipe path to connect start and end points.

There are some other settings that can be defined by the user, such as pre-defined pipe racks and routing rules. It is also possible to create a point in a space

as a routing point, and a certain pipe will be routed automatically through that point.

It is important to keep in mind that PDMS Router does not provide any other automation beyond the de facto routing of pipes. This means that the designer has to perform all the manual tasks needed for PDMS Router to work. In order to reduce the human involvement in the tasks necessary for the PDMS Router, a research using PDMS Router has been done by Calixto, Bordeira, Calazans, Tavares, and Rodriguez [2009].

Another attempt to use the commercial software was done by Il Roh, Lee, and Choi. Rather than using a generic search algorithm, Il Roh et al. [2007] chose to improve the function that is available in the commercial 3D CAD software TRIBON and IntelliShip systems. There are 2 main parts in their research. The first one is the improvement of the pipe routing function in TRIBON and IntelliShip systems. In principle, to route pipes they need to define the pipe tray and then those pipes can be routed automatically. The second part of their research is to rapidly modify the pipes that are already routed when the hull structure is changed.

Both the original Tribon M3 automatic pipe routing and the two researches above that have been done to improve its standard functionality are not focusing on performing fully automatic pipe routing as we intended.

3.5 Summary

This chapter starts with the explanation of the basic outline of the functional framework of our proposed methodology. The type of information that is needed is described and the source and how to get it into the methodology is also briefly discussed.

The second part of this chapter reviews the previous researches that have been done in the field of automatic pipe routing. We found many interesting approaches but none of them led to an approach that satisfies our targets, i.e. fully automatic routing applicable for detailed design phase and for complex, real ship situations in a practically applicable methodology.

While reviewing, we found the answer to the first research question. That question asks for the phase that we should concentrate our effort on; in the pre-contractual or detail design phase, and the reason why we choose to that. In his work, Zuurmond [2004] routes the pipes in a machinery room of a real ship. What he did is the approximate routing so the result can be found almost immediately. This approach is not suitable to be used for detail design phase. However, from Subsection 1.3.1 we knew that during the pre-contractual phase, approximate routing is sufficient. Therefore, we can say that to solve the approximate routing in the pre-contractual phase, we can simply adopt that method. For the sake of widespread applicability, we focus on solving the pipe routing problem in the detail design phase.

Chapter 4

Related Work

In the first part of Chapter 3 we have discussed five main points that need to be investigated; the shortest path problem, combinatorial optimization, the knowledge base, the differences of behavior according to area type, and objective function. In this chapter the first two points will be discussed along with the problem of model simplification. The third until the fifth points will be discussed in Chapter 5.

We begin with surveying previous studies of each subject, then comparing those studies to find the most suitable solution and investigate how to improve the existing solution to fulfill our functional framework requirements.

4.1 Shortest Path Problem

In graph theory, the shortest path problem is the problem of finding a path between two vertices (nodes) in a graph such that the sum of the weights of its constituent edges is minimized. An example is finding the shortest way to get from one city to another on a road map, shown in Fig. 4.1; in this case, the vertices represent cities and the edges represent the segments of the road and are weighted by the travel distance.

For example, if we need to answer the question "What is the shortest travel distance to drive from Hardinxveld to Amsterdam?", we might use the graph in Fig. 4.1 and use one of the shortest path algorithms to find the optimal solution. Many researches have been done to solve the shortest path problem and it is becoming one of the most prominent generic problems in various fields. One of the reasons for this is that essentially any combinatorial optimization problem can be formulated as a shortest path problem. Thus, this class of problems is extremely large and includes numerous practical problems that have nothing to do with actual shortest path problems.

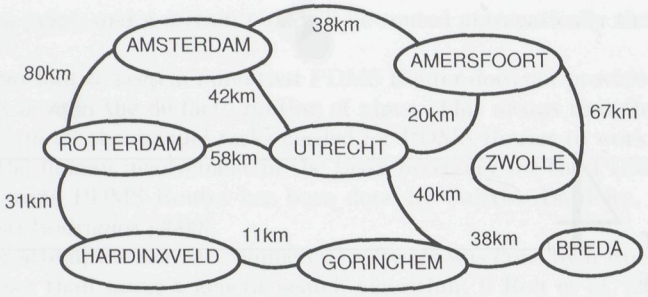


Figure 4.1: *Example of a road map*

Many approaches have been made ranging from using the deterministic (exact) algorithm, subsequently introducing the heuristic part to improve the algorithm, and recently researches exclusively using heuristic algorithms to solve it.

In this section, we walk through those algorithms and choose the algorithm that suits our requirements. There are two criteria to be considered. The first one is the performance of the algorithm itself. There are some important issues that need to be addressed to measure the algorithm performance to solve the shortest path problem; always find a solution if it exists, always find the optimal solution, and use limited resources in terms of computer memory and time. In our methodology, pipes can be routed freely in the area, therefore the selected algorithm must be able to be implemented to route pipes in a free space.

We also consider the flexibility of those algorithms to be extended and adapted in our methodology. In chapter 2, we have discussed some practical knowledge that help us define the basic capabilities that must be utilized in our methodology. We have discussed that there are some practical aspects that need to be addressed beyond the shortest length and minimum number of bends. To have pipes routed nicely in parallel, to route pipes close enough to the steel construction for ease of installing the pipe support, and to follow the marine classification guidance, are more important than only trying to route pipes such that they have minimum length.

The nature of a shortest path algorithm is to find the path between start and end points that has a shortest distance. To make the shortest path algorithm choose the path that satisfies the important practical aspects above, we need to modify the environment to fit the algorithm, for example by using a different weighted cost, or using potential energy techniques. Because of that, in addition to four requirements above, the selected algorithm must also allow implementation to find the path in that type of environment.

As a part of our methodology, we also need to have a fast function to detect that a certain pipe can be routed at all. For this functionality the only important requirement to choose the shortest path algorithm is that the selected algorithm

always finds a solution if it exists. This will be discussed in chapter 5.

In the next few sections, some approaches to solve the shortest path problem are discussed. We start with the deterministic approaches, followed by the heuristic methods to solve this problem. After that, a comparison between various methods to solve s simple shortest path problem will carried out.

4.2 Deterministic Approaches

4.2.1 Graph Traversal Algorithms

In graph theory, one of the most fundamental tasks in an algorithm is visiting the vertices and edges of a graph in a systematic order. There are at least three different traversal techniques that are frequently used; Depth-first search, Breadth-first search, and Best-first search.

Breadth-first Search

The breadth-first search (BFS) algorithm is an uninformed search that systematically visits all the vertices of a graph until a goal vertex is found or all vertices are visited. This algorithm starts at a designated start vertex and then examines the neighbours of that vertex and puts it in the queue stack. Once a vertex has been examined, it is marked as explored. After all neighbours have been examined, it visits all of them one by one and examines their neighbours. This process is repeated until all vertices of the entire graph have been visited, or until the goal vertex has been reached. In other words, it visits the vertices of a graph uniformly across the breadth of the frontier of its search, visiting all vertices at distance (d) from the start vertex before looking for vertices at distance $(d+1)$. For the order of the search, BFS algorithm uses a First-in First-Out (FIFO) queue stack.

We use the graph of Fig. 4.2 to illustrate the Breadth-first search algorithm. In this example, the start vertex is Hardinxveld, and we would like to find a path to Amsterdam. Before we use BFS algorithm, we define an order of visiting the neighbours. Fig. 4.2.a and b illustrate BFS algorithm with visiting order from the right to the left side and from the left to the right side respectively.

In Fig. 4.2.a, BFS starts from Hardinxveld and examines its neighbours, Gorinchem and Rotterdam, marks it as explored and stacks it in a queue. After that BFS visits Gorinchem and examines its neighbors Breda, Utrecht, and Hardinxveld. Since Hardinxveld was already marked, it won't be explored further. Breda and Utrecht are then marked and added to the queue stack. The next vertex in the queue stack is Rotterdam. From all three neighbours of Rotterdam, only Amsterdam is unmarked. Therefore, Amsterdam is marked and added to the queue stack. Eventually, our goal vertex has been reached, but in this example, we continue to run the algorithm to build the complete tree.

Breda is now in the top of the queue stack, but all of its neighbours are marked. From Utrecht, the search is continued and results in Amersfoort to be marked and

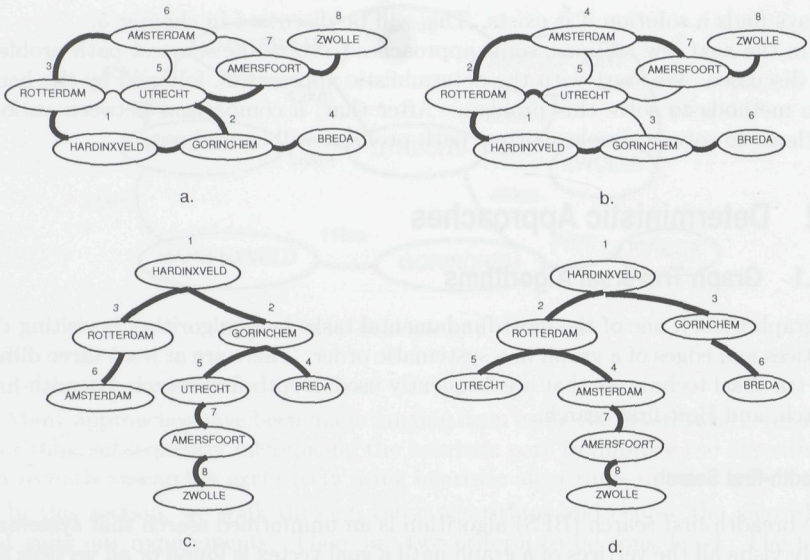


Figure 4.2: *Breadth-first search on the road map tree*

added to the queue. Since all edges from Amsterdam lead to the marked vertices, no vertex is added to the queue stack. Then it visits Amersfoort and then marks Zwolle. The resulting Breadth-first search tree is shown in Fig. 4.2.c. Fig. 4.2.d shows the tree in the left to right visiting order.

By comparing the BFS tree in Fig. 4.2.c and Fig. 4.2.d, it can be seen that the BFS algorithm is highly sensitive to the visiting order of the vertices. If the graph is connected, BFS will find a solution because it explores all vertices.

For unit-step cost, BFS is optimal. In general, Breadth-first search is not optimal since it always returns the result with the fewest segments between the start vertex and the goal vertex. As in our example above, if the graph is a weighted graph and has costs associated with each step, a goal next to the start does not have to be the cheapest goal available. This problem can be solved by improving Breadth-first search to uniform-cost search, which considers the path costs. Nevertheless, if the graph is not weighted and all step costs are equal, Breadth-first search will find the nearest and the best solution.

The BFS algorithm begins with straightforward initialization that requires $O(1)$ time. In the worst case, the algorithm needs to visit all vertices before the goal is reached. This require time $O(V)$ with V the number of vertices. The algorithm also examines all edges of each vertex, and since there are two vertices connected by one edge, the total number of examinations is 2 times the number of edges, $O(E)$ with E the number of edges. We know that $E_{max} <= V_{max} - 1$,

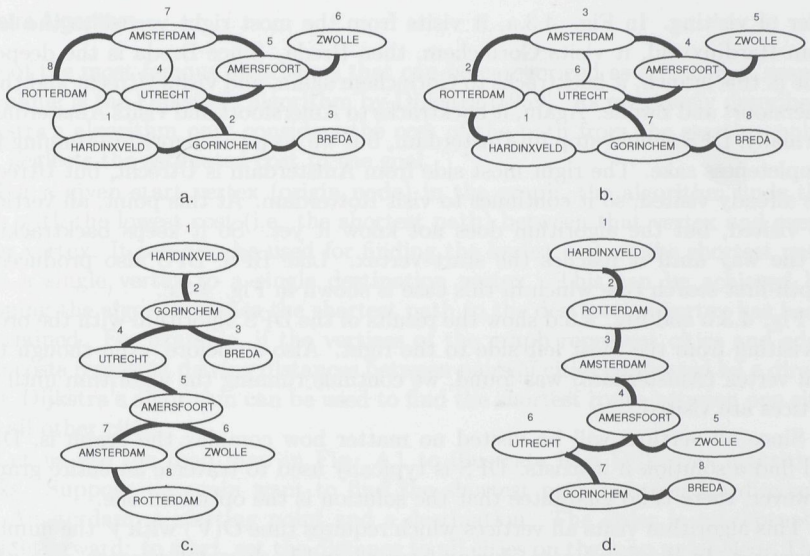


Figure 4.3: *Depth-first search on the road map tree*

so $O(V+E)$ can be simplified to $O(V)$. Since it uses FIFO stack, it only requires $O(1)$ time for the dequeue process. In total, the time complexity of BFS algorithm is $O(V)$.

Furthermore, BFS needs to memorize the state (marked or unmarked) of all vertices. It also needs a queue stack memory with the maximum number of the stack equal to the number of all vertices. So the space complexity is $O(V)$.

Depth-first Search

Another fundamental search algorithm in graph theory is the Depth-first search algorithm. Like the BFS algorithm, Depth-first search (DFS) is also an uninformed search, so the visiting order of vertices can be arbitrary. The main difference with BFS algorithm, which explored all the neighbors of a given vertex at a time, is that the DFS algorithm explores only one neighbor at a time, and then explores a path in a graph as far as possible until a goal vertex is found, or until it hits a vertex that has no children. When it is no longer possible to go forward, the algorithm backtracks one level and then tries again to go deeper. This process repeats until all vertices of the entire graph have been visited.

To show how the DFS algorithm works, we use the graph of Fig. 4.1 and define Hardinxveld as the start vertex and Amsterdam as the goal vertex. However in this example, we let the DFS algorithm run until all vertices are visited.

As shown in Fig. 4.3.a and Fig. 4.3.b, this algorithm highly depends on the

order of visiting. In Fig. 4.3.a, it visits from the most right vertex to the left. From Hardinxveld, it visits Gorinchem, then Breda. Since Breda is the deepest node in this branch, it backtracks to Gorinchem again, and visits Utrecht and then Amersfoort and Zwolle. Again, it backtracks to Amersfoort and visits Amsterdam. Normally, DFS should stop in Amsterdam, but we keep the algorithm running for completeness sake. The right most side from Amsterdam is Utrecht, but Utrecht was already visited, so it continues to visit Rotterdam. At this point, all vertices are visited, but the algorithm does not know it yet. So it keeps backtracking all the way until it reaches the start vertex. Like BFS, DFS also produces a Depth-first search tree which in this case is shown in Fig. 4.3.c.

Fig. 4.3.b and Fig. 4.3.d show the results of the DFS algorithm with the order of visiting from the most left side to the right. Also as before, even though the goal vertex (Amsterdam) was found, we continue running the algorithm until all vertices are visited.

Since all vertices will be visited no matter how complex the graph is, DFS will find a solution if it exists. DFS is typically used to traverse an entire graph. However, there is no guarantee that the solution is the optimum one.

This algorithm visits all vertices which requires time $O(V)$ with V the number of vertices. In the worst case, during forward and backtracking search each edge is examined two times which requires $O(E)$ with E the number of edges. We know that E is always less than V, such that the total time needed is $O(V)$.

For the sake of backtracking, the DFS algorithm needs to memorize all vertices which requires $O(V)$ memory space.

Generic Best-first Search

Unlike both Depth-first search and Breadth-first search, Best-first search explores the graph not uniformly following the depth or breadth of the graph, but it expands the vertex with the best value. Normally, Best-first search is used on a weighted graph and uses a "heuristic evaluation function" that estimates the minimum cost from any vertex to the goal.

In the formal terminology, $g(n)$ represents the cost of the path from the starting point to any vertex n, and $h(n)$ represents the heuristic estimated cost from vertex n to the goal. Best-first search balances the two as it moves from the starting point to the goal. In each step of the iterative procedure, it examines the vertex n that has the lowest $f(n) = g(n) + h(n)$.

It is crucial to choose an appropriate heuristic function because it affects the behaviour and performance of the algorithm. At one extreme, if $h(n)$ is extremely high relative to $g(n)$ (or simply neglecting $g(n)$), then only $h(n)$ plays a role. In this case, this algorithm becomes the Greedy Best-first search algorithm, which is sometimes called the Greedy algorithm. At the other extreme, if $h(n)$ is 0, then only $g(n)$ plays a role, and this turns into Dijkstra's algorithm. In between, if both $h(n)$ and $g(n)$ are included in the algorithm, we get the A* algorithm (see below).

Dijkstra Algorithm

One of the most famous algorithms that can be categorized as a Best-first search technique is the Dijkstra's algorithm by Dijkstra [1959]. As previously described, Dijkstra's algorithm only considers the cost of the path from the starting point and neglects the estimated cost to the goal.

For a given start vertex (origin node) in the graph, the algorithm finds the path with the lowest cost (i.e. the shortest path) between that vertex and every other vertex. It can also be used for finding the lowest cost or the shortest path from a single vertex to a single destination vertex. This can be achieved by stopping the algorithm once the shortest path to the destination vertex has been determined. For example, if the vertices of the graph represent cities and edge path costs represent driving distances between pairs of cities connected by a direct road, Dijkstra's algorithm can be used to find the shortest route between one city and all other cities.

Let us use the road map in Fig. 4.1 to illustrate how Dijkstra's algorithm works. Suppose we again want to find the shortest path between Hardinxveld and Amsterdam, a starting point and a destination. The order is conceptually straightforward: to start, set the distance to all cities on the map unlabeled. This is done not to imply that there is no distance, but to note that that intersection has not yet been examined. Some variants of this method set the distance to infinity on all cities. In Fig. 4.4.a, Amsterdam as a destination is marked with blue color, and Hardinxveld is marked with green colour to show that it is currently the best vertex and will be explored.

Now, at each iteration, select a current city. For the first iteration, the current city will be the starting point (Hardinxveld) and the distance to it (the Hardinxveld's label) will be zero. For subsequent iterations (after the first), the current city will be the closest unvisited city to the starting point.

From the current city, the algorithm updates the distance to every unvisited city that is directly connected to it. This is done by determining the sum of the distance between an unvisited city and the value of the current city, and relabeling the unvisited city with this value if it is less than its current value. In effect, the city is relabelled if the path to it through the current city is shorter than the previously known paths.

Dijkstra's algorithm uses Breadth-first search at this stage, because it examines every neighboring city before it moves on to the next iteration and puts a label to the examined cities, as shown in Fig. 4.4.b.

Fig. 4.4.c and d shows that Hardinxveld is marked as visited (red colour) and Gorinchem acts as the current city because it has the lowest label (11), and Fig. 4.4.d shows the updated distance label of Gorinchem's neighbors.

In Fig. 4.4.e and f, Rotterdam acts as the current city since it now has the lowest value, and the algorithm labels Utrecht and Amsterdam. As we can see that the new value of Utrecht is bigger than the previous value ($89 > 51$), so the link from Rotterdam to Utrecht is not a valid path. Also, Amsterdam has been

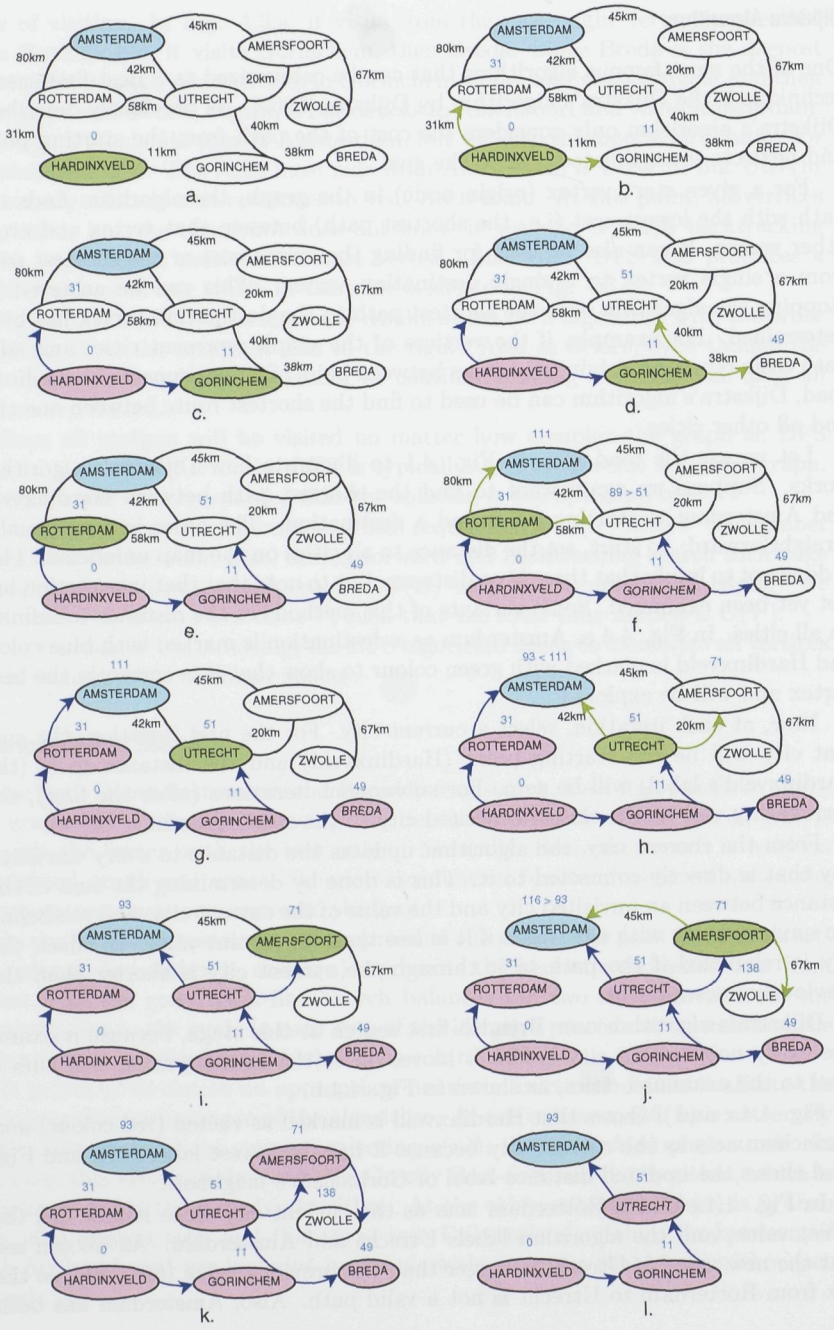


Figure 4.4: *Dijkstra's algorithm*

labelled at this point, but because there are still vertices having lower value than the current value of Amsterdam, the iteration will continue.

In the next iteration, Breda is marked as visited vertex because it does not have an unvisited neighbour. Thus, Utrecht has the next best value and it yields labels on Amersfoort and Amsterdam. The new value of Amsterdam is lower than its previous one ($93 < 111$). The value of Amsterdam is therefore updated to 93 and the link from Rotterdam to Amsterdam is not valid anymore as shown in Fig. 4.4.g to i.

The next best city is Amersfoort, and its neighbors, Amsterdam and Zwolle, are labeled. The new value of Amsterdam is bigger than its previous value ($109 > 93$) which will cause to maintain the old link between Utrecht and Amsterdam, and Amersfoort is marked as visited. The next best city is Amsterdam, and since Amsterdam is our goal vertex, the algorithm stops. The result of Dijkstra'a algorithm is shown in Fig. 4.4.l.

Of note is the fact that this algorithm makes no attempt to direct "exploration" towards the destination as one might expect. Rather, the sole consideration in determining the next "current" intersection is its distance from the starting point. In some sense, this algorithm "expands outward" from the starting point iteratively, considering every vertex that is closer in terms of the shortest path distance until it reaches the destination. When understood in this way, it is clear how the algorithm necessarily finds the shortest path. However, it may also reveal one of the algorithm's weaknesses: its relative slowness in some topologies.

The main advantage of Dijkstra's algorithm is that it always finds the shortest path. However, because the number of vertices in a pipe-routing application are extremely large, it takes relatively much calculation time and occupies much memory as it needs to memorize the state of all vertices which requires $O(V)$ space of memory.

The time complexity of Dijkstra's original algorithm is $O(|V|^2)$, because in the worst case it needs to visit all vertices requiring time $O(V)$, and on each iteration it needs to find the vertex with the best value that requires again $O(V)$. In 1984, Fredman introduced the use of Fibonacci Heap as a min-priority queue, and it improves the performance to $O(VlogV)$. This is asymptotically the fastest known single-source shortest-path algorithm for arbitrary directed graphs with unbounded nonnegative weights.

Greedy Algorithm

The Greedy algorithm is the fastest of all varieties of the Best-first search technique, and it is well known as Best-first search algorithm itself. The Greedy algorithm works in a similar way as Dijkstra's algorithm, except that it uses the estimation (called a heuristic) of how far from the goal any vertex is. Instead of selecting the vertex closest to the starting point, it only considers the estimated distance to the goal. It runs much quicker than Dijkstra's algorithm because the heuristic function guides the algorithm towards the goal quickly. For example, if

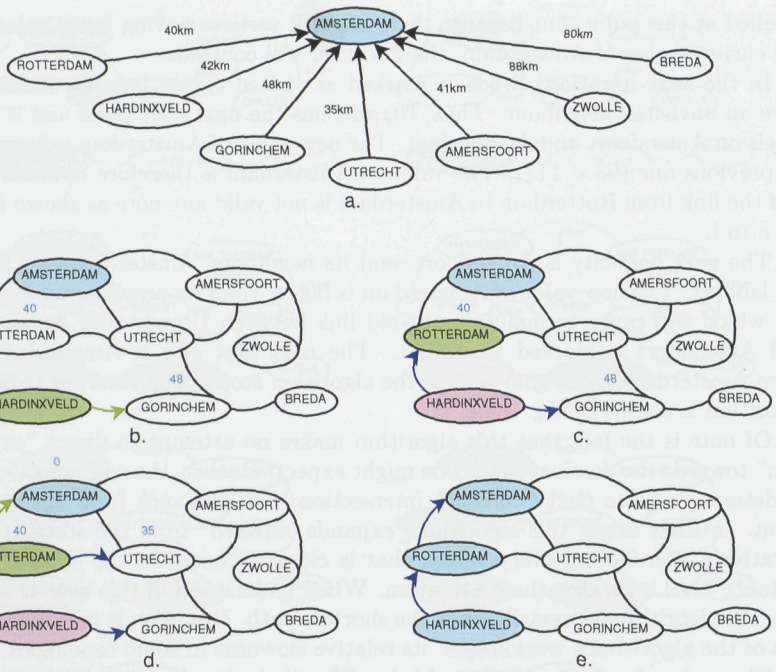


Figure 4.5: *Greedy algorithm*

the goal is to the north of the starting position, Greedy algorithm will tend to focus on paths that lead northwards.

Fig. 4.5 illustrates how the Greedy algorithm works to find a shortest path from Hardinxveld to Amsterdam. We introduce the actual distance (not a driving distance but "as the crow flies") between Amsterdam (as the goal vertex) and other cities, as the heuristic values of the algorithm (Fig. 4.5.a).

For the first iteration, it starts to find a city that is directly connected to Hardinxveld and selects the city closest to Amsterdam. Fig. 4.5.b shows that Rotterdam is closer to Amsterdam than Gorinchem.

The Greedy algorithm chooses Rotterdam as its next current city. In the next iteration, it compares Utrecht and Amsterdam, and it obviously chooses Amsterdam because Amsterdam is the goal as shown in Fig. 4.5.d.

In general cases, Greedy algorithms mostly (but not always) fail to find the globally optimal solution, because they usually do not operate exhaustively on all the data. In terms of time complexity, the worst case of this algorithm is the same as Dijkstra's original algorithm, $O(|V|^2)$, but in practice it mostly visits a lower number of vertices compared to Dijkstra's.

A* Algorithm

Historically, the A* algorithm is an improvement of Dijkstra's algorithm that introduces the heuristic part in its cost calculation. Hart, Nilsson and Raphael first described the A* algorithm in Hart et al. [1968]. This algorithm combines the pieces of information that Dijkstra's algorithm uses (favoring vertices that are close to the starting point) and the heuristic cost (favoring vertices that are close to the goal).

Using the same example, Fig. 4.6 illustrates how the A* algorithm works to find the shortest path from Hardinxveld to Amsterdam. The heuristic value $h(n)$ that is shown in Fig. 4.6.a describes the actual distance ("as the crow flies") from Amsterdam to other cities. For a better explanation, in this figure the total fitness value $f(n)$ is shown as partial $g(n) + h(n)$.

It starts by examining the direct neighbors of Hardinxveld, and compares the total value $f(n)$. Fig. 4.6.c shows that the fitness value of Gorinchem is smaller than Rotterdam's, so Gorinchem is visited by the algorithm. Then it examines Breda and Utrecht and labels both with fitness value 129 and 86 respectively.

The next best city is Rotterdam. The A* algorithm examines Amsterdam and Utrecht. For Amsterdam, it simply labels it, but for Utrecht, since that city had been examined before, the old fitness value needs to be compared with the new one. In Fig. 4.6.e, we can see that the new value is higher than the old one, which means that the total cost from Hardinxveld to reach Utrecht is cheaper through Gorinchem rather than through Rotterdam, so the link from Rotterdam to Utrecht will not be used.

Utrecht has the best fitness value, so the A* algorithm starts to examine Utrecht's neighbours, Amersfoort and Amsterdam, and gives a label for both cities. Amsterdam already has a label, so the algorithm compares both fitness values. As we can see in Fig. 4.6.g, the new fitness value is smaller that the old one, so the valid link to Amsterdam is through Utrecht instead of Rotterdam.

Fig. 4.6.h shows that Amsterdam has the best fitness value. It means that the solution is found as shown in Fig. 4.6.i.

The characteristic of the A* algorithm is highly sensitive to the behaviour of the heuristic function that is used. If $h(n)$ is always lower than (or equal to) the cost of moving from n to the goal, then the A* algorithm is guaranteed to find a shortest path. The lower $h(n)$ is, the more the algorithm expands and the slower it becomes.

If $h(n)$ is exactly equal to the cost of moving from n to the goal, then the A* algorithm will only follow the best path and never expands anything else, making it extremely fast. Although we can't make this happen in all cases, we can make it exact in some special cases. It is satisfying to know that given perfect information, the A* algorithm will behave perfectly. If $h(n)$ is sometimes greater than the cost of moving from n to the goal, then the A* algorithm is not guaranteed to find a shortest path, but it can run faster than Dijkstra or Greedy.

As in Dijkstra' algorithm, to improve the performance of the A* algorithm,

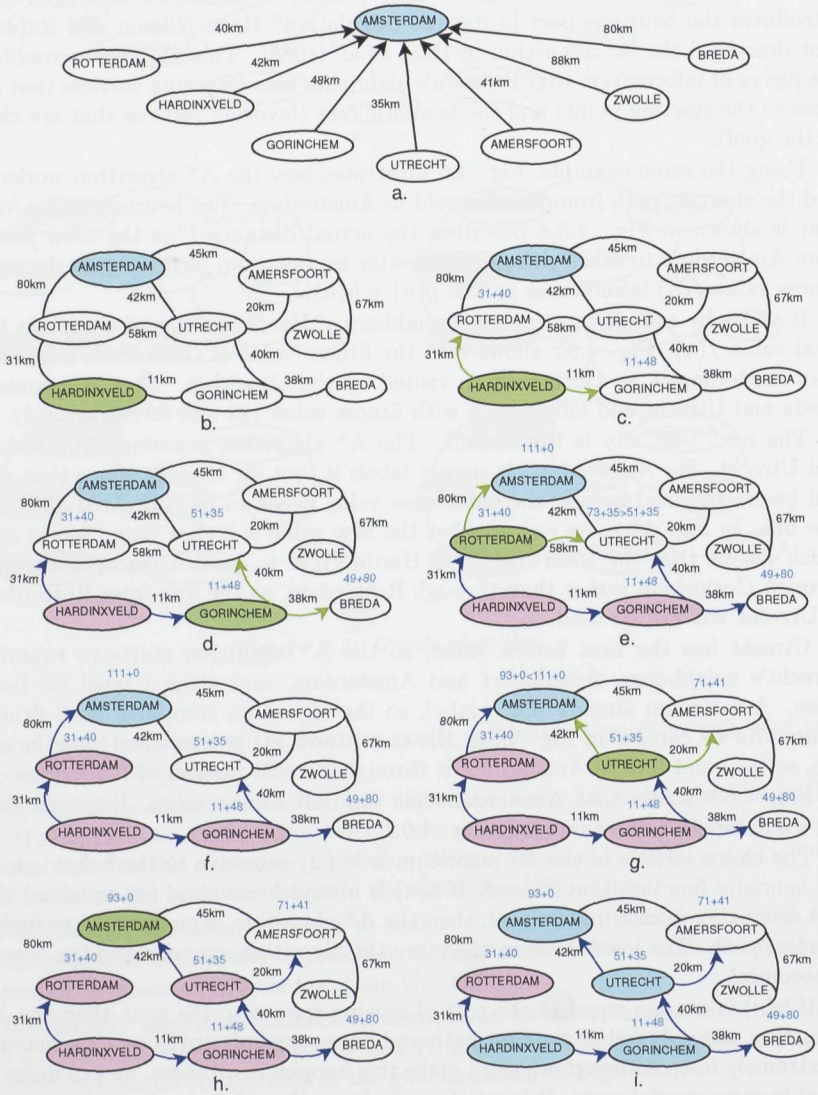


Figure 4.6: *A* algorithm*

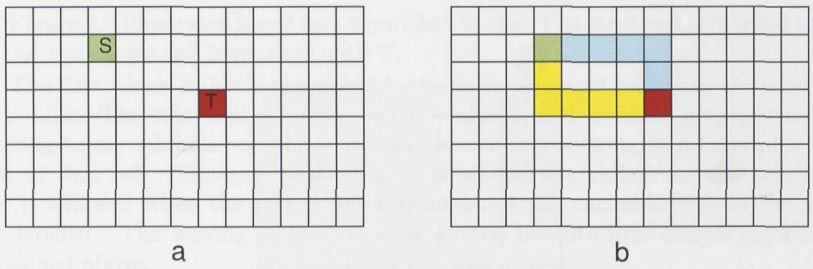


Figure 4.7: *Shortest path in a free space*

Fibonacci Heap is used as min-priority queue. The time and space complexity of the A* algorithm is the same as Dijkstra's algorithm, but since the heuristic part directs the search toward the destination, the number of vertices that are visited are smaller than for Dijkstra's. However, there is a possibility that the path that is found is not optimal.

4.2.2 Maze Algorithm

In previous examples, we used a graph as the environment. In a graph, it is assumed that the exact path (or every edge) that connects two vertices is already pre-defined. In contrast, two vertices can be connected in many ways in a free space. Fig. 4.7 shows that there are two solutions that connect node S with node T with the same distance and number of turns.

Seen from another point of view, in fact a graph and a maze share the same principle. In a maze, the space is divided into cells, and each cell has its own associated cost. A graph is consists of vertices and edges that connect one vertex with another. In principle, a cell is the same as a vertex, and an edge represents the associated cost.

Starting from this subsection, we move on to survey shortest path algorithms to find a path between nodes in a free space. In this subsection, the space is divided into cells, and each cell has its own associated cost. Some cells are marked as obstacles. The goal of any algorithm we study is to find the path that connects the start cell (containing the start point) with the end cell. Fig. 4.8 shows an example of a grid of cells in 2D space. In this example, it is shown that each cell has eight direct neighbors, with only four of them directly accessible.

Lee's Algorithm

The most common maze algorithm was proposed by Lee [1961], and is called Lee's algorithm. Lee presented a wavefront approach to solve the shortest path problem. This approach is similar to Dijkstra's algorithm.

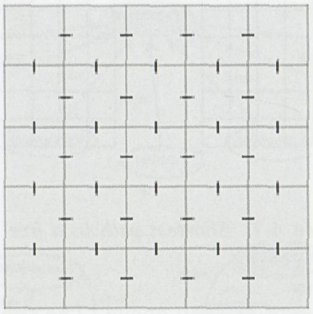


Figure 4.8: *Grid of cells*

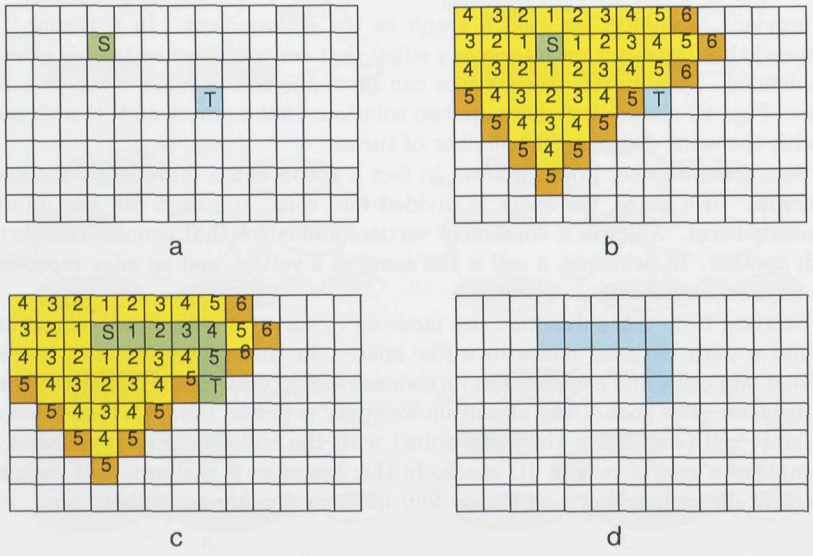


Figure 4.9: *2D Lee's algorithm*

Figure 4.9 illustrates how Lee's algorithm works. The start cell is marked with S, and the target cell is marked with T.

The first phase in Lee's algorithm is the wave propagation phase as shown in Fig. 4.9.b. The start cell is labeled with a value 0, and it starts propagating by walking to an unlabeled neighbor cell and marks this with its own label plus the cost of that cell. This propagating process should not violate an obstacle cell, and it will end when the target cell is reached, or all reachable cells in the grid are labeled. The waving process runs as in the breadth-first search algorithm explained above.

After the propagating process completes, it will do the backtracking phase. This starts from the target cell, and it will move to the next neighboring cell that has a lower value. All cells that are encountered during the backtracking are kept as the result path. The backtracking phase can be seen in Fig. 4.9.c.

If a solution exists, this algorithm guarantees to find it. If every cell cost is equal, the lowest number of steps is equal to the minimum cost, therefore the propagation process can be stopped after it reaches the target. However, if the cell cost is not uniform there is a possibility that the lowest step path is not the minimum cost, therefore the propagation process must be continued until all cells are labeled. In general, in the non uniform cost environment this algorithm requires more time to find the optimal solution. The example shows that the algorithm needs to explore many cells before it finds the solution.

If an obstacle exist, Fig. 4.10 shows that Lee's algorithm needs to explore an even larger amount of cells.

Beside the high time complexity, Lee's algorithm also uses a lot of memory, because it needs to save the state of all visited cells. In the worst case, Lee's algorithm needs to memorize all cells in the space.

A* Algorithm

In the previous section, we have surveyed how the A* algorithm can be used to find the shortest path between vertices in a graph. We also have seen that the A* algorithm outperformed Dijkstra's algorithm in terms of the calculation time and memory usage, and if we wisely choose the heuristic function then the solution that is found by the A* algorithm is optimal.

Now, we investigate the performance of the A* algorithm to find the shortest path between nodes in a free space. The A* algorithm requires the space to be divided into a grid of cells in the same manner as when we use Lee's algorithm. Also, as in Lee's, A* will perform two phases of searching. First it performs front wave phase. Then after the goal is reached, it performs backtracking to the start node.

The A* algorithm in a free space has fitness value $f(n) = g(n) + h(n)$, where $g(n)$ represents the cost that is needed to travel from the start node to node n, and $h(n)$ represents the estimated cost from node n to the goal node. The difference

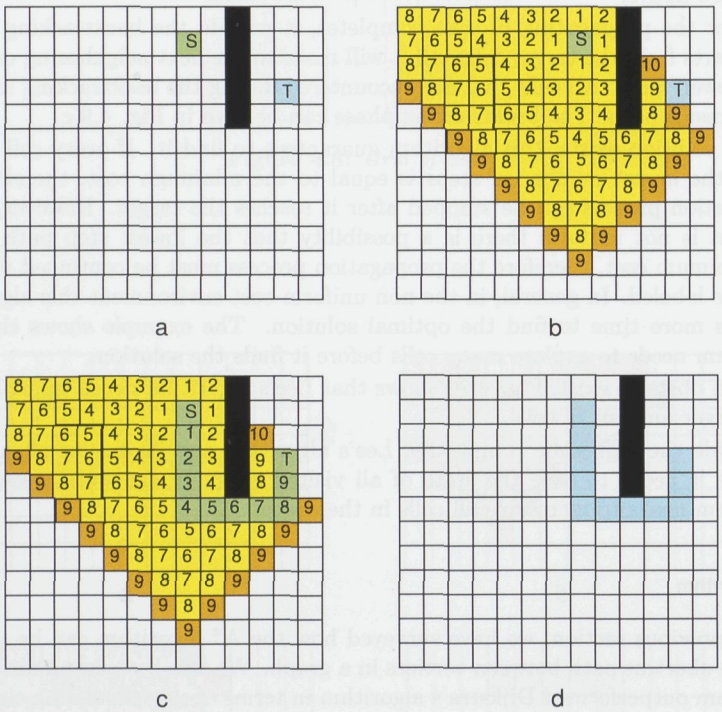


Figure 4.10: *2D Lee's algorithm with obstacle*

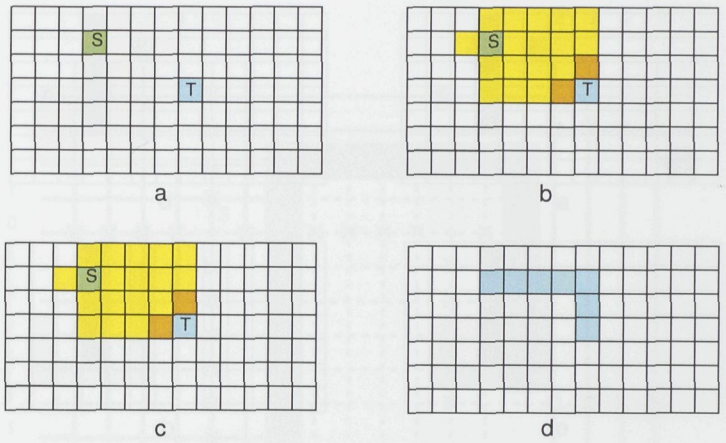


Figure 4.11: *2D A* algorithm*

is that normally in a graph, the value of $h(n)$ can be pre-calculated, while in a free space, the calculation of $h(n)$ must be done on the fly.

Figure 4.11 and 4.12 show the performance of the A* algorithm to solve the same problem as Lee's algorithm in the previous subsection. Comparing those figures with Fig. 4.9 and 4.10, A* is the obvious winner.

However, if the obstacle is more complex, as shown in Fig. 4.13, the A* algorithm still needs to observe many cells before it can find a solution, and in the worst case, it needs time and memory as large as Lee's.

4.2.3 Line search algorithm

By using a maze algorithm, the shortest path problem in a free space can be solved trivially. However, if the space is loaded with complex obstacle, even the A* algorithm will suffer. In this subsection, we survey line-search algorithms that should perform better than a maze algorithm in term of calculation time and memory usage.

Mikami-Tabuchi's Algorithm

The first line-search algorithm was proposed by Mikami and Tabuchi [1968]. In a two dimensional environment, this method starts with generating four lines (two horizontal and two vertical) through the starting point S and target point T. These lines are extended until they hit obstructions or the boundary of the space. If a line generated from S intersects a line generated from T, then a connecting path is found. If they do not intersect, they are identified as trial lines of $level_0$. These lines are stored in temporary storage for further processing.

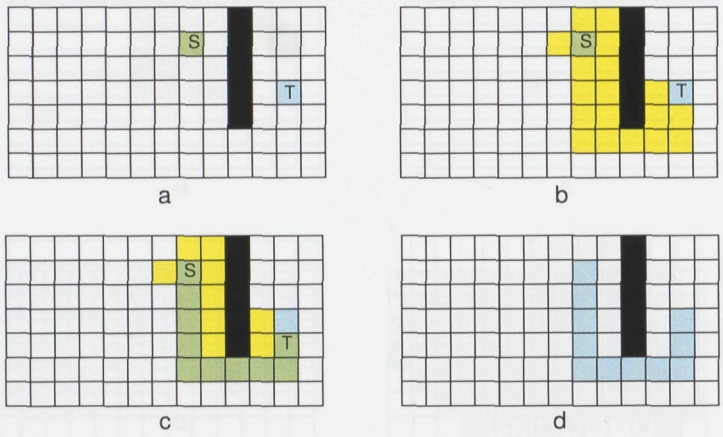


Figure 4.12: *2D A* algorithm with obstacle*

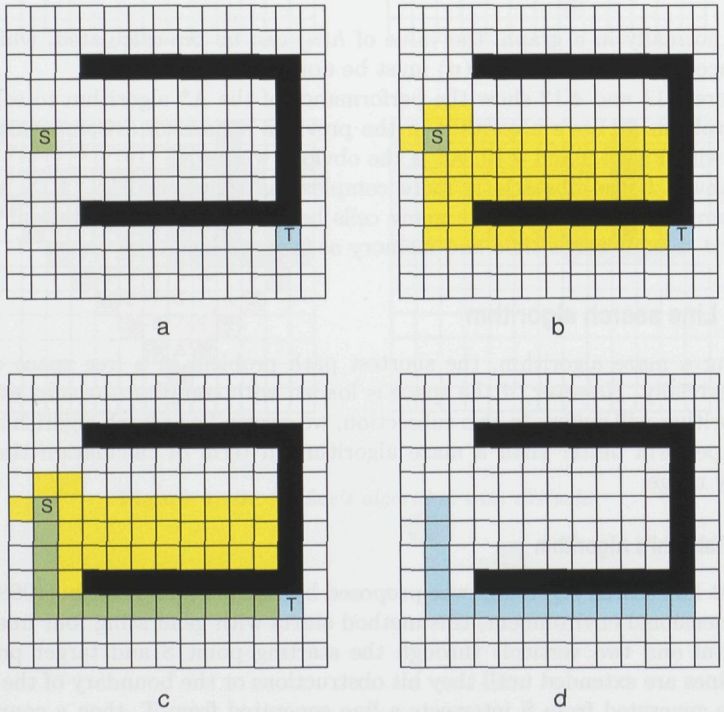


Figure 4.13: *2D A* algorithm with obstacle*

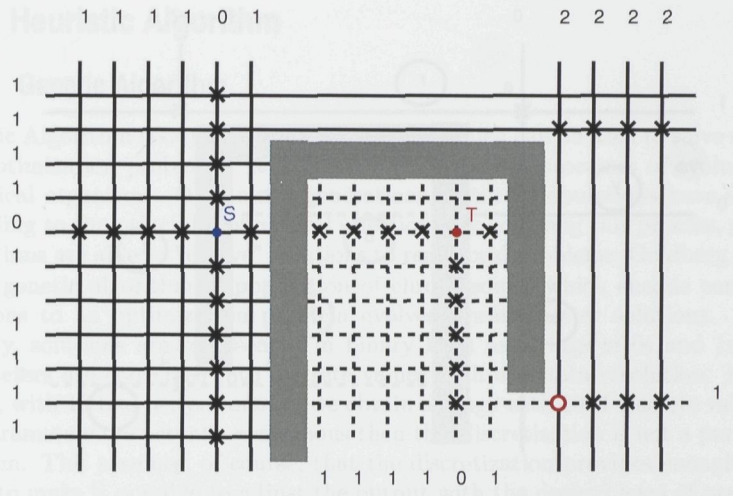


Figure 4.14: *Mikami-Tabuchi's algorithm*

After that, the algorithm starts the iteration procedure. For each step i of iteration, it creates trial lines for each grid $level_i$, one at a time. Along the trial line, all its base points are traced. From these base points, new trial lines (of $level_{i+1}$) are perpendicularly generated to the trial lines of $level_i$. If a trial line of $level_{i+1}$ intersects a trial line (of any level) from the other terminal point, the connecting path can be found by backtracking from the intersection point to S and T. Otherwise, all trial lines of $level_{i+1}$ are added to temporary storage, and the iteration procedure repeated.

We use Fig. 4.14 to illustrate Mikami-Tabuchi's algorithm. At the beginning, two lines (horizontal and vertical) are generated from the start point S and the target point T as trial lines $level_0$. Since those lines are not connected, it generates new trial lines $level_1$ from S and also from T. At this stage, there is still no intersection between the trial lines from S and T. Therefore it starts to generate trial lines $level_2$. In Fig. 4.14, the intersection between trial line $level_2$ from S and trial line $level_1$ from T is shown by a red circle. Then the algorithm performs backtracking from the intersection point to both S and T.

Even though this algorithm guarantees to find a path if it exists, it does not guarantee finding the optimal path. However, the calculation time and memory requirements are significantly less than for the A* algorithm.

Escape Algorithm

Hightower [1969] proposed an escape algorithm that modifies the Mikami-Tabuchi algorithm. Similar to Mikami-Tabuchi's algorithm, this method starts with two

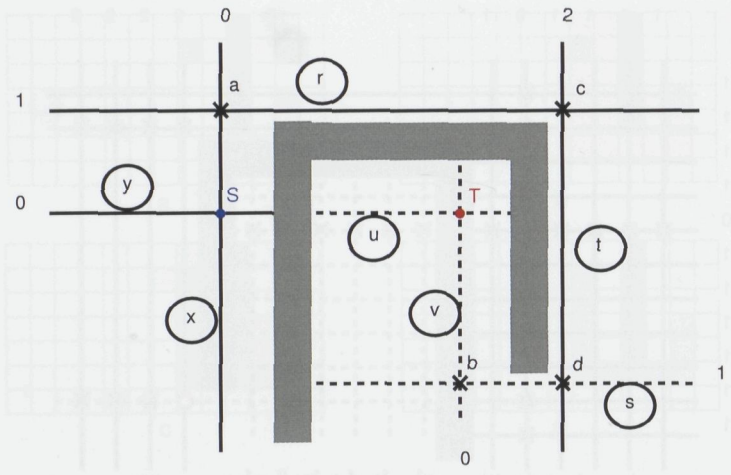


Figure 4.15: *Escape algorithm*

perpendicular lines through the starting point S. However, instead of using all line segments perpendicular to a trial line, it considers only those lines that can be extended beyond the obstacle which blocked the preceding trial line. The intersection point between horizontal and vertical lines is called the escape point, and one line segment only has one escape point.

Fig. 4.15 illustrates how the escape algorithm finds a path between the start point S and the target point T. It starts by generating two lines (horizontal and vertical) from S (line x and y) and T (line u and v), and those lines are called trial lines $level_0$. Then it starts establishing the new trial lines $level_1$. In contrast with Mikami-Tabuchi's algorithm, Hightower considers only a line that can be extended beyond the obstacle that blocks the previous trial line. In this example, the trial line $level_1$ from S is line r because it can be extended beyond the obstacle that blocks line y, and the escape point is point a. In the same manner, the $level_1$ trial line from T is line s, and the escape point is point b. Then line t is found as the trial line $level_2$ from S. This line intersects with line s, so point d is the intersection point between trial lines from S and T. From here, the algorithm performs back tracking to S and T, and the path S - a - c - d - b - T is found.

The escape algorithm is faster and uses less memory space compared to Mikami-Tabuchi's algorithm, but it cannot guarantee to find a solution.

4.3 Heuristic Algorithm

4.3.1 Genetic Algorithm

Genetic Algorithm (GA) is an adaptive method which can be used to solve search-and-optimization problems. It is based on the genetic processes of evolution of biological organisms. Over many generations, natural populations have evolved according to the principles of natural selection. By adopting this process, genetic algorithms are able to "evolve" solutions to real world problems, Goldberg [1989].

In genetic algorithm, a population of chromosomes which encode candidate solutions to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary form as strings of 0s and 1s. The parameters are converted into discrete values with a certain resolution. For example, with 10 bits per parameter, we obtain a range with 1024 discrete values. If the parameters are actually continuous then this discretization is not a particular problem. This assumes, of course, that the discretization provides enough resolution to make it possible to adjust the output with the desired level of precision.

If some parameters are discrete values then the coding issue becomes more difficult. For example, suppose there are exactly 1500 discrete values which can be assigned to variable X_i. We need at least 11 bits to cover this generating 2048 discrete values. Since there are only 1500 values, the 548 unused bit patterns may result in no evaluation. Also some parameter settings may be represented twice so that all binary strings result in a legal set of parameter values. Solving such coding problems is usually considered to be part of the design of the evaluation function.

The evolution usually starts from a population of randomly generated chromosomes and happens in generations. In each generation, the fitness of every chromosome in the population is evaluated, multiple chromosomes are stochastically selected from the current population (based on their fitness) and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

In genetic algorithm, reproduction is one of the most important phases. During this phase, first the parents must be selected and then crossover occurs. In actual biological evolution, the genes from parents form in some way the whole new chromosome. The newly created offspring can then be mutated. Mutation means, that the elements of the DNA are changed a bit. These changes are mainly caused by errors in copying genes from parents.

In genetic algorithms, there are several methods for selection of the chromosomes from a population for reproduction; such as Roulette wheel selection, Boltzmann selection, Tournament selection, and Rank selection. The most pop-

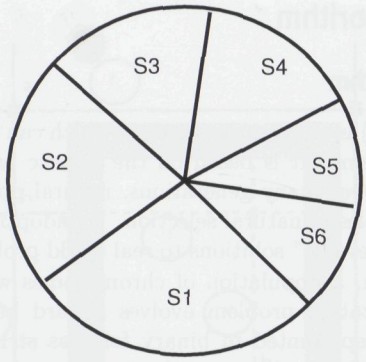


Figure 4.16: *Roulette wheel selection*

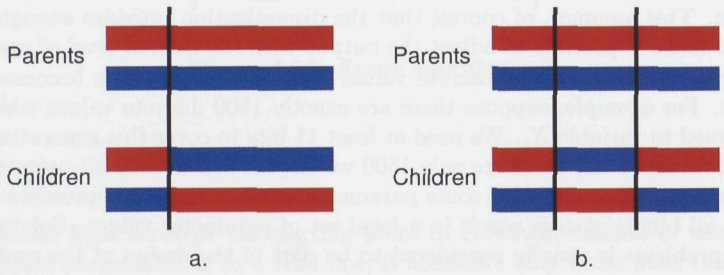


Figure 4.17: *One point and Two points crossover techniques*

ular one is the Roulette wheel selection which is based on the fitness value of all chromosomes. For example, suppose there are six chromosomes in a genetic algorithm population and that they are sorted based on their fitness value. Fig. 4.16 illustrates the Roulette wheel. It shows that the proportion of the wheel that is allocated to a particular chromosome differs according to the fitness value of that chromosome. Then the selection process is similar to a Roulette wheel in a casino. The wheel is rotated and stopped randomly, and the selected chromosome is chosen as the new parent. The probability for the chromosome with biggest fitness value to be chosen is bigger than for the others. However, there is still a probability that the relatively unfit chromosome is selected.

After two or more chromosomes are selected, the crossover process is started. Crossover is a genetic operator used to vary the chromosomes from one generation to the next. Crossover is a process of taking more than one parent chromosome and produce a child chromosome from them. Two of several techniques of crossover are shown in Fig. 4.17. Fig. 4.17.a shows one point crossover technique, and Fig. 4.17.b shows two points crossover technique.

Beside crossover, mutation also might occur. Mutation is a genetic operator used to maintain genetic diversity from one generation of a population of algorithm chromosomes to the next. Mutation occurs during evolution according to a mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search. There are some different types of mutation. The simplest one is a bit string mutation, that flips one bit at random position from 0 to 1 or vice versa.

The notion of evaluation and fitness are sometimes used interchangeably. However, it is useful to distinguish between the evaluation function and the fitness function used by a genetic algorithm. The evaluation function provides a measure of performance with respect to a particular set of parameters only. The fitness function transforms that measure of performance into an allocation of reproductive opportunities. In the canonical genetic algorithm, fitness is defined by f_i/f where f_i is the evaluation value associated with chromosome i and f is the average evaluation value of all the chromosomes in the population. However, depending on the selection method that is used, different formulas may be used.

The key factor of genetic algorithm is defining the evaluation function. Normally, developing an evaluation function can sometimes involve developing a simulation. The evaluation function value determination must also be relatively fast. This is typically true for any optimization method, but it may particularly pose an issue for genetic algorithms. Since a genetic algorithm works with a population of potential solutions, it incurs the cost of evaluating this population.

There are many ways to utilize genetic algorithm to solve the shortest path problem. One of them is by dividing the space into a grid, and we define a chromosome as a turning grid location of the path. For our case of pipe routing such location would indicate a pipe bend. Then the number of chromosomes in a population is equal to the maximum number of turning locations. Based on that, the evaluation function is defined as the shortest path between start and target points. Also to avoid a collision with any obstacle, we need to add a large penalty value in the evaluation value if there is a segment of the path that collides with an obstacle. Genetic algorithm then evolve and eventually will give a set of chromosomes as the result.

For example, the evaluation function could be the number of nodes in a path plus the number of turning points plus a very large value if the path collides with an obstacle.

Fig. 4.18 shows some snapshots of genetic algorithm evolution to find the shortest path between node S and node T. In Fig. 4.18.b there are three turning points found by the algorithm (marked as blue nodes). The path that was founded contains collisions with the obstacles, so the evaluation value is very large.

Fig. 4.18.c shows an evolution snapshot of the genetic algorithm that found six turning points. The evaluation value is 20 nodes + 6 turning points. Then after several evolutions, the optimal solution is solved by the algorithm as shown in Fig. 4.18.d (16 nodes + 3 turning points).

One of disadvantages of using genetic algorithm to solve the shortest path

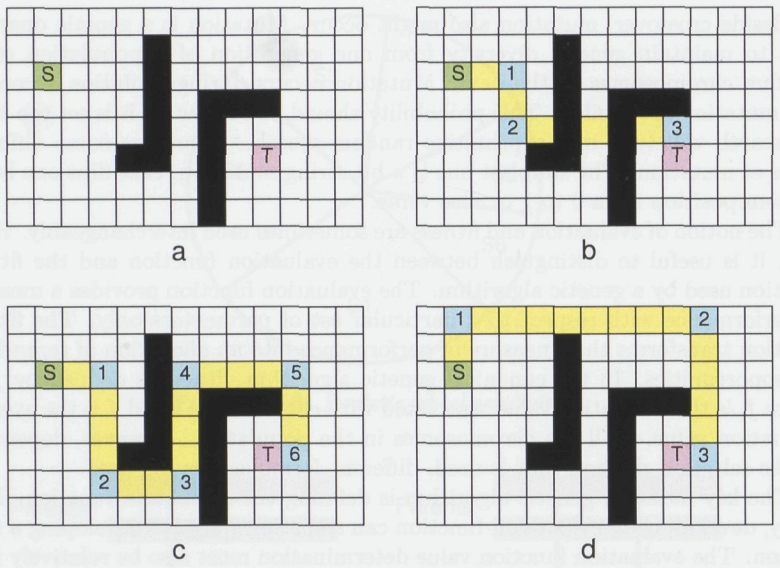


Figure 4.18: *Genetic algorithm search*

problem is that this algorithm needs many generations of a population before it finds a good solution. In every generation, the algorithm needs to re-calculate its evaluation value. In the previous example, during the calculation of the evaluation value, every segment of the path must be tested against all obstacles, and collision detection is relatively expensive. Beside that, there is no guarantee that the solution is the optimum one.

On the other hand, the method that we use in the previous example does not need to memorize the state of cells. So it requires less memory space than Lee's or the A* algorithm.

4.3.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) was first introduced and originally developed by Kennedy and Eberhart [1995]. It refers to an algorithm that is used to find optimal solutions of numerical and qualitative problems. It emerged from earlier experiments with algorithms that modeled 'flocking behavior' seen in many species of birds. PSO has been used in the last few years to solve different kinds of optimization problems, such as training a neural network model, Salerno [1997], tracking and optimizing dynamic systems, Eberhart and Shi [2001], power system control, Yoshida et al. [2001], scheduling of manufacturing systems, Jerald et al. [2004], and tuning PID parameters in an AVR system, Gaing [2004].

Standard PSO

In the original framework, each individual in PSO, called a *particle*, moves in the search space with a velocity that is dynamically adjusted according to its own experience and its companions experience. As each particle moves through the problem space, it evaluates the fitness function, and memorizes its best position (the position giving its own best fitness value *pbest*) as a *pbestx*. Each particle also memorizes the best global solution (*gbest*), which is the best fitness value among all particle in the population, and its position (*gbestx*), obtained so far by any particle in the population based on the fitness function.

While moving toward its *pbest* and *gbest* location, each particle, at each time step, changes its velocity and its current position according to:

$$v_i = \alpha v_i + c_1\sigma_1(p_i - x_i) + c_2\sigma_2(p_g - x_i) \tag{4.1}$$

$$x_i = x_i + v_i \tag{4.2}$$

where

v_i	velocity of the i^{th} particle;
α	inertia weight;
c_1, c_2	acceleration constant;
σ_1, σ_2	random numbers in the range $[0...1]$;
x_i	position of the i^{th} particle;
p_i	*pbest* of the i^{th} particle;
p_g	*gbest* of population, index g represents the index of the best particle among all particles in one population.

The first part of (4.1) represents the *inertia weight* which was first introduced by Shi and Eberhart [1998]. This term serves as a memory of the previous velocity. A large inertia weight favors exploration and a small inertia weight favors exploitation. The second part is the *cognition* component, representing the exploiting of its own experience. The third part is the *social* component, representing the shared information and mutual cooperation among particles.

There are two types of boundaries; velocity maximum *Vmax* and position boundaries x_{min} and x_{max}. If the velocity according to (4.1) exceeds *Vmax*, then it is limited to *Vmax*. In same manner, if position x_i in (4.2) exceeds the position boundaries, then the particle is placed on the boundary. *Vmax* needs to be chosen wisely, because it influences the convergence of the search. If *Vmax* is too high, particles might move too fast, passing a good solution. If *Vmax* is too small, the particles may not sufficiently explore the search space Fan and Shi [2001]. Early experience with particle swarm optimization indicated to set the acceleration constant c_1 and c_2 equal to 2.0, and *Vmax* equal to 20% of the total range of the variable. The standard PSO algorithm is shown in Fig. 4.19, Kennedy and Eberhart [1995]-Fan and Shi [2001].

```
Initialize PSO
for j = 0 to number of iteration do
      for i = 0 to number of particle do
            f_i^* = f(x_i)
            if f_i^* < f(p_i) then
                  f(p_i) = f(x_i) and p_i = x_i
            if f(p_i) < f(p_g) then
                  f(p_g) = f(x_i) and p_g = x_i
            v_i = αv_i + c_1σ_1(p_i − x_i) + c_2σ_2(p_g − x_i)
            x_i = x_i + v_i
      end
end
f(p_g) and p_g are the outputs
```

Figure 4.19: *The Standard PSO Algorithm*

Improved Version of the PSO Algorithm

Results show that PSO finds good solutions much faster than other algorithms, Angeline [1998], Peram et al. [2003], Tao et al. [2004], Krink et al. [2002] and Tao et al. [2003], however for some type of problem, Angeline [1998] shows that after a few number of iterations the quality of solutions can not be improved. Especially in strongly multi-modal problems, PSO may suffer from premature convergence Tao et al. [2003].

To overcome that problem, much work to improve PSO has been done. The main work was on parameter modification, increasing diversity, and variations of the PSO algorithm.

PSO needs predefined parameters, i.e. swarm size, maximum velocity, weight inertia, individual and social factor. The ability to find a globally optimum solution relies greatly on the setting of these parameters. Tao et al. [2004] proposed an adaptive PSO by choosing parameter to increase stability and avoid premature convergence. Other works on the PSO parameters are described in Shi and Eberhart [2000] and El-Gallad et al. [2002].

The diversity of the swarm needs to be high, while low diversity would lead to fitness stagnation of the swarm. If this happens close to a the local optimum region, the swarm can be trapped in a local solution. Løvbjerg and Krink [2002] introduced self-organized criticality (SOC) to maintain diversity of the swarm, by resetting particles that are too close to each other. Another approach to reset the swarm to increase diversity was proposed by Clerc [1999] by defining a no-hope convergence criterion and a re-hope method so that, from time to time, the swarm re-initializes its position, according to some gradient estimations of the objective function and to the previous re-initialization.

Modification of the PSO algorithm itself has been done to achieve better and faster convergence. Most of it hinges on mixing the PSO algorithm with other

optimization methods. The combination of PSO with hill climbing is used by Lim et al. [2003] to solve the bandwidth minimization problem. Another hybrid approach introduced by Krink and Løvbjerg [2002] apply GA, PSO and hill climbing simultaneously.

Variations of the Standard PSO

Another approach to modify the PSO algorithm is by using different types of probability distribution to generate the random number. Krohling et al. [2004] use a truncated Gaussian probability distribution. Truncation is done for the values of the random number less than (-1) and greater than 1. To assure that the PSO method converges, the random numbers σ_1, σ_2 have to be positive. Because of that requirement, the generated random numbers are mapped to generate random numbers in the interval [0,1].

To ensure the convergence of the PSO, Clerc and Kennedy [2002] introduced a *constriction factor* K that replace the stochastic terms $c_1\sigma_1$ and $c_2\sigma_2$ into 0.729. Another approach called Gaussian PSO was proposed by Krohling [2004]. It uses the absolute value of the Gaussian probability distribution $N(0,1)$ for automatic generation of the stochastic terms of PSO which has a mean value equal 0.798. So, there is no more need to specify the accelerating constant c_1 and c_2. Furthermore, the momentum term is not used by setting the inertia weight α equal zero and therefore the maximum velocity $Vmax$ is no longer necessary. The Gaussian PSO only has the swarm size and the boundary of the search space as its parameters.

However, Kennedy [2005] stated that the inertia term adds several unique characteristics to the particle swarm, and it should be maintained. Based on this, the Gaussian PSO with inertia term is used as one variant in this thesis. The velocity term in the Gaussian PSO was then modified to:

$$v_i = \alpha v_i + |Rand_1|(p_i - x_i) + |rand_2|(p_g - x_i) \tag{4.3}$$

where $Rand_1$ and $Rand_2$ are the random numbers generated by taking the absolute value of the Gaussian probability distribution $N(0,1)$.

To solve the shortest path problem using this algorithm, we can use exactly the same method that was used for the genetic algorithm.

In some literature, it is mentioned that PSO normally converges faster than the genetic algorithm. However, it still needs quite a large number of generation before it can find the solution. Since the calculation of the evaluation value is expensive, it still takes a lot of computation time. Also, as in genetic algorithm, the solution from PSO is not guaranteed to be the optimal solution.

4.3.3 Ant Colony Optimization

The ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. The fundamental idea is inspired by the foraging behavior of real life ant

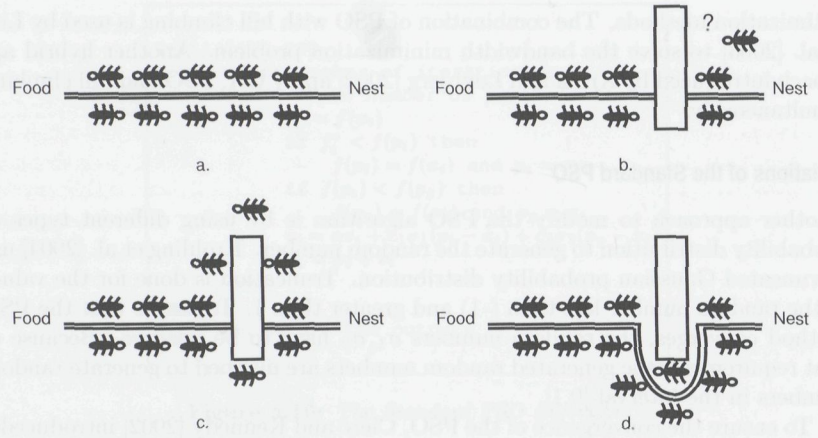


Figure 4.20: *Ant colony optimization*

colonies in which individual ants deposit a substance called pheromone on the path while moving from the nest to the food sources and vice versa. Thereby, a pheromone trail is formed through which individual ants are aided to smell and select their routes. The paths with higher pheromone doses would be more likely to be selected by other ants bringing on further amplification of current pheromone deposits, which leads to a positive feedback process. Due to this interesting behavior of ants, after some time the shortest path from the nest to the food source or vice versa would be formed.

Fig. 4.20 illustrates the ant colony optimization. Fig. 4.20.a shows an established path between the nest and the food. Then if we put an obstacle as shown in Fig. 4.20.b, the path becomes obsolete. Fig. 4.20.c shows how the colony of ants finds two paths to go around the obstacle. Since the path below the obstacle is shorter than the upper path then the times that is needed to travel through the lower path is smaller. Then after a while, the number of ants that using the lower path is larger, therefore the pheromone trail is stronger in the lower side. Then the new shortest path is found.

The ant colony needs to register the amount of pheromone on each visited cell. In the worst case, all cells are visited by the ants. For a graph, the space complexity is $O(E)$ with E is the number of edges in graph. In a grid space, it becomes $O(C)$ with C the amount of cells in the grid, the same as Lee's or the A* algorithm.

On every repetition, a colony of ants explores a graph or a grid until the target cell is reached. Comparing with Lee's or A*, the time that is needed by the ant colony on one repetition is shorter. However since it needs many repetitions, it takes much longer time before a solution is found.

Table 4.1: *Comparison of algorithm*

	Maze		Line		Heuristic	
	Lee	A*	Mikami	Hightower	GA	PSO
Find path?	5	5	5	3	5*	5*
Optimum?	5	4	3	2	4*	4*
Time Avg	$67ms$	$30ms$	$13ms$	$11ms$	$2min$	$1.5min$
Space	$O(C)$	$<< O(C)$	$O(L_M)$	$<< O(L_M)$	$O(P_{GA})$	$O(P_{PSO})$
Weighted?	Yes	Yes	No	No	Yes	Yes

The implementation of ant colony optimization to solve the shortest path problem is normally applied on a graph or a map with predefined boundaries to speed up the searching process and also to minimize the memory usage.

In the same way as for other heuristic methods, the solution from ant colony optimization might not be the optimal one.

4.4 Comparison of Algorithms

In the beginning of this section, we have defined our criteria to measure the performance of the algorithms; always find a solution if it exists, always find the optimal solution, and use small resources in terms of computer memory and shorter computation time. We also consider the flexibility of those algorithms to be extended and adapted to our methodology.

In order to compare the performance, we create a simple 3D uniform weight grid type environment and use it as the test case to measure the performance of the shortest path algorithm mentioned above.

Since this test is only a rough comparison, we use the standard variety of each algorithm, for example the genetic algorithm variance that we test is its standard form.

For the genetic algorithm, we choose to use 50 different populations, and the maximum number of regenerations is 100 times. Each population consists of 6 chromosomes, because the maximum number of bends that we expect is 6. The choice to use the roulette wheel as the selection method and use the two point crossover method.

We also use 50 different populations and a population size of 6 members for the particle swarm optimization. The maximum number of iterations is also 100.

For both PSO and GA, the iteration process stops if at least one valid solution has been found and the result is no longer improved after 5 further iterations.

Table 4.1 summarizes the performance of the algorithms. For the first two criteria, the range is between 1 and 5, with 5 meaning always and 1 meaning never. For heuristic algorithms, an asterix means that this grade can be achieved by running the algorithm with the number of generations less than 100.

Mikami-Tabuchi and Hightower algorithms cannot be implemented in a weighted environment and do not have a good performance to find the optimum solution, which means that our methodology cannot use it as the detail routing algorithm. However, due to its speed and low memory usage, they can potentially be used to test whether a certain pipe can be routed or not. The table shows that even though the Hightower algorithm has a better performance in term of memory usage, it does not always find a solution, so it won't be considered anymore, and we choose to utilize Mikami-Tabuchi in our methodology for that function.

The other four algorithms are suitable to be used in a weighted environment, and they also always find a solution if it exists. First, let's take a look at the heuristics algorithm; genetic algorithm and particle swarm optimization. Comparing the performance between heuristic algorithms is not trivial. There is a famous theory introduced by Wolpert and Macready [1997] in connection with the problems of search and optimization. They say there is "no free lunch" (NFL) in search and optimization. In short, NFL says that on average, all optimization algorithms have the same performance, but for a certain kind of problem and condition, one algorithm might be better than the other.

In the shortest path problem, both GA and PSO use the same implementation. In our example case, PSO always finds a solution in fewer iteration compared to GA. If we use the optimized variety of GA or PSO, there is a possibility that the solution can be found faster, however it is still incomparable to the speed of the deterministic algorithm to find the solution.

In terms of memory usage, for precise routing with a small cell size and a small number of pipes, GA and PSO implementation is better than the deterministic algorithm.

As we mentioned in the previous subsection, it takes many generations for heuristic algorithms to find a good solution. So in terms of time complexity, they are much slower than the exact algorithm; Lee's and the A* algorithm.

For completeness sake, we also tried to use GA and PSO to route a group of pipes. As a result, we found out that both methods are able to route a small group of pipes, but the time that was needed is much longer than for the deterministic algorithm. Also, both failed to route a group of pipes if the group contained more than 5 pipes.

If we only consider the certainty to find the optimum solution, Lee's algorithm is the winner. However the speed and significantly lower memory usage of A* algorithm leads us to select this as our main algorithm.

4.5 Beyond the Shortest Path Problem

4.5.1 Multiple Nodes

Most of the shortest path algorithms are basically only concerned with finding the shortest path that connects two nodes, while in practice, more than 70% of

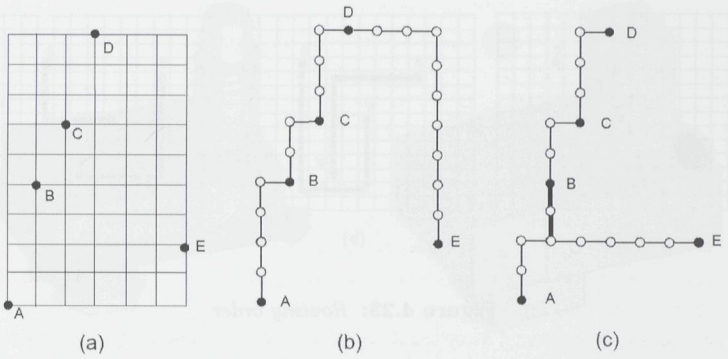


Figure 4.21: *Multiple nodes*

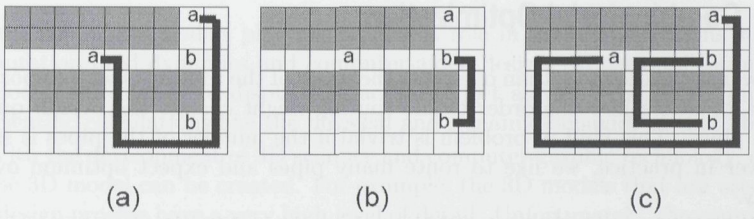


Figure 4.22: *Mutually intervening case*

pipes have a branch. There is a specific algorithm to solve this optimally, called the Steiner Tree Problem, Bern and Graham [1989]. However, it is not as flexible as the A* algorithm that we adopted in our methodology.

We use Fig. 4.21 to illustrate the multiple nodes routing. Let us say that we have five nodes to be connected as shown in Fig. 4.21.a. Fig. 4.21.b is the result by performing the A* algorithm four times to connect A to B, B to C, C to D, and D to E. Fig. 4.21.c shows the result by using Steiner Tree.

In chapter 5 we will discuss a modified A* to solve the branching problem.

4.5.2 Mutually Intervening Case

In practice, it might happen that two or more pipes block each other when they are optimally routed. As an example, Fig. 4.22 shows that if pipe a is optimally routed, pipe b is blocked. Also in the other way around, if pipe b is optimally routed, it blocks pipe a.

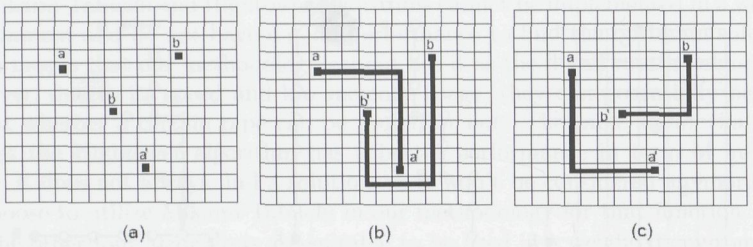


Figure 4.23: *Routing order*

In chapter 5, we will look in more detail into this problem and how our methodology solves it.

4.6 Combinatorial Optimization

If we want to route more than one pipe, the order of the routing is very important. Fig. 4.23 shows how the order of the routing might change the overall quality of the pipes. This kind of problem is trivial if the number of the pipes is small. However in practice, we like to route many pipes and expect optimum overall quality.

To solve this combinatorial problem, we need to utilize the heuristic optimization method. One of the most famous methods is the variation of the Particle Swarm Optimization algorithm, called Discrete PSO (DPSO), Clerc [2004].

The Discrete PSO can be formalized as follows:

$$v_i = \alpha v_i \oplus c_i(p_i - x_i) \oplus c_2(p_g - x_i) \tag{4.4}$$

$$x_i = x_i + v_i \tag{4.5}$$

As can be seen by comparing eq.(4.1) with eq.(4.4) and eq.(4.2) with eq.(4.5), there is no formal difference between classical PSO and DPSO. However slightly different rules are imposed. The search space S is the finite set of all sequences of the pipes to be routed. The position p_i is one of the possible sequences. The interesting part is the velocity, since the meaning of movement of the particle is not the same as in the classical PSO. In DPSO, the velocity is in a form of a list of transpositions. For example, $v = (2, 5)$ means that if this velocity is applied to a position $p_i = (0, 1, 2, 3, 4, 5)$, it generates a new position $p_i = (0, 1, 5, 3, 4, 2)$.

In his work, Clerc [2004] develops DPSO and proves the performance of this algorithm to solve the asymmetric Traveling Salesman Problem, since TSP is well known as one of the most important test grounds for the combinatorial problem.

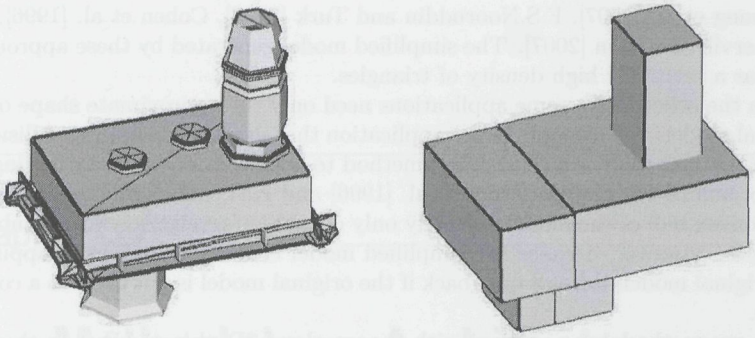


Figure 4.24: *Example of model simplification*

4.7 Model Simplification

Three dimensional models play an important role in many applications such as computation fluid dynamics and computer-aided design. Those applications use 3D models not merely to display the actual object as well as possible, but also for scientific calculations like the physical and dynamic behavior of that object.

With the development of 3D scanner and computer graphic technology, a very precise 3D model can be created. For example, the 3D models that are used in a ship design process have a very high level of detail. Unfortunately, the complexity of these models, measured by the number of triangles, seems to grow faster than the ability of the hardware to calculate and render it interactively. Put in another way, the level of detail of those models always seems to exceed the level that we can afford.

To overcome this discrepancy the algorithm to simplify the 3D models has been investigated for years. Most of them are dealing with reducing the number of polygons, Garland and Heckbert [1997], Gotsman et al. [2002], Hua-hong et al. [2007], F.S.Nooruddin and Turk [2003], Cohen et al. [1996], Hjelmervik and Leon [2007], Barber et al. [1996], and Kirkpatrick and Seidel [1986]. Another way of 3D model simplification is by translating the polygon soup (a group of unorganized triangles, with generally no relationship whatsoever) to volume visualization (*voxel*) Kaufman [1994].

Generally there are two main objectives of model simplification, for faster rendering and/or faster computation time of the physical aspect of the model. In practice, one application might require a different level of detail compared with other applications.

In some applications, the high level of detail is needed to maintain the similarity of the shape of the original model and the simplified model. For this purpose, we can use approaches of Garland and Heckbert [1997], Gotsman et al. [2002],

Hua-hong et al. [2007], F.S.Nooruddin and Turk [2003], Cohen et al. [1996], and Hjelmervik and Leon [2007]. The simplified model generated by these approaches still has a relatively high density of triangles.

On the other hand, some applications need only the approximate shape of the original models, for example in the application that needs to detect the collision of two or more objects. An established method to achieve this goal is by finding the convex hull of the model Barber et al. [1996] and Kirkpatrick and Seidel [1986]. The convex hull of the model normally only consists of a relatively low number of triangles. However, because the simplified model is always a convex wrapping of the original model, it has a drawback if the original model is not close to a convex shape.

In our methodology, we deal with thousands of 3D objects. Due to the very large and complex calculation, and the fact that most pipes are routed in an orthogonal way, we need a simplified model that only contains a small number of cuboids for faster calculation. For this purpose, those approaches that are mentioned above are not suitable to be used.

In chapter 5, we will discuss our method to simplify the 3D model as shown in Fig. 4.24.

4.8 Summary

In this chapter, we have reviewed and compared the existing shortest path algorithms to identify the algorithm that should be adopted in our methodology. Based on our experiment described in Section 4.4, we have decided to use the A* algorithm to find the shortest path in the single pipe router module of our methodology. We also will use the Mikami-Tabuchi algorithm for the blockage checker module, due to the fact that this algorithm can find a solution faster and use less memory than the A* algorithm.

During our experiment, we found out that the heuristic algorithm is not suitable to route many pipes at the same time. However, it works nicely to route a group of 2-3 pipes. This leads us to further investigate it to solve the mutually intervening problem that will be discussed in detail in Chapter 5.

As described in the previous chapter, since our methodology routes pipes one by one, the problem of optimizing the quality of a group of pipes becomes a combinatorial problem. In this chapter, we also chose the heuristic algorithm that will be used to solve the combinatorial problem.

Beside that we also described some other difficulties beyond the shortest path problem, and our solution to these issues will be presented in Chapter 5.

Chapter 5

The Methodology Architecture and its Implementation

5.1 Introduction

Chapter 3 discussed the core functionality that must be implemented in our pipe routing methodology. These requirements are based on the current pipe routing process in practice that has been discussed in Chapter 2.

Since a few decades ago, the pipe routing problem has been an interesting research subject, and some selected researches that have most relevance for our methodology have been reviewed.

In Chapter 4, we have reviewed and compared some optimization algorithms in more detail. The comparison results led us to select the suitable optimization algorithms to be used in the methodology. Some other important matters such as model simplification and combinatorial optimization were also explained or touched upon.

In Fig. 3.1, the outline of the proposed methodology architecture is shown from the functionality point of view. In our methodology, the function for data

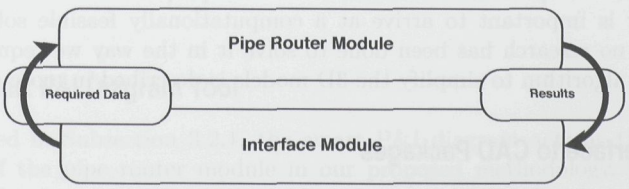


Figure 5.1: *The Outline of Proposed Methodology Architecture*

retrieval and to send result back to CAD software are combined into one module called the interface module. Therefore, Fig. 3.1 can be represented as Fig. 5.1 that shows that the architecture of our proposed methodology consists of two main parts; the interface module and the pipe router module. In this chapter, we will look into both modules in more detail.

While the interface module is scientifically less interesting, it is an integral part of our methodology since this research investigates and validates a methodology that in principle should lend itself for practical application. For that reason we must also dwell on it at some length.

5.2 Interface Module

As described in Chapter 3, the proposed methodology is used together with a 3D CAD software package. First it retrieves the required data from the CAD software packages, and then sends back the result to them. All functions for data retrieval and to send back the results are implemented in the interface module of the methodology.

The data that are required to perform the pipe routing is already discussed in Section 3.2 and the conclusions are summarized in Section 3.3. It is also mentioned that some data are not available in the CAD software, therefore the proposed methodology must consist of some other tools to generate them.

The first tool that is needed is the constraint editor tool. This tool is a simple tool for a user to manually add an object constraint, such as a NoGo area, or a virtual obstacle. The details of an object constraint will be discussed later in this chapter. However, because it is basically a computer programming problem, it will not be discussed in detail in this thesis. What is important however, is that the methodology can handle such arbitrary constraints or NoGo area.

As mentioned above, some of data are available in CAD software. However, most of the time the format of those data cannot immediately be used in our methodology. Therefore it is required to include an interface between our proposed methodology and the CAD software.

Following our discussion in Section 3.3, the smart P&I diagram tool must also be included in our methodology.

In Section 4.7 we have discussed the importance of model simplification before we use them for optimization calculation in the proposed methodology. Since this matter is important to arrive at a computationally feasible solution, and apparently no research has been done to solve it in the way we require, in this section the algorithm to simplify the 3D models is described in more detail.

5.2.1 Interface to CAD Packages

Since most of the data that are required by our methodology are already available in the CAD software packages, we would like to make use of it. In today situation,

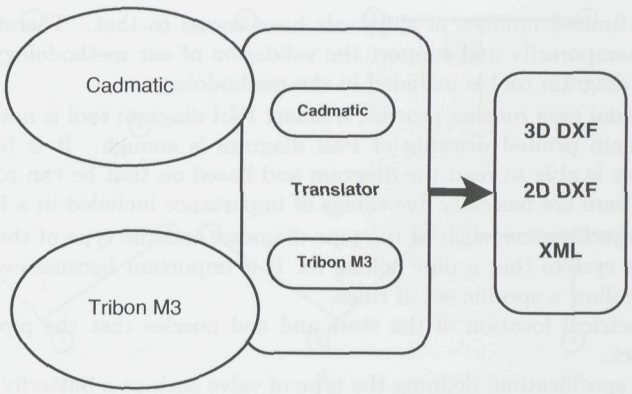


Figure 5.2: *The Interface to CAD software packages*

most of the CAD software packages have an export function to dump the data from their system to be used by external applications. However, this is not always a trivial case, because not every data can be dumped using the standard export functions. Moreover, every CAD software package uses its own kind of standard, and stores its data in a different way.

Let's take a look at our case which is important for validating the methodology and its implementation. At this moment, our laboratory implementation with both Cadmatic and Tribon M3 software packages. Even though both can be used to design a ship, they are fundamentally different in terms of internal data. Also, their export functions are totally different, and will produce an output in different formats.

To tackle this problem, it is decided that in our methodology we will use a common type of data format, the DXF format. As widely known, DXF has become the de facto ASCII standard file format for CAD drawing exchange.

The data are divided into three different categories; the 3D volume data are converted to 3D DXF, the 2D drawing data are converted to 2D DXF, and data that contains the information are restructured into an XML file. As shown in Fig. 5.2, the required data from the CAD software packages will be converted to the common format. Further details are not necessary to appreciate the validation in Chapter 6.

5.2.2 Smart P&I Diagram Tool

As described in Subsection 3.2.1, the smart P&I diagram acts as the primary guidance of the pipe router module in our proposed methodology. Currently, even though some commercial 3D CAD software vendors already implement a concept of smart P&I diagram in their software packages, it is still relatively new

and only a limited number of shipyards have access to that. Therefore, to fill in the gap temporarily and support the validation of our methodology, a simple Smart P&I diagram tool is included in the methodology.

In a manual pipe routing process, a Smart P&I diagram tool is not necessary. A simply plain printed drawing of P&I diagram is enough. It is because the pipe engineer is able to read the diagram and based on that he can route a pipe correctly. There are basically five things of importance included in a P&ID:

1. Pipe specification, such as the pipe diameter and the type of the pipe.
2. Piping system that a pipe belong to. It is important because every system must follow a specific set of rules.
3. Geometrical location of the start and end nozzles that the pipe needs to connect.
4. Valve specification; defining the type of valve such as a butterfly valve, and defining the valve location type (see below).
5. Branch location type (see below).

Without implementing a Smart P&I diagram tool, the task to define this input to the router module will take a lot of time. This would unduly hinder our validation work and also raise question with regards to the practical applicability of our methodology. The first three items listed above can be seen immediately in a plain P&I diagram, but the fourth and the fifth can only be seen by the experienced pipe engineer.

Valves

Two main things are described in a Smart P&I diagram regarding a valve. The first one is the technical type of valve. It is important since during the pipe routing process the valves will be placed automatically by the pipe router module. This feature also prevents a pipe engineer to make a mistake like shown in Fig. 2.6

The second part is the valve location type. In practice, a pipe engineer places a valve in two different ways. The first one applies to a certain kind of valves: a pipe engineer must define the location of that valve in space and after that routes the pipe through that location. This must be done if the valve belongs to the group of control valves. We call this valve a fixed valve. For other types of valve, normally the pipe will be routed first, then after that the valve that belong to that pipe will be placed in that path. This kind of valve is called a floating valve.

Branches

Even though pipe branch cases are neglected in many studies about automatic pipe routing, in practice most of the pipes in a ship have a branch. For that reason, branches must be included in any relevant methodology. As with the valve location type, the branch types are divided into two types; a fixed branch and a floating branch.

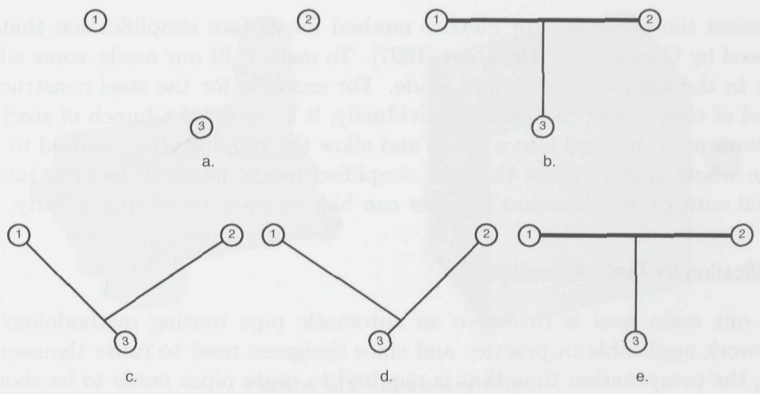


Figure 5.3: *Branch type*

Fig. 5.3 shows the difference between a fixed branch and a floating branch. Assume there are three nozzles that must be connected as shown in Fig. 5.3.a. Fig. 5.3.b shows the example of a fixed branch. In a fixed branch there is a master pipe and a child pipe. In this case, the pipe between nozzle 1 and nozzle 2 is routed, as a master pipe, then nozzle 3 is routed to the master pipe, as a child pipe.

In a floating branch, all nozzles have the same priority. Thus, the master pipe can be the pipe that connect nozzle 1 and nozzle 3 as shown in Fig. 5.3.c or the master pipe is a pipe between nozzle 2 and nozzle 3 as shown in Fig. 5.3.d or between nozzle 1 and nozzle 2 as shown in Fig. 5.3.e.

5.2.3 Simplification of the 3D Model

The 3D models that are imported from the CAD software package have a high level of detail. Not only an entity that has a large volume is complex, even a small bolt is quite complex. As mentioned in Section 4.7, generally there are two main objectives of model simplification, faster rendering and/or faster computation time of the physical aspect of the model.

Simplification for Faster Rendering

In any viable methodology, this kind of simplification is needed to improve the performance of the graphic user interface (GUI) part. As described in Chapter 3, not all data can be retrieved automatically from the CAD software, some data such as a NoGo area is still needed to be entered manually. To make this task easier, the GUI part of the methodology must have a good performance.

In Section 4.7, it was mentioned that many researches have been done to simplify the 3D model, and some of them are useful for our purpose. We decided to

implement the quadric error metrics method for surface simplification that was proposed by Garland and Heckbert [1997]. To make it fit our needs, some adjustments in the implementation are made. For example for the steel construction, instead of simplifying the model individually, it is better if a bunch of steel constructions are combined into a group and allow the simplification method to work on the whole group. After that the simplified model needs to be split into the original entities again so that the user can hide or show every single entity.

Simplification for Faster Calculation

Since our main goal is to derive an automatic pipe routing methodology and framework applicable in practice and since designers need to route thousands of pipes, the computation time that is required to route pipes needs to be short.

As we already discussed in Chapter 4, one of the most time consuming parts in the pipe routing process is the collision avoidance. To avoid the collision between the pipe that is being routed with other objects, such as the pieces of equipment and the pipes that were already routed, the collision detection routine must be performed in each step.

To test the collision detection between two real tessellated objects is computationally very expensive since basically we need to test the collision detection for each triangle in the model. To tackle this problem many researches have been done to speed up the collision detection routine. The common way is by simplifying the 3D model into a collection of shapes that is more simple to be tested, such as a bounding box.

In the next subsection, our proposed method to simplify the 3D model is described in detail.

Approximate Orthogonal Simplification of a 3D Model

The simplest way to simplify the 3D model is by constructing its axis-aligned boundary box ($AABB$). However if the model is complex, the $AABB$ will not represent the model correctly. As shown in Fig 5.4, the original model in Fig 5.4a cannot be represented adequately with its $AABB$ in Fig 5.4b.

A better 3D model representation can be achieved by voxelizing the model, Kaufman [1994]. A voxel representation of a model is a regular grid of cells, in which each cell (voxel) contains a density value in the range of zero to one. Normally a voxel-value of zero is representing a portion of unoccupied space and a value of one is representing a voxel that is inside the model. The result of the voxelization method can represent the original model much better than a boundary box. However to get the best results, we need to have a large number of voxels.

Another way to get a better 3D model representation is by representing the model as a collection of cuboids. This approach was used by Zuurmond [2004] to have a simple but quite representative model compared with the boundary

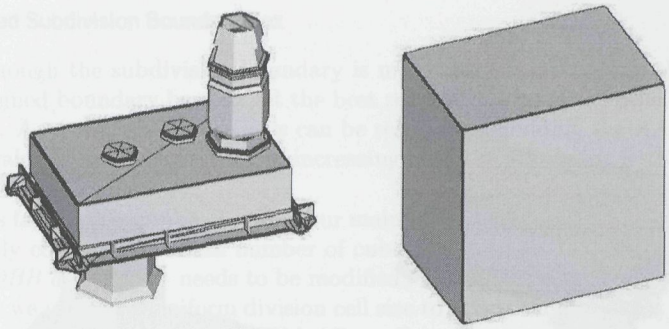


Figure 5.4: *Boundary Box*

box approach. Basically, this approach is done by constructing the subdivision boundary box (*SDBB*) from the model rather than the *AABB*. However, the creation of cuboid models are done manually.

The method to construct the *SDBB* is almost similar with the voxelization method. First, we divide the *AABB* boundary box into a grid of cells, but instead of finding the density of each cell, we construct the simplified model in a different way. The steps of this method are as follows:

Algorithm 1: Subdivision boundary box

1. *Construct the AABB of the model.*
2. *Divide the AABB into n parts in each X, Y, and Z axis, resulting in a uniform grid of cells (n x n x n).*
3. *For each cell, find the polygons of the model that lie inside or intersect with that cell.*
4. *Construct the AABB of the polygons in step 3).*
5. *Clip the polygon AABB with the cell itself.*
6. *Loop to step 3) until all cells in the grid are processed.*

This process is illustrated in Fig. 5.5. From the original model, shown in the upper left of Fig. 5.5, we construct the *AABB*. Then we apply the *step 2* in the *algorithm SDBB* with n equal *3*. It divides the *AABB* to 3 parts in each axis uniformly (cell sizes $X_1 = X_2 = X_3; Y_1 = Y_2 = Y_3; Z_1 = Z_2 = Z_3$) and creates 27 uniform cells that can be seen in the lower left part of Fig. 5.5. Then *steps 3 - 5* are applied to each cell. The final result is the simplified model that is shown in the right most part of Fig. 5.5.

As shown in Fig. 5.5, the *SDBB* is able to represent the original model much better than the *AABB*, while maintaining an acceptable number of cuboids.

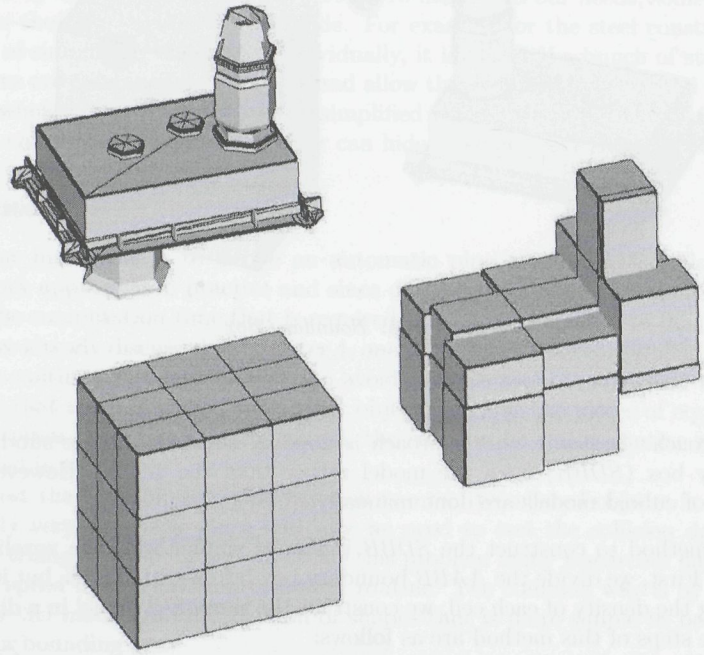


Figure 5.5: *Subdivision Boundary Box*

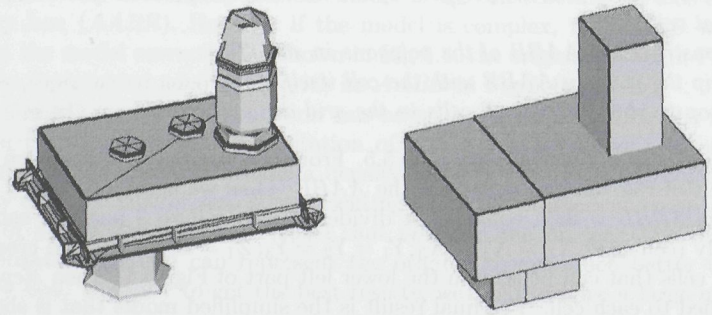


Figure 5.6: *Optimized Subdivision Boundary Box*

Optimized Subdivision Boundary Box

Even though the subdivision boundary is more representative than the ordinary axis-aligned boundary box, to get the best result, the size of the cells needs to be smaller. A smaller size of the cells can be reached by dividing the $AABB$ using a larger value of n, at the expense of increasing numbers of cuboids in the simplified model.

This fact is unfavorable, because our main objective is to get a simplified model that only consists of a small number of cuboids. To overcome this problem, *step 2* in *SDBB algorithms* needs to be modified. Instead of using a uniform division cell size we use a non-uniform division cell size to divide the $AABB$. It means that the cell sizes $(X_1 \dots X_n), (Y_1 \dots Y_n), (Z_1 \dots Z_n)$ are not necessarily equal.

Fig. 5.6 shows the comparison between the original model with the original *SDBB* using the uniform division cell size, and the *Optimized SDBB* using the non-uniform division cell size. As can be seen with proper division cell size, the shape of the simplified model is closer to the original model and also contains a smaller number of cuboids. This method will be called as the optimized subdivision boundary box method.

Moreover, because this method uses the boundary box for each cell that intersects with the original model, it is guaranteed that the simplified model always covers each whole triangle in the original model.

However, the output of this method is very sensitive to the variation of the division cell size and to get the best result it needs to have the correct size of each division. To find the best value manually is not trivial. Fortunately, the problem to find the best division size can be solved using the heuristic optimization method.

Optimizing using Tribes-D

Particle Swarm Optimization (PSO) was introduced in Subsection 4.3.2 as one of the heuristic optimization methods that can be used to solve optimization problems in various fields. Also it was shown in Subsection 4.4 that a generic PSO can be used to solve the shortest path problem.

The classical PSO is very dependent on parameter values. The values such as the weight of velocity inertia and acceleration constants need to be tuned, and it requires much time to find the optimal value. Tribes-D, on the other hand, is a parameter-less particle swarm optimization algorithm. Basically we only need to define the range of particles, and the maximum number of evaluations.

In principle, Tribes-D divides swarms in tribes. In the beginning, the swarm consists of one tribe with one particle inside. At each time step, the best particle of each tribe, called shaman, acts as an informer of other particles in that tribe. For the shaman itself, the informer is selected at random among other shamans in all tribes or in the archive.

From time to time, the quality of each tribe is checked. If the quality is bad, a new particle is generated, and if it is good and has enough particles the worst

particle is removed. The quality of the swarm is also measured, and if it is bad a new tribe is added, and if it is good and has enough tribes the worst tribe is removed.

The *Tribes-D* algorithm that is used was adapted from the C source code of *Tribes-D* by Clerc [2008]. The difference concerns the random generator. Original *Tribes-D* uses the *KISS* random number generator, while we use the *Mersenne Twister* random generator.

Implementation of Approximates Orthogonal Simplification

We implement the optimized subdivision boundary box method and utilize *Tribes-D* to find the best division cell size. The main goal is to minimize the volume of the simplified model while maintaining a low number of cuboids. Put in another way, we would like to maximize the difference between the volume of the *AABB* of the original model and the sum over all volumes of each cuboid in the simplified model. The fitness value is defined as follows:

$$f_1 = \left\{ VolBB - \sum_{i=1}^{Cub} (VolCuboid_i) \right\} \tag{5.1}$$

where $Cub \in 1 \ldots n^3$, $VolCuboid_i$ is the volume of each cuboid i, and $VolBB$ represents the volume of $AABB$.

The Tribes-D algorithms is used to optimize the division cell size. In this case, it optimizes $(X_1 \ldots X_n), (Y_1 \ldots Y_n), (Z_1 \ldots Z_n)$, with n the number of division, to minimize the fitness functions f_1. In our implementation, we choose to define n equal to 3 or 5 depending on the actual size of the model and maximum fitness evaluation is 10000 times.

The performance of the approximate orthogonal simplification method is measured by the ability to find the solution that visually represents the original model.

As a comparison, we compare the result of this method with the simplified model using a simple voxelization method with two type of voxel-values, zero and one.

Comparison with Voxelization Method

We have tried the approximate orthogonal simplification method to find the simplified model of a pump model. To improve that results, we would like to capture the quality variance in relation to the desired number of cuboids. It means that the optimization problem becomes a multi-objective optimization, since it has to optimize the subdivision interval and also the total of cuboids. Therefore, the Tribes-D was modified so it can be used to find the solution of a multi-objective optimization problem. For this purposes, we introduce another objective function:

$$f_2 = \{Cub\} \tag{5.2}$$

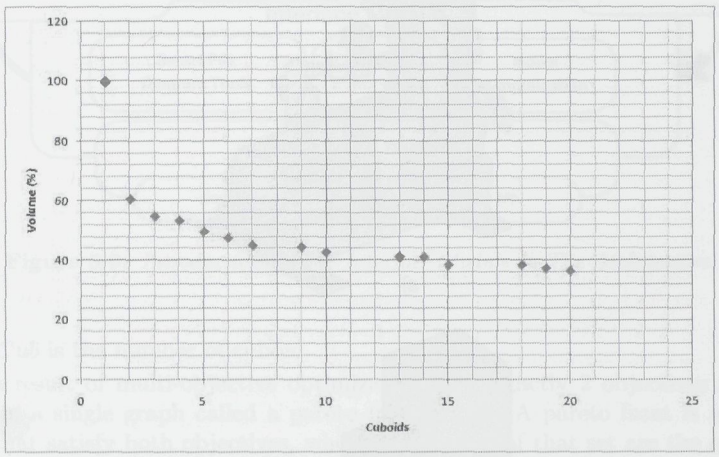


Figure 5.7: *Pareto Front*

Table 5.1: *Comparison results proposed method and voxelization*

Method	Cuboids number	Percentage volume
Approximate orthogonal simplification Method	10	42.58
Voxelization	546	34.32

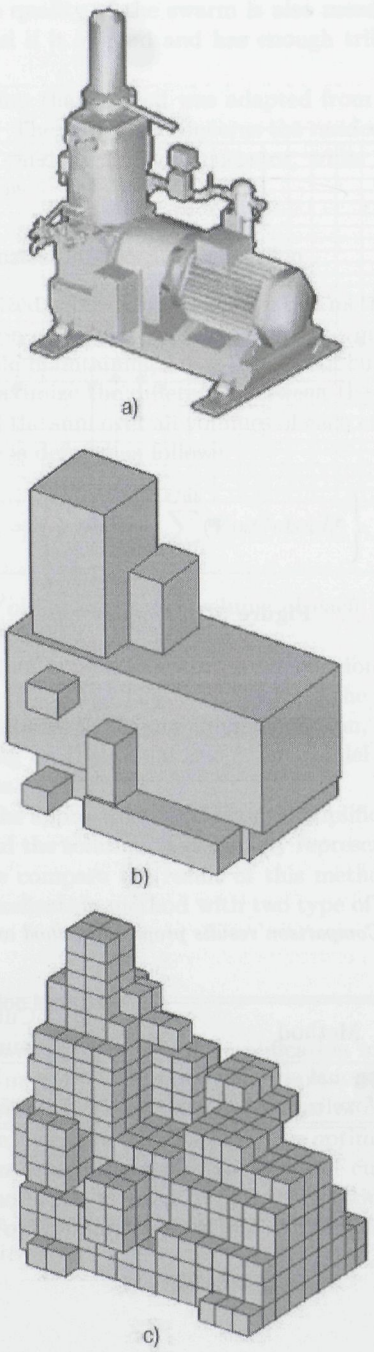


Figure 5.8: *Comparisons*

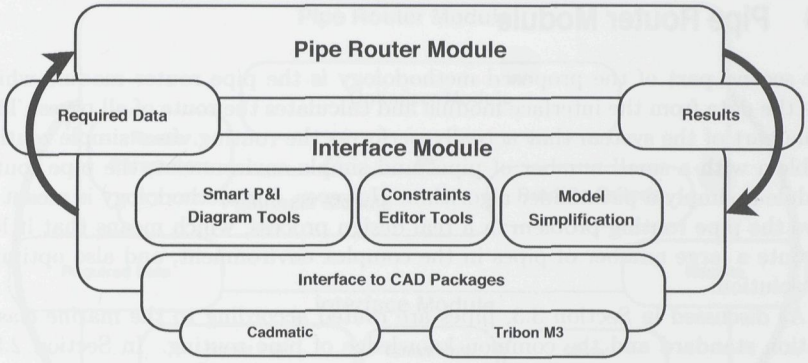


Figure 5.9: *Introduction of The Interface Module in The Architecture*

where *Cub* is the number of cuboid.

The result of multi-objective optimization with exactly 2 objectives can be shown in a single graph called a pareto front graph. A pareto front is a set of result that satisfy both objectives, where all member of that set are the optimal solution for each combination of both objectives. Fig. 5.7 shows that Tribes-D found the pareto front in almost all possible number of cuboids. As a comparison, this pump model consists of 2768 triangles.

Furthermore, based on the pareto front, we can easily select the level of detail of the simplified model. In Fig. 5.8 we can see the comparison of the original model (5.8.a), the simplified model using the approximate orthogonal simplification algorithm with the number of cuboids equal to 10 (5.8.b), and the simplified model using simple voxelization (5.8.c).

Concerning the shape of the simplified model, both methods are able to generate a representative simplified model. The summary is shown in Table 5.1. It shows that using the simple voxelization method, we can easily get a more representative model. However, the number of cuboids that are generated by the simple voxelization method are excessive for our purposes.

5.2.4 Summary of the Interface Module

In this section, we have defined four important parts of the interface module:

1. Constraint editor tool
2. Interface to CAD software packages
3. Smart P&I diagram tool
4. Model simplification tool

Fig. 5.9 shows the extension of the architecture with more detail Interface Data part.

5.3 Pipe Router Module

The second part of the proposed methodology is the pipe router module which uses the data from the interface module and calculates the route of all pipes. This is the part of the system that actually performs the routing. In a simple routing problem with a small number of pipes and simple environment, the pipe router module is simply a path finder algorithm. However, our methodology is meant to solve the pipe routing problem in a real design process, which means that it has to route a large number of pipes in the complex environment, and also optimize the solution.

As discussed in Section 3.3, pipes are routed according to the marine classification standard and the common knowledge of pipe routing. In Section 2.5.1 and 2.6, we have discussed the common knowledge and strategy to route pipes in ship. In this section, it will be formulated in more detail in the form of routing criteria, and will be used as the objective function of the pipe router module.

After the routing criteria are defined, the routing process can be performed by finding the path for every pipe. In a simple situation, the path finding problem can be solved by simply selecting one of the shortest path problem algorithms that have been discussed in Chapter 4. However as described above, in the real ship environment, this problem cannot be solved only by implementing one shortest path algorithm. Therefore, we construct a pathfinder module that is capable to handle the general pipe path finding problem.

Basically the pathfinder module contains several optimization algorithms, and as discussed in Chapter 4, each algorithm has its own advantages and disadvantages. Also, providing each optimization algorithm with the correct parameters is very important to ensure the quality of its result. As discussed in Section 2.4.3, the routing behavior of the pipe router module in the proposed methodology depends on the area in which the pipes are placed. In the optimization problem, the variation in the routing behavior can be achieved by using different optimization parameters.

The main objective of the pathfinder module is to find the optimized path for every pipe. However, since each pipe is routed and optimized one by one, the combined solution might not be the optimized solution as a group. The solution then might not be good enough because our main goal is to optimize the combined solution. Therefore, an optimizer module is needed to ensure that the group solution is optimized.

As described above, the pipe router module is no longer a simple shortest path algorithm, but it become a complex module, as depicted in Fig. 5.10 that consists of four main parts:

1. "Routing criteria" is the measurement of the validity and the quality of the routed pipes.
2. "Pathfinder module" is the part of the pipe router module that finds the path of pipes.
3. "Optimizer module" evaluates the quality of pipe routes that are routed

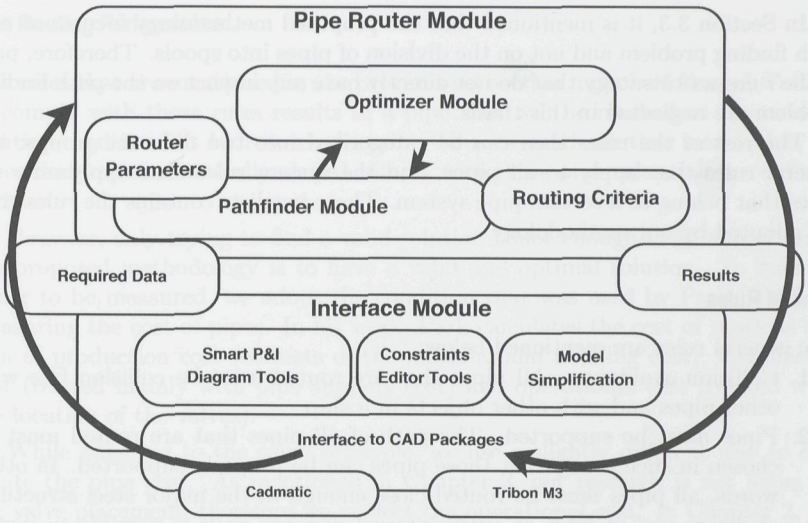


Figure 5.10: *Introduction of The Pipe Router Module in The Architecture*

by the single-router module to ensure that those routes are optimized as a group.

4. "Router parameters" are the fixed parameters that are used by the single-router module to perform its tasks. These parameter values depend on the location where the pipes are routed.

These modules will be discussed in the next four sections.

5.3.1 Routing Criteria

According to their characteristic, the routing criteria to verify the quality of the solution can be categorized as two part:

1. Pipe routing rules,
2. Pipe route performance.

The pipe routing rules consist of criteria that have to be fulfilled; otherwise the solution cannot be used. If the result has passed this first criterion, then it can be evaluated using the pipe route performance criterion.

Pipe Routing Rules

In Subsection 2.5.1, we have discussed the rules that must be followed by a pipe engineer when he routes a pipe in a ship. Also the common strategy to route pipes in a ship has been described in Subsection 2.6.

In Section 3.3, it is mentioned that the proposed methodology focuses on the path finding problem and not on the division of pipes into spools. Therefore, part of the rules and strategy that do not directly have any impact on the path finding problem are neglected in this thesis.

The rest of the rules then can be categorized into two different groups; the general rules that apply to all pipes, and the system rules that apply only for pipes that belong to a certain pipe system. These two lists contains the rules that are adopted by our methodology.

General Rules

The general rules are mentioned below:

1. Collision avoidance. All pipes that are routed must be collision free with other pipes and with other objects in a ship.
2. Pipes must be supported. The path of all pipes that are routed must be chosen in such a way that those pipes can be properly supported. In other words, all pipes must be routed close enough to the major steel structural elements and at least at every interval distance so it is possible to install pipe supports on that pipe.
3. Parallelism. For ease of installation and maintenance, all pipe paths that are close to one another and running in the same direction should be routed in parallel.
4. Layering longitudinal and transverse direction. For much the same reason as for the previous rule, if it is possible pipes are better routed in imaginary pipe racks, and the imaginary racks that have a different direction should be arranged into layers.

System Rules

The system rules should be implemented in such a way that they are easy to be added or edited by a user. The system rules that are implemented:

1. Sea Cooling Water system must be placed as low as possible.
2. Bilge and Ballast system must be placed as low as possible.
3. Fresh Water system must not be routed through the oil tank.
4. Fuel Oil system must not be routed through the fresh water tank.
5. Fuel Oil system must be routed as far as possible from the electrical component (i.e. switchgear) or combustion engine. Especially not above it.
6. Fuel Oil Transfer system must slope down according to the flow direction.
7. Draught Measuring system must slope down according to the flow direction.
8. Lubrication Oil Bow thruster system must slope down according to the flow direction.
9. Dirty Oil and Sludge system must slope down according to the flow direction.
10. Air, Filling, and Sounding system must be routed as straight as possible, and the maximum bends should be limited to less than 30 degree angles.

Pipe Routing Performance

The rules that are mentioned in the previous subsection must be followed. Failing to comply with those rules results in a pipe that will be marked as a non-valid solution, and it must be re-routed again. Thus, a valid solution means that all pipes that are automatically routed comply with those pipe rules that are defined above.

However, only trying to find a valid solution is not enough. The aim of using the proposed methodology is to have a valid and optimal solution. To make it easier to be measured, we adopt the condition that was used by Park [2002] by measuring the cost of pipes. In his work, Park calculates the cost of pipes as the sum of production cost (consists of the material and bending cost), installation cost (related mainly with pipe support cost) and operational cost (related with the location of the valves).

While adhering to the same principle, we use a slightly different way to calculate the pipe cost. As mentioned in Chapter 3, our research is not focus on the valve placement, therefore we neglect the operational cost. In Chapter 2, we have mentioned that the cost of the pipe consists of three elements; pipe material, production and installation cost. Starting from this point, the pipe material and production costs are combined into a single category, the production cost.

Production Cost

From the two elements mentioned above, the production cost is the most obvious part. By calculating the part of each pipe completely, the cost figure can be estimated. The raw material for a pipe consists of a straight pipe. Before it can be installed in the ship, it must be processed according to the pipe spool sketch. It needs to be cut, bent, flanged and if required painted or coated. In total, the production cost of a pipe is the sum of raw material cost, the cost of the production process, and the cost of pipe treatment.

The cost for the raw material of a pipe depends on:

1. Pipe material
2. Pipe length
3. Pipe diameter
4. Pipe thickness
5. Type of pipe bends

The cost to produce a pipe depends on:

1. Raw material of that pipe
2. Pipe surface treatment
3. Type of pipe bends
4. Type of pipe treatment or coating

Installation Cost

When we discuss the installation cost of a pipe, it means that we discuss the pipe spools of that pipe. Basically the installation cost depends on two aspects; the first is whether the pipe is installed during the pre-outfitting stage or during the outfitting stage. The latter costs are around 3 times higher than the first. Beside that, the cost of installation of a pipe depends on the weight of that pipe, the type of coating that the pipe has, the ease to support that pipe spool, and also the shape of the pipe spool. For example, a simple pipe spool that only has one bend but with a length of 2 meter for each legs, can be difficult to install.

Calculating the installation cost is not trivial, it is very sensitive to the labor cost, labor skill, weather, equipment and how well the pipe spool is designed. The version used for the analysis in this thesis includes a simplified approach, which totally depends on the output generated by the router module.

Based on those assumptions, to calculate the installation cost can be done by calculating the man-hours. In his book, Page (1999) describes in depth the man-hours for piping production and installation, and it can easily be adapted to the current situation.

Pipe Cost Value

In this research, we would like to combine the production and installation cost of a pipe into a single pipe cost value. For the straight pipe, the total pipe cost can be estimated to yield a single value. However, we need to differentiate the cost of pipe bending depending on the type.

There are two different types of pipe bending; the bending that can be done by a pipe bending machine, and the type involving the use of pipe elbows. In addition to those two types, in practice there is another case when there are two bends in a pipe and the distance between two bending points is too close to be bent by the pipe bending machine. There are two solutions if this happens: using two elbows or making the two bends using the pipe bending machine while allowing more distance between the two bending points and then cut the excess pipe and weld it back together. The second option requires additional pipe cutting and welding. However, it is still cheaper than using two elbows.

5.3.2 Pathfinder Module

The pathfinder module is a part of the pipe router module that actually finds the path of every pipe. This module routes all pipes one by one, and tries to optimize each pipe by using the routing criteria as its objective functions.

In the routing process, this module uses pre-defined parameters from the router parameters module, and follows the pipe order list from the optimizer module.

Basically, this module contains three separate parts; the blockage checker, the single pipe router and the hybrid back-tracker.

Blockage Checker

As discussed before, besides the data that can be retrieved automatically from the CAD software packages, there are also some areas that must be defined manually by the user of the methodology. Thus, there is a possibility that this areas lead to blocking some pipes. If this situation occurs, the methodology may fail to find a solution.

To prevent this, it is required that the system can perform the blockage test. For this purpose, one of the shortest path algorithms that have been discussed in Chapter 4 can be used. Since the main purpose of the blockage checker is to test whether a pipe can be routed or not, it is not necessary to use the shortest path algorithm that guarantees to find the shortest path, so long the algorithm always finds a solution if it exists. Thus, the fastest computation time is the only criterion that is considered.

In Section 4.4, we selected the shortest path algorithm from Mikami and Tabuchi [1968]. The Mikami-Tabuchi algorithm always finds a solution if it exists, requires less memory than others, and is significantly faster than the A* or Dijkstra algorithm.

Single Pipe Router

The single pipe router is the part that actually routes all pipes. The router get a set of parameters from the routing parameter module according to the location type of the pipe. During the routing process, the pipes are routed one by one according to the list of pipes that are generated by the optimizer module. Then, the result is evaluated using the routing criteria, and the router will send it back to the optimizer module. Based on that, the optimizer module will optimize the new pipe order to be used by the single pipe router. This iterative process will continue until the globally optimized set of pipe paths is founded.

As discussed in Chapter 4, the shortest path algorithm that is suitable to be used by the single pipe router in our proposed methodology is the A* algorithm. There are four reasons for choosing this algorithm over the others; the A* algorithm always finds a solution if it exists, the A* algorithm is faster and uses less memory than Dijkstra, the A* algorithm most likely will find the shortest path, and since it works in a grid environment, the A* algorithm can be used in a weighted environment.

The basic approach to the single pipe router is as follows:

- unobstructed space is decomposed into discrete elements,
- the elements are treated as grids,
- the optimization algorithm walks through the possible path,
- an efficient and valid path through the graph is found,
- if a valid path of a pipe cannot be found, the hybrid back-tracker is invoked,
- the above steps will be repeated until all pipes have been routed or terminate if there is no solution

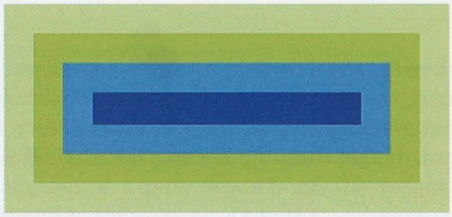


Figure 5.11: *Attraction area*

Figure 5.12: *Magnet area*

Geometrical Constraints

Section 2.4.4 described four types of object constraints. That categorization is based on the way a pipe engineer judges an object or an area when he routes a pipe. However, that categorization is not sufficiently detailed to be a guidance for a computer algorithm. Thus, it needs to be categorized into more detail, and it will be referred as a geometrical constraint.

A geometrical constraint is a virtual area in a three dimensional space. In this thesis, we always work in 3D environment, therefore the terms area means a 3D volume. This virtual area represents a constraint value that will affect the environment where it is located. In our methodology, there are eight basic types of geometrical constraint:

- **Routing point**

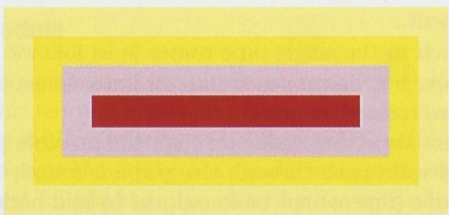


Figure 5.13: *Distraction area*

Figure 5.14: *Special type area*

When a pipe is routed and a routing point exists, that pipe must be routed through that point. A routing point can be added manually by the user or it can be added automatically. For example, if the pipe that is routed contains a fixed location valve, that pipe must be routed through the location of that valve.

- **Obstruction area**
  Obstruction area is a virtual area where no pipes are allowed to be routed through it. This area has an infinite constraint value. Every hard object constraint, such as an equipment, is an obstruction area.
- **Sink area**
  This virtual area is a desirable area for a pipe to be routed in or through. Thus, a pipe that is routed in this virtual area will get a bonus. For example, the area near a major steel construction can be categorized as a sink area, because it will be easier to install a pipe support.
- **Rough area**
  Contrary to a sink area, a rough area is a virtual area where a pipe will get a penalty if it is routed there.
- **Attraction area**
  As shown in Fig. 5.11, an attraction area is the more complex form of a sink area. The actual area is only the dark blue part, but there is also a bonus effect to the surrounding area outside the attraction area itself. The dark blue area has the highest bonus, and for the area further away from it, the bonus is smaller.
- **Magnet area**
  Magnet area, as shown in Fig. 5.12, is almost the same as an attraction area. The difference is that the actual area is an obstruction. Thus, area surrounding the obstruction is preferable, but the actual area itself is a forbidden area. As such the magnet area is a combination of obstruction

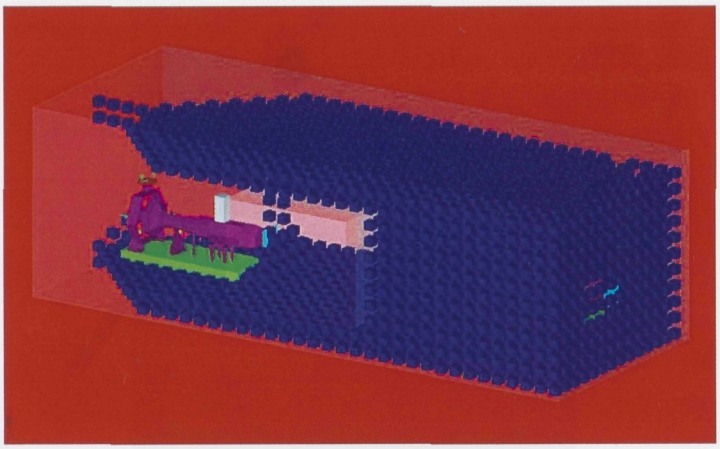


Figure 5.15: *Without attraction area*

area and attraction area.

- **Distraction area**
  In contrast to an attraction area, a distraction area gives a penalty to the pipe that is routed through this area and the surrounding area. Fig. 5.13 shows that the actual attraction area is red, which represent the highest penalty area, and the penalty will be smaller the farther the location is away from it.
- **Special type**
  Beside the basic geometrical constraint above, a combination of two or more basic constraints is also possible. For example, the left figure in Fig. 5.14 shows a NoGo area and the right figure shows a stair area from side view. The center of a NoGo area is an obstruction area, but the edge of it is a rough area. The construction part of a stair is an obstruction area, and the top part is a rough area.

Fig. 5.15 and 5.16 shows an example how the attraction area affects the search process. As shown in Fig. 5.15, for the environment without an attraction area, the search algorithm needs to explore a larger volume than if there is an attraction area in the floor.

Geometrical Constraint Parameters

Each type of geometrical constraint has different influence directions. Some of them influence all 3 directions X, Y and Z, while others only influence 2 directions depending on its orientation.

All of them have a maximum distance of influence, called the distance parameter. They also have min and max parameters to set the minimum and max-

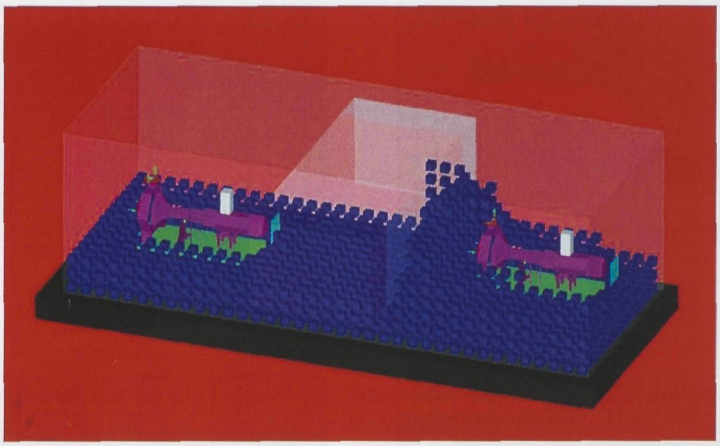


Figure 5.16: *With attraction area in the floor*

imum weight value depending on the distance.

Some of the geometrical constraints have a linear weight dependence on the distance, while others are non linear.

In case two or more areas of geometrical constraint have an intersection, different rules are applied depending on the types. In some combination, the values resulting from the constraints are multiplied. There is also a dominant geometrical constraint which if its influenced area intersects with another's, only the value from that constraint is used.

Grid Decomposition

The unobstructed space is decomposed into grid of cells, with the size of each cell 5 mm by 5 mm by 5 mm. This small cell size is used to improve the quality of the routed pipes. However, there is an important disadvantage by using a small cell size in the grid. As discussed in Chapter 4, the memory that is needed to store a grid of cells is very large, especially in a large three dimensional space when the cell size is small. Fortunately, in most of the cases, the A* algorithm does not need to explore all cells in the space to find the solution. Based on this knowledge, the cells are created during the routing process.

Every cell has a weight value. The weight value of a cell is the penalty or bonus factor that will be used to calculate the objective cost value of the pipe that is routed through that cell. When a cell is created, a weight value will be assigned to that cell based on its location in the three dimensional space. As previously described, each location in the space is affected by the constraint value of the geometrical constraints that lie in that location. Therefore, the cell's weight value depends on the geometrical constraints that lie on or near by that cell. If there

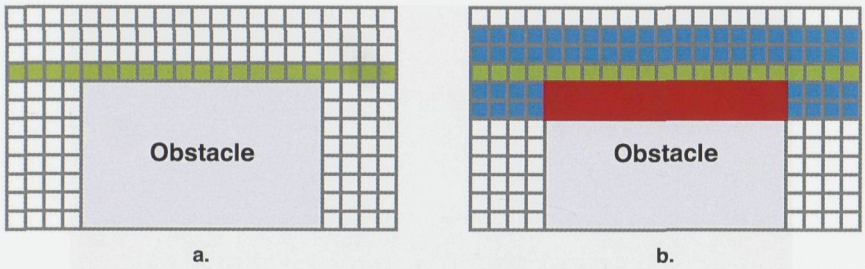


Figure 5.17: *Collision detection is needed*

are more than one geometrical constraints, the weight value is the combination of those geometrical constraints. The objective cost value is a multiplication of the pipe cost value with the cell's weight value.

The geometrical constraints that are automatically created by the rules constraints are based on the pipe that is currently being routed. Thus, the constraints might be different between pipes that belong to different groups to which different rules apply.

Reduce Collision Detection Usage

After the grid of cells is built, the A* algorithm can be used to find the shortest path. As described in Chapter 4, the A* algorithm finds the optimized path by exploring the grid, cell by cell. However, since the cell size is smaller than the diameter of the pipe that is being routed, the path that was found by the A* algorithm might not be valid, because the real pipe might have a collision with an obstacle. For example, Fig. 5.17.a shows the path that was found by the A* algorithm. However, as can be seen in Fig. 5.17.b, the solution is not valid because the actual pipe has a collision with the obstacle, as depicted in red.

To prevent this problem, the collision detection routine must be triggered every time the A* algorithm explores a cell. This solution however, has a big disadvantage because a collision detection routine is computationally expensive, and the total computation time that is needed to find the optimized path becomes longer.

Fortunately, this situation can be avoided by manipulating the obstacle before the process to build the grid of cells is started. Fig. 5.18 shows how it works. In Fig. 5.18.a, as shown in pink color, before the cells are created the obstacle was virtually extended. Then, when the grid is built, that virtual area will not be used as a cell. Therefore, the optimized path that was found by the A* algorithm is always collision free. It is important to ensure that the extended area has the proper size; enough to prevent the collision but not too large. Thus, the expansion size must be adjusted according to the size of the pipe that is being routed.

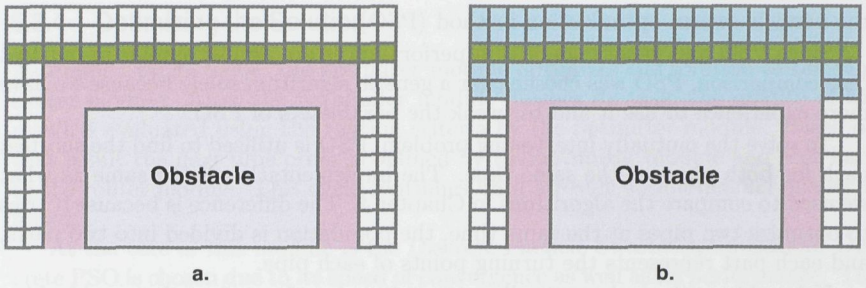


Figure 5.18: *Collision detection is not needed*

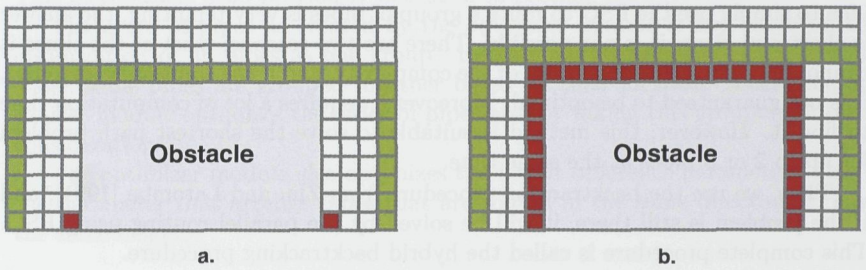


Figure 5.19: *Backtracking*

Hybrid Backtracking

In the pathfinder module, a set of pipes is routed one by one according to the order of routing that was prepared by the optimizer module. Because of that, there is a possibility that the path of one pipe is blocked by the other pipes that were previously routed. This situation is described by Zhu and Latombe [1991] in their work, and they suggest a solution by performing a backtracking process.

The backtracking procedure is shown in Fig. 5.19. As can be seen in Fig. 5.19.a, there is no way to route the red pipe because it is blocked by the green pipe. In this case, those two pipes will be re-routed to find the solution as shown in Fig. 5.19.b.

Using the backtracking procedure will solve most of the pipe blocking problems. However, this procedure cannot solve the mutually intervening case that was mentioned in Subsection 4.5.2. To solve this special case by utilizing the deterministic procedure is possible. However, since it is very hard to predict the environment of a three dimensional space that causes the mutually intervening case, a robust deterministic procedure will be very complex and re-routing of many pipes might be required. Thus, we propose to solve this problem by using one of the heuristic optimization methods that have been discussed in Chapter 4,

the particle swarm optimization method (PSO).

Since PSO and genetic algorithm performances are almost similar in our test case comparison, PSO was chosen over a genetic algorithm solely because we have more experience to use it and to tweak the parameters of PSO.

To solve the mutually intervening problem, PSO is utilized to find the shortest path for both pipes at the same time. The implementation is the same as what we used to compare the algorithms in Chapter 4. The difference is because it tries to optimize two pipes at the same time, the population is divided into two parts, and each part represents the turning points of each pipe.

Using this method, the mutually intervening case of two pipes that are blocking each other can be solved. Then, the question arises, why did we not implement this method to find the paths for all pipes at the same time? Even though the idea to simply use the PSO to route a group of pipes is very tempting, the answer is short and clear; it is not possible. There are two reasons; most of the times it cannot find any solution because of the complexity, and if the solution were found, it is not guaranteed to be optimal. Moreover it requires a lot of computation time to find it. However, this method is suitable to solve the shortest path problem for up to 2 or 3 pipes at the same time.

Thus, we use the backtracking procedure from Zhu and Latombe [1991], and if the problem is still there, it will be solved by the parallel routing using PSO. This complete procedure is called the hybrid backtracking procedure.

5.3.3 Quality Measurement

Subsection 5.3.1 discussed the criteria of pipe routing that must be complied with for every routed pipe. It was also mentioned that the pipe must be routed as efficient as possible in a way that the pipe cost value is as low as possible.

However, the quality of pipes not only depends on the minimum cost, but we must also consider the non quantitative aspects such as the possibility to install a support and the parallelism of the pipes.

To measure a non quantitative aspect is not trivial. Therefore, we need to convert that aspect into a quantitative value. In our methodology, we do this by giving a different weight value for each grid cell. In Subsection 5.3.2 we have described the various types of geometric constraints that affect the grid cell's coefficient weight. Also we have explained that the geometric constraints can be defined manually but also automatically created. For example, the magnet constraints are created for every plate and stiffener to attract pipes to be routed nearby the steel construction. The special type constraints are automatically created for the pipe that had been routed, so the next pipes are attracted to it and run as parallel to it as possible.

The quality of the routed pipes then can be calculated using the combination between the pipe cost value and the weight of grid cells. During the optimization process, this function is used as the objective function.

5.3.4 Optimizing the Solution

As implied by its name, the optimizer module optimizes the solution of the pipe router module. After the pathfinder module finishes its task, the quality of the result is evaluated using the routing criteria by the optimizer module. Based on that result the next pipe order is defined by the optimizer module and sent back to the router module. This cycle continues until a result with a defined quality is found.

At the core of this module is the Discrete Particle Swarm Optimization. Discrete PSO is chosen due to its speed of convergence as well as its performance. The Discrete PSO version that is used here is based on the method developed by Clerc [2004] with some adjustment on the objective function and problem formulation. This method has been previously discussed in Section 4.6.

The order of pipes as created by the optimizer module is not based on the optimization of all pipes as one group. Instead, during the first rough routing stage, some pipes are grouped together based on their location. Then, the optimizer module optimizes the order of pipes also by taking this grouping into the consideration.

The optimizer module also optimizes the height difference parameter. This is the parameter that arrange pipes that are routed in the same direction to have the same height.

5.3.5 Routing Parameters Behavior

Pipe routing in a ship is hard. However there are some behavior patterns that can be identified. An example is that in a certain area a group of pipes always runs in the same direction, so for this area pipes normally are arranged in parallel.

In Subsection 2.4.3 we have discussed three different types of working area from the pipe engineer point of view.

1. Machinery type area
2. Accommodation type area
3. Technical type area

This categorization is adopted in our proposed methodology by creating three different parameter sets to be used by the pathfinder module.

5.4 Implementation

Fig. 5.20 shows the expanded version of the outline methodology architecture that was shown in Fig. 3.1.

The proposed methodology has been implemented for research testing and validation purposes as a software package. The current version of the proposed methodology is implemented using C# programming language tools, and it can be operated under Microsoft Windows operating system using Microsoft .NET 4, and also under Linux operating system using Mono. However the interface part of

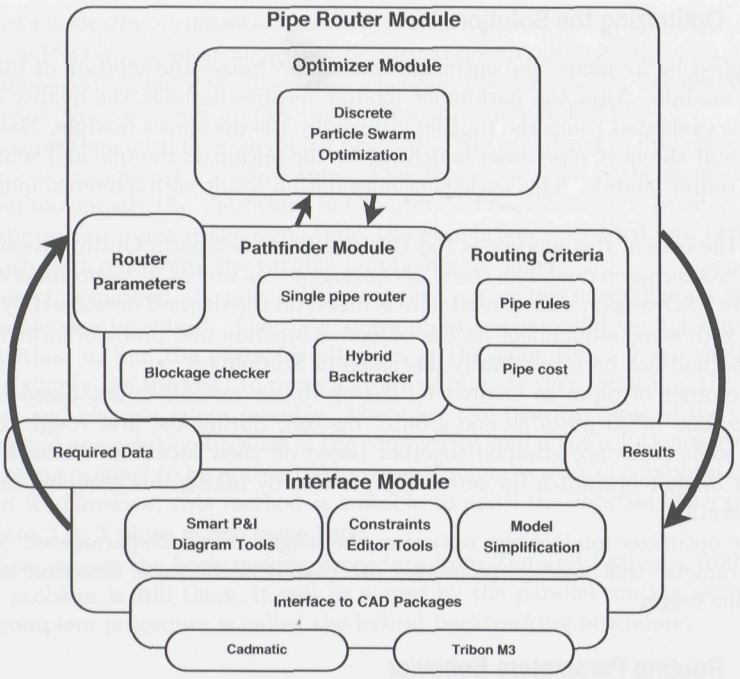


Figure 5.20: *The Complete Architecture of Proposed Methodology*

the proposed methodology must be operated under Microsoft Windows because it has to communicate with the existing CAD software which mainly works under Windows.

One of the design requirements states that the proposed methodology has to be expandable. Because of that the modularity concept is used. This is implemented by dividing the proposed methodology into three main parts, the core of which contains the pipe router module and the optimizer module. The interface contains the interfacing between the proposed methodology and existing CAD software, and also the P&I diagram tools. Finally, the third part contains the knowledge rules that represent the criteria that are used as guidance for the core part.

A mySql database engine is used as the database engine of the proposed methodology. The main reasons for using it are its very good performance and its freeware nature. One other practical reason is the small size and easy installation of mySql.

Fig. 5.21 shows the flow inside the pipe router module. It starts by reading the data from the interface module and than performs the blockage checker to ensure that the initial environment does not block any pipe. The blockage checker itself has been discussed in Subsection 5.3.2. If a pipe is blocked, it gives a notification

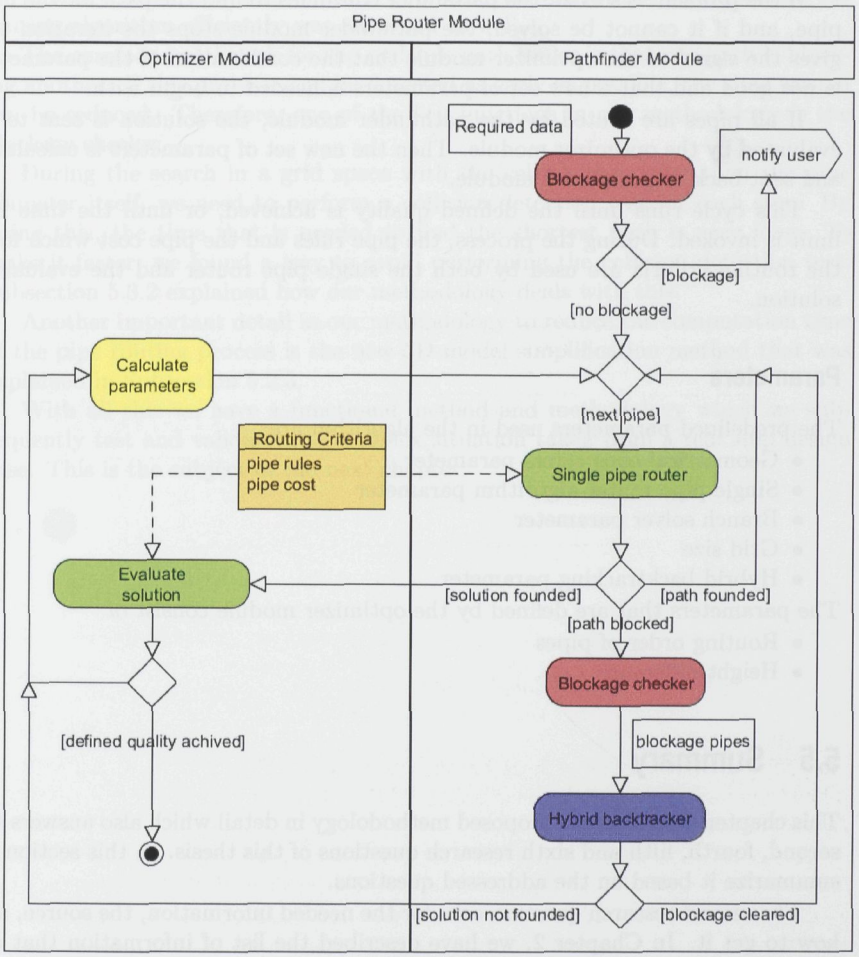


Figure 5.21: *Interconnection inside Pipe Router Module*

to the user.

Than, using the standard parameters, the single pipe router module starts to route pipes one at a time until all pipes are routed. If there is a pipe that cannot be routed, the blockage checker is triggered again to find the routed pipe that blocks the current pipe. Based on that, the hybrid backtracker tries to solve the blockage problem. The hybrid backtracker was discussed in Subsection 5.3.2.

If the problem is solved, the pathfinder continues to find the path for the next pipe, and if it cannot be solved, the pathfinder module stops the iteration and gives the signal to the optimizer module that the combination of the parameters is not good and that a new set of parameters is needed to begin with.

If all pipes are routed by the pathfinder module, the solution is sent to be evaluated by the optimizer module. Then the new set of parameters is calculated and sent back to pathfinder module.

This cycle runs until the defined quality is achieved, or until the time flag limit is invoked. During the process, the pipe rules and the pipe cost which form the routing criteria are used by both the single pipe router and the evaluation solution.

Parameters

The predefined parameters used in the algorithm are:

- Geometrical constraints parameter
- Single pipe router algorithm parameter
- Branch solver parameter
- Grid size
- Hybrid backtracking parameter

The parameters that are defined by the optimizer module consist of

- Routing order of pipes
- Height difference

5.5 Summary

This chapter describes our proposed methodology in detail which also answers the second, fourth, fifth and sixth research questions of this thesis. In this section we summarize it based on the addressed questions.

The second research question asks for the needed information, the source, and how to get it. In Chapter 2, we have described the list of information that are needed to perform the pipe routing process. In Subsection 5.2.4 we have shown that most of the information can be retrieved from the CAD software, and that we also need additional tools such as the smart P&I diagram tools.

The fourth research question asks which common knowledge of the pipe routing process should be adopted in our proposed methodology, and to what extent we should adopt it. Subsection 5.3.1 answered this question by translating the common knowledge into the routing criteria.

Subsection 5.3.3 answers the fifth research question that concerns the way to evaluate the quality of the routed pipes quantitatively. In this subsection, we have explained how the translation from the non quantitative to the quantitative value was made and that our methodology uses this as the objective function.

In this chapter the sixth research question, relating to how to implement the routing algorithm efficiently was answered as well.

The common bottleneck for the multiple path finder is that one path is blocking another. If this problem can be detected as early as possible, the time wasted can be reduced. Therefore, one of the key functions in our methodology is the blockage checker.

During the search in a grid space with the cell's size smaller than the pipe diameter itself, we need to perform a collision detection test for each step. By doing this, the time that is needed to find the shortest path is very long. To make it faster, we found a way to avoid performing the collision detection test. Subsection 5.3.2 explained how our methodology deals with this.

Another important detail in our methodology to reduce the computation time of the pipe routing process is the new 3D model simplification method that was explained in Subsection 5.2.3.

With all this we have a functional method and methodology which we subsequently test and validate in a complex situation taken from a real ship design case. This is the subject of the next chapter.

Chapter 6

Pipe Routing Methodology Validation

6.1 Introduction

In the previous chapters, we have described the needs to automate the pipe routing process and its practical aspects. Then as a means to both investigate and prove our proposed methodology, the automatic router software package was developed as a laboratory.

In order to validate the proposed methodology, we need to use the validation method from Seepersad et al. [2006] called the validation square. This validation technique is chosen because the traditional approach of formal, rigorous and quantifiable validation is problematic for the proposed methodology.

The essence of the validation square is covered by two primary tasks:

- Structural validation of the proposed methodology
- Performance validation of the proposed methodology

6.2 Structural Validation

As explained by Seepersad et al. [2006], there are three complementary facets in structural validity:

1. Internal consistency of each parent construct
2. Internal consistency of the method
3. Appropriateness of the example problem

In the next subsections each facet is described and validated.

6.2.1 Internal Consistency of each Parent Construct

There are two parts that must be shown to validate this facet. The first one is that the requirements for the outcome and the process. The second one is to establish the internal consistency proposed methodology in its entirely.

In Chapter 2, we have described the criteria for pipe routing that contain the requirements for both the outcome and the process. In Chapter 5, we have shown the complete architecture that translates the pipe routing methodology into a set of methods that subscribe to the same requirements. The interviews with pipe routing experts, the discussions with them on the process and also the problem analysis show that the requirements are determined.

In Chapter 5, we have also shown how the methodology architecture was built, and we explained every part that was used to build it. The reasons of choosing those parts were also described in that chapter, and it was shown that each part was selected such that the part requirements were met. With this, the internal consistency of each part of the proposed methodology is satisfied.

6.2.2 Internal Consistency of The Method

To show that the proposed methodology satisfies the internal consistency, we chose to use the flowchart method. By using the flowchart in Fig. 5.21 in Chapter 5, it was shown that all input required for one of the methods is always available from a previously called method.

6.2.3 Appropriateness of The Example Problem

In order to verify the performance of our proposed methodology, we have implemented it into the automatic router software package. Then, we chose to use this implementation to solve the pipe routing problem in a machinery room in a complex ship.

Even though there is no formal statement claiming that the machinery room is the most difficult one, it can be easily established from literature that the characteristic of the pipe routing problem in a machinery room in a ship generally considered to be the most challenging one. Indeed, it has been the main target of many other research to improve the pipe routing process. Based on that, we conclude that the selected example problem is appropriate.

6.3 Performance Validation

As for the structural validation, the performance validation also consists of three facets:

1. The usefulness of the method for the chosen example problem
2. The demonstrated usefulness being linked to applying the methodology
3. The methodology being useful for domains that are broader

6.3.1 The Usefulness of The Method for The Chosen Example Problem

To establish the usefulness of the methodology, it should be applied to a representative example problem. As explained above, the proposed methodology was applied to solve the pipe routing problem in a machinery room in a complex ship.

The first step in establishing the usefulness is to determine how the result for our example problem should be measured; what aspects are needed to be examined and compared.

As discussed in the previous chapters, many researchers have made a lot of effort to tackle the pipe routing problem. In order to prove that their algorithm works, they use their own criteria to measure it. Since most of them merely focus on developing the routing algorithm itself, the validation usually is performed by showing that the algorithm has the capability to route pipes in the defined environment, and then they claim that the algorithm is validated. Some others include more detail in their validation criteria, by not only minimizing the length and number of bends, but also considering the branches. Ultimately, some of them compare the pipes that are routed by their routing algorithm with the pipes that are routed manually. They not only compare the total cost (using total meter-inch and number of bends), but also compare the path of those pipes in the space. The main argument is that since the automatic pipe router algorithm is created by mimicking how a person routes the pipes, both results are expected to have the same path, or at least almost the same.

In our research, beside minimizing the total cost, we have tried to include most of the practical aspects, for example, all pipes must be routed according to the marine standard. There was a point where we were also tempted to compare the path of the pipes that are automatically routed with the one that are routed by a pipe engineer. However, we decided only to make a comparison of the cost to show that the quality of the automatic pipe routing is on the same level with the pipes that are routed manually in terms of cost.

The main reason why we choose not to compare the path of the pipes is because we believe that the argument above was not completely correct. That argument is true if we only route one pipe from one nozzle to another. Then the path of the pipe most likely is the same for both automatic and manual routing. This is because both methods will route the pipe as efficiently as possible. However, there are many pipes that need to be routed in a ship, and it is impossible to route all pipes if we only consider minimizing the cost of the pipe that is currently being routed. Every pipe engineer must consider that all pipes are routed efficiently as a group of pipes. In practice, there are no exact rules to tackle this condition. Thus, every pipe engineer must think to solve this problem individually. Then as a result, the combination of pipes that are routed by a different pipe engineer might not be the same.

In short, it is a fact that there is no standard of how a combination of pipes must be routed in a ship. Thus, it is no longer interesting to compare every single path of the pipes that are routed automatically and manually.

However, in other aspects, we still need to make a comparison between the result of automatic and manual routing. For example, we still need to compare the total cost of the pipes, but merely as a measurement that the total cost of the pipes that are automatically routed are acceptable. Beside that, in a certain difficult area, for example in a very small area that contains many pipes in it, we also would like to show that the automatic algorithm is capable to route pipes in that area, and compare it with the way a pipe engineer handles that kind of situation.

Moreover, rather than focusing on comparing the automatic result with the manual result, we are more interested to show that the proposed methodology has the capability to solve the pipe routing problem in a ship. There are four aspects that are used as measurement criteria:

1. The capability of the methodology to route pipes in a difficult area of a ship, i.e. in a machinery room.
2. The routed pipes must comply with the marine standard.
3. The routing algorithm must produce high quality pipe paths, with regard to the practical aspects. Especially the parallelism and the possibility to install pipe supports for every pipe.
4. Show that the total cost of the routed pipes are reasonably optimized.

Test Case: The Machinery Room

In order to establish the usefulness of the proposed methodology, we have implemented it into the automatic pipe router tools package. This package includes the simple smart P&I Diagram tools, library tools, and the automatic router tools.

The automatic pipe router tools package is used to route the pipes in the most difficult area of a ship. We choose to use it in an area that is considered to be a difficult area by a pipe engineer, rather than to route all pipes in a ship. This is simply because in other areas, the manual routing can be done conveniently using 3D CAD software.

To satisfy the usefulness of our proposed methodology in the machinery room, we need to prove that the measurement criteria are satisfied. However, it is not trivial to measure every criterion from the test case. For example, to show criterion number four that the cost of the pipes that are routed automatically is reasonable, we have to compare many pipes, but to prove the measurement criterion number three, we have to investigate a specific group of pipes.

As the test case, we test our proposed methodology to route pipes in the machinery room in a ship. In this case, all pipes in this part of the compartment will be routed automatically. By proving that the proposed methodology capable to route all pipes in this compartment, we prove that (part of) the first criterion is satisfied.

Then the pipe cost of the results of both manual and automatic routing are compared, by only considering the production and installation cost as described in Subsection 5.3.1. By showing that the total cost similar to the value when the

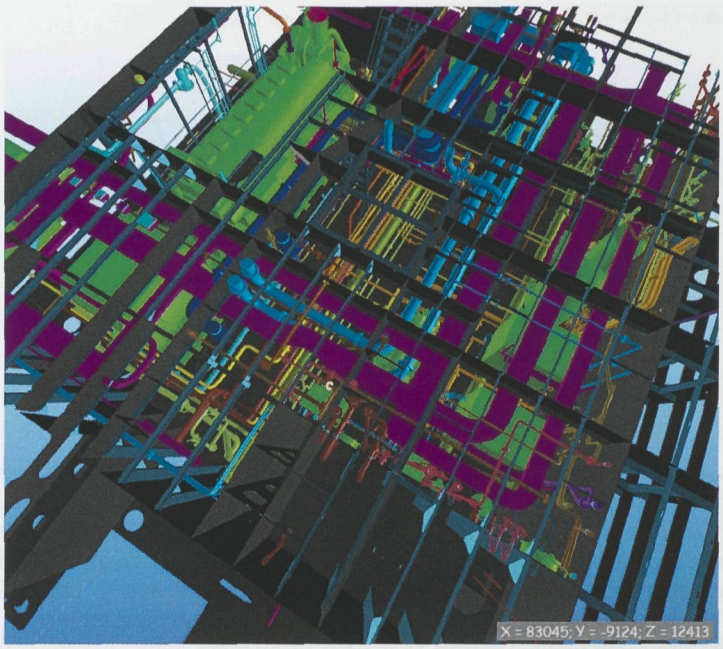


Figure 6.1: *Machinery room*

pipes are routed manually, we validate criterion number four.

The automatic routing process was performed in a HP Z400 computer, with Intel Xeon CPU @ 3.2GHz, and 24GB RAM. The operating system is Windows 7 x64. This computer is also used as the database server using mySQL 5.5 which consumed 8GB of RAM. Therefore, the computer memory that can be used by the automatic pipe routing was limited to 12GB.

The machinery room is the most common test case in automatic pipe routing research area. The main reason is because the machinery room is considered to be the most difficult area for a pipe engineer.

As our test case, we use the machinery room of a real vessel. Fig. 6.1 shows the original situation that was imported from CAD software.

Model Preparation

As mentioned in the previous chapter, our methodology uses the simplified 3D model from the CAD software. The simplification of the 3D model is performed during the 3D model import process.

There are two parts in the simplification process; for the plates in the steel construction and for other parts. Steel plates are simplified by settings the thickness

Table 6.1: *Calculation time*

Run	Time (Minutes)	Remarks
1	3233	
2	3015	
3	5760	stopped
4	2683	
5	2658	memory overload
6	4373	
7	3554	
8	2745	
9	3111	
10	2890	

to zero. For the other parts, they are simplified into a small number of boundary boxes using the simplification method that was discussed in Subsection 5.2.3.

During the simplification process, the basic geometry constraints are created automatically, i.e. the attraction constraint areas are also created in the steel construction part. The intention of this is to attract a pipe to choose a path nearby the steel construction for the ease of installing the pipe support.

There are some manual setups that need to be done. One of them is to place some valves that are required to be arranged as a group in a certain location.

Discussion of the result

We have run the implementation of the methodology to route the pipes in the machinery room 10 times, and the best result is shown in Fig. 6.2. This is achieved with 5 mm cell size, which means that the center pipe is routed in a 5 mm grid space. For example, the automatically routed pipe path can only lies in coordinates that are a multiple of 5 (8255, -555, 645), and cannot be in coordinate (8257, -557, 643).

The smaller the grid space size is, the better the chances of the methodology to find the solution. However, it needs more memory and longer computation time.

As shown in Fig. 5.21 in Chapter 5, the pipe router module contains some heuristic parts, in the optimizer module and hybrid backtracking part. Therefore, on each run we expect to get a different solution.

Table 6.1 shows the calculation times that were needed by the methodology. As we can see, in run #3 and run #5 the methodology failed to find the solution. In run #3, the methodology was stopped because it still did not find the complete solution after 4 days. The limitation of 4 days was chosen based on the previous two runs that found the solution in about 2 days time. However, without the limitation, the methodology might also able to find the complete solution in run

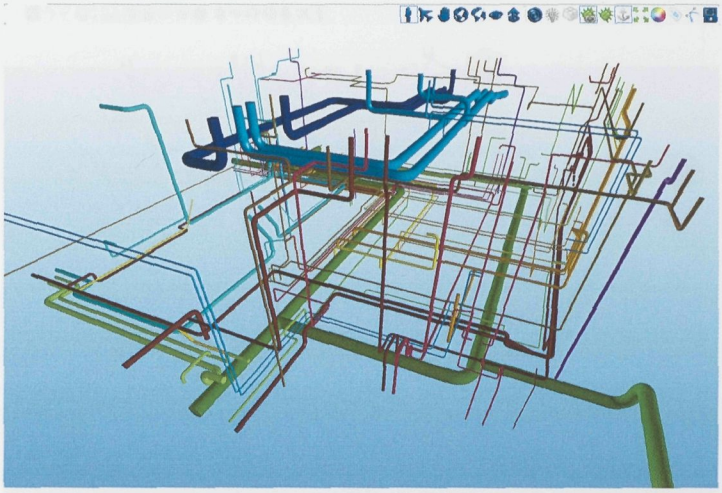


Figure 6.2: *Automatically routed*

#3.

In run #5, the methodology crashed due to memory overload. There are some aspects that might generate this problem. There is a possibility that another application running on the computer was causing this. However, since it is only a research implementation, at this moment the automatic pipe routing software package is not yet optimized for real production in terms of robustness, such as memory management and prevention of crashes caused by the mistake of the operator. Here we only focus on the algorithm to solve the pipe routing problem.

If we conclude that the methodology failed on run #3 and #5, the success ratio is 80% which can be considered as acceptable at this stage. Therefore, the 5 mm grid space is good enough for our present purpose. Considering only the runs that were able to find the complete solution, the average time needed to route 144 pipes in the engine room is 3200 minutes.

At this moment our focus in not to minimize the calculation time. But it is important to show that the proposed methodology is able to automatically route many pipes within an acceptable time, and 3200 minutes to route 144 pipes in an engine room can be claimed to be an acceptable result.

To shorten the calculation time and make the success ratio higher, we need to focus on the improvement of the software package, and keep the proposed methodology intact. However, this is not our main focus because it is a software engineering problem.

Also, with the rate of hardware improvement in the past years, we can safely make an assumption that in the near future the increasing computational power of the computer will make the calculation time shorter. The fast growth in com-

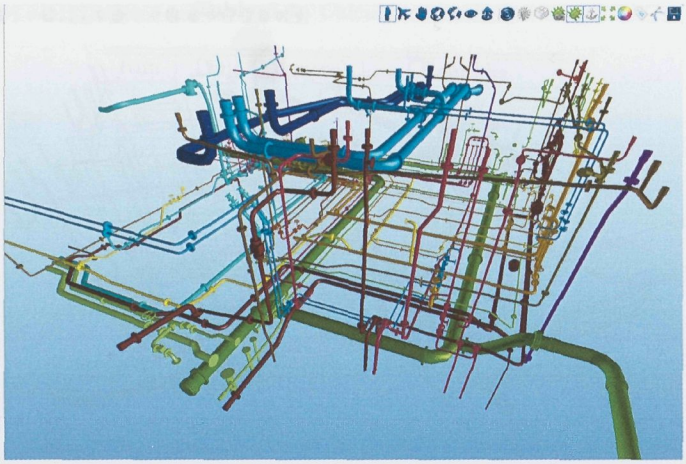


Figure 6.3: *Manually routed*

putation power is one of the main motivation to route pipes automatically; the time that is needed to do the manual routing is basically almost the same from year to year, but the computation time to perform automatic pipe routing will grow shorter and shorter.

Our main focus was to translate the way a pipe engineer routes the pipes in a very difficult area of the ship into the procedural methods that can be done by the computer. This is very interesting and challenging and until today, there is a general assumption that only experienced pipe engineers are able to route pipes properly in the engine room.

Fig. 6.3 shows the pipes in the machinery room that are routed manually. If we compare it with Fig. 6.2, even though there are important similarities, the differences between those two results can be noticed easily. As we mentioned earlier, those differences are expected, since there are many ways to route the group of pipes.

One of the reasons for the differences is that the routing algorithm in the proposed methodology is more strict with respect to parallelism, while in practice, pipes might not have to be routed in parallel that strictly. This can be seen in Fig. 6.4 and Fig. 6.5.

Beside that as shown in Fig. 6.6, to ensure the minimum number of bends, the automatic routing algorithm is strict with respect to maintaining different heights for X and Y direction, while in manual routing, shown in Fig. 6.7, a pipe engineer is more flexible in this respect.

It is interesting to compare the total cost to make sure that the result of our proposed methodology is on par with the manual routing result. Table 6.2 shows the comparison of the pipe cost per system. The cost value in Table 6.2 is

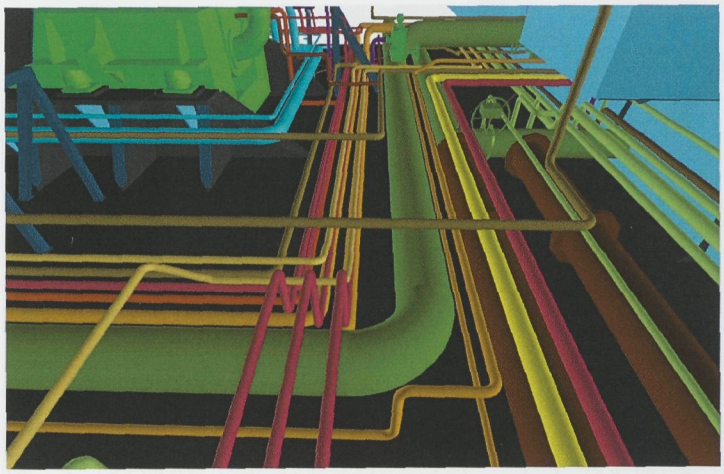


Figure 6.4: *Parallelization in automatic routing*

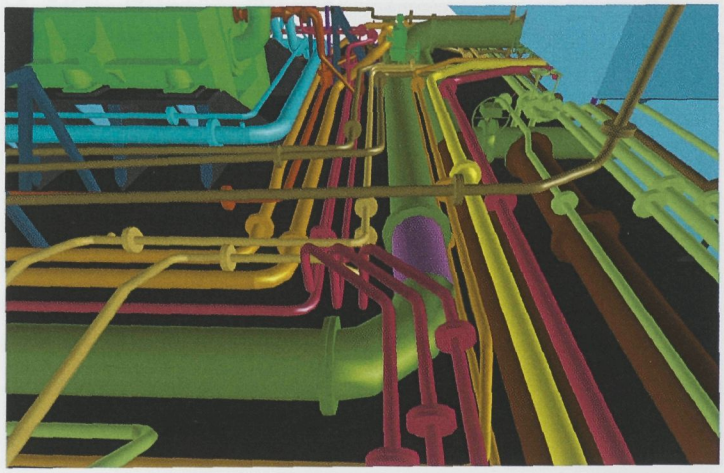


Figure 6.5: *Parallelization in manual routing*

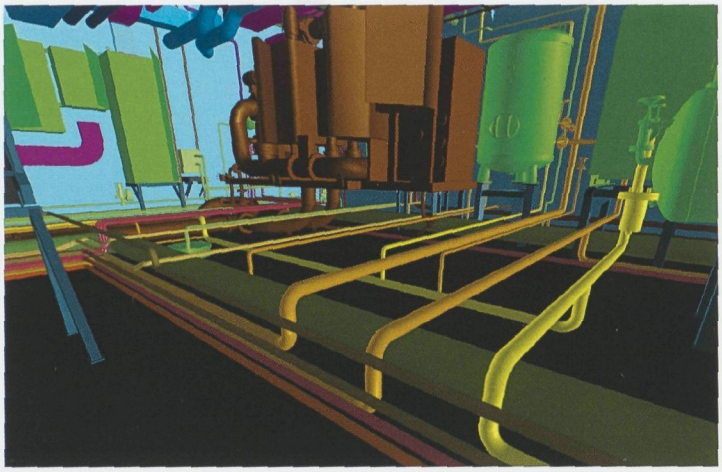


Figure 6.6: *Maintaining difference heights for different direction*

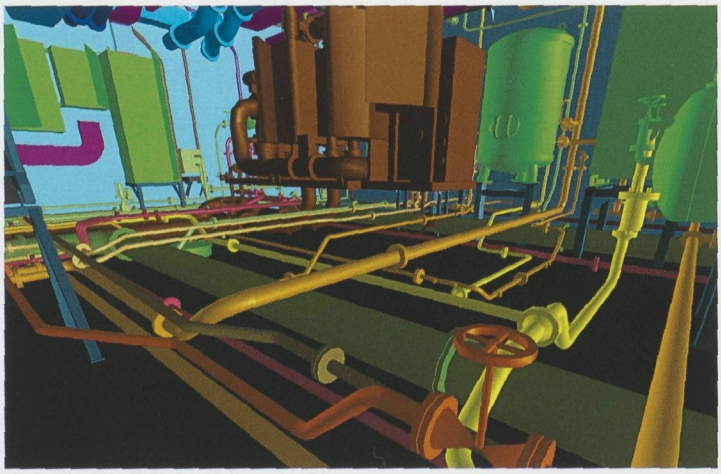


Figure 6.7: *Difference in heights more flexible for manual routing*

System	Number of pipes	MeterInch auto	MeterInch manual	Price auto	Price manual
Air and Sounding	10	1464	1387	5802	5446
Ballast	6	15274	14232	97145	98540
Bilge	18	7522	7068	37575	36150
Chilled Water Return	4	5461	5042	10732	10296
Control Air	5	1621	1551	4665	5403
Deck Scupper	7	1386	1219	7671	7051
Driptray	13	5058	4715	17817	17664
Exhaust Gas	1	153	121	662	625
FO Service	17	5770	5617	20199	21514
FO transfer	17	8947	8511	19485	20538
Fresh Coolingwater Auxiliaries	12	9389	8894	38006	37387
Fresh Coolingwater Engines	13	5940	5903	15618	17770
Greywater	1	58	57	337	330
LO Service	1	669	542	2479	2475
LO Transfer	2	407	404	1405	1799
Sanitary Fresh Water	1	110	105	454	442
Sludge	3	1316	1288	3327	3117
Starting Air	5	3048	3210	8647	10127
Working Air	8	6309	6026	13382	14153
Total	**144**	**79902**	**75892**	**305408**	**310827**

Table 6.2: *Routed pipes comparison*

calculated based on the pipe cost value that was used as part of the objective cost function as described in Chapter 5. It must be noted that the pipe cost value is not the same as the objective cost value that was used during the optimization process. The pipe cost value is a constant value per pipe in every location, while the objective cost value varies depending on the cell location, and each cell has a weight value that depends on the geometric constraints.

Also, since our methodology does not split pipes into spools, we exclude the cost of the pipe flanges, with the assumption that both manually and automatically routed pipes have almost the same number of flanges.

As shown in Table 6.2, the total meter inch of the automatically routed pipes is around 5% higher than the pipes that are routed manually. This result is reasonable because the current routing algorithm uses 90 degree of bending angle as its first choice. It only will choose other bending angles for 3 reasons; the distance between two bending points is shorter than the minimum requirement, the pipe is a member of a pipe system that requires the pipe path to have a slope and depends on the predefined bending parameter, to select a cheaper bend type.

The total estimated cost in automatic routing is around 1.7% lower than the total cost of manual routing. This is also a reasonable result since the routing algorithm intends to route pipes as parallel as possible. Also the consistency of maintaining constant height for X and Y direction reduces the number of bends that are needed.

Since we only compare the estimated cost of the pipes, we are not claiming that the automatically routed pipes are definitively cheaper than the manually routed pipes. Nevertheless, from this comparison, we can claim that the solution obtained by the proposed methodology is quite good and that the validity of the cost criterion is established.

In a previous subsection we mentioned that for operational reasons, some of the valves must be located nearby in an accessible area. Fig. 6.8 and Fig. 6.9 show that the proposed algorithm is also capable to route pipes with this condition.

Since all criteria are met, it proves that the proposed methodology is able to route pipes in difficult areas of a ship. Thereby we demonstrated its usefulness.

6.3.2 The Demonstrated Usefulness is Linked to Applying The Methodology

During our attempt to demonstrate the usefulness of the methodology in the previous subsection, we applied it and as a corollary did also prove that the usefulness is linked to the utilization of the proposed methodology.

6.3.3 The Methodology is Useful for Domains that are Broader

In order to identify what can be categorized as broader domains relevance to our research we need to look back to its main objectives. These were to identify the expertise of pipe engineers and translate it into a set of procedures to route pipes

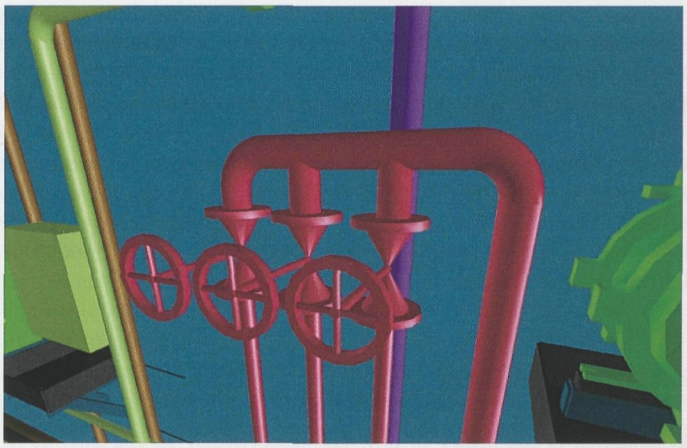


Figure 6.8: *Predefined valves group*

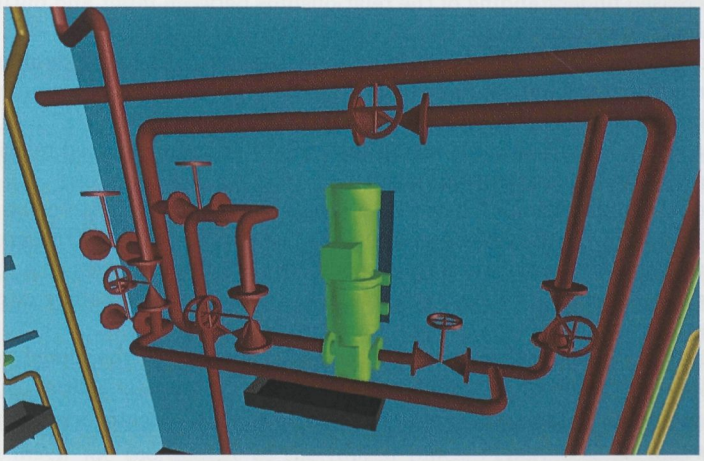


Figure 6.9: *Predefined valves group*

in a ship. From that, it was clear that our broader domain is the pipe routing problem in an arbitrary ship.

As mentioned above, we took the machinery room as the example problem. It was also stated that the machinery room is the most difficult part of the ship. Based on this, it can easily be deduced that if the methodology is able to solve the pipe routing problem in that room, it must be able to be used in other areas in a ship also.

Indeed, our research addresses pipe routing. There are also other distribution systems that need to be routed: ducts, cables and walkways for example. While these distribution channels obey different rules, our methodology can most likely be tailored to these situations by implementing these rules. Then the relevant domain for our research is indeed considerably broader.

6.4 Sensitivity Analysis

Sensitivity analysis is the study of how the uncertainty in the output of a mathematical model or system (numerical or otherwise) can be apportioned to different sources of uncertainty in its inputs, Saltelli et al. [2008].

Pannell [1997] says that sensitivity analysis can be useful for a range of purposes, including:

1. Testing the robustness of the results of a model or system in the presence of uncertainty.
2. Increased understanding of the relationships between input and output variables in a system or model.
3. Uncertainty reduction: identifying model inputs that cause significant uncertainty in the output and should therefore be the focus of attention if the robustness is to be increased (perhaps by further research).
4. Searching for errors in the model (by encountering unexpected relationships between inputs and outputs).
5. Model simplification - fixing model inputs that have no effect on the output, or identifying and removing redundant parts of the model structure.
6. Enhancing communication from modelers to decision makers (e.g. by making recommendations more credible, understandable, compelling or persuasive).
7. Finding regions in the space of input factors for which the model output is either maximum or minimum or meets some optimum criterion.

Our methodology consists of many parameters, that are categorized into three categories:

1. **Routing algorithm parameters;** As described in chapter 5, our methodology utilizes the hybrid optimization method. Therefore many parameters need to be tuned properly. Both the general and system rules that are explained in subsection 5.3.1.1 are included in this category.

2. **Geometry constraints;** The parameters of the influenced area of the geometry constraints must be chosen properly to ensure that the pipes are routed as intended.
3. **Objective criteria;** This is related to the cost of the pipes. Differences in how pipe cost is calculated might affect the way pipes will be routed.

During the validation of the methodology, we used the optimized parameters that are tuned properly.

To measure the parameter sensitivity of our methodology, we show how a certain parameter affects the behavior of the methodology. Since the number of parameters is large, only some of them are chosen to be varied.

Since the core of the routing algorithm has been discussed in detail in chapter 4 and 5, it is not interesting anymore to try to change these parameters. Also, we will maintain the pipe routing rules that are used as parameters.

In the routing algorithm, basically to create pipes that are easily installed, we would like to route pipes as orthogonally as possible. However, there are conditions that require that the pipe should be routed non orthogonally, e.g. when there are two consecutive bends, and the distance between two bends is smaller than a minimum value. For this kind of situation, the algorithm allows that particular part of the pipe to be routed non orthogonally. This parameter needs to be tuned and as part of the sensitivity analysis, we vary this parameter and compare the results.

In order to show the importance of the geometry constraints parameters, we try to route the pipes with modified parameters for the influenced area of the attraction, magnet, and distraction area.

6.5 Sensitivity Analysis

As described in the previous section, the sensitivity analysis is performed by varying certain parameters of second category, the geometry constraints. In this case, to isolate the effect of each parameter, the routing process is performed individually for each parameter change.

The first parameter that is tried is the orthogonal restriction parameter. This parameter decides whether the pipe is allowed to be routed non-orthogonally or not. The second parameter is the attraction coefficient of the geometric constraints. As discussed in Chapter 5, this coefficient is used to influence whether the pipe is routed as close as possible to a steel structure or as far as possible from a particular object.

During the parameter tuning, we have found that the effect of both parameters is non linear for the quality of the routed pipes. Therefore, we only show the extreme lowest and highest parameter values. The results are compared with the optimal solution.

Table 6.3 shows the summary of the comparison with the optimal solution. Strictly orthogonal means that every pipe must be routed orthogonally, except

	Length	Cost
Optimal	100,00%	100,00%
Strictly orthogonal	100,03%	104,35%
No restriction orthogonal	failed	failed
Very high attraction coef.	102,33%	110,97%
Very low attraction coef.	failed%	failed%

Table 6.3: *Comparison with the results for optimal parameter settings*

if the distance between two bending points is shorter than allowed. In the other way around, the non restriction orthogonal means that the pipes are allowed to be routed non-orthogonally without any restriction at all. In the case of a very high attraction coefficient, the objective cost value is low if the pipe is routed close to a geometric constraint. Vice versa, with a very low attraction coefficient, the objective cost value is not influenced by the distance between pipe and geometric constraint.

Fig. 6.10 shows the example of pipes that are routed with the strictly orthogonal parameter. As can be seen in that figure, the distance between two bending points are too close to each other. In that case, it is required to use two elbows rather than using a single pipe with two bends, which in terms of cost, can be between 4-6 times more expensive than the normal pipe bending as shown in Fig. 6.11.

In the other way around, allowing pipes to be routed non-orthogonally without any restriction might prevent the proposed methodology to find a complete solution. In our case, the methodology failed to find a complete solution if there is no restriction at all for the orthogonality. Apparently the optimization procedure or the parameters in this case need to be adjusted. Alternatively we could consider starting the optimization from the standard case and slowly relax the orthogonality parameter and find a solution in that way. We did not pursue that possibility in this thesis.

To achieve the optimal result as shown in Fig. 6.11, the restriction parameter must be chosen properly. The parameter tuning must be done properly and it is not a trivial task. As we can see in Table 6.3, the cost result using the strictly orthogonal parameter is around 4% higher than the optimal one. Meanwhile, the non restriction orthogonal parameter case failed to find the solution. From this fact, it can be deduced that it is safer to choose the restriction parameter value that is too high.

The variation of the geometry constraints parameters is also interesting. By lowering the distance of influence in the attraction and magnet area, some of the pipes are routed too far away from the steel construction. Therefore it is very hard to put the pipe supports on it, and in most of the cases during operations pipe vibration may occur. In our test, the low attraction coefficient value failed to find the complete solution.

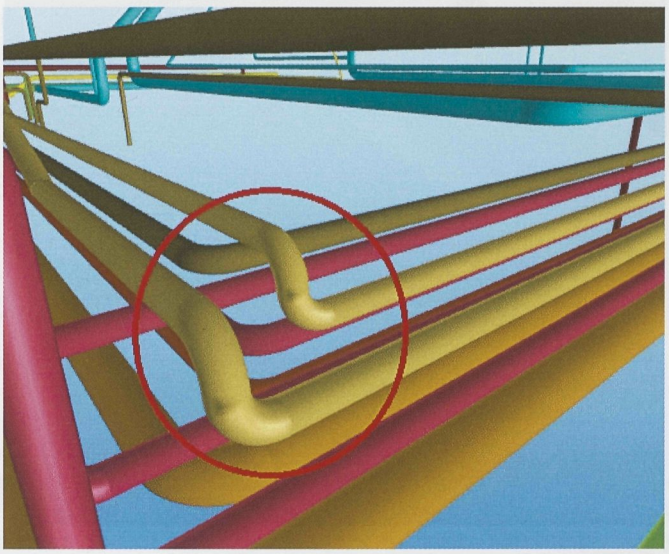


Figure 6.10: *Only allowed orthogonal pipe*

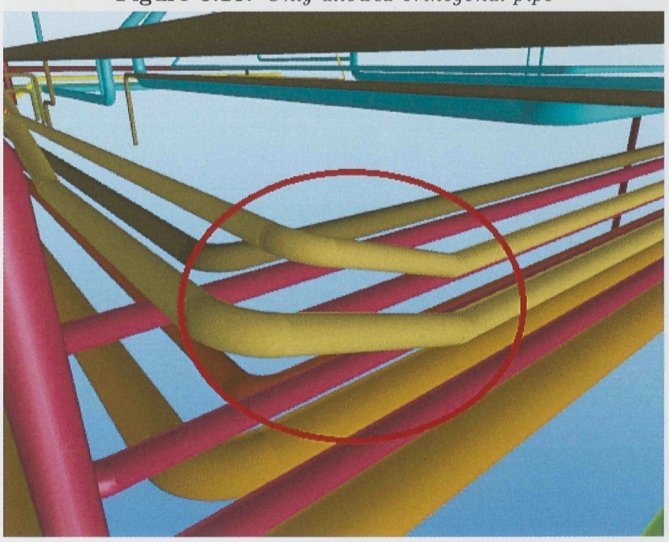


Figure 6.11: *Pipes is allowed to be routed non orthogonally to allow a cheaper bending*

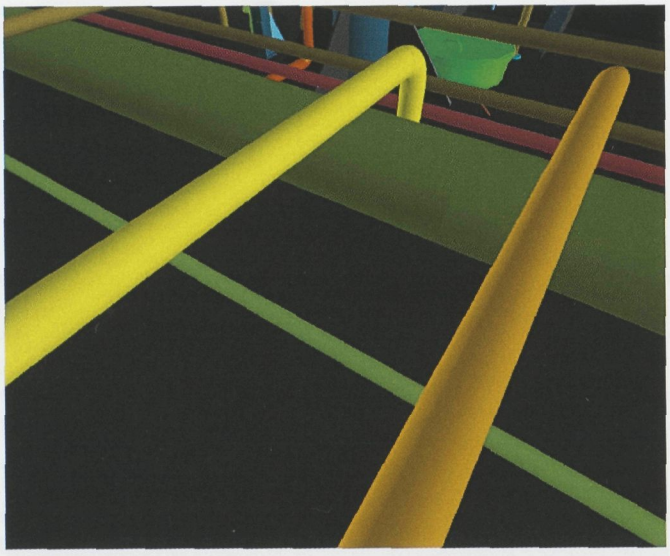


Figure 6.12: *Optimal distance of influence and coefficient in geometry constraints area*

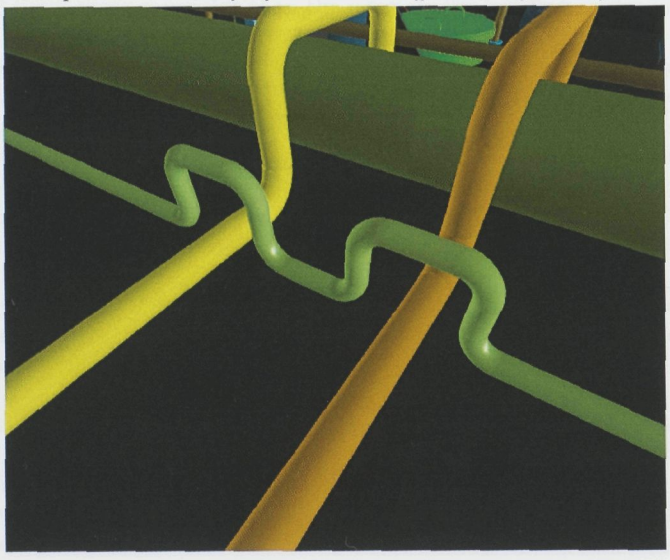


Figure 6.13: *Large distance of influence and coefficient of the geometry constraints area*

However, if this coefficient is too large, the solution won't be good anymore. As can be seen in Fig. 6.13 and Fig. 6.12, by using the very large attraction coefficient, it produces many unnecessary bends. Table 6.3 shows that the cost is more than 10% higher.

Therefore, the importance to have an optimal parameter for the attraction coefficient is higher than the restriction orthogonal parameter.

6.6 Summary

In this chapter we have validated our proposed methodology and its implementation using the validation square from Seepersad et al. [2006]. This evaluation aims to answer our main research objectives of this thesis:

1. to what extent can the expertise of a pipe engineer be identified and translated into procedures that lend themselves to be computerized?
2. if we build a pipe routing methodology based on the results of the first question and combine it with advanced optimization techniques in a practically applicable method, how good will it perform?

In this experiment, we measure the performance of our proposed methodology to route pipes in the most difficult area in a ship. We have compared the result to pipes that were routed manually by a pipe engineer. The comparison proves that the methodology is able to perform the pipe routing process and is on par with the quality of a pipe engineer result. Our experiment shows that the methodology is able to implement the common knowledge.

We also learn from this experiment that the success ratio of the tools that were built from the methodology is only 80%. Since we found that the cause of the error probably was on the software engineering part, this number is good enough to prove that the methodology is valid, but not yet perfect.

At the end of this chapter, we also performed some experiments by varying some of the key parameters to identify the methodology's sensitivity to these parameters. The results shows that great care must be taken to use the cored parameter settings. Further work in this area is recommended.

Chapter 7

Conclusions and Recommendations

7.1 Research Questions

The work reported in this thesis is motivated by the main research objectives below:

1. to what extent can the expertise of a pipe engineer be identified and translated into procedures that lend themselves to be computerized?
2. if we build a pipe routing methodology based on the results of the first question and combine it with advanced optimization techniques in a practically applicable method, how good will it perform?

In the course of experiments conducted to answer the question, we raised the following six sub-questions:

1. On which design phases should we focus and what is the reason for that?
2. What information is needed to perform the pipe routing process, who is responsible to provide it and how can one get it?
3. What is the common knowledge to route pipes that is used as guidance by a pipe engineer?
4. Which knowledge from Research Question #3 should be adopted in our proposed methodology and to what extent can the current practical knowledge be absorbed into a programmable methodology?
5. How to evaluate the quality of a set of routed pipes quantitatively?

6. How to efficiently implement the routing algorithm in a given 3D environment?

We provide conclusions to those research questions in Section 7.2, and add some recommendations in Section 7.3.

7.2 Conclusion

In order to stay competitive in the shipbuilding industry, European shipyards need to be more efficient in design and production processes. One of the useful directions is to be more innovative in the pipe routing process. A lot of effort has been done in the past to be able to route pipes automatically in a ship. However to date automatic routing is not applied nor possible in ship design.

As much tacit basic knowledge on the pipe routing process exists, we decided to absorb the expertise of the pipe engineer and translate it into procedures that lend themselves to be programmed and executed in a computer. We started by establishing the information that is needed to route pipes, how to get it and who is responsible for that. This step provided answers to our research question #2 and the result is described in Chapter 2. By having that knowledge, in Chapter 5 we could focus on the selected information that is needed for our methodology.

To translate the expertise of a pipe engineer, we made an inventory of the common knowledge of the pipe routing process. This step was not easy to be done, especially since currently there is no official guideline for a pipe engineer to route pipes in a ship. We performed many interviews experts in pipe routing to absorb their knowledge. During the interview period, the pipe routing guideline is compiled by v.d. Berg [2009]; the experienced pipe engineer team leader. The interview results are formulated as the common knowledge of pipe routing. Thus, with regards to research question #3, we should refer to the whole of Chapter 2.

Then we survey the state of the art in automatic pipe routing and in Section 3.6 we described some of the previous researches that have most relevance for our problem domains. While reviewing those researches, we found the answer to the research question #1. From Subsection 1.3.1 we knew that during the contractual phase, approximate routing is sufficient. Combining suitably selected existing methods with the completeness of our methodology, e.g. the ability to extract the required data easily from CAD software and automatically perform the cell generation, we can build the pipe routing application for the pre-contractual phase easily. For the sake of widespread applicability however, we placed our focus on the scientifically much more interesting and demanding detail design phase.

With regards to research question #4, Subsection 5.3.1 provided the answer. The essential common knowledge with direct impact on the path finding problem is selected and reformulated as general and system rules.

To perform the optimization process we defined the objective function that needs to be minimized. As described in Chapters 2 and 5, the objective in the

pipe routing process can be divided into two categories. First is the quantitative value that consists of pipe production and installation cost. This is relatively easy to be calculated, and since our research attempt to use the practical environment, the actual total pipe cost from the pipe contractor is used. The second part is of non quantitative nature. It concerns issues that involve parallelism and aesthetics considerations.

With regards to research question #5, we conclude that the best way to measure the quality of the pipes is by assigning quantitative measures to them. The quantification process was done by giving a weight to every cell based on the geometric constraints reflecting the space and objects surrounding it, then combine the pipe cost value with the cell's weight. The results are used as the objective function.

Routing pipes is hard. Implementing the expertise of pipe engineers into computer procedures is even harder. In this thesis, we proposed a methodology and methodology to answer that challenge. It does not merely focus on the algorithm to find the pipe path, but also includes the data preparation step and, more generally, its embedding in the practical design process.

In Chapter 5, we described our proposed methodology in detail. We started with the interface module that contains three important parts; the functionality to extract data from CAD software, the smart P&ID tools and the model simplification. The last part, the model simplification, is very important to ensure that the whole automatic routing process can be done within an acceptable time. We established a novel method to automatically simplify the 3D model heuristically to ensure that the model is simple yet sufficiently detailed.

In our methodology, we reformed the 3D space into a grid of cells. Balancing between the quality of the routed pipes and the computation time, the cell size was fixed at 5 mm by 5 mm by 5 mm. The fact that the cell size is smaller than the diameter of pipes raises the need of collision detection. We know that collision detection is computationally expensive, therefore we introduced the expanding obstacle method as described in Chapter 5. Even though this method looks simple, the performance of the shortest path algorithm significantly improved.

For the path finding itself, we investigated both deterministic and heuristic shortest path algorithms. We made a comparison by implementing the standard form of 5 well-known shortest path algorithms, and selected the A* algorithm as the main single pipe routing algorithm and the Mikami-Tabuchi algorithm as the blockage tester.

Since our goal is to route many pipes, there is a possibility that two or more pipes are blocking each other. We call this the mutually intervening problem. We tackled this by implementing the particle swarm optimization to route multiple pipes at the same time.

Since our methodology basically routes pipe one at a time, the order of pipes is very important. To optimally order the pipes, we implemented the discrete particle swarm optimization which minimized the objective values of the routed pipes.

In Chapter 6, we have applied the methodology to route pipes in the machinery room of a ship. The results prove that our proposed methodology is able to perform automatic routing processes in a real ship. The results are excellent and support detailed engineering research questions. We also showed that we can build the methodology in such a way that it uses the selected common knowledge of the expert pipe engineers. In this way, it is knowledge based.

In the course of this application we have also validated the methodology using the validation square method.

7.3 Recommendations

As we mentioned in the previous section, we have proved that our proposed methodology can be used to solve automatic pipe routing problems in difficult areas of a ship. However, our work is not complete, since there are some important aspects that have not been addressed in this thesis.

To improve the proposed methodology, there are at least three main points that should be focused on in future research. The first point concerns the number of common pipe routing knowledge rules that should be included in the methodology. As mentioned in Chapter 5, not all of the common knowledge is integrated in the methodology described in this thesis. By including more items, the quality of the routed pipes may be further increased. However, the selection must be done carefully, by only selecting the most important ones that have most influence.

The result from our proposed methodology is only in the form of pipe routes. It still requires manual work to split it into pipe spools. Therefore, it is also very interesting to complement the methodology with the method to split pipes into spools and use the result as an input for the research of Wei [2012] to generate the assembly sequence automatically.

As explained in Chapter 6, our methodology uses 90 degree angles as default, and only chooses other bending angles if required. This situation is good enough for steel pipes, which form the majority of the pipes in a ship. However, a PVC pipe might require to have different bending angles, and the current behavior of the methodology is not yet entirely suitable for that situation.

Another interesting subject for further research is to extend our methodology to route HVAC and cable trays in a ship.

Bibliography

ABS. *Guidance notes on the application of ergonomics to marine systems.* The American Bureau of Shipping, 1998.

P. J. Angeline. Using selection to improve particle swarm optimization. In *Proc. IEEE Congress on Evolutionary Computation*, pages 84–89, May 1998.

Andi Asmara and Ubald Nienhuis. Optimum routing of piping in real ships under real constraints. In *Proc. Int. Conference on Computer and IT Application in Maritime Industry*, pages 271–281, April 2008.

Pipework Design User Guide. AVEVA, 2007.

C B Barber, D P Dobkin, and H T Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, (22), 1996.

Marshall W. Bern and Ronald L. Graham. The shortest-network problem. *Scientific American*, 260:84–89, 1989.

Ewerton E.S. Calixto, Paula G. Bordeira, Hugo T. Calazans, Cesar A. C. Tavares, and Marco T. D. Rodriguez. Plant design project automation using an automatic pipe routing routine, 2009.

M. Clerc. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In *Proc. IEEE Congress on Evolutionary Computation*, volume 3, pages 1951–1957, Washington, DC, July 1999.

M. Clerc and J. Kennedy. The particle swarmexplosion, stability, and convergence in a multidimensional complex space. 6:58–73, February 2002.

Maurice Clerc. Discrete particle swarm optimization. In Godfrey C. Onwubolu and B. V. Babu, editors, *New optimization techniques in engineering*, pages 219–240. Springer, 2004.

Maurice Clerc. Tribes-d, 2008. URL `http://clerc.maurice.free.fr/pso/\#TRIBES-D`.

J Cohen, A Varshney, D Manocha, G Turk, H Weber, P Agarwal, F P Brooks, and W Wright. Simplification envelopes. In *In: Rushmeier, H. (Ed.), SIGGRAPH 96 Conference Proceedings. Annual Conference Series. ACM SIGGRAPH*, pages 119–128. Addison Wesley, 1996.

Edgar W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959. ISSN 0029-599X.

R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proc. IEEE International Symposium on Micro Machine and Human Science 6th*, pages 39–43, Nagoya, Japan, October 1995.

R. C. Eberhart and Y. Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proc. IEEE Congress on Evolutionary Computation*, pages 94–97, Seoul, Korea, May 2001.

A. I. El-Gallad, M. E. El-Hawary, A. A. Sallam, and A. Kalas. Enhancing the particle swarm optimizer via proper parameters selection. In *Proc. Canadian Conference on Electrical and Computer Engineering*, pages 792–797, 2002.

H. Y. Fan and Y. Shi. Study of Vmax of the particle swarm optimization algorithm. In *Proc. of the Workshop on Particle Swarm Optimization*, Indianapolis, IN: Purdue School of Engineering and Technology, April 2001.

X. Fan, Y. Lin, and Z. Ji. The ant colony optimization for ship pipe route design in 3d space. In *World Congress on Intelligent Control and Automation*, October 2006.

R. Edward Freeman and David L. Reed. Stockholders and stakeholders: A new perspective on corporate governance. *California Management Review*, Vol 25 Issue 3:88–106, 1983.

F.S.Nooruddin and G Turk. Simplification and repair of polygonal models using volumetric techniques. *IEEE Trans. Visual Computer Graphics*, 9:191–205, April 2003.

Z. L. Gaing. A particle swarm optimization approach for optimum design of pid controller in avr system. 19:384–391, June 2004.

Michael Garland and Paul S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '97, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997. ISBN 0-89791-896-7.

A.M. Gibbons. *Algorithmic Graph Theory*. Cambridge University Press, 1985. ISBN 9780521288811.

D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Kluwer Academic Publishers, Boston, MA, 1989.

C Gotsman, S Gumhold, and L Kobbelt. Simplification and compression of 3d meshes. In *Tutorials on Multiresolution in Geometric Modelling*, pages 319–361. Springer, 2002.

S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: a hierarchical structure for rapid interference detection. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 171–180, 1996. ISBN 0-89791-746-4.

R. Guirardello and R. E. Swaney. Optimization of process plant layout with pipe routing. *Computers and Chemical Engineering*, 30:99–114, 2005.

R.L. Harrington. *Marine Engineering.* The Society of Naval Architects and Marine Engineers, 1992.

Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Systems Science and Cybernetics*, 4(2):100–107, 1968.

David W. Hightower. A solution to line routing problems on the continuous plane. In *Proc. 6th Design Automation Workshop (IEEE)*, pages 1–24, 1969.

J Hjelmervik and J.-C. Leon. Gpu-accelerated shape simplification for mechanical-based applications. In *Proc. IEEE Int. Conference on Shape Modeling and Applications*, pages 91–102, June 2007.

Chen Hua-hong, Luo Xiao-nan, and Ling Ruotian. Mesh simplification algorithm based on quadrangle collapse. In *Proc. IEEE Int. Conference on Image and Graphics*, pages 960–965, August 2007.

Teruaki Ito. A genetic algorithm approach to piping route path planning. *Journal of Intelligent Manufacturing*, 10:103–114, 1999.

J. Jerald, P. Asokan, G. Prabaharan, and R. Saravanan. Scheduling optimisation of flexible manufacturing systems using particle swarm optimisation algorithm. *The Int. Journal of Advanced Manufacturing Technology*, 2004.

Sang-Seob Kang, Se-Hyun Myung, and Soon-Hung Han. Design expert system for auto-routing of ship pipes. In *Proc. Pacific Conference on Manufacturing*, October 1996.

Sang-Seob Kang, Se-Hyun Myung, and Soon-Hung Han. A design expert system for auto-routing of ship pipes. *Journal of Ship Production*, 15:1–9, 1999.

A Kaufman. Voxels as a computational representation of geometry. In *in The Computational Representation of Geometry. SIGGRAPH '94 Course Notes*, page 45, 1994.

J. Kennedy. Why does it need velocity? In *Proc. IEEE Swarm Intelligence Symposium*, pages 38–44, June 2005.

J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, NJ, December 1995.

David G Kirkpatrick and Raimund Seidel. The ultimate planar convex hull algorithm. *SIAM J. Comput*, (15), 1986.

J. Klein Woud and D. Stapersma. *Design of propulsion and electric power generation systems.* IMAREST, 2003.

T. Krink and M. Løvbjerg. The lifecycle model: Combining particle swarm optimisation, genetic algorithms and hillclimbers. In *Proc. International Conference on Parallel Problem Solving from Nature*, pages 621–630, Granada, Spain, September 2002.

T. Krink, J. S. Vesterstrom, and J. Riget. Particle swarm optimisation with spatial particle extension. In *Proc. IEEE Congress on Evolutionary Computation*, volume 2, pages 1474–1479, May 2002.

R. A. Krohling. Gaussian swarm: A novel particle swarm optimization algorithm. In *Proceedings IEEE Conf. on Cybernetics and Intelligent Systems*, pages 372–376, Singapore, December 2004.

R. A. Krohling, F. Hoffmann, and Ld. S. Coelho. Co-evolutionary particle swarm optimization for min-max problems using gaussian distribution. In *Proc. IEEE Congress on Evolutionary Computation*, pages 959–964, June 2004.

C. Y. Lee. An algorithm for path connections and its applications. *IEEE Trans. on Computers*, EC-10:346–365, 1961.

E.V. Lewis. *Principles of Naval Architecture.* Society of Naval Architects, 1988. ISBN 978-9991181417.

A. Lim, Jing Lin, and Fei Xiao. Particle swarm optimization and hill climbing to solve the bandwidth minimization problem. In *Proc. The Fifth Metaheuristics International Conference*, Kyoto, Japan, August 2003.

Myung ll Roh, Kyu-Yeul Lee, and Woo-Young Choi. Rapid generation of the piping model having the relationship with a hull structure in shipbuilding. *Advances in Engineering Software*, 38:215–228, 2007.

M. Løvbjerg and T. Krink. Extending particle swarm optimisers with self organised criticality. In *Proc. IEEE Congress on Evolutionary Computation*, pages 1588–1593, Honolulu, Hawaii, 2002.

Lawrence Markosian, Philip Newcomb, Russell Brand, Scott Burson, and Ted Kitzmiller. Using an enabling technology to reengineer legacy systems. *Communications of the ACM*, 37(5):58–70, 1994.

Y. Matsui, H. Takagi, S. Emori, N. Masuda, S. Sasabe, C. Yoshimura, T. Shirai, S. Nioh, and B. Kinno. Automatic pipe routing and material take-off system for chemical plan. In *Conf. on Design Automation*, pages 121–127, 1979.

K. Mikami and K. Tabuchi. A computer program for optimal routing of printed circuit connectors. In *Proceedings of IFIPS*, volume H47, pages 1475–1478, 1968.

R. Newell. An interactive approach to pipe routing in process plants. *J. Information Processing*, pages 121–127, 1972.

T. A. J. Nicholson. Finding the shortest route between two points in a network. *The Computer Journal*, 9:275–280, 1966.

David J. Pannell. Sensitivity analysis: strategies, methods, concepts, examples. *Agricultural Economics*, 16:139–152, 1997.

Jin-Hyung Park. *Pipe-Routing Algorithm Development for A Ship Engine Room Design.* PhD thesis, University of Washington, 2002.

Jin-Hyung Park and Richard L. Storch. Pipe-routing algorithm development: case study of a ship engine room design. *Expert Systems with Applications*, 23: 299309, 2002.

T. Peram, K. Veeramachaneni, and C. K. Mohan. Fitness-distance-ratio based particle swarm optimization. In *Proc. IEEE Swarm Intelligence Symposium*, pages 174–181, April 2003.

Patrick William Rourke. *Development of a Three-Dimensional Pipe Routing Algorithm.* PhD thesis, Brunel University, 1975.

J. Salerno. Using the particle swarm optimization technique to train a recurrent neural model. In *Proc. IEEE International Conference on Tools with Artificial Intelligence*, pages 45–49, Newport Beach, CA, November 1997.

A. Saltelli, Marco Ratto, Terry Andres, Francesca Campolongo, Jessica Cariboni, Debora Gatelli, Michaela Saisana, and Stefano Tarantola. *Global Sensitivity Analysis: The Primer.* 2008.

Sunand Sandurkar and Wei Chen. Gaprus - genetic algorithms based pipe routing using tesselated objects. *The Journal of Computers in Industry*, 1998.

Carolyn C. Seepersad, Kjartan Pedersen, Jan Emblemsvag, Reid Bailey, Janet K. Allen, and Farrokh Mistree. The validation square: How does one verify and validate a design method? In Kemper E. Lewis, Wei Chen, and Linda C. Schmidt, editors, *Decision Making In Engineering Design*. ASME, New York, 2006.

Y. Shi and R. C. Eberhart. A modified particle swarm optimization. In *Proc. IEEE International Conference on Evolutionary Computation*, pages 69–73, Piscataway, NJ, 1998.

Y. Shi and R. C. Eberhart. Comparing inertia weight and constriction factors in particle swarm optimization. In *Proc. IEEE Congress on Evolutionary Computation*, pages 84–88, San Diego, CA, May 2000.

Cai Tao, Pan Feng, and Chen Jie. Adaptive particle swarm optimization algorithm. In *Proc. IEEE Congress on Intelligent Control and Automation*, volume 3, pages 2245–2247, June 2004.

Li Tao, Wei Chengjian, and Pei Wenjang. Pso with sharing for multimodal function optimization. In *Proc. IEEE Int. Conference on Neural Networks and Signal Processing*, volume 1, pages 450–453, December 2003.

D. A. Taylor. *Introduction to Marine Engineering*. Elsevier Butterworth-Heinemann, 1996.

Mario v.d. Berg. Guidelines piping design. Internal Document IHC Merwede O&M, 2009.

Zheng Wang and Jon Crowcroft. Analysis of shortest-path routing algorithms in a dynamic network environment. *SIGCOMM Comput. Commun. Rev.*, 22(2): 63–71, April 1992.

Glenn E. Wangdahl, Stephen M. Pollock, and John B. Woodward. Minimum trajectory pipe routing. *Journal of Ship Research*, 18:44–49, 1974.

Yan Wei. Automatic generation of assembly sequence for the planning of outfitting processes in shipbuilding, 2012.

David Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1997.

H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control in electric power systems. In *Proc. IEEE Congress on Evolutionary Computation*, pages 87–93, Seoul, Korea, May 2001.

David Zhu and Jean-Claude Latombe. New heuristic algorithms for efficient hierarchical path planning. *IEEE Transactions on Robotics and Automation*, 7: 9–20, 1991.

R.J. Zuurmond. Automatisch routeren van pijpleidingen in schepen. Master's thesis, Technische Universiteit Delft, 2004.

Summary

Pipe routing consumes a large part of the total required effort in the ship design process. In current practice, it takes around 30-40 thousands man hours to route pipes of a middle size complex ship. Reducing the time of this process will have a large impact in total engineering cost.

Judged by the results, the current process produces an excellent solution although an objective, quantitative assessment of this is difficult given the fact that there is no mathematically determined optimum known. So, in order to speed up the process, the current process was investigated, adopted and subsequently translated into the computer procedures.

The main objective of this thesis is to create a pipe routing methodology that can be used in ship detail design process in practice. The methodology consists of the functional framework, the architecture and its implementation.

In order to do it properly, the pipe routing process was investigated. Since no formal documentation that explains how pipe routing process should be done in practice is available, this knowledge was gained by carrying out many interviews and discussions with experienced pipe engineers and other stakeholders in the pipe routing process.

Beside the know-how on how a pipe engineer route pipes in ship, those interviews and discussions provide us with the criteria of pipe routing in ship.

Based on that knowledge, the functional framework of our methodology was derived. The elements needed by the functional framework were investigated and reviewed, starting with reviewing the previous research attempts on improving pipe routing process. Next the well known algorithms needed by each part of the functional framework were compared and selected.

After all elements had been completed, the architecture of pipe routing methodology was built. Finally it was implemented into a prototype software package that can be used to route pipes in a real ship.

The validation of the proposed methodology was carried out using the validation square technique by performing the structural and performance validation. The implementation of the proposed methodology was used to solve the pipe rout-

ing problem in a machinery room of a real ship. The parameter variances was also performed and compared.

As conclusion the pipe routing methodology was developed and validated on the difficult area of a complex ship. As shown by the quality of the test case result, the main objective that was previously mentioned is fulfilled indeed.

Samenvatting

Routing van pijpleidingen beslaat een groot deel van de totale benodigde inspanning in het scheepsontwerpproces. In de hedendaagse praktijk zijn 30-40 duizend manuren vereist om de pijpleidingen van een middelgroot, complex schip te routeren. Vermindering van de tijd die benodigd is voor dit proces, zou een grote impact hebben op de totale engineering kosten.

Wanneer men naar de resultaten van het handmatige routeerproces kijkt, valt op dat dit zeer goed is, hoewel een objectieve, kwantitatieve maatstaf moeilijk aan te leggen is omdat er geen wiskundig bepaald optimum bekend is. Daarom werd, om het proces wezenlijk te versnellen, het huidige proces als uitgangspunt genomen en onderzocht. Vervolgens werd dit vertaald in algoritmen die door een computer opgelost kunnen worden.

Het hoofddoel van dit proefschrift is om een pijp-routing methodiek te ontwikkelen, die in de praktijk gebruikt kan worden, namelijk gedurende het detail-ontwerpproces. De methodiek bestaat uit het zg. functionele raamwerk, de architectuur en de implementatie.

Ten behoeve van deze ontwikkeling werd het pijp-routing proces onderzocht. Aangezien er geen formele documentatie van het proces bestond waarin praktijkvoorschriften worden beschreven, is deze kennis verkregen door middel van vele interviews en discussies met ervaren pijpschetsers en andere belanghebbenden in het pijp-routing proces.

Het resultaat van deze interviews en discussies is een vastgelegde rationale betreffende de manier waarop een schetser de pijpen routeert en een set formele criteria.

Van deze kennis kon het functionele raamwerk van de beschreven methodiek worden afgeleid. De elementen benodigd voor het raamwerk werden onderzocht, te beginnen met een onderzoek naar eerdere pogingen om het pijp-routing proces te verbeteren. Vervolgens werden alle bekende algoritmen die van toepassing zouden kunnen zijn op de verschillende onderdelen van het functionele raamwerk vergeleken, en de meest geschikte oplossingen geselecteerd.

Nadat alle afzonderlijke functionele elementen ontwikkeld waren, werd de ar-

chitectuur van de methodiek gemaakt. Tenslotte werd alles geïntegreerd in een testversie van een softwarepakket dat gebruikt kan worden om daadwerkelijk automatisch pijpen te routeren in een echt schip.

De validatie van de voorgestelde methodiek werd uitgevoerd met behulp van de zg. validation square techniek (validatiekwadranten), die kijkt naar de structurele aspecten van validatie, maar ook kijkt naar de geleverde (ontwerp)prestaties van de methodiek.

Tenslotte werd de pijp-routing methodiek ontwikkeld en gevalideerd op een moeilijke ruimte van een complex schip. De kwaliteit van het resultaat van de testcase laat zien dat het onderzoeksdoel, zoals dat eerder werd vermeld, inderdaad wordt behaald.

Acknowledgments

This research was conducted within the CE3P project (Concurrent Engineering, Production, Planning and Pricing) and the Integraal Samenwerken project. The pipe routing research subject was carried out in IHC Offshore and Marine who employed me to do this research. I gratefully acknowledge management of IHC Offshore and Marine for this opportunity and support.

I would like to thank the doctoral committee for your time and effort reviewing this thesis. My gratitude for your comments, suggestions and critiques has improve the quality of this thesis.

This research project would not have been possible without the support of many wonderful people. I wish to express my gratitude to my promotor, prof.dr.Ir. Ubald Nienhuis who was abundantly helpful and offered invaluable assistance, support and guidance, not only work related but also helped me to get through life in The Netherlands. I would like to thank him for the trust he gave me, the independence he allow me in conducting this research, his experience he shared with me and his advices and support he always offered me when I needed them.

Other most valuable support has come from Teus van Nordennen who has given me the opportunity, trust and support to conduct this research. He was also the one who kept me in balance between practical and scientific worlds.

I would to thank Jenny Coenen who helped me a lot, especially during the beginning of my research and also her willingness to proofread this thesis.

Since the beginning of my journey, I received many supports and assistances from a large group of people. There are too many of you to mention but I'd especially like to thank the following people.

Frank Dekker and Gerco Brand who gave me the first insight about pipe routing in practice, introduced me to the CAD software package and provided me with many kinds of data through all the research. Mario van den Berg who shared his expertise in the pipe routing process and especially for compiling the document on pipe routing in practice.

The engineering department from IHC O&M in general and Gert Rook, Sjaak van Dorsten, Bert Rissema, Wim van Mannen, Cor Molenaar, Gerard Smouter,

Andre Dubbeldam, Krijn Ooms, Ilja van Deurzen, Henk Scheep, Peter de Rek, Nuur Nuur, Ken Schie and Edwin Keizer in particular. The discussion on pipe routing process in practice provided invaluable knowledge for this research.

Peter Wagenaar, your insight on pipe cost calculation during the pre-contract phase is highly appreciated.

Jeroen Kaarsemaker and other members of process management department, thank you for the discussion that we had during these past years. Especially Reinier Zuurmond who started this research project and gave me a solid base to start with.

From IHC Piping, Peter Gelderbloom and Mark Whitney for the knowledge on how pipes are produced, Ad Hazelaar and Edwin van Leeuwen who showed me how to calculate pipe cost in practice, and especially Edwin, thanks a lot for the data.

Also my colleagues in TU Delft, in particular Erik Ulijn for sharing his office with me, Ria Nieuwland-Jobse for her helping me with many administrative things, Jan Jaap Nieuwenhuis for letting me use some of his typesettings, Guus van der Bles, late Hugo Grimmelius, Bart van Oers, Robert Hekkenberg, Jeroen Pruyn, Wei Yan, Elena Moredo for the company and the discussions we have had over the past years.

Many thanks to Dina Shona Laila for reviewing this manuscript.

My parents and family back home in Indonesia, who understand my choice to live far away from all of you.

Last, but most certainly not least, I would especially like to thank my awesome family for the love, patience, support, and constant encouragement. Without them I would not come this far. Erin, Faza, Gafa, Kai, this is for you.

Curriculum Vitae

Andi Asmara was born in Jakarta, Indonesia on February 18, 1974. After finishing his highschool in SMA 70 Jakarta in 1992, he studied Automation and Control Engineering from Electrical Department at Bandung Institute of Technology. He graduated in 1996, and his BSc. final project title is 'Application of Programmable Logic Controller for Analog and Discrete System'.

He worked as an Automation Product Engineer at Schneider Electric Indonesia from 1996 to 1999. In 1999, he became a co-founder of Ekakarsa Cita Dinamika, an automation system integrator company which was responsible for programmable logic controller system during the commissioning and warranty period at Musi Pulp Mill.

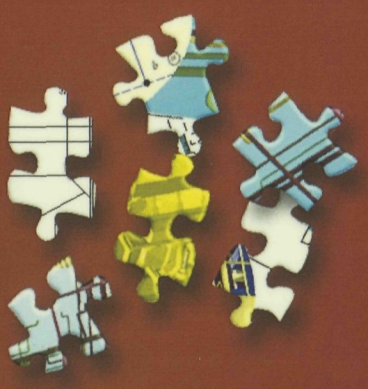
In 2002, he continued his education in automation and robotics program at Universität Dortmund, Germany. He graduated in 2005 as master of science in automation and robotics, and his MSc. thesis title is 'Accelerated Co-evolutionary Particle Swarm Optimization for Computed-Torque Controller parameter tuning for Robot Manipulator'.

In 2005, he started with his PhD research as a researcher at Merwede Shipyard (now IHC Offshore and Marine) and carried out one of the sub-project of the CE3P project that was done in co-operation with the TU Delft. Within CE3P, he focused at the automatic pipe routing project. After the CE3P project finished in 2009, the automatic pipe routing project was continued in the Integraal Samenwerken project.

After the pipe routing research finished in 2010, but while still writing his dissertation, he continued to work in IHC Offshore and Marine and also worked on other sub-project in Integraal Samenwerken to develop the 3D Information Viewer.

He is currently employed as one of the project managers in the One IHC Program inside IHC Merwede.

13W34870/ T1/ 9789065623263



DAP Delft Academic Press

9 789065 623263