Optimal System Design with Geometric Considerations

by

Kwang Jae Lee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2014

To Jinsook

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

$\mathbf{d}_i$	vector of all dimensions of component i;size of vector d_i can vary depending on i
$\mathbf{x}_g$	design variable vector for optimization problem for packaging
$\mathbf{x}_p$	design variable vector for optimization problem for functionality
C_i,C_c	geometric space occupied by i-th component and container, respectively; they are sets of points
$C_i \cap C_j$	intersection of C_i and C_j
d_{ij}	j-th dimension of component i
f_g	objective function for optimization problem for packaging
f_p	objective function for optimization problem for functionality
i	index for i-th component; i=1,...,N
L_{pipe_i}	length of i-th pipe
L_{pipe}	total length of pipes
N	number of components
N_{pipe}	number of pipes
P_{comp}	power consumed by component
Ti	Tower i; i=1,2
$TiCAC$	charge air cooler in tower i
$TiFan$	fan in tower i
$TiPj$	pump j in tower i
$TiRj$	radiator j in tower i
$Vol(C_i)$	volume of C_i

Running Example

$\dot{m}$	mass flow rate
$\dot{V}$	volumetric flow rate
A	area
a, b, c	pressure drop coefficients
C	heat capacity rate
C_f	friction coefficient
C_p	specific heat
C_r	the ratio of minimum to maximum fluid heat capacity rate (C_{min}/C_{max})
d	diameter
f	friction factor
H	height
h	convection heat transfer coefficient
I	electric current
K_{loss}	loss coefficient
L	length
N	number of revolutions per minute (rpm)
NTU	number of transfer units
p	pressure
q	heat generation rate per heat transfer rate
Re	Reynolds number
T	temperature
t	thickness
U	overall heat transfer coefficient
V	voltage
α	scale factor
ϵ	effectiveness of heat exchanger

η	efficiency
μ	dynamic viscosity
ω	angular velocity
ρ	density
σ	Stefan-Boltzmann constant
τ	torque

Subscripts and Superscripts in Running Example

act	active area
air	air
c	cold
CAC	charge air cooler
cap	capacity
co	coulomb
$comp$	component
$cond$	condenser
$cool$	coolant
eng	engine
ext	external
gen	generator
h	hot
$heat$	heat source
i	input
int	internal
$louv$	louver
min	minimum
mot	motor
o	output

oil	oil cooler
pb	power bus
r	ratio
rad	radiator
ref	reference
T/S	thermostat
tc	turbo charger

ABSTRACT

Optimal System Design with Geometric Considerations

by

Kwang Jae Lee

Co-Chairs: Panos Y. Papalambros and Michael Kokkolaras

System design is tied to both functionality and geometric realization. The former is pertinent to system performance, and the latter is related to packaging. Packaging is an optimization process that finds a desirable placement for the system components within a given space. When the components do not fit into the allocated space at the packaging stage, the design engineers must make modifications that can affect the performance of the system. The modification of a component can also affect the geometry and positions of other components in the system. These changes might lead to an infeasible layout. Therefore, optimizing the system performance considering packaging is desirable.

Packaging problems and solution methods have been studied in many applications, such as electrical circuit layout, glass or metal cutting, truck loading, trunk packing, rapid prototyping (RP), architectural floor plan layout, routing, and mechanical component layout. Packaging problems in a mechanical system design are more challenging than 2D applications such as circuit layout and the metal cutting problem; this is due to a larger design space and increased complexity of geometry. Complex 3D geometry leads to increased computational time for interference checking, which is inevitable for finding a feasible layout. Detailed 3D CAD models, however, are not required or not available at the preliminary design stage. Therefore, abstract representation of the components is necessary during the layout process. Abstract models should balance accuracy of geometry representation and rapid computation capturing designers intent.

This dissertation presents a computational environment for addressing the combined packaging and optimal system design. The packaging problem also includes pipe generation because pipe routing is also important problems in mechanical system design. The simulation model of a thermal management system for heavy duty series hybrid electric vehicles is used to demonstrate the usefulness of the proposed framework.

CHAPTER I

Introduction

1.1 Motivation

Increasing market demands on smaller and more compact products with the same, often even better, performance have been making system deign more challenging. To meet those market requirements, a system that is tied to both functionality and geometric realization as depicted in Figure 1.1 is designed. The former is pertinent to system performance, and the latter is related to packaging.

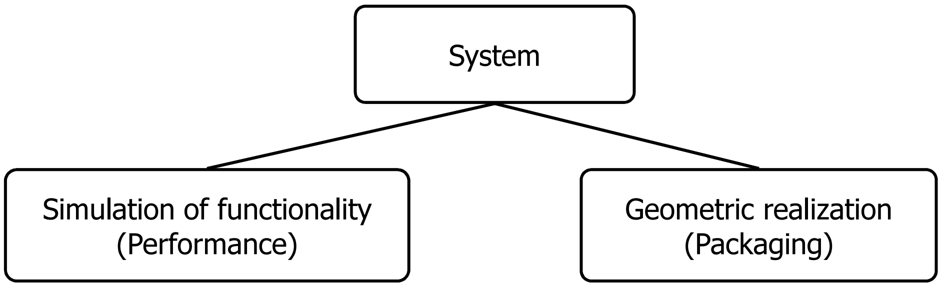

System
Simulation of functionality (Performance)
Geometric realization (Packaging)

Figure 1.1: System design

Designing a system to meet system performance requirements is a very fundamental and important engineering activity from conceptual design to detailed design of each component. Optimization models for system performance have been developed to attempt to find the best design while satisfying the requirements. These models are generally more than one for a system because a system has many disciplines. In those system design models, avoiding interference among the components of a system is crucial because interference makes the system design infeasible, so constraints are generally included in the form of either simple upper and lower bounds or analytic equations for some dimensions. Those constraints, however, are not generally formulated for all possible contacts, which lead sometimes to an invalid design at the end. Also, other components that exists in a system but are not considered in a specific performance optimization model are sometimes ignored during interference checking. The later the problem is found during the product development process, the higher the cost to fix it will be.

Packaging, the other aspect of system design, is an optimization process that finds a desirable placement for the system components within a given space such that a set of objectives is optimized, while satisfying spatial constraints. In the literature, packaging is also referred to as packing, layout design, configuration design, and spatial engineering. Packaging problems have been studied in many applications, such as electrical circuit layout, glass or metal cutting, truck loading [31], trunk packing [18, 21, 74], rapid prototyping (RP) [3, 37–40], architectural floorplan layout [55], routing [36, 64, 69, 71], and mechanical component layout. Packaging problems in mechanical system design are more challenging than 2D applications such as circuit layout and the metal cutting problem; this is due to a larger design space and increased complexity of geometry. Complex 3D geometry leads to increased computational time for interference checking, which is inevitable for finding a feasible layout. Detailed 3D CAD models, however, are not required or not available at the preliminary design stage. Therefore, abstract representation of the components is necessary during the layout process. These abstract shapes are used to check for interferences between components during the optimization, and are finally replaced with original geometry for more accurate but time-consuming computation. Abstract models should balance accuracy of geometry representation and rapid computation capturing designers' intent.

Packaging problems deals generally with the fixed shapes of components, which are given from the component design. Although the shapes of components are changed during optimization in several research [20, 80], their shapes are not linked with their performance. Although the changing shapes should be taken into account in mechanical packaging problems, one reason why there are no link between the component geometry and the its performance is that because the packaging problem is already a hard problem to solve, the research mainly focuses on developing the methodology to find the better solutions more quickly.

Packaging is a very significant process in the product design. When the components do not fit into the allocated space at the packaging stage, the design engineers must make modifications that can affect the performance of the system. The modification of a component can also affect the geometry and positions of other components in the system. These changes might lead to an infeasible layout. Therefore, optimizing the system performance considering packaging is desirable.

Pipe routing plays an important role in packaging problems because the space that is necessary for pipes' shapes and the additional space for assembly are not negligible. In addition to no overlapping among the components, there should be feasible paths for pipes to be a feasible layout for a system.

1.2 Running Example : Thermal Management System for a Heavy Duty Tracked Series Hybrid Electric Vehicle

This section presents the thermal management system for a heavy duty tracked series hybrid electric vehicle(SHEV) that is used as a running example throughout the dissertation.

1.2.1 Introduction

The simulation model of the thermal management system for a heavy duty SHEV is developed to investigate the thermal responses and power consumptions of the system by Park and Jung [28, 60, 61]. Note that all the figures, tables, and data about the model in this section are adopted from [28, 60, 61].

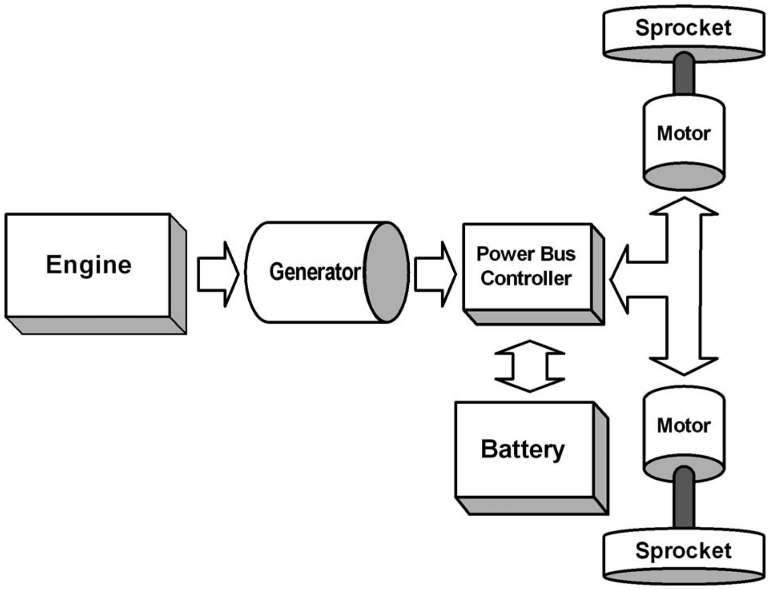


Figure 1.2: Schematic of a series hybrid vehicle propulsion system

Figure 1.2 shows the main components of the SHEV powertrain system modeled in the example. The SHEV powertrain system consists of an internal combustion engine, a generator, a power bus, a high voltage battery pack, and two drive motors.

In SHEVs, all the engine power is converted to electricity and it is stored in the battery or directly used by the motor. The arrows in the figure indicate the directions of power flows that depend on the vehicle driving modes, which are discharging mode, charging mode, and braking mode. In the discharging mode, the battery is the prime power source. When the power demand from vehicle exceeds the battery capacity, the engine is activated to supplement the power demand. In charging mode, the engine/generator is the prime power source. If the State of Charge (SOC) of the battery is lower than the lower limit, the engine supplies additional power to charge the battery. Once the power demand from the vehicle is determined by the controller, the engine is operated at the most efficient operating point to maximize the fuel economy. In braking mode, regenerative braking is activated to absorb the braking power. If the braking power required by the vehicle is larger than the capacity of the motor or the battery, friction braking is used. This power flow between the powertrain components is managed by the power bus, which includes the inverter and the voltage-boosting converter.

Although the SHEV is an attractive platform for heavy duty military ground vehicles compared with conventional propulsion systems because it offers several advantages such as improved fuel economy, better acceleration performance, low acoustic signature, and exportable electric power, SHEVs need additional components such as a generator, drive motors, a large battery pack, and a power bus, all of which require proper thermal management. Therefore, dedicated cooling circuits for the hybrid components are required due to considerable heat rejections and different cooling requirements of the components.

1.2.2 Vehicle Simulation

The cooling system simulation requires component operating conditions as a function of time to simulate the thermal response of the cooling system when the vehicle is driven over a driving cycle. Based on the configuration of vehicle components and the power management modes, a vehicle model with a SHEV propulsion system is configured employing vehicle-engine simulation (VESIM), which was previously developed at the Automotive Research Center (ARC) at the University of Michigan and the model is used to acquire the operating conditions of powertrain components of the SHEV. The acquired operating conditions of components are provided as input data to cooling system simulation.

The specifications of the virtual SHEV simulated in the example are summarized

Table 1.1: Specification of the selected SHEV

Component	Type	Specification
Vehicle	Tracked SHEV	20 ton
Engine	Turbocharged diesel	300 kW
Generator	Permanent magnetic	300 kW
Motor	ac induction	2 * 150 kW
Battery	Lead-acid	18 Ah/120 modules
Maximum speed	(Governed)	72 km/h

in Table 1.1. A turbocharged diesel engine is chosen as the power source due to better efficiency against the spark ignition engine and lower cost against the gas turbine. The rated engine power is determined based on the power (kW) to weight (ton) ratio of 15. Generator and motor capacities are determined to convert all the power supplied by the engine to electricity. Two alternating current (ac) induction type electric motors are used to drive two separate tracks of the vehicle and a lead-acid battery is selected. The maximum vehicle speed is limited by the track dynamics and durability, and the speed is assumed to be governed at 72 km/h, which is the typical maximum speed of compatible tracked vehicles.

The capacity of a cooling system should be enough to remove all of the heat generated by the heat sources under extreme operating conditions. Three conditions are evaluated and the sizes and capacities of the cooling system components are determined to meet the cooling performance requirements under the most severe condition. The three severe conditions are grade load, maximum speed, and off-road conditions, which are listed in Table 1.2. The vehicle simulation results shows that the grade load condition is the most severe condition for the cooling system, so the grade load condition is used for cooling system design and evaluation.

Table 1.2: Driving conditions

Condition	Grade load	Max. speed	Off-road
Vehicle speed (km/h)	48	72	48
Road profile	7%(uphill)	flat	Figure 1.3
Ambient temp.(°C)	40	40	40

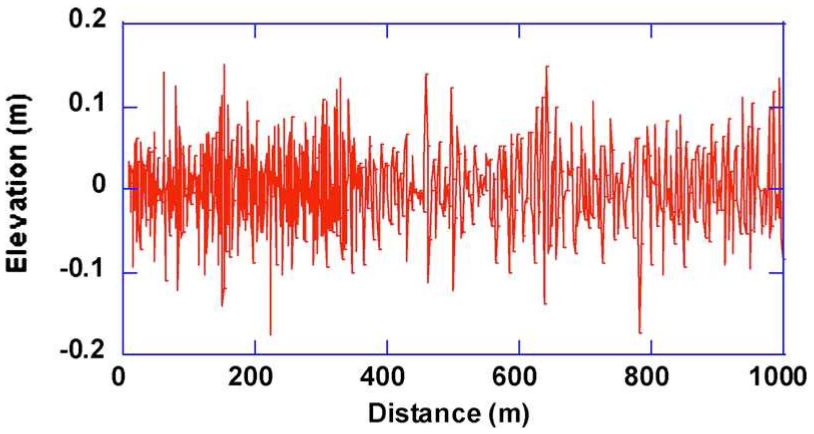


Figure 1.3: Off-road profile

1.2.3 Cooling System Modeling

Cooling system modeling is twofold: development of component models and design of system architectures. A cooling system has many components such as coolant pumps, fans, radiators, thermostats, and heat sources. Each component is modeled to predict its thermal response and/or power consumption.

1.2.3.1 Component Modeling

Thermodynamics-based component models are developed and then integrated into cooling system architectures. The components can be categorized into three groups, depending on the function in the cooling system: heat source, heat sink, and media delivery components. Each component model has the submodels of heat transfer, pressure drop, flow rate, and heat generation.

Heat Source Components

Internal combustion engine, electric generator, drive motors, and power bus are the main heat source components that are considered in the example. Lumped thermal mass model is used for the temperature calculation of all heat source components and the temperature of each component is calculated from the balance of heat generation by the component, heat transfer to the coolant, and heat transfer to the ambient. The heat transfer to the ambient includes convection and radiation heat transfer modes. Each model is summarized and described in Table 1.3.

Heat generation of the engine is modeled with a look-up table type module. Engine heat rejection rate and brake specific fuel consumption data are measured as a function of engine speed and load from the engine dynamometer test and used to build the engine module. Heat generation by the generator or motor is calculated from the efficiencies of the generator and motor. The efficiency lookup tables of the generator and motor are adopted from the library of ADVISOR, which is a vehicle simulation package.

The heat generated by the power bus is calculated based on the power delivered to the electric components through the power bus and the power bus efficiency, as summarized in Table 1.3. The power delivered through the power bus depends on the power management mode. In normal mode, all of the power from the generator is supplied to the motors through the power bus to propel the vehicle. In charging mode, the power supplied by the engine is consumed both by the motors and battery. Thus, the power used to recharge the battery and the power supplied to the motors are the total power delivered through the power bus. In braking mode, the power regeneration by the motors is the total power delivered through the power bus.

Each heat source component has its pressure drop model and the pressure drop across the heat source component is used to calculate the coolant flow rate and power consumption of the coolant pump. Experimental correlation is used for coolant pressure drop across the engine. The coolant pressure drop across the electric component is calculated by assuming that the coolant path in the component is a smooth pipe with an equivalent length [76].

Table 1.3: Model summary of heat source components

Component	Heat generation model	Transient thermal model	Pressure drop model
Engine	Map-based performance model $q_{eng} = f(N_{eng}, \tau_{eng})$	Lumped thermal mass model $\frac{dT_{comp}}{dt} = \frac{q_{comp}-q_{int}-q_{ext}}{\rho C_p}$ $q_{int} = (hA)_{int}(T_{comp} - T_{cool})$ $q_{ext} = (hA)_{ext}(T_{comp} - T_{ext}) + \sigma A_{ext}(T_{comp}^4 - T_{ext}^4)$	Experimental correlation: $\Delta p = a\dot{V} + b\dot{V}^c$
Generator	$q_{gen} = \tau_{gen} \times \omega_{gen}(1 - \eta_{gen})$		Flow in smooth pipe [76] Laminar: $\Delta p = \frac{128\mu L\dot{V}}{\pi d^4}$ Turbulent: $\Delta p = 0.241L\rho^{3/4}\mu^{1/4}d^{-4.75}\dot{V}^{1.75}$
Power bus	Battery is charged and motor is propelling: $q_{pb} = (\frac{1-\eta_{pb}}{\eta_{pb}})(\lvert VI\rvert + \frac{\tau_{mot}\times\omega_{mot}}{\eta_{mot}})$ Motor is propelling: $q_{pb} = (\frac{1-\eta_{pb}}{\eta_{pb}})(\frac{\tau_{mot}\times\omega_{mot}}{\eta_{mot}})$ Motor is generating: $q_{pb} = (1-\eta_{pb})(\eta_{mot}\times\tau_{mot}\times\omega_{mot})$		
Motor	Motor is propelling: $q_{mot} = \tau_{mot} \times \omega_{mot}(\frac{1}{\eta_{mot}} - 1)$ Motor is generating: $q_{mot} = \lvert\tau_{mot} \times \omega_{mot}\rvert(1 - \eta_{mot})$		

Heat Sink Components

Heat sink components are heat exchangers that reject heat to the ambient air. The thermal resistance concept based on the two-dimensional finite difference method (FDM) developed by Jung and Assanis [43] is used for the modeling of the radiator. The radiator core is divided into small control volumes along the tube to take into account the significant air temperature change as well as the local variations in the properties and heat transfer coefficient. Discretized forms of the mass and energy conservation equations are derived for a two-dimensional staggered grid system. The overall heat transfer coefficient in the energy conservation equation is derived using the thermal resistance concept.

The same modeling technique is also used for the charge air cooler. The only difference between a charge air cooler and a radiator is that heat is transferred from the compressed charge air to the coolant in a charge air cooler while heat is transferred from the coolant to the cooling air in a radiator.

An air conditioning (AC) condenser rejects the heat from the passenger compartment to the cooling air. A heat addition model is used for the condenser of the AC system. The heat rejection rate from the AC condenser is assumed to be constant.

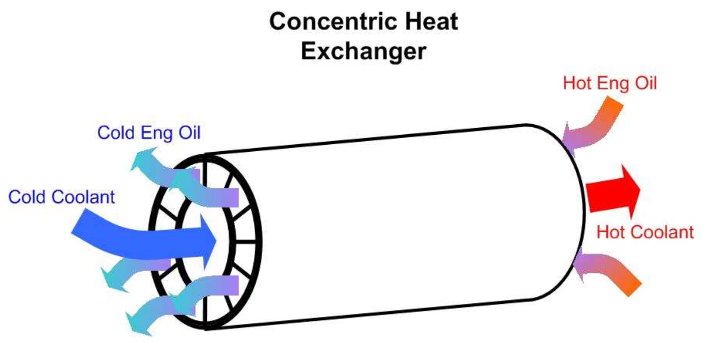


Figure 1.4: Schematic of oil cooler

The oil cooling system has an oil circuit including an oil pump and a heat exchanger between oil and coolant. Effectiveness-number of transfer units (effectiveness-NTU) method [41] is employed for the oil cooler and a performance databased model is employed for the oil pump. Figure 1.4 shows the schematic of the oil cooler.

The models for heat sink components are summarized in Table 1.4.

Media Delivery Components

The function of a media delivery component is delivering and controlling the heat transfer fluids such as coolant and cooling air. Media delivery components include coolant pump, cooling fan, and thermostat.

The coolant pump model calculates the coolant flow rate based on the pump operating speed and the total pressure drop along the cooling circuit. The coolant flow rate is calculated with the pump performance map, which consists of flow rate, pressure rise, and pump speed. A conventional cooling system with a mechanical pump is used for the engine cooling circuit. In the cooling circuits of electric heat sources, pumps and fans driven by electric motors control the component temperature by managing the motor speeds. The benefits of the controllable electric pump over

Table 1.4: Model summary of heat sink components

Component		Heat generation model	Heat transfer model	Pressure drop model
Radiator		N/A	Thermal resistance concept 2D FDM [43]	Water side: $C_f = (0.79 \ln Re_D - 1.64)^{-2}$ Air side: $C_f = 11.9 Re^{-0.39} (\frac{H_{louv}}{H_{fin}})^{0.33} * (\frac{t_{louv}}{H_{fin}})^{1.1} H_{fin}^{0.46}$
Condenser		Heat from A/C module is assumed to be constant	Heat addition model	
Charge air cooler		N/A	Thermal resistance concept 2D FDM	
Oil Cooler	Heat source	Map-based performance model $q_{oc} = f(N_{eng}, \tau_{eng})$	Heat addition model	
	Heat exchanger	N/A	Heat exchanger model (effectiveness-NTU method) [41] $NTU \equiv \frac{UA}{C_{min}}$ $\epsilon \equiv \frac{1-\exp(NTU(C_r-1))}{1-C_r \exp(NTU(C_r-1))}$ $q = \epsilon C_{min}(T_{h,i} - T_{c,i})$	Flow in smooth pipe Laminar: $\Delta p = \frac{128 \mu L \dot{V}}{\pi d^4}$ Turbulent: $\Delta p = 0.241 L \rho^{3/4} \mu^{1/4} d^{-4.75} \dot{V}^{1.75}$
	Oil pump (flow rate model)	Performance data-based model $\dot{V} = f(N_{pump}, \Delta P_{pump})$ $\Delta P_{pump} = \Delta P_{heat} + \Delta P_{bypass} = \Delta P_{heat} + \Delta P_{rad}$		

the mechanical pump were studied by Cho *et al.* [15] in a cooling system of a medium duty diesel engine.

The thermostat in the engine cooling circuit is a three way valve, which prevents overcooling by channeling the coolant to the radiator or to the bypass circuit. The valve opening is determined by the temperature and hysteresis characteristics of the thermostat. The thermostat temperature is calculated by a lumped thermal mass model. The coolant flow rates to the bypass circuit and radiator circuit are determined at the point where the pressure drops of two circuits are equal to each other.

The cooling fan model is similar to the pump model. The cooling fan model calculates the cooling air flow rate based on the fan speed and total pressure drop across radiators. To calculate the cooling air flow rate, a performance map that consists of flow rate, pressure rise, and fan speed is also used.

The summary of the models are shown in Table 1.5.

Table 1.5: Model summary of media delivery components

Component	Flow rate model	Transient thermal model	Pressure drop model
Pump	Performance data-based model $\dot{V} = f(N_{pump}, \Delta P_{pump})$ $\Delta P_{pump} = \Delta P_{heat} + \Delta P_{bypass} = \Delta P_{heat} + \Delta P_{rad}$	N/A	N/A
Cooling fan	Performance data-based model $\dot{V} = f(N_{fan}, \Delta P_{fan})$ $\Delta P_{fan} = \Delta P_{grill} + \Delta P_{cond} + \Delta P_{rad1} + \Delta P_{rad2}$	N/A	N/A
Thermostat	Modeled by a pair of valves $\Delta P_{rad} = \Delta P_{bypass}$ $\dot{V}_{total} = \dot{V}_{bypass} + \dot{V}_{rad}$	Lumped thermal mass model	$\Delta P_{circuit\_rad} = \Delta P_{pipe\_rad} + \Delta P_{T/S\_valve} + \Delta P_{rad} = f_{rad}\frac{L_{rad}}{d_{rad}}\frac{\rho \dot{V}_{rad}^2}{2} + K_{loss}\frac{\rho \dot{V}_{rad}^2}{2} + \Delta P_{rad}$ $\Delta P_{circuit\_bypass} = \Delta P_{pipe\_bypass} + \Delta P_{T/S\_valve} = f_{bypass}\frac{L_{bypass}}{d_{bypass}}\frac{\rho \dot{V}_{bypass}^2}{2} + K_{loss}\frac{\rho \dot{V}_{bypass}^2}{2}$

1.2.3.2 Cooling system architectures

Compared with conventional vehicles, SHEVs have additional heat source components with different operating temperatures. Furthermore, the operations of heat source components are not synchronized due to complicated power flow and power management modes. To satisfy the various requirements of the additional components, the cooling system requires more sensors, controllers, and cooling circuits. Multiple fans may be used combined with multiple cooling circuits. Therefore, the architecture of a SHEV cooling system should be carefully designed to ensure efficient operations of pumps and fans. The battery pack is assumed to be cooled by the compartment AC system due to its low operating temperature and the heat from the battery pack is considered to be dissipated through the AC condenser.

The component models are integrated into three architectures and they are evaluated with different driving scenarios [28, 61]. The simulation results says that architecture C performs better than other two candidate architectures do in term of the power consumption and the temperature fluctuations of the components.

Architecture C, shown in Figure 1.5, is created based on the consideration of the power management modes of the SHEV. In contrast with the conventional cooling system, the heat source components in the SHEV do not always operate simultaneously because they operate independently depending on the driving condition and power management mode. Accordingly, the components do not generate heat simultaneously. When the engine operates, the engine accessories and generator also operate; thus, the heat should be removed from these components simultaneously. However, the operation of the power bus or motors is not synchronized with the engine operation. Therefore, architecture C is configured by grouping the components that work together in one circuit, as illustrated in Figure 1.5. Thus, the engine, generator, charge air cooler, and oil cooler are integrated in one cooling tower and the power bus and motor are integrated into another cooling tower in architecture C. As can be expected, the temperature fluctuations of the components in architecture C are much smaller than those in the other architectures because every electric component has its own cooling circuit.

1.2.4 Performance Requirements

As shown in Table 1.6, each heat source component has its own control target temperature, which is the maximum allowable temperature that should be maintained

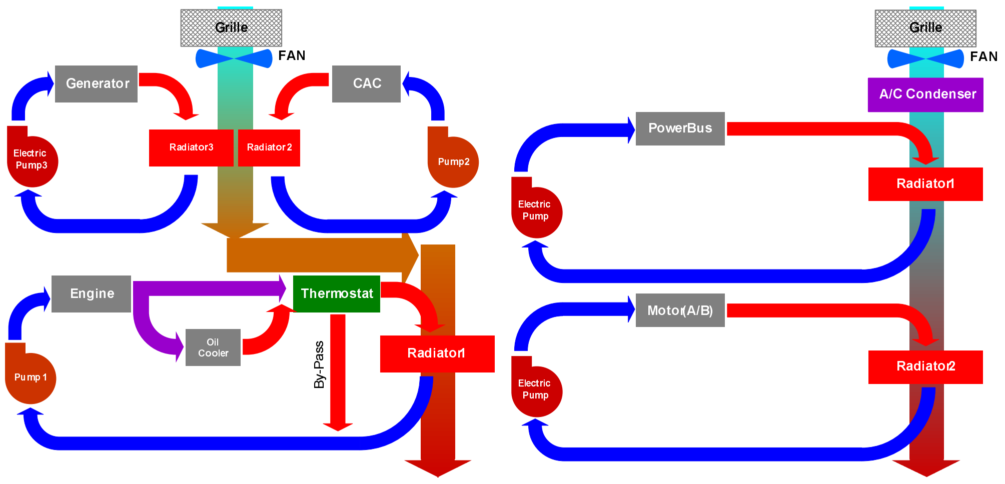


Figure 1.5: Schematic of cooling system architecture C

by the cooling system. As described earlier, the final sizes of the pump and radiator are determined for the grade load condition. The sizes of the pump and radiator are scaled so that the cooling circuit can control the component temperature under the cooling target temperature of the component. The temperature distribution in a heat source including the engine, generator, motors and power bus should be minimized by the cooling system because large temperature distribution can deteriorates the durability of the heat source component. Thus, the coolant temperature change across the heat source component should be limited to minimize the temperature distribution. The coolant temperature change can be controlled by changing the pump and radiator sizes. Larger radiator increases the coolant temperature change, while larger pump decreases the coolant temperature change. Thus, the pump and radiator sizes are tuned for the coolant temperature change not to exceed 10°C.

Table 1.6: Control target temperatures of the components

Component	Control target temperature (°C)
Engine	120
Motor	95
Generator	95
Oil cooler	125
Power bus	70
Battery	45

1.2.5 Geometric Requirements

Cooling system size is basically limited by vehicle dimensions. Among the cooling system components, the radiator is most affected by vehicle dimensions because it occupies a large space to exchange heat with ambient air and should be open to ambient for heat rejection. Thus, taking the specifications of existing vehicles in the equivalent class into consideration, radiator frontal size is limited within a $1.2 \times 0.6m^2$ rectangle.

1.2.6 Baseline Design

Cooling system design has two constraints that need to be satisfied: cooling performance and packaging. Even though a larger cooling system can offer better cooling performance, the cooling system size is limited by the packaging space in a vehicle. Therefore, a cooling system should be carefully designed to satisfy both constraints. Radiator and pump sizes are the main design variables that determine the capacity of a cooling system. A scaling method is developed for the initial estimation of radiator and pump sizes. The preliminary sizes of the pump and the radiator in each cooling circuit are scaled from a referenced cooling system of a conventional vehicle based on the amount of heat generation from a heat source component.

The heat rejection from a radiator is proportional to coolant flow rate and it is also proportional to the product of the radiator frontal area and the temperature difference between the ambient air and coolant

$$q_{rad} \propto \dot{m}_{cool} \tag{1.1}$$

$$q_{rad} \propto A_{rad} \Delta T \tag{1.2}$$

where ΔT is the temperature difference between the coolant and the ambient air.

Therefore,

$$q_{rad} \propto \dot{m}_{cool} A_{rad} \Delta T \tag{1.3}$$

or

$$\dot{m}_{cool} A_{rad} \propto \frac{q_{rad}}{\Delta T} \tag{1.4}$$

In Eq. 1.4, the heat rejection from the radiator (q_{rad}) can be replaced by the heat generated by the component (q_{comp}) in the cooling circuit because all of the heat generated by the component should be rejected at the radiator. Therefore,

$$\dot{m}_{cool} A_{rad} \propto \frac{q_{comp}}{\Delta T} \tag{1.5}$$

Assuming that the coolant flow rate is proportional to the pump capacity, the pump capacity and the radiator size are scaled from referenced pump capacity and radiator based on the following scale ratio:

$$\dot{m}_{pump\_cap,ref} A_{rad,ref} : \dot{m}_{pump\_cap} A_{rad} = \frac{q_{comp}}{\Delta T}_{ref} : \frac{q_{comp}}{\Delta T} \tag{1.6}$$

The control target temperature of the heat source component in the cooling circuit is used as the coolant temperature when calculating the temperature difference (ΔT) in Eq. 1.6 because the coolant temperature is limited by the control target temperature. To estimate the sizes of pump and radiator, the same scaling factor (α) for the pump and the radiator is used

$$\dot{m}_{pump\_cap} = \alpha \dot{m}_{pump\_cap,ref}, A_{rad} = \alpha A_{rad,ref} \tag{1.7}$$

From Eqs. 1.6 and 1.7, the scaling factor can be found as

$$\alpha^2 = (\frac{q_{comp}}{\Delta T})/(\frac{q_{comp}}{\Delta T})_{ref} \tag{1.8}$$

Once the scaling factor is estimated using Eq. 1.8, the size of the scaled radiator is examined whether it is within the packaging constraints. If the radiator size is out of the range, the radiator size is reduced down to the limit and the pump size is rescaled based on the scale ratio of Eq. 1.6. After the preliminary sizing is completed as explained above, final sizes of pumps and radiators are refined until all component temperatures can be controlled lower than their control target temperatures.

1.3 Dissertation Objectives

This dissertation seeks to develop the framework that integrates geometric and packaging considerations with system functionality considerations. The optimization problem is formulated for both problems. This integrated model is much more complex than the original performance model because it includes the additional objective functions such as compactness and serviceability, design variables such as positions and orientations, and constraints including a non-overlapping constraint. Also, other components in a system that are not used in the optimization model might need to be geometrically modeled to compute interference with existing components even though their sizes do not change.

In addition, because implementation is also an important issue to do research more efficiently, one practical goal is to develop an integrated computational environment that can handle geometry as well as performance optimization. This environment includes computational geometry libraries, computer graphics (CG), and optimization algorithms such CFSQP (C code for Feasible Sequential Quadratic Programming), Simulated Annealing (SA), and Genetic Algorithm (GA).

Although pipe routing problems are hard to solve, the generation of pipes and cables is discussed and formulated into a packaging problem. It is selected among the additional necessary space requirements for other purposes such as maintaining, assembly, and piping because it is a very important in mechanical and electrical system design. Given a layout, the method to estimate the feasibility and length of pipes is proposed.

1.4 Dissertation Overview

The subsequent chapters of this dissertation are organized as follows. Background information about packaging and the developed computational environment are presented in Chapter II. Chapter III provides the optimization formulation for the combined packaging and optimal system design and the results. Also, the solutions to the implementation issues for integration are explained. Chapter IV explains a routing problem in a system design. Chapter V concludes with a summary, contributions, and future work.

CHAPTER II

Geometric Realization in System Design

This chapter discuss geometric realization in system design. This discussion includes how to represent the shape of a component in a packaging problem, previous works in 3D packaging problems, and explanation of the computational environment.

2.1 Introduction

When an initial configuration is determined at the early stage of product design. The one of the most important question is: “Do the components fit into the container?”. To answer this question, we first need to understand the question. This question can be translated into a optimization problem that is to find an optimal positions and orientations of components without overlapping within a container. This is mathematically a packing problem, which is known to be NP-hard (non-deterministic polynomial-time hard) in computational complexity theory even with 2D rectangular shapes [77].

According to computational complexity theory, P is a set of problems that can be solved in polynomial time, and NP is a set of problems for which a solution can be verified in polynomial time. NP-hard is a class of problems that are, informally, at least as hard as the hardest problems in NP. Generally, we think of problems that are solvable by polynomial time algorithm as being tractable, or easy, and problems that require superpolynomial time as being intractable, or hard.

Before investigating packing problems, the issues related to component-level realization is discussed.

2.2 Component-level Realization

More powerful engine normally requires more space. Although many efforts to reduce the size and weight are made by engineers, it is generally true that more performance requires more space to be installed in a system.

In system design, components are not necessarily modeled with dimensions depending on what information is required in design problem for functionality. For example, the fan model in the running example calculates the cooling air flow based on the fan speed and pressure drop, which is based on a map. No information about dimensions is available.

At a component level, there are several ways to link performance value with dimensions. If we know that how to design a component mathematically and geometrically, components shapes are designed to meet that performance. This could be an another optimization problem, which is to find dimensions to meet the target performance of a component. The result of this optimization problem is the shape of the component, which can be used for packaging with or without generating abstract geometry. When the model or formulations to design a component are not available, the geometric size of the component would be assumed based on the catalog or available previous design. For some components such as A/C condenser in the running example, constant numbers are enough to measure performance. These components' shapes are important to evaluate geometric performance because they exist in a container and occupy the space even though neither the size changes nor they move.

In the case of dimensions are already design variables in a performance optimization problem, then geometry can directly be created from those dimensions. For example, the dimensions of radiators in the running example are design variables of the performance optimization problem. This imply that geometry exists even when formulating an analysis model. However, this does not mean that all dimensions are used to generate the shape for packaging. For instance, dimensions of fins are not necessary for a packaging purpose.

Sometimes shapes need to be purely guessed when any information is not available. Also, resizing from the existing model is also possible. The existing models means not performance analysis models, but geometric models that are already created in a CAD modeler. The author believes that the most promising method is to build a database for specific application so that we can use the previous models and experiences. For example, a company can build its own database of geometry for packaging purpose when building and archiving the CAD models for their products.

2.2.1 Abstract Model

It is very important to decide how to represent components' geometry because it affects the problem complexity, problem formulation, and computational performance of the optimization. If CAD models are available, these models can be used in the optimization problem because CAD models contains all geometric information. At the conceptual design stage, CAD models may not be available. Also, these models have many detailed geometry that is not necessary for packaging purpose. So, the abstract models need to be built. There are two questions: The first one is how to generate them, and the second is how to represent them. Although the abstract models are simpler shapes than the original ones, their geometric representation still needs to be selected.

In many engineering applications, the difference abstract models from the original CAD model are used for different purpose to reduce the computational time. Figure 2.1 shows three different abstract models for the same part. The original CAD model

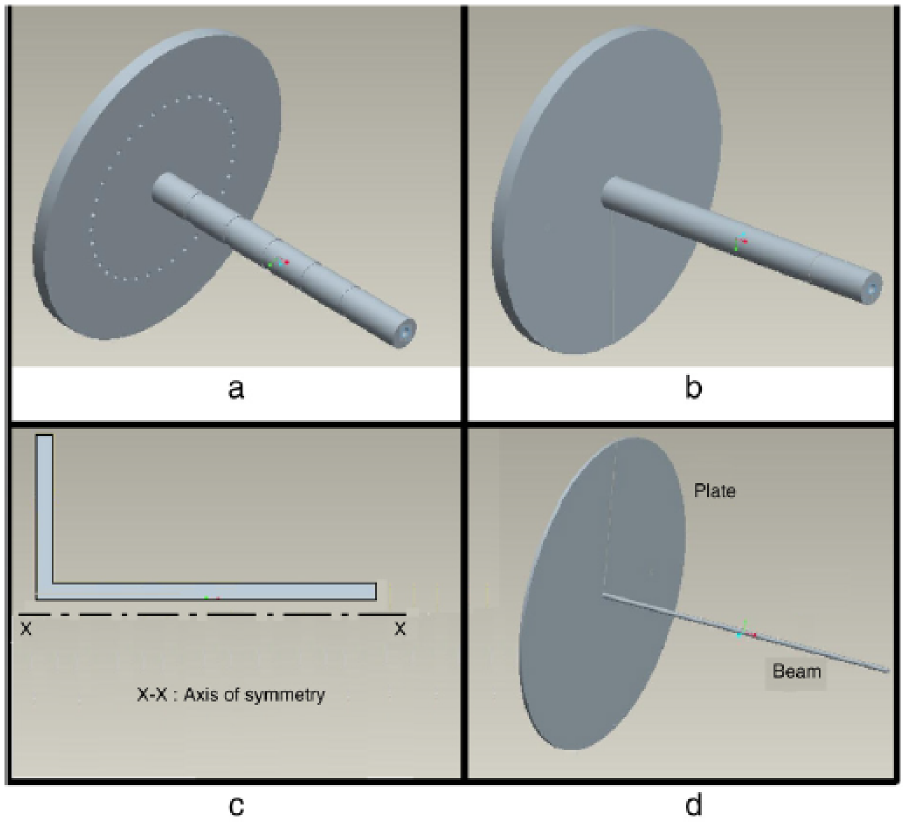


Figure 2.1: Model simplification example reproduced from [73]

of an axisymmetric part with several grooves and holes is shown in Figure 2.1(a). Figure 2.1(b) is the simplified model after removing notches and tiny holes, which can be used for an application like rigid body simulation where small holes and grooves play a negligible role in determining the inertia tensor and the collision contact points. Figure 2.1(c) is the another simplified model after removing notches and tiny holes and dimension reduction and exploiting symmetry of the part, which can be used for an application such as thermal analysis. Figure 2.1(d) the simplified part composed of a beam and a plate element which can be used in structural analysis. All these simplified instances reduce the computational time significantly while affecting the respective simulation results negligibly as compared to the full solid model. Unfortunately these simplification cannot be fully automated, and requires knowledge and experience.

The models in Figure 2.1 are all for the simulations. Components' geometry for packaging should be different from the original shapes even when the original shapes are already abstract. The shapes for packaging do not need the small holes or detailed geometry as shown in Figure 2.1(b); however, removing the detailed geometry would not be enough for packaging. Additional geometry would be required because of vibration, assembly, and other purpose. These geometry can be modeled and added to the abstract model.

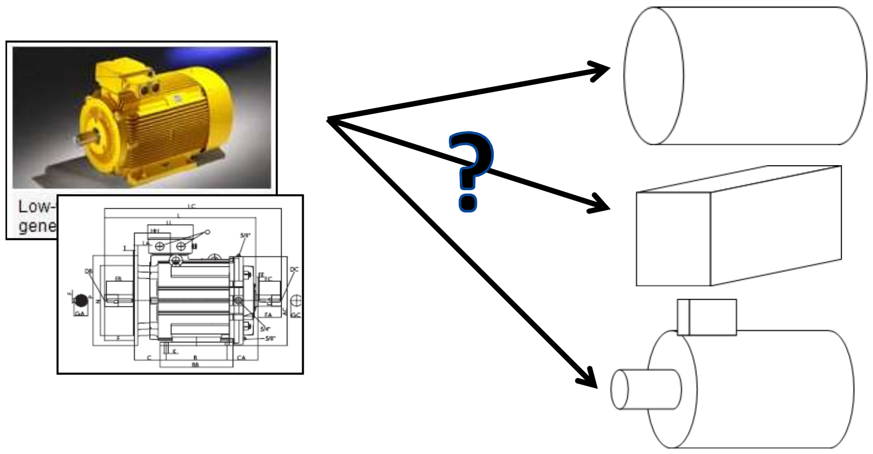


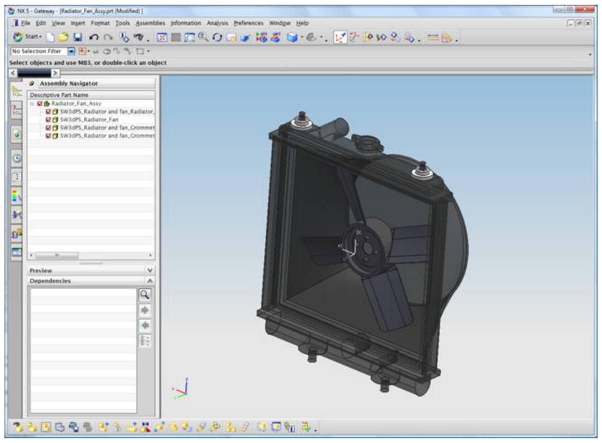
Figure 2.2: Abstract representation example

Figure 2.2 shows that why building abstract model from the original shape cannot be automated. Knowledge in application is required to extract key parameters and define the abstract geometry. This cannot be done automatically, but can be semi-automatic by selecting the pre-defined shapes from the database for abstract geometry.

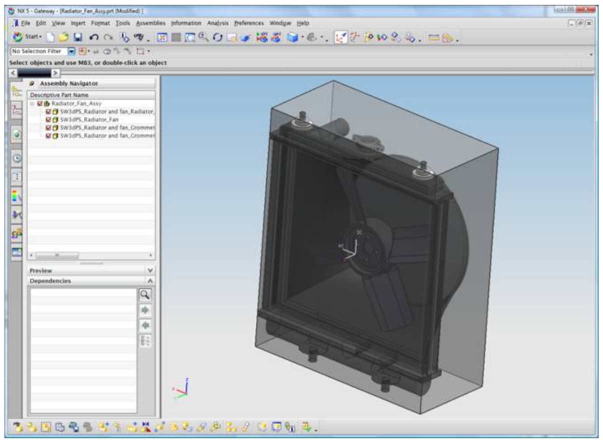
Building another model for packaging either automatically or manually seems to be the way to create and store the abstract geometry. Instead of saving in the separate files, creating one more solid body in the same file is suggested so that when the original shapes change, the abstract model also can be updated accordingly.

2.2.2 Bounding Volumes

As explained in the previous section, detailed geometry is not necessary for a layout design. The first possible abstract model would be the bounding volume as an envelope. As shown in Figure 2.3, the envelope of an assembly, as well as that of a single component, can be created.



(a) Radiator assembly



(b) Created envelope

Figure 2.3: An example of the envelope of an assembly

This envelope is the one of the bounding volumes, which is axis-aligned bounding box(AABB). Figure 2.4 shows five of the most common bounding volume types:sphere, axis-aligned bounding box(AABB), oriented bounding box(OBB), eight-direction discrete orientation polytope(8-DOP), and convex hull. Note that all the bounding volumes are convex.

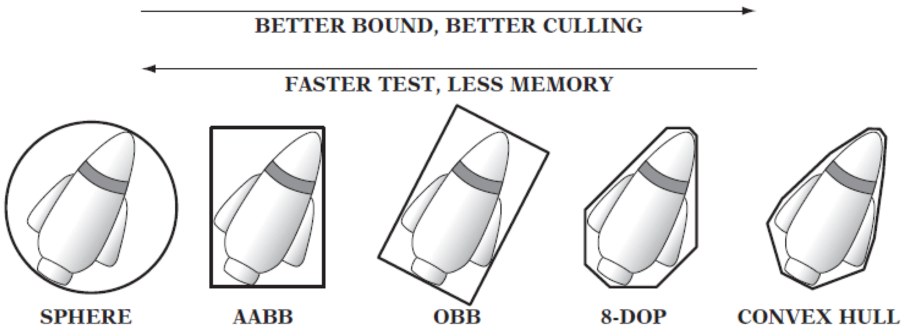


Figure 2.4: Types of bounding volumes reproduced from [22]

Bounding volumes are sometimes good abstract models, and very efficient to compute interference. For concave parts, however, the bounding volumes are not sufficient, which can loose the space that can be useful for packaging or piping.

In this dissertation, the automation generation of AABB and OBB is implemented. This functionality can be useful for some components, but the shapes are not restricted to those bounding volumes in this dissertation.

2.2.3 Convex and Concave Shapes

A set $S \subseteq \Re^n$ is convex if, for every point $\mathbf{x}_1$,$\mathbf{x}_2$ in S, the point

$$\mathbf{x}(\lambda) = \lambda \mathbf{x}_2 + (1 - \lambda)\mathbf{x}_1, 0 \leq \lambda \leq 1 \tag{2.1}$$

belongs also to the set. If a set is not convex, it is concave.

Handling convex shapes has advantages in computation.

- Separation by a hyper plane
- Local optimal for minimum distance is a global optimum.

The key idea behind bounding volumes is to precede expensive geometric tests with less expensive tests that allow the test to exit early. Because of these good properties, some collision detection algorithms are allowed only convex shapes as inputs. Bounding volumes in Section 2.2.2 are all convex shapes, which is not coincident.

When we have concave shapes in the problem, partitioning the concave shape into convex shapes could be considerable. The problem of partitioning a nonconvex polyhedron into a minimal number of convex pieces is known to be NP-hard [13].

Although convex shapes are desirable, assuming all geometry in a problem are convex looks impractical. A container can always be concave. In this research, the components shapes are not assumed as convex although it is preferred.

2.2.4 Parametric and Non-parametric models

This is not the case where shapes are complex or not. The original and/or abstract shapes can be parametric or non-parametric. If the original model does not change in size, non-parametric model, such as mesh model, can be used as long as a CAD modeler can handle, which means a CAD modeler can import the non-parametric model data into its own geometry database. Mesh data is good approximation and would be good for interference check in terms of computation time, but modification of geometry is not easy. Since the sizes of the components inevitably change in this research, parametric models are chosen. Even though parametric model is used, not all dimensions are used to build an abstract model for packaging. For instance, a diameter for a hole that is used for performance computation can be unnecessary for packaging purpose. Also, distances among components are sometimes design variables of a system optimization problem depending on the objective function. For instance, in an optimization problem that uses CFD analysis, a system optimum is obtained by varying distances among components. These design variables can directly be used in a packaging problem as additional constraints or constant parameters. In any cases, parametric models are preferred when optimization problems for functionality as well as packaging problems are considered together.

2.2.5 Geometric Representation

Geometric representation is closely related to the interference check. All possible geometric representations are not covered here, such as implicit solid modeling, constructive solid geometry (CGS), and subdivision surface because these representations are not generally used in the recent packaging research. Details regarding each representation can be found in [50].

- **Boundary representation.** A boundary representation (B-Rep) expresses a solid object by its boundary surfaces, which are geometric entities such as vertices, edges, and faces. This B-Rep is the most popular representation of 3D geometry in a commercial solid modeling system. The models are parameterized, so the size of the object can easily be changed by modifying the dimensions of the solid model. Research directly utilizing the commercial CAD system employs this representation [20, 29, 47, 51, 52, 54, 63, 67].

- **Triangle mesh.** In a mesh model, the faces are usually triangles, quadrilaterals, or other simple convex polygons. However, triangles are most commonly used. This representation can be viewed as a special case of B-Rep, of which surfaces are polygons. In packaging, most research makes use of STL file format that contains the coordinates of the triangle vertices with normal vectors because this file format is supported by many other software packages [19, 24, 30, 32, 37–40, 64]. Employing this representation, we can make use of the efficient collision detection algorithm developed in the computer graphics area, but cannot easily change the component size due to lack of parametric information. Additionally, generating STL files from the CAD model should be taken into consideration in the computation time when components' shapes are changing during the optimization.

- **Voxel.** The voxel, also known as exhaustive enumeration [50], represents a solid model by the collection of small cubes. These cubes are called voxels. This method is used because computation of interference check is not expensive [21, 74]. However, more memory is required to represent a solid model more accurately. Also, this approach is efficient as long as the rotation angle of the objects is restricted to multiples of 90 degrees. Otherwise, generating a voxel model is required at each iteration.

- **Octree.** The octree representation uses a recursive subdivision of the space of interest into eight octants, and utilizes memory more efficiently than the voxel model. Many researchers implemented this representation in their work [3, 4, 11, 18, 45, 78, 80, 82, 83] because of the capability of controlling the level of details by adjusting the precision level and the efficiency in the interference check.

If geometric shapes are limited to 3D primitives such as spheres, boxes, and cylinders, parametric representations are basically used because in addition to position and orientation of the components, the shapes are mathematically defined with a few additional dimensions. Although other representation such as triangular mesh can be used for these shapes, parametric representation is normally used as long as all the shapes of components are limited to those shapes.

The shapes are not restricted to 3D primitives in this research, and B-Rep representation is chosen because the size of a component should be easily changed.

2.3 System-level Realization

System-level realization corresponds to a packaging problem. As explained, it is a NP-hard problem, so many heuristics have been developed.

2.3.1 Packaging Problem

Packaging problems can be divided into two categories by the objective function used in the problem. In the first category, compactness or packing efficiency is a major criterion; however, this is not the case in the second category. Problems in the first category are also referred as "compact packing" [74] or "pure packing" [17]. Figure 2.5 shows a typical example of compact packing. Compactness is generally measured as the ratio between the sum of the volumes of all objects and the volume of the convex hull or the bounding box enclosing all objects. Aladahalli *et al.* uses

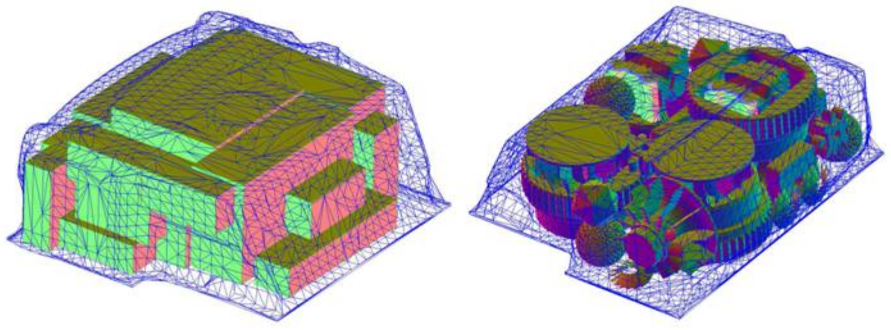


Figure 2.5: An example of compact packing reproduced from [74]

the minimum height as the objective function for RP applications [3]. Dickinson and Knopf proposed an alternative moment based metric for 2D and 3D packing, termed the point moment metric [17].

In most mechanical packaging problems, additional objectives such as survivability, maintainability, routing costs, and specific position of center of gravity and additional geometric and performance constraints are considered and evaluated. This research is to handle a general layout problem, not "compact packing".

2.3.2 Non-overlapping Constraint

Interference detection is a well-understood problem in the field of robotics and computer graphics [45]. The computation cost of objective functions and other constraints differs from problem to problem; however, evaluating the interference between components is usually expensive. Moreover, this routine is indispensable in packaging problems to obtain feasible results. Therefore, researchers have chosen the geometric representation described in Section 2.2.5 primarily for computation efficiency of interference. Teng *et al.* analytically calculates the interference because objects are simplified as cuboid and cylinder [72].

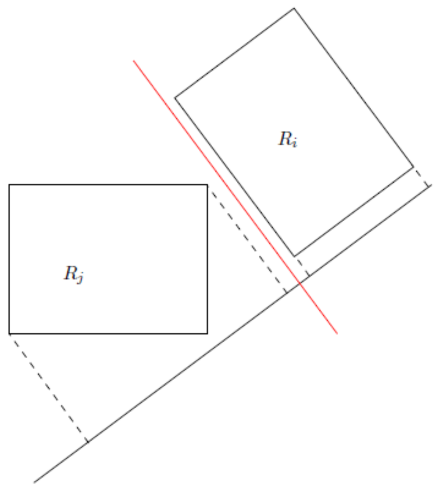


Figure 2.6: Separating axis adapted from [21]

Separate axis theorem is used to determine whether or not two convex objects are intersecting, as shown in Figure 2.6 . By the theorem for convex polyhedra, these normals are either the ones of the two polyhedra or are parallel to the cross product of an edge of the first polyhedron and an edge of the second one or vice versa. In the special case of rectangles, those normals are exactly the directions of the edges of the rectangles. This can reduce the computational time, but this theorem is only applicable for convex shapes. This shows that the appropriate selection of geometry types used in the packaging problem affects the performance of interference check routine.

Types of queries are possible for interference depending on how to model the non-overlapping constraint in the optimization model. First, interference detection or intersection testing problem: answering the Boolean question of whether two objects, A and B, are overlapping at their given positions and orientations. Boolean intersection query is both fast and easy to implement and are therefore commonly used. However, sometimes a boolean results is not enough and the parts intersection must be found. The problem of intersection finding is a more difficult one, involving finding one or more points of contact. If the objects penetrate, some applications require finding the penetration depth. The penetration depth is usually defined in terms of the minimum translational distance: the length of the shortest movement vector that would separate the objects. Computing this movement vectors is a difficult problem, in general. The separation distance between two disjoint objects A and B is defined as the minimum of the distances between points in A and points in B. A more general problem is that of finding the closest points of A and B: a point in A and a point in B given the separation distance between the objects. Note that the closest points ar not necessarily unique; there may be an infinite number of closest points.

2.3.3 Topology Representations

Topology of a layout is the term to describe the different layout depending on the relative positions among components. When the method to represent the layout is possible, we can generate the different layout by manipulating the representation.

Sequence pair is developed to address the compact packing problem in electric industry the layout design of such VLSI systems. Basically, packing problem, or floor plan, is to find the location of rectangular shapes in a rectangular container. Because of these simple shapes, calculation cost is not expensive to check if there is interference. Therefore, exploring the large search space is more important in this area. In addition to sequence pair, there are more methods that can solve the same problem, which is a combinatorial problem, such as O-tree, B-tree [25] or [44].

Figure 2.7 shows the representation of layout using sequence pair. Sequence pair represents a topology of layout with a pair of sequences of all components, Γ_+ and Γ_-.

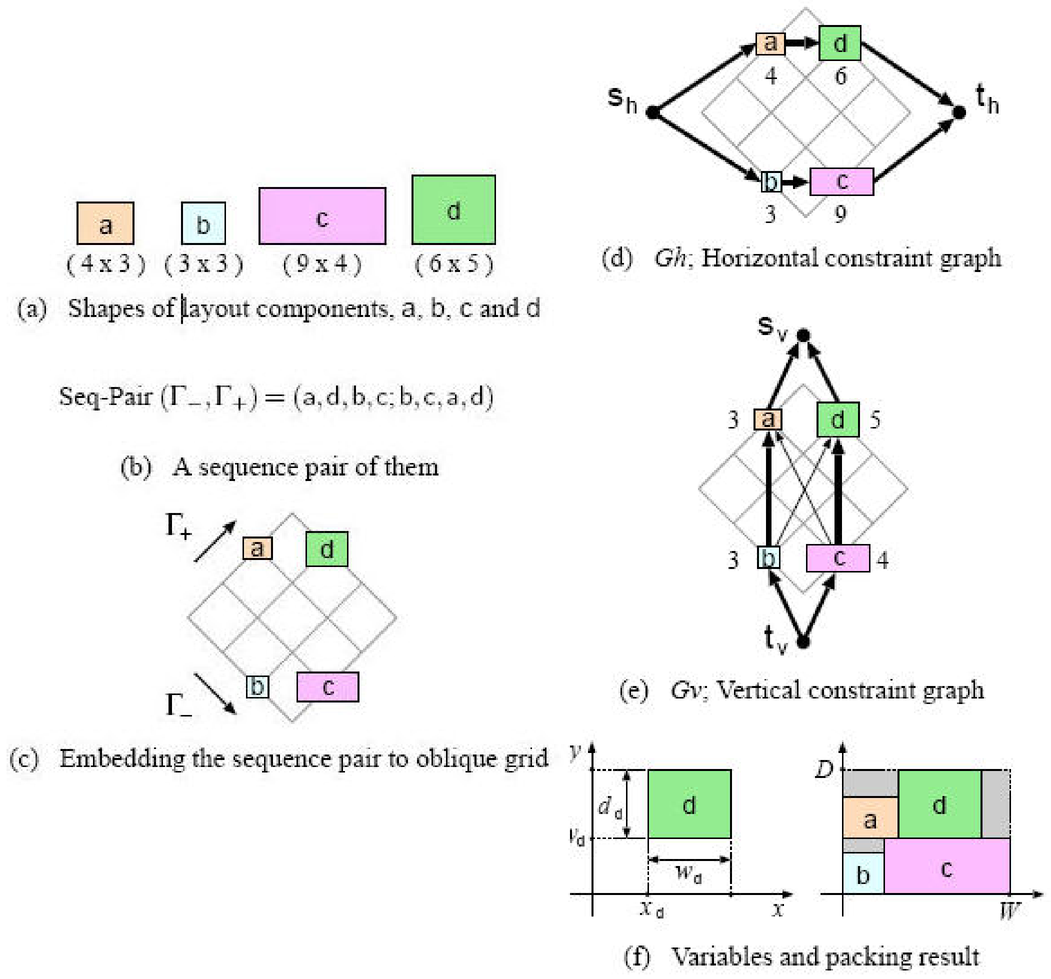

a
b
c
d
(4x3)
(3x3)
(9x4)
(6x5)
(a) Shapes of layout components, a, b, c and d
Seq-Pair (Γ−,Γ+) = (a,d,b,c;b,c,a,d)
(b) A sequence pair of them
Γ+
Γ−
(c) Embedding the sequence pair to oblique grid
sh
th
4
6
3
9
(d) Gh; Horizontal constraint graph
sv
tv
3
5
3
4
(e) Gv; Vertical constraint graph
y
dd
yd
wd
xd
x
D
W
(f) Variables and packing result

Figure 2.7: Layout representation by sequence pair reproduced from [25]

In Figure 2.7 shows the detailed steps from the given $\Gamma_+ = (adbd)$ and $\Gamma_- = (bcad)$ to the layout result, but the important fact is that one sequence pair generates one non-overlapping layout. By permutating the each sequence, the layout of different topology is generated. This concept is currently extended to the rectilinear shapes such as L or T shapes and to the 3D block packing problem. Fujiyoshi *et al.* reviewed the representations for 3D packing [26].

A tree structure is also used to represent the layout topology. Figure 2.8 shows an example of O-tree representation. To encode the O-tree, T and π are needed. A string T identifies the tree branching structure and a permutation π is the labels of the tree nodes. $(T, \pi) = (00110100011011, adbcegf)$ represents the tree structure in

Figure 2.8 (a). By permutating (T, π), different layouts are created. This tree type representation is extended to 3D packing problem [26].

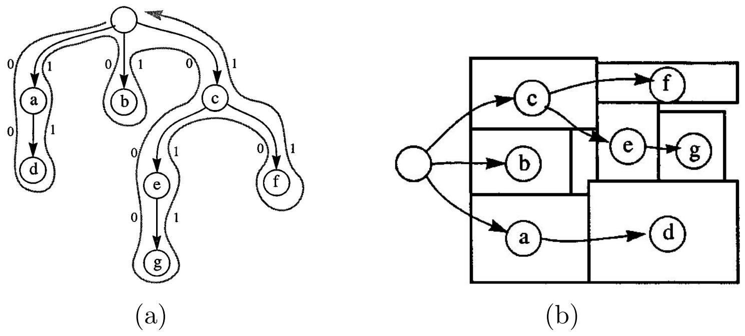


Figure 2.8: O-tree and its corresponding layout reproduced from [34]

This approach can be applied only to compact packing problems, which are not the typical case for a mechanical system. Also, when the components shapes are rectangular in 2D and box in 3D, we can represent the layout with the relative positions of the blocks to be placed into a rectangular container. As mentioned earlier, assuming the shapes as only blocks is not practical. Also, when system performance is being considered, the layout, even initial with very simple geometry, is mostly selected. Initial position and size are mostly chosen

The methods to represent the topology of a layout are surveyed, but to the author's knowledge, these methods cannot be applied to mechanical system design.

2.3.4 Pipe Routing

Layout design of pipes, hoses, and electrical cables in a system is very important because they have their own volumes, which require space to be installed and are not negligible in most cases. So, these additional components, which are generally represented by lines in performance model, should be included in a packaging problem. The details of a pipe routing problem such as characteristic, previous work, algorithms, and formulation in a packaging problems are presented in Chapter IV.

2.3.5 Solution Methods

Extensive research has been done in developing packing algorithms for 2D and 3D problems. Cagan *et al.* (2002) [10] performed a survey of computational approaches to 3D layout problems, and Blouin *et al.* (2004) [7] presented a review of research on configuration design and packaging optimization at Clemson University.

As explained in Chapter I, because packing problems are known to be NP-hard problems, different methods are developed. With problem-specific heuristics, we can probably get close to the optimal answer. Heuristic methods like simulated annealing or greedy approaches can be used to quickly find a solution with no guarantee that it will be the best one.

This section presents solution methods used in the literature focusing on the 3D packaging problem in mechanical engineering.

2.3.5.1 Heuristic Approaches

Heuristic algorithms typically generate acceptable solutions for a specific application, not for a general free-form packaging problem.

Blouin *et al.* presented a hybrid GA for three-dimensional packing optimization where the final number of packed items was unknown [8]. According to this study, the GA is considered 'hybrid' by the fact that the genome is composed of two different parts (combinatorial and non-combinatorial parts), each requiring a different set of specific genetic operators. Packing and orientation were design variables encoded into genomes, and a heuristic algorithm was used to locate the items. Several heuristics were presented, and the issues related to computational efficiency were discussed. The container and all items were modeled as rectangular prisms with discrete 90-degree rotations.

Tiwari *et al.* proposed the algorithm for packing three-dimensional free-form objects inside an arbitrary enclosure to maximize packing efficiency [74]. The design variables consist of two parts: the packing sequence and the orientations. An extension of the original bottom-left-fill (BLF) heuristic to three dimensions is used. The geometry of the container and the objects were provided in the form of STL files, and were voxelized to perform fast collision detections.

2.3.5.2 Gradient-based Algorithm

Landon and Balling solved a 3D container loading problem using a gradient-based method, and calculated gradients of mass properties explicitly [46].

A gradient-based algorithm can generally reduce the computation time to find an optimal solution, but usually converges to the nearest local optimum. The algorithm is sensitive to the initial design and deterministic, so multiple runs with different starting points are necessary for multi-modal problems. When explicit gradients are not available, finite-difference approximations may be used. In spite of several drawbacks, this algorithm is used in the packaging problem in conjunction with other stochastic algorithms [20, 47, 72] such as a genetic algorithm (GA) and simulated annealing (SA) to find the improved solution.

Dong *et al.* showed the gradient-based algorithm performed more efficiently for the local search compared to the genetic algorithm. This research incorporated shape morphing into a vehicle layout problem [20]. A parameterization-based morphing method and a mesh-based morphing method were implemented. In addition, a bi-level optimization formulation was developed and examined. At the system-level, positions and orientations of the components are design variables, while minimizing the given objective function of the layout design problem. At the component level, the shape of the component is changing to minimize the overlap between other components and the container. In addition, a genetic algorithm and a gradient-based algorithm were evaluated for both the system and the component level problem to assess the computational performance.

2.3.5.3 Genetic Algorithm

A GA is both a search algorithm based on natural selection and a stochastic algorithm requiring no gradient information. In a GA, design variables are encoded into chromosomes by a fixed length string. Ikonen *et al.* applied a GA to a RP packing problem with triangle mesh representation (STL format) [38]. In their application, design variables are the packing sequence and the orientation of the components with 45 degree increments around each coordinate axis. Ikonen *et al.* also presented a GA for packing in a RP machine with objects having cavities and holes [39, 40]. In addition to the two design variables in the previous work [38], the five 'attachment points' for each part are specified by a human operator to describe how parts relate to each other.

Grignon *et al.* developed a method to solve 2.5 dimensional packing problems using a GA [31]. The term 2.5 means that the vertical dimension is discretized into several non-overlapping layers. The method was applied to the placement of fixed size rectangular boxes in a rectangular volume to obtain a desired location of the center of gravity. An Order Crossover Operator was used to allow change of the box placing sequence. The rotation of the boxes was also taken into consideration by assigning a rotation bit to each box. A penalty was calculated if boxes completely or partially extended outside the boundary of the container.

Grignon and Fadel addressed the problem of multi-objective three-dimensional free form packaging [30, 32]. Tessellated data representation (STL format) was used for the geometry representation. Population sets, instead of a population of individual points, were employed in a GA. As a result, a Pareto set of a multi-objective problem was obtained by a single run of the algorithm. Three objective functions (compactness, balance, and maintainability) were selected to test the method. This method was applied to both a car engine and a satellite configuration problem.

Miao *et al.* described a configuration optimization method based on a multiple objective GA [51]. The method used in this research is called NSGA-II (Non-dominated Sorting Genetic Algorithm). Two conflicting objectives were considered: vehicle dynamic performance and ground clearance. The vehicle packaging analysis, used to calculate the ground clearance, is performed followed by dynamic performance computation. The commercial CAD kernel, ACIS [1], was utilized to analyze interference among the components. In this study, the effort was focused on obtaining the Pareto set by a single run of the algorithm.

Miao *et al.* continued the work presented in [51] [52]. Three objectives were considered: vehicle dynamic performance, maintainability, and survivability. A swap operator specifically constructed for packaging problems was presented, while a crossover operator was improved. These enhancements were made to show improved GA efficiency when searching for the Pareto set. As presented in [51], the effort focused on obtaining the Pareto set by a single run of the algorithm.

Miao and Fadel proposed a “packing GA” designed specifically to address the packing problems [53]. New encoding methods and GA operators were illustrated. The packing GA was compared to two other GAs on an 8-box packing problem to show the packing GA has a better chance to find the global optimum.

Gantovnik *et al.* and Miao *et al.* applied the packing GA developed in [53] to the FMTV (Family of Medium Tactical Vehicle) case study with three objectives: maintainability, survivability, and vehicle dynamic performance, and two constraints: overlap between components and the ground clearance [29, 54]. This work showed better performance of the packing GA than the traditional binary GA in solving packaging problems.

Sandurkar *et al.* presented the GAPRUS (Genetic Algorithm based Pipe Routing Using .STL files) to solve the pipe routing problem, which in this study minimizes the total length of the pipes and the number of bends, while satisfying the collision free constraint [64]. Due to the GA, a set of solutions, which are the coordinates of bend locations and the number of bends, is generated

Sun and Teng proposed a two stage approach for the layout design of a satellite module employing a GA for the global layout and Ant Colony Optimization (ACO) algorithms for detail [67].

2.3.5.4 Simulated Annealing

SA is a stochastic technique based on the analogy between simulating the annealing process and solving the optimization problems.

Szykman and Cagan introduced a SA to generate a 3D component packing layout [68]. The approach presented in this research includes an adaptive annealing schedule, a component move set, and a dynamic move selection strategy. Objects are limited to blocks and cylinders because of efficient interference check.

Kolli *et al.* adopted the optimization framework proposed by [68] using a multi-resolution model realized through an octree representation [45]. Generating a 3D layout with components of arbitrary geometry and orientation is possible in reasonable time.

Szykman and Cagan proposed an algorithm using SA to solve the constrained 3D component layout design [70]. The spatial constraint language is implemented to represent spatial constraints. The algorithm is applied to a cordless power drill case study, but has several limitations. Blocks and cylinders are used for the shapes of components and containers, and rotations are limited to multiples of 90 degrees.

Cagan *et al.* presented a general extension to the previous work [45, 68–70] [11]. An approach to 3D component placement is presented and illustrated through various test cases and applications using octree.

SA has been applied to many applications such as layout and routing of heat pump [71], the layout design of a submergible boat [42], and routing of a chemical production plant [69]. Hills and Smith also applied SA to the system having the ability to handle the layout and routing, concurrently [36]. Smith *et al.* combined SA and knowledge-based systems to produce layouts for made-to-order products [66]. Campbell *et al.*presented an application of producing a final layout of an embedded wearable computer example with hierarchical heat transfer analysis [12].

2.3.5.5 Extended Pattern Search

Yin and Cagan introduced an extended pattern search algorithm for 3D component layout design [78, 82]. Extensions to basic pattern search are implemented to help the algorithm to converge to optimal solutions by escaping inferior local minima. Employed extensions include randomized search orders, constraint related search directions, occasionally allowed step-jumps, and strategically used swapping moves. Components are represented by the octree model. The algorithm tested the problems originally solved by the SA in [11, 45] and showed better performance. Yin and Cagan examined four different heuristics for generating pattern directions in the extended pattern search [79].

Ding and Cagan used the extended pattern search to the trunk packing problem with several extensions made to address special issues to the problem [18]. The extensions introduced in this research are to select components based on probabilities, to add more degrees of freedom, and to swap components in and out of the trunk. Furthermore, the RP packing problem, minimizing the height of the components, is addressed using extended pattern search [3].

Another application is the layout design of an automobile transmission [80, 83]. Shapeable octrees are presented to take into account the shapeability of the components and to minimize the overlap evaluation cost. In the example, the sizes of the clutches are varied during the optimization, but the shapes do not change.

2.3.5.6 Objective Function Effect Based Pattern Search

The objective function effect based pattern search is presented [2, 4–6]. This algorithm decreases the step sizes of the patterns based on the expected change in objective functions value due to that pattern. Estimating the effect of pattern moves on the objective function is expensive, so the sensitivity metric specialized for the layout problems is developed and applied. Results on 3D component layout problems show that the algorithm performs better than the conventional generalized pattern search.

2.3.5.7 Rubber Band Analogy

Fadel *et al.* introduced the methodology for solving the packing problem using a rubber band analogy. The convex hull was employed to determine the direction of forces applied to a single component, and a motion can occur from the application of such forces [24]. Dong *et al.* extended the original rubber band analogy proposed by [24], presenting new technological developments [19]. The method is enhanced by adding two operators: volume relaxation and temporary retraction. This approach is capable of finding the local optimum as opposed to obtaining the global optimum.

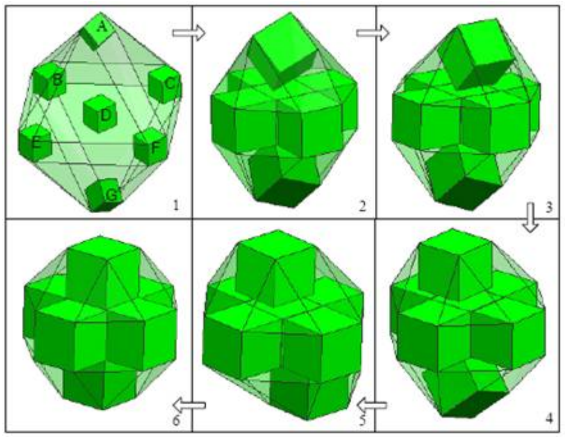


Figure 2.9: Rubber band packing reproduced from [19]

2.4 Computational Environment

This section describes the developed computational environment to conduct the research.

2.4.1 Introduction

This research focuses primarily on the component geometry, in addition to the system performance optimization, presenting new consideration regarding necessary functions and implementation suggestions.

First, required capabilities of the computational environment are listed.

1. **Create geometries.** In general packaging problems, geometries are given in specific file format. However, creation of geometries are sometimes required in this research because geometries may not be given. For this reason, the component shape may have to be assumed and represented by primitives such as box, cylinder, and sphere or by abstract geometry such as bounding box and convex hull. Furthermore, when the bounding box or convex hull is calculated, these geometries may also need to be created in the system.

2. **Modify geometries.** In this research, the component shape is expected to be changing during the optimization. This function can be achieved by modifying dimensions, if dimensions exist in the model.

3. **Load/Import existing CAD files.** Components can be designed in heterogeneous systems, so they may inevitably exist in various file format. For this reason, neutral formats such as IGES (Initial Graphics Exchange Specification) or STEP (Standard for the Exchange of Product model data) is possibly used, so the function of importing a neutral file format is required.

4. **Transform components.** This is a basic routine expected in packaging prob-

lems. The 4 by 4 transformation matrix containing translation and rotation information is generally applied to the components, which can be a single solid body or an assembly. Computation routine of this transformation is necessary.

5. **Handle assembly structures.** An assembly can be treated as one component during the packaging(for example, a radiator assembly consisting of radiator, fan, and shroud). Moreover, an assembly naturally represents the hierarchy of the system. When hierarchical optimization methodology such as ATC is employed for the system design in the future, this capability is expected to be very advantageous.

6. **Calculate mass properties.** Mass properties such as volume, moment of inertia, and center of mass are important quantities in the packaging problems. Calculation of these quantities should be available based on the type of geometric representation.

7. **Check interference.** As described in Section 2.3.2, this is a key function required in the computational environment.

8. **Visualize geometry.** This function is not optional because visualizing and examining the packaging result on the screen are very important to the design engineers.

Next, implementation is also an important issue to research more efficiently. Several options exist to meet the requirement listed above. First, development of the software could be done from scratch; however, given the time constraints and human resource this is not possible. The second available option is to develop the system with the CAD kernel and the available graphic library. A geometric modeling kernel is a 3D solid modeling software component used in computer-aided design packages. The most famous commercial CAD kernels are Parasolid [62] and ACIS [1]. As for the graphic library, OpenGL [58] is widely used for the graphic library, and the special visualization environment [35] provided with the kernel can be integrated. In this approach, the system can be tailored for a specific research purpose. The drawback is the time element required in the process of integrating the library and with other routines such as importing neutral CAD files (procedures not directly related with the research). The final option is to customize the existing commercial CAD molder. Most CAD systems provide an API (application programming interface) in C, C++, or JAVA languages. This approach enables us to fully utilize the functions already developed in the CAD system and develop our own routines. For example, we can use the built-in interference check routine as well as deploy additional algorithms, if necessary.

The last approach is selected for this research due to its efficiency in developing the system. Even though it still requires much time, the time involved is comparatively less than other methods. NX V6 [56] and Microsoft Visual Studio 2005 are chosen as a CAD modeler and a development tool, respectively. Another advantage of using NX V6 API is the software can be run in batch mode, which is a very important feature when being integrated with other systems such as MATLAB, iSIGHT [86], and OPTIMUS [57]. iSIGHT and OPTIMUS are commercial optimization tools, which can be used in the future.

2.4.2 Wrapper Class

The object oriented wrapper class using C++ is developed for ease of project development, depicted in Figure 2.10. Direct use of the NX API functions in the packaging optimization problem is possible. However, the wrapper class provides higher level capabilities and easier development. Moreover, it allows focus on specific problems by the user instead of understanding the CAD system. Individuals with less knowledge of the CAD system and API can still perform their research in virtue of this class.

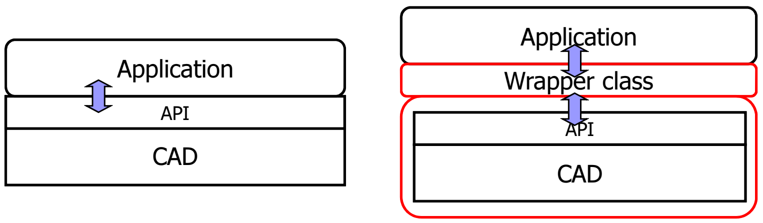


Figure 2.10: The wrapper class architecture

Figure 2.11 shows the C++ class hierarchy developed thus far, which is expected to be expanded as the research progresses.

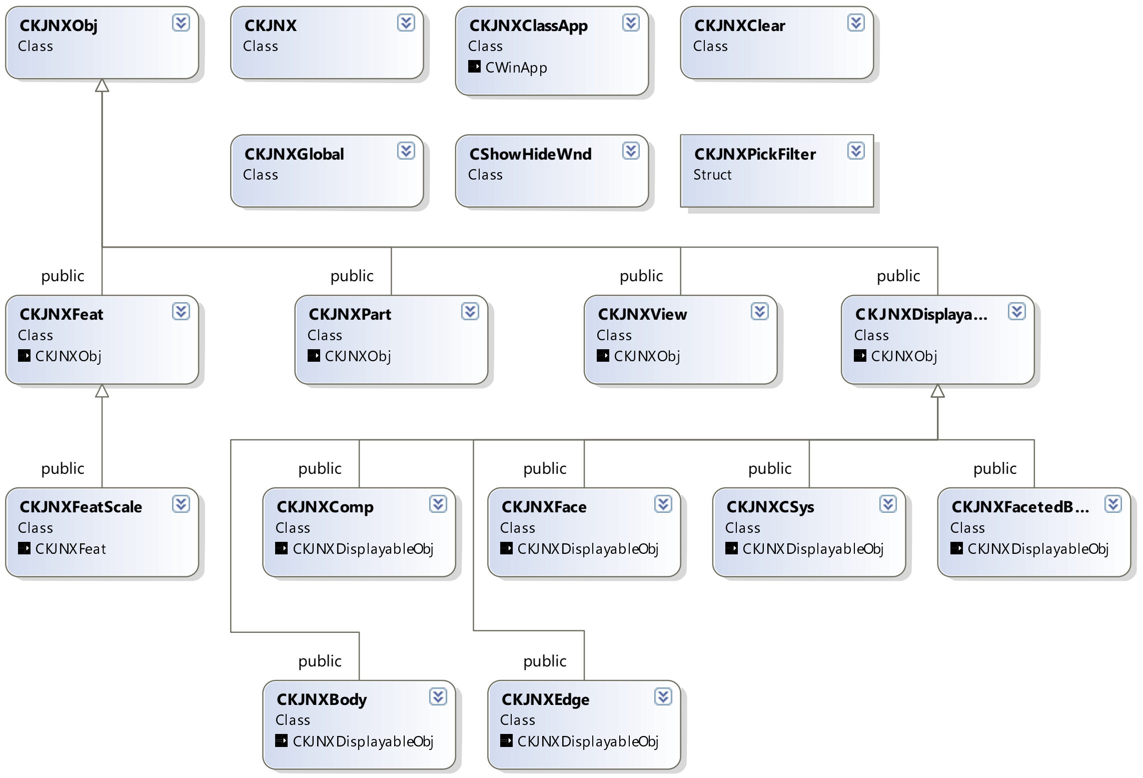


Figure 2.11: Class hierarchy

2.4.3 Integrated Optimization Algorithms

Considering most popular graphic library and API supported by CAD modeler, C++ is chosen to implement the computational environment, which means the optimization codes written in C or C++ is required. Three algorithms are integrated in the system. Additional algorithms can be developed or adopted from other research work, if needed.

2.4.3.1 CFSQP

CFSQP (**C** code for **F**easible **S**equential **Q**uadratic **P**rogramming) V2.5d [9] is integrated into the system. This is a gradient-based algorithm is sensitive to the initial value, but usable in the local search. CFSQP is similar general SQP except that it generates a feasible design at every iteration. When being integrated, the code is modified to be compiled in C++ environment because CFSQP is built in C. Also, the wrapper C++ class is implemented for ease of use.

2.4.3.2 Simulated Annealing

In addition to the gradient based algorithm, SA, which is one of the derivative-free algorithms described in Section 2.3.5, is linked. Whitehead implemented and tested successfully his own MATLAB code on the hybrid electric vehicle simulations; therefore, his MATLAB code is converted into C++ class [75].

2.4.3.3 Genetic Algorithms

GA, another derivative-free algorithm, is also integrated. The source code is downloaded from Kanpur Genetic Algorithms Laboratory [27], then its C++ class is created. Binary and real variables can be used in the code. Constraints also can be handled. Not that all constraints must be positive null form, $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$, and normalized.

2.5 Summary

In this section, geometric realization at both component-level and system-level are reviewed, and some conclusions are made.

Parametric models are used even for abstract geometry of the original CAD data, which allows us to modify it easily. Bounding volumes are not generally enough for abstract geometry in mechanical system design because they are convex. Assuming shapes of all the components in a mechanical system are convex looks impractical. So, although creating AABB and OBB are implemented, the shapes are not restricted to bounding volumes. Solution methods of packaging problems are reviewed, and the computational environment is explained.

In the next chapter, the process to solve the running example and the result is explained. The implementation issues and the solutions are also discussed.

CHAPTER III

Geometry Consideration in System Design Optimization

As can be seen in the running example, the shapes of the components are not often considered depending on which aspect of a system is considered. Before formulating and solving the integrated problem, optimal design for functionality and geometric requirements are presented using the running example.

3.1 Introduction

The schematic diagram in Figure 1.5 and the Simulink model in Figure 3.1 presents the relationship among the components in terms of thermal performance. An optimization problem for functionality is formulated, and uses the Simulink simulation model to compute thermal properties of the system. Also, an optimization model for packaging is necessary to see if there is a feasible layout solution, and better or optimal solution to that. Because the information of the shapes and locations of the components are not available, the abstract models are assumed base on existing models, experience, and market availability, and modeled in a commercial CAD software with dimensions so that the size changes in the simulation model are easily applied to the shapes in the packaging problem.

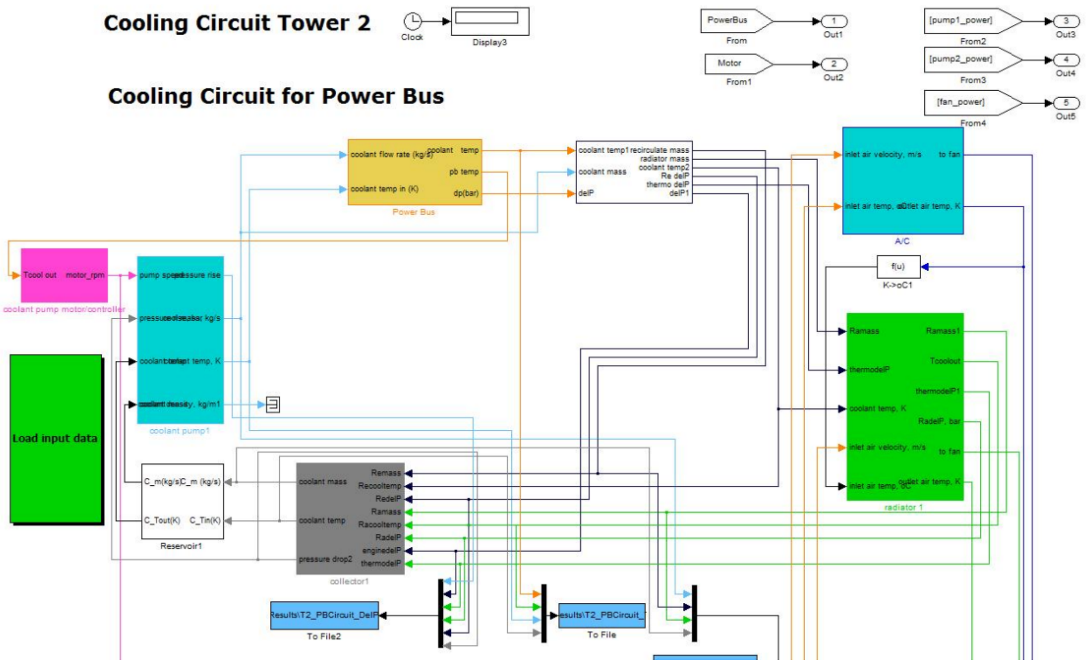


Figure 3.1: Simulink model of the running example

3.2 Elementary Example

Before executing the optimization problem of the running example, an elementary problem with the known optimal solution is formulated to verify the computational environment. Figure 3.3 shows the shapes of components and a container in the problem. d_{ij} denotes the jth dimension of the component i. These components are not yet modeled in a CAD system, which means these shapes are abstract as in the early stage of a design process. Note that the shape of C_1 is simple, but concave.

To formulate an optimization problem, the initial layout is assumed as Figure 3.3. Because the shapes are rectangular, so the non-overlapping constraints can be analytically formulated.

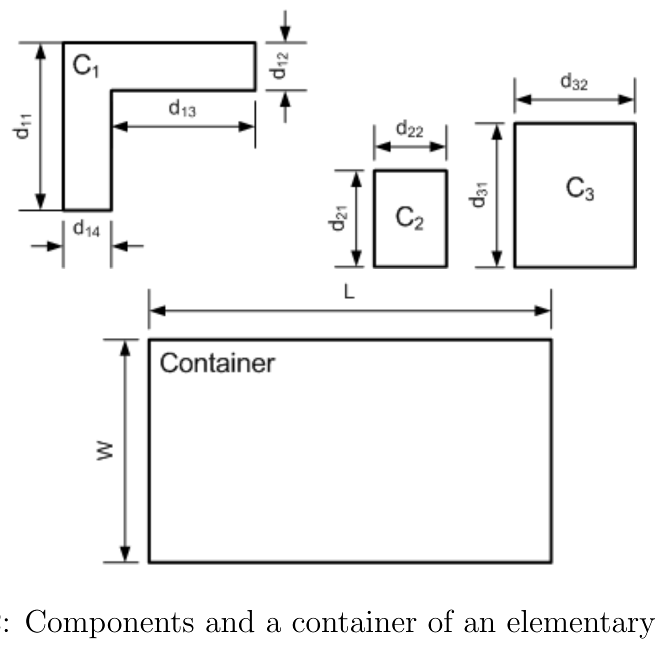


Figure 3.2: Components and a container of an elementary example

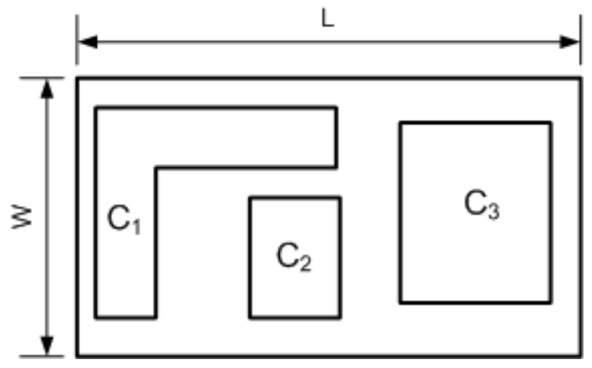


Figure 3.3: Initial layout for an elementary example

The optimization formulation is:

$$
\begin{aligned}
\min_{\mathbf{x}_p} \quad & f_p(\mathbf{x}_p) = -d_{11} - d_{21} * d_{22} \\
\text{subject to} \quad & g_1 = max(d_{14} + d_{13}, d_{14} + d_{22}) + d_{32} - L \leq 0 \\
& g_2 = max(d_{12} + d_{21}, d_{11}) - W \leq 0 \\
\text{where} \quad & \mathbf{x}_p = (d_{11}, d_{21}, d_{22}) \\
& 60 \leq d_{11} \leq 100 \\
& 10 \leq d_{21} \leq 80 \\
& 30 \leq d_{22} \leq 100
\end{aligned}
\tag{3.1}
$$

Table 3.1: Elementary problem parameters

dimension	value (mm)
L	150
W	100
d_{12}	20
d_{13}	80
d_{14}	20
d_{31}	90
d_{32}	40

The parameters are listed in Table 3.1, and the optimal design is known as $\mathbf{x}_{p*} = (100, 80, 90)$. When the MATLAB code that calculates the $\mathbf{x}_p$, g_1, g_2, the optimal values is $\mathbf{x}_{p*} = (100.000000, 79.999937, 90.000000)$. The result seems to be acceptable in most cases, but because the calculations are basic and simple, further investigation is made to make sure it is really numerical errors. It is found out that this error come from the number of digits for values written in the input and output text files. For some FEM softwares, the number of digits are fixed according to their own file format, but since no restriction on the number of digit in the proposed framework, 16 digits are recommended for double precision values.

Equation 3.1 is relatively simple, but several things should be noted in this formulation.

- The shapes are not real, these are abstract.
- The shapes are simple, even though the component one is concave. They are rectilinear, which makes building formulations for non-interference much easier.
- If this layout is not feasible for some reasons, the whole formulation needs to be updated based on the new layout.
- The two constraints are basically a special case of non-interference constraint because of the shapes and layout.

- Once we setup the problem with CAD models, we can test different layouts by changing the location of the components to check feasibility with a general formulation for non-interference constraint, which uses geometry information.

Although this elementary example is formulated to verify that the proposed framework is correctly working, it is also used to check the pipe routing routine before applying to the running example.

In the following section, the implementation issues and optimization problem for thermal performance are explained.

3.3 Optimal Design for Functionality

Engineers make a great effort to meet, minimize, or maximize many functional requirements of a system. Thermal requirement in the running example is the one of functional requirements that should be maximized by the system. The running example has many thermal analysis models that are developed to compute various thermal properties. Using those models, an optimization model is developed to find the optimal solution to maximize thermal performance.

3.3.1 Software Implementation

Analysis models are generally developed on various computational platforms such as MATLAB, Simulink, or CAE softwares. The platform on which analysis models are built directly influences the selection of the platform for optimization codes. For example, when the models are built on MATLAB, which is widely used in many research fields, the optimization tool in MATLAB such as fmincon could be the first choice. fmincon is the optimization code that can solve a minimization problem with nonlinear constraints. Even with some models developed on other commercial CAE tools, MATLAB can still be good because MATLAB provides many methods to interface with other softwares. Also, MATLAB has many solvers in Optimization Toolbox and Global Optimization Toolbox. On the other hand, commercial optimization softwares can be used to solve the problem such as Isight or Optimus because these softwares provides interfaces to many softwares including MATLAB and CAE tools. Since the simulation models in the running example is built using Simulink, fmincon in MATLAB should be the one of the possible choices. If it is easy to handle geometry and compute interference in MATLAB, MATLAB would be the computational environment. As explained in Section 2.4, however, a commercial CAD modeler using C/C++ API is selected as a computational environment, so the methods to interface with the MATLAB and Simulink needs to be found and implemented in the computational environment.

3.3.1.1 Integration with MATLAB and Simulink

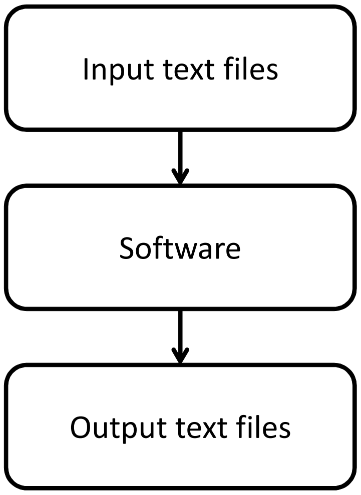


Figure 3.4: Execution of a software in batch mode

Figure 3.4 shows the execution of a software in batch mode. A program takes input files that are normally text files, processes the data, and produces output files that are also usually text files. Running the softwares in batch mode is the one of the most widely used to interface with other software packages. This method is also useful to run the softwares that takes a lot of time to produce the results. This is why almost all commercial CAE tools support the batch mode execution. So, the method to run MATLAB and Simulink in batch mode is firstly investigated because, when it is possible, the integration would be easy.

Batch mode for MATLAB and Simulink

The method to run MATLAB script files in batch mode should be first established because the Simulink model of the running example uses the workspace variables such as ambient temperature, look-up table data, and driving condition. Therefore, the Simulink model should be called from the MATLAB script.

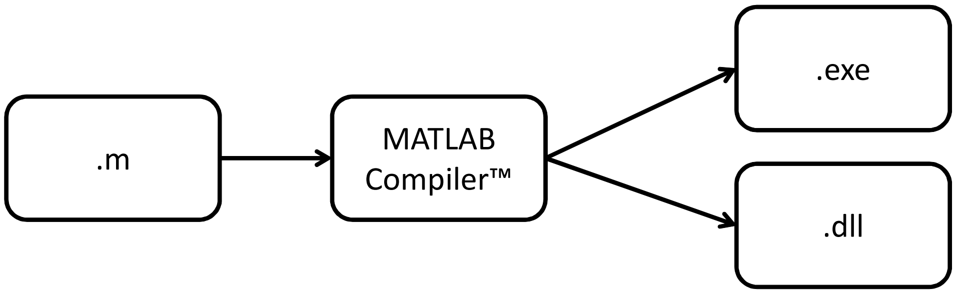


Figure 3.5: MATLAB Compiler™

MATLAB Compiler™ can automatically package MATLAB applications into standalone applications or C/C++ libraries, as appeared in Figure 3.5. This compiler, however, does not support the functions that need to call the Simulink model from MATLAB Compiler™ such as `sim()` and `simset()`.

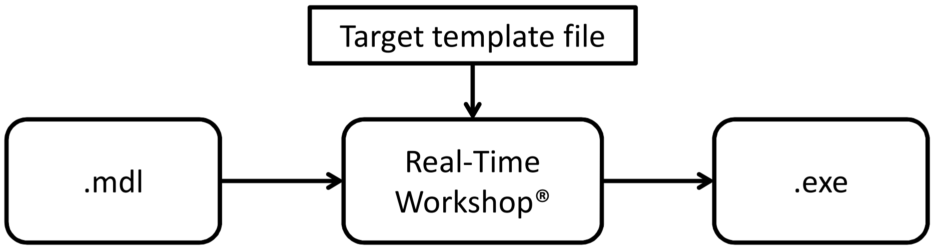


Figure 3.6: Real-Time Workshop®

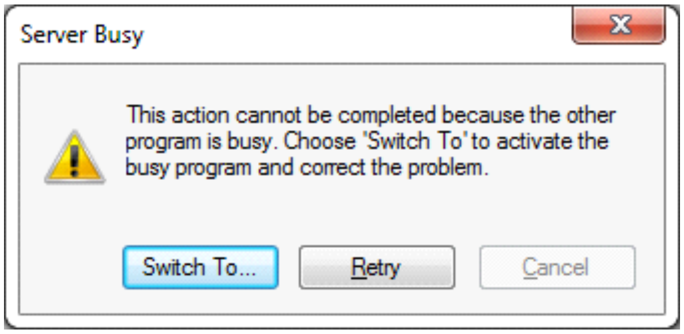
Real-Time Workshop® (Simulink Coder™ in the current release) generates standalone C code from Simulink models. Real-Time Workshop® uses target template files to translate the Simulink models into C code. The target templates specify the environment on which the generated code will run. Rapid Simulation (RSim) target is selected because RSim target generates fast, standalone simulations that allow batch parameter tuning and loading of new simulation data from a standard MATLAB MAT-file without needing to recompile the Simulink model. Unfortunately, one of the limitations of Rsim target is that it does not support Fortran S-functions, which is a big problem because the S-functions in the running examples are all built with Fortran language. Instead of porting manually the Fortran codes to C codes, f2c utility is used [23], then the S-functions are updated and re-compiled with C codes. The simulation results are compared with the previous results that uses Fortran S-functions.

One more step is left to use MATLAB Compiler™ and Real-Time Workshop® with RSim target. To be able to read the input values from the text file and apply those values to the Simulink model, `‘AddTunableParamInfo’` should be on. Also, to use `‘AddTunableParamInfo’`, inline parameters must be enabled in Simulink. The problem is that the matrix of output values for the 2-D look-up table block, such as 'fan performance' block, contains an expression, which does not support code generation of an expression with a tunable variable when inline parameters are switched on. Tunable variables is necessary because the values in the Simulink models are changed at the optimization running time, not at compile time. Although all the implementation and test are performed with a Simulink model that has a C S-Function, some issues are still left to be addressed when it applying the running example model. So, other approaches are examined.

MATLAB Engine

It is found out that MATLAB provides the interface for C/C++ or Fortran program, which MATLAB Engine library. Engine programs are standalone C/C++ or

Fortran programs that communicate with a separate MATLAB process through a Microsoft Component Object Model (COM) interface on Microsoft Windows systems. MATLAB provides a library of functions that allows you to start and end the MATLAB process, send data to and from MATLAB, and send commands to be processed in MATLAB. This library is directly integrated with the computational environment. This approach is slower than the previous compile approach for a single run because it connects to MATLAB, run MATLAB codes, and disconnect. Although it takes some time to start MATLBA process, it works well. When, however, this routine is frequently called during optimization, the OLE Server Busy dialog box randomly displays, as shown in Figure 3.7.



This action cannot be completed because the other program is busy. Choose 'Switch To' to activate the busy program and correct the problem.

Figure 3.7: OLE Server Busy dialog box

If the COM call takes longer than the previously specified time, the client application displays the OLE Server Busy dialog box. This problem can be addressed by either increasing the pending time or disabling the busy dialog box from appearing after the COM call times out. This solution looks straightforward and working, but it is called from an MFC DLL, the call fails. As explained in Section 2.4, NX is selected among commercial CAD modelers. When the application is developed using NX API, its form is DLL, so the solution cannot be applied.

Proposed MATLAB interface

Figure 3.8 shows the proposed framework to call MATLAB functions from the developed computational environment that uses NX API. The application writes the input file that contains the design variables for the optimization problem, and executes RunML.exe that is the standalone executable program that reads the input text file, and calls MATLAB functions via COM to calculate an objective function and constraints. Although this framework is a little slower than both batch mode only or directly integrated with MATLAB Engine library, it is much more stable.

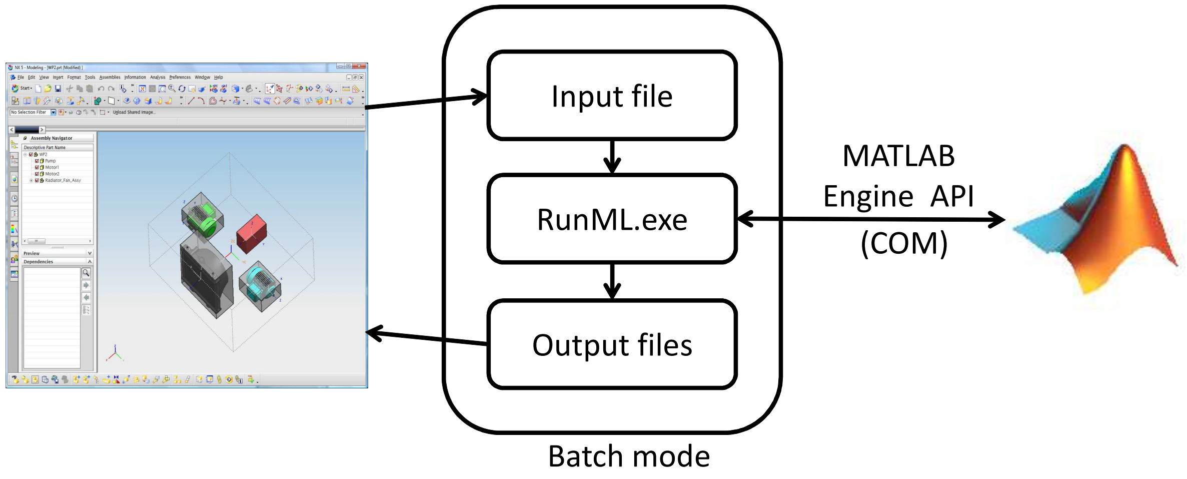


Figure 3.8: Proposed framework for MATLAB interface

3.3.2 Surrogate Model Development for Radiators

Before discussing the objective function and constraints of the thermal performance optimization problem, the surrogate models for radiators are explained. Because simulation time of the running example takes long time for one run, it is not suitable for an optimization study. Then, the implementation issues are discussed. Not only building optimization formulation and solving it itself is not easy, but also integrating the optimization code and analysis models is not trivial.

The radiator model in the running example is developed using Finite Difference

Method(FDM) with staggered grid system [43]. Computational cost of this FDM model is relatively expensive than other analysis models in the running example. In addition, this FDM model is used in 5 radiators and a CAC with different input parameters, which makes computational cost more expensive. So, it would be better to have the surrogate models for radiators. Figure 3.9 shows the shape of a radiator and its core.

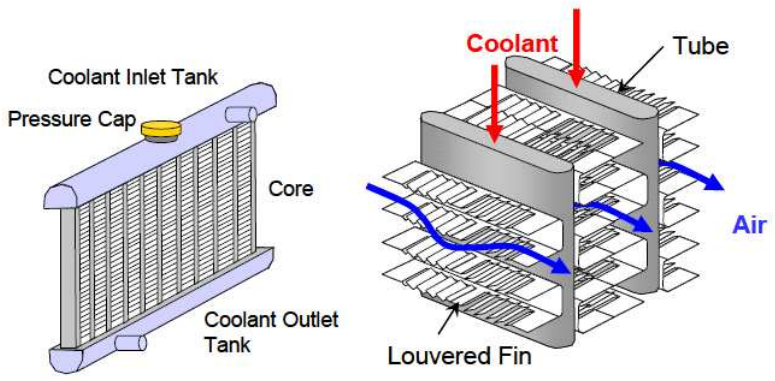


Figure 3.9: Structure of a radiator and its core reproduced from [43]

The FDM model has 23 inputs and 5 outputs. As shown in Figure 3.10, the input parameters include the dimensions of core, tube, and fins, temperature and mass flow rate of coolant, and velocity and temperature of air. Among 23 inputs, 7 variables are selected to generate the surrogate model because other values are either constant or calculated based on the given values. Among the input parameters, the range of some values such as temperature and mass flow rate of coolant cannot be estimated without running the simulation model because these values are internally calculated during the vehicle simulation. Vehicle simulations are performed with the three different input values over the 30 minutes driving cycle, which are the baseline design, and ±10% from the baseline design. The ranges of input variables are chosen based on the simulation results. The input parameters and ranges are summarized in Table 3.2. Note that not all 7 input variables are not used for some models. For example, the core size of CAC is constant in the simulation, so those variables are excluded.

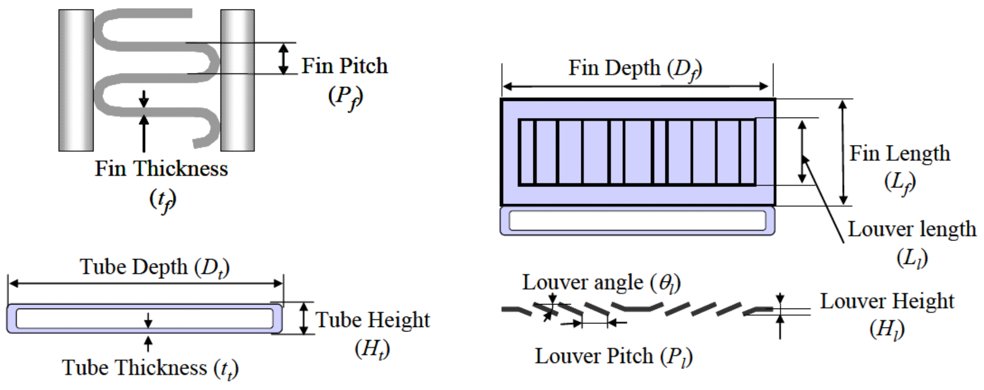


Figure 3.10: Design parameters of the radiator core adapted from [43]

Table 3.2: Input and output variables for DOE study

Inputs	Outputs
core width	
core height	heat dissipation (q_{rad})
core thickness	outlet air temperature (T_{air})
coolant mass flow	outlet coolant temperature(T_{cool})
inlet coolant temperature	air pressure drop (ΔP_{air})
inlet air temperature	coolant pressure drop (ΔP_{rad})
inlet air velocity	

The 8000 design points are prepared for DOE study using Latin Hyper Cube method, and the points that generate numerical errors are excluded. Each surrogate model is created for each radiator model including CAC model. Two thirds of the points are used to make the surrogate model and other one third are used to test the surrogate model.

3.3.2.1 Neural Network

A radial basis neural network model is widely used to approximate functions. Figure 3.11 shows the architecture of radial basis networks, which consist of two layers: a hidden radial basis layer of S^1 neurons, and an output linear layer of S^2

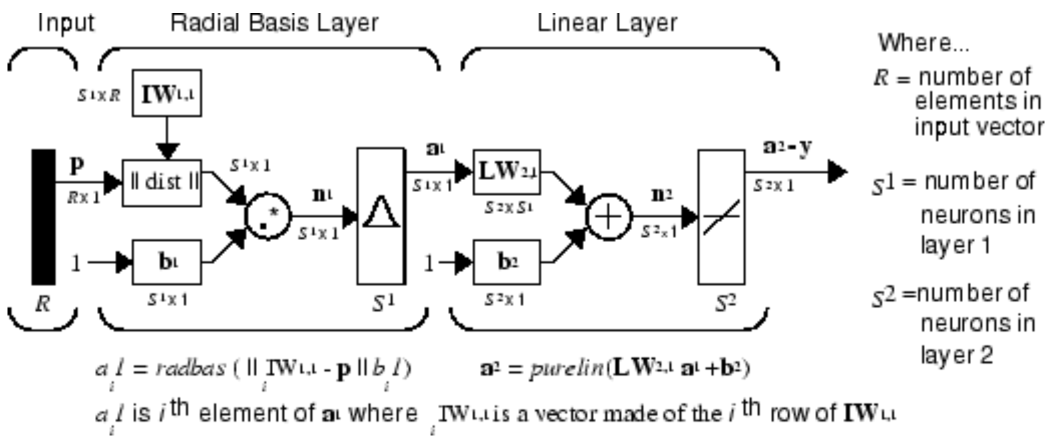

Input
Radial Basis Layer
Linear Layer
Where...
R = number of elements in input vector
S1 = number of neurons in layer 1
S2 = number of neurons in layer 2
IW1,1
|| dist ||
b1
LW2,1
b2
p
a1
n1
n2
a2 = y
a1_i = radbas (|| iIW1,1 - p || b1_i)
a2 = purelin(LW2,1 a1 + b2)
a1_i is i th element of a1 where iIW1,1 is a vector made of the i th row of IW1,1

Figure 3.11: Architecture of radial basis networks reproduced from [48]

neurons. The transfer function for a radial basis neuron is

$$radbas(n) = e^{-n^2}$$

As can be seen Table 3.3, the neural network models fit well.

Table 3.3: R-squared values for neural network model

component	q_{rad}	T_{air}	T_{cool}	ΔP_{air}	ΔP_{rad}
T1R1[†]	0.99	0.98	0.95	0.90	0.98
T1R2	0.99	0.91	0.99	0.93	0.99
T1R3	0.97	0.88	0.98	0.91	0.99
T1CAC	0.98	0.99	0.93	0.99	0.96
T2R1	0.99	0.99	0.99	0.74	0.99
T2R2	0.99	0.99	0.99	0.73	0.99

[†] T1R1 denotes radiator 1 in tower 1

Because all the details of the network are implemented in Matlab Neural Network Toolbox, we generally do not need to write to lines of code. Instead, we just use newrbe() and newrb() functions to train the network, and can obtain the outputs with sim() function. This looks obvious, but there is a computation time issue with sim() function. The computation time is improved about 60% (from 51 min to 21 min) when one simulation is run over 30min driving cycle. Although it is improved quite a lot, running one simulation still takes 20 min.

sim() is a general function to simulate a neural network, not only for a radial basis network. There are unnecessary codes when it is sure that a radial basis network is only used.

So, instead of `y = sim(net, pn);`, the following code is used.

```
IW = net.IWp{1,1};
B1 = net.b{1};
b1 = B1(1);
LW = net.LW{2,1};
b2 = net.b{2};
n = b1*dist(IW,pn);
h = exp(-(n.*n));
y = LW*h + b2;
```

The computation time on Xeon E5150, 2.66GHz (DualCore), 5GB RAM is shown in Table 3.4.

Table 3.4: Computation time of vehicle simulation

Model	Time (min)†
Original Model	51
Neural Network	21
Revised Neural Network	5.1

† 30 min driving cycle

The calculated results such as power consumption of fan and pumps, and temperature of components are compared with the original MATLAB code and verified.

3.3.2.2 Polynomial Function Approximation

In addition to the neural network model, polynomial surrogates using second-order and interaction terms are built for each radiators. While R-squared values are similar, computation time is improved from 21min to 32 seconds when one simulation is run over 30min driving cycle. Therefore, the polynomial model is selected as the surrogate models for the radiators in this dissertation.

Table 3.5: R-squared values for polynomial model

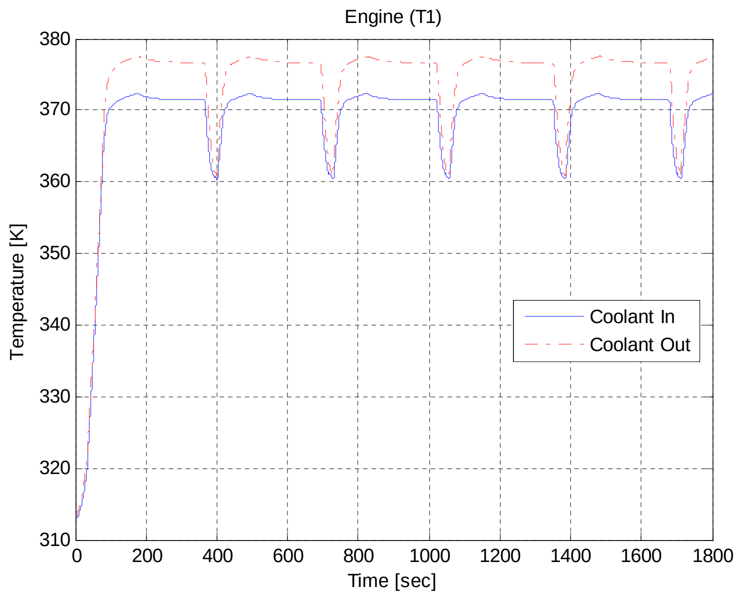
component	q_{rad}	T_{air}	T_{cool}	ΔP_{air}	ΔP_{rad}
T1R1	0.97	0.94	0.83	0.89	0.96
T1R2	0.98	0.84	0.98	0.92	0.99
T1R3	0.92	0.81	0.95	0.91	0.99
T1CAC	0.99	0.99	0.94	0.99	0.99
T2R1	0.98	0.99	0.96	0.74	0.99
T2R2	0.98	0.99	0.96	0.74	0.99

3.3.3 Formulation

The optimization problem for functionality is to determine the sizes of pumps, fan, and radiators such that overall power consumption by fan and electrical pumps during the driving cycle is minimized, subject to constraints on packaging requirements for radiators and the target temperatures for components. The formulation is:

$$\begin{aligned}
\min_{\mathbf{x}_p} \quad & f_p(\mathbf{x}_p) \\
\text{subject to} \quad & g_1 = 0.8 * S_{T1R1X} + 0.3 * S_{T2R1X} - W_c \leq 0 \\
& g_2 = T_{gen} - T_{gen\_target} \leq 0 \\
& g_3 = T_{eng} - T_{eng\_target} \leq 0 \\
& g_4 = T_{CAC} - T_{CAC\_target} \leq 0 \\
& g_5 = T_{oil} - T_{oil\_target} \leq 0 \\
& g_6 = T_{pb} - T_{pb\_target} \leq 0 \\
& g_7 = T_{mot} - T_{mot\_target} \leq 0 \\
\text{where} \quad & f_p(\mathbf{x}_p) = P_{T1} + P_{T2} \\
& P_{T_1} = P_{T1P1} + P_{T1P2} + P_{T1P3} + P_{T1Fan} \\
& P_{T_2} = P_{T2P1} + P_{T2P2} + P_{T2Fan}
\end{aligned} \tag{3.2}$$

The objective function and all constraints except g_1 are computed by the vehicle simulation. As explained , the vehicle simulation results shows that the grade load condition is the most severe condition for the cooling system, so the grade load condition is used for cooling system evaluation. Figure 3.12 shows the vehicle simulation results for the fan and engine in tower 1 over 30 minutes driving cycle. Because of the grade load condition, the simulation results shows the repetition pattern, so 10 minutes is used to evaluate the cooling system to reduce computation time.



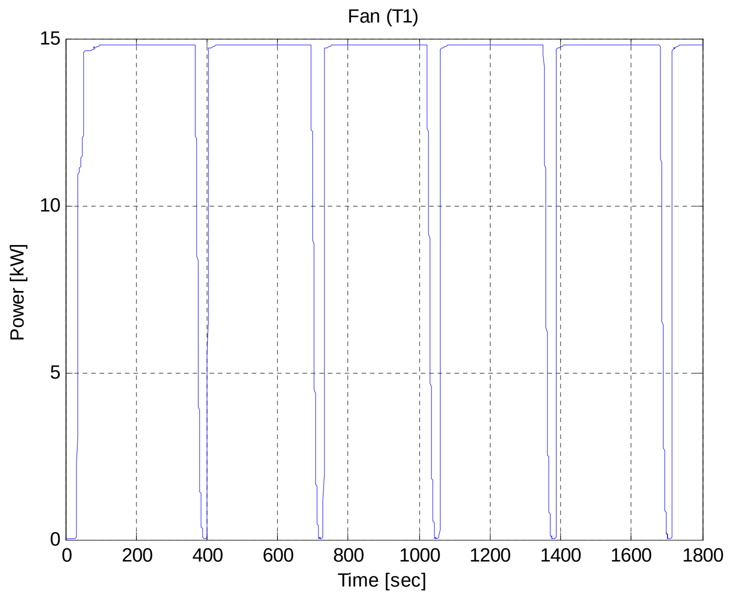


Figure 3.12: Results of vehicle simulation with baseline design

Objective function

The objective function is to minimize the power consumed by the fans and pumps in the cooling system.

Design Variables

The design variables for functionality are the sizes of the radiators and pumps. The 10% changes from the baseline design are used. The size of the pump is not the actual geometry size but the scaling factor of the pump's capacity. The design variables that appear in Equation 3.2 are described in Table 3.7 along with their lower and upper bounds. The first eight design variables are for tower 1, and the rest are for tower 2.

Table 3.6: Design variables for optimization problem for functionality

variable	lower bnd.	upper bnd.	description
x1 = S_{T1P1}	0.9	1.1	scaling factor for T1P1
x2 = S_{T1P2}	0.9	1.1	scaling factor for T1P2
x3 = S_{T1P3}	0.9	1.1	scaling factor for T1P3
x4 = S_{T1R1X}	0.9	1.1	scaling factor for width of T1R2
x5 = S_{T1R1Z}	0.9	1.1	scaling factor for thickness of T1R1
x6 = S_{T1R2X}	0.9	1.1	scaling factor for width of T1R2
x7 = S_{T1R2Z}	0.9	1.1	scaling factor for thickness of T1R2
x8 = S_{T1Fan}	0.9	1.1	scaling factor for T1Fan
x9 = S_{T2P1}	0.9	1.1	scaling factor for T2P1
x10 = S_{T2P2}	0.9	1.1	scaling factor for T2P2
x11 = S_{T2R1X}	0.9	1.1	scaling factor for width of T2R1
x12 = S_{T2ACZ}	0.9	1.1	scaling factor for thickness of T2AC
x13 = S_{T2R1Z}	0.9	1.1	scaling factor for thickness of T2R1
x14 = S_{T2R2Z}	0.9	1.1	scaling factor for thickness of T2R2
x15 = S_{T2Fan}	0.9	1.1	scaling factor for T2Fan

Constraints

The temperatures of the components that should be maintained by the cooling system are the constraints. Constraints except g1 are about performance. g1 is for

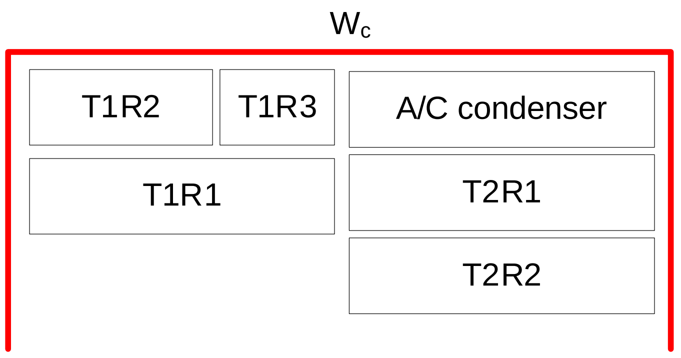


Figure 3.13: Layout of radiators in the running example

packaging for radiators, which is what can be done with this performance model. It is not sure whether other components can also be placed in a container.

g_1 is a non-overlapping constraint between radiators and a container. The analytic formulation is possible because several implicit assumptions are made on the shape and location of the radiators. First, the shapes of the radiators and container are rectangles. Second, the radiators does not rotate. Third, a packaging constraints for the height and thickness of the radiators are not necessary. The assumption for height is valid because the height of the radiators is constant during optimization with baseline design. The assumption for thickness is basically that thickness is so small compared to other dimensions that it should no problem for packaging. This seems to be fair, but needs to be checked later with real geometry. Fourth, other components in the system can be located without overlapping. This assumption is problematic, but it is impossible to formulate the packaging constrains for all components at this point because information of the shapes and locations of the components are not available. This is why a packaging problem needs to be formulated with real geometry that is even abstract shape.

W_c in g_1 is the width allowed for radiators in tower 1 and 2, as shown in Figure 3.13. Its values is 1.2(m), and is a parameter for the optimization problem. Only the width and thickness of radiators vary during optimization.

The target temperatures for the components in g_2–g_7 are listed in Table 1.6. The temperature distribution in a heat source including the engine, generator, motors and power bus should be minimized by the cooling system because large temperature distribution can deteriorate the durability of the heat source component.

3.3.4 Results

An optimal solution obtained by CFSQP. Power consumtion is improved by 25% while satisfying the temperature constraints.

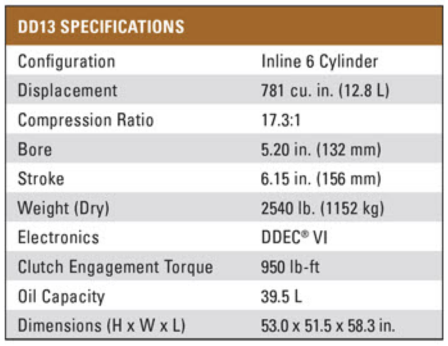
Table 3.7: Optimization results for functionality

variable	initial design	optimal design
x1 = S_{T1P1}	1.0	0.90
x2 = S_{T1P2}	1.0	0.90
x3 = S_{T1P3}	1.0	1.05
x4 = S_{T1R1X}	1.0	1.08
x5 = S_{T1R1Z}	1.0	0.91
x6 = S_{T1R2X}	1.0	0.92
x7 = S_{T1R2Z}	1.0	1.08
x8 = S_{T1Fan}	1.0	0.90
x9 = S_{T2P1}	1.0	1.10
x10 = S_{T2P2}	1.0	0.95
x11 = S_{T2R1X}	1.0	1.09
x12 = S_{T2ACZ}	1.0	1.00
x13 = S_{T2R1Z}	1.0	1.09
x14 = S_{T2R2Z}	1.0	0.98
x15 = S_{T2Fan}	1.0	0.90

3.4 Optimal Design for Geometric Requirements

3.4.1 Geometry for Components

As a first step, the shape for each component is needed. The components are used in the running examples does not exist, so the shapes and sizes are assumed based on the reference model that used in the performance model or similar type from catalogs as much as possible. Note that the computational environment are designed to incorporate the change of geometry and size. These shapes and sizes are selected to show the validation of the proposed framework and can be updated at any time.



DD13 SPECIFICATIONS	
Configuration	Inline 6 Cylinder
Displacement	781 cu. in. (12.8 L)
Compression Ratio	17.3:1
Bore	5.20 in. (132 mm)
Stroke	6.15 in. (156 mm)
Weight (Dry)	2540 lb. (1152 kg)
Electronics	DDEC® VI
Clutch Engagement Torque	950 lb-ft
Oil Capacity	39.5 L
Dimensions (H x W x L)	53.0 x 51.5 x 58.3 in.

Figure 3.14: Image and specification of Detroit DD13 Engine reproduced from [16]

Engine rooms size is selected based on the vehicle spec. The shapes is assumed as a box shape. The look-up table for the engine used in the running example is generated for a heavy duty, inline, six cylinder, turbocharged, and intercooled diesel engine with the data from engine dynamometer tests [61]. As shown in [28], turbochared diesel engines are widely used in the similar tracked vehicles as in the running example. Based on the specification of the baseline diesel engine described in [61], the similar models from the Detroit Diesel Corporation to find the size for the engine. The specification of DD13 engine is found in [16]. The sizes for other components are either found in the simulation model or assumed. The initial sizes are summarized in Table 3.8

3.4.2 Software Implementation

The input text file format is developed to make geometry optimization problem more easily, as shown in 3.15. Although the computation of objective function and constraints is hard to implement with text file, but the initial values for position, orientation, and dimensions can be easily changed. Also, the degree of freedom for com-

Table 3.8: Initial sizes and degree of freedom for the components

no.	name	size(W*H*L† or D*L‡(mm))	degree of freedom
1	container	2000 * 1500 * 2500	-
2	engine	1270 * 1016 * 1346	tx
3	generator	800 * 300	(relative to engine)
4	CAC	200 * 200 * 76	tx,ty,tz
5	oil cooler	75 * 1000	tx
6	T1R1	800 * 600 * 50.67	-
7	T1R2	200 * 600 * 50.67	(relative to T1R1)
8	T1R3	600 * 600 * 50.67	(relative to T1R1)
9	T1Fan	500 * 100	(relative to T1R1)
10	T1P3	200 * 150	tx,ty,tz
11	power bus	365 * 238 * 380	tx,ty,tz
12	motor(A/B)	405 * 241	-
13	T2R1	300 * 600 * 17.73	-
14	T2R2	300 * 600 * 50.67	(relative to T2R1)
15	T2AC	300 * 600 * 76	(relative to T2R1)
16	T2Fan	200 * 50	(relative to T2R1)
17	T2P1	200 * 150	tx,ty,tz
18	T2P2	200 * 150	tx,ty,tz

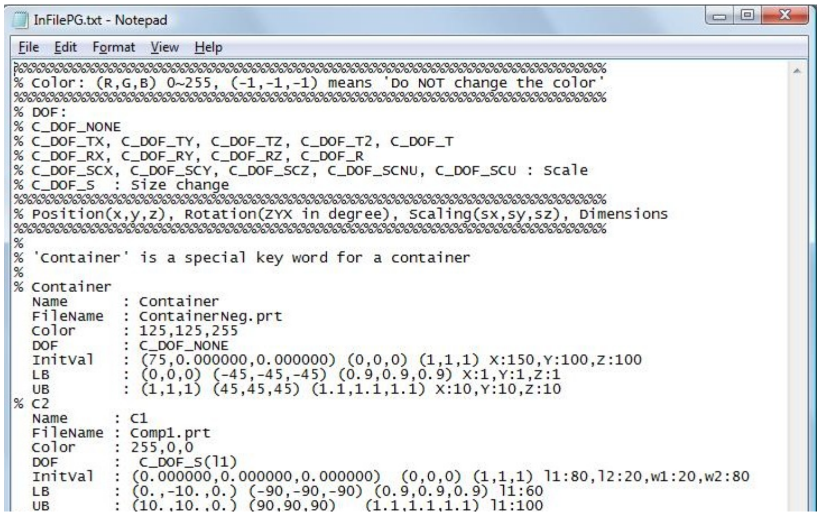
† for box shape components
‡ for cylinder shape components

ponents can be also easily modified. Without this input file, when the DOF changes, the number of design variables changes, then the program needs to be compiled. This enables engineers to run the optimization problem more easily with different initial values with the same objective function and constraints. Furthermore, this capability can explore different layout without modifying the codes.

Also, the same format of a log file is created for CFSQP, GA, and SA to record the optimization, then it can be visually played, and engineers can review the results.

3.4.3 Formulation

Compactness of the system is formulated for packaging purpose. With experts' input improves the accuracy or quality of constraints and objective function. But, the most important thing is that the proposed environment and process can handle



```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Color: (R,G,B) 0~255, (-1,-1,-1) means 'Do NOT change the color'
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DOF:
% C_DOF_NONE
% C_DOF_TX, C_DOF_TY, C_DOF_TZ, C_DOF_T2, C_DOF_T
% C_DOF_RX, C_DOF_RY, C_DOF_RZ, C_DOF_R
% C_DOF_SCX, C_DOF_SCY, C_DOF_SCZ, C_DOF_SCNU, C_DOF_SCU : Scale
% C_DOF_S  : Size change
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Position(x,y,z), Rotation(ZYX in degree), Scaling(sx,sy,sz), Dimensions
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% 'Container' is a special key word for a container
%
% Container
  Name      : Container
  FileName  : ContainerNeg.prt
  Color     : 125,125,255
  DOF       : C_DOF_NONE
  InitVal   : (75,0.000000,0.000000) (0,0,0) (1,1,1) X:150,Y:100,Z:100
  LB        : (0,0,0) (-45,-45,-45) (0.9,0.9,0.9) X:1,Y:1,Z:1
  UB        : (1,1,1) (45,45,45) (1.1,1.1,1.1) X:10,Y:10,Z:10
% C2
  Name     : C1
  FileName : Comp1.prt
  Color    : 255,0,0
  DOF      :  C_DOF_S(l1)
  InitVal  : (0.000000,0.000000,0.000000)  (0,0,0) (1,1,1) l1:80,l2:20,w1:20,w2:80
  LB       : (0.,-10.,0.) (-90,-90,-90) (0.9,0.9,0.9) l1:60
  UB       : (10.,10.,0.) (90,90,90)   (1.1,1.1,1.1) l1:100
```

Figure 3.15: Example of an input file for packaging problem

almost all the geometric requirements at both component and assembly level.

$$
\begin{aligned}
\underset{\mathbf{x}_g}{min} \quad & f_g(\mathbf{x}_g) \\
\text{subject to} \quad & \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} Vol(C_i \cap C_j) \leq 0 \\
\text{where} \quad & f_g(\mathbf{x}_g) = d_1 + d_2 + 10 * d_3 + d_4 + d_5 + d_6 \qquad (3.3) \\
& d_1 = \|\mathbf{P}_{gen} - \mathbf{P}_{T1R3}\| + \|\mathbf{P}_{T1R3} - \mathbf{P}_{T1P3}\| + \|\mathbf{P}_{T1P3} - \mathbf{P}_{gen}\| \\
& d_2 = \|\mathbf{P}_{CAC} - \mathbf{P}_{T1R2}\| + \|\mathbf{P}_{T1R2} - \mathbf{P}_{eng}\| + \|\mathbf{P}_{eng} - \mathbf{P}_{CAC}\| \\
& d_3 = \|\mathbf{P}_{eng} - \mathbf{P}_{T1R1}\| \\
& d_4 = \|\mathbf{P}_{pb} - \mathbf{P}_{T2R1}\| + \|\mathbf{P}_{T2R1} - \mathbf{P}_{T2P1}\| + \|\mathbf{P}_{T2P1} - \mathbf{P}_{pb}\| \\
& d_5 = \|\mathbf{P}_{motA} - \mathbf{P}_{motB}\| + \|\mathbf{P}_{motB} - \mathbf{P}_{T2R2}\| + \|\mathbf{P}_{T2R2} - \mathbf{P}_{T2P2}\| \\
& +\|\mathbf{P}_{T2P2} - \mathbf{P}_{motA}\| \\
& d_6 = \|\mathbf{P}_{eng} - \mathbf{P}_{oil}\|
\end{aligned}
$$

$\|\cdot\|$ denotes L^2-norm.

Objective function

The objective function is to minimize the summation of distances among components in the same cooling circuits including oil cooler. Since the distance between the engine and radiator 1 in tower 1 plays an important role for system compactness, the weight 10 is applied to it.

Design Variables

The positions and orientations of the components are the design variables. Some parts such as radiators and fan can be grouped together. As explained earlier, the number of design variables are automatically calculated from the DOFs in the input file. For the running example, it has 17 design variables, as listed in Table 3.8. The shape of the components are parameters for the packaging problem.

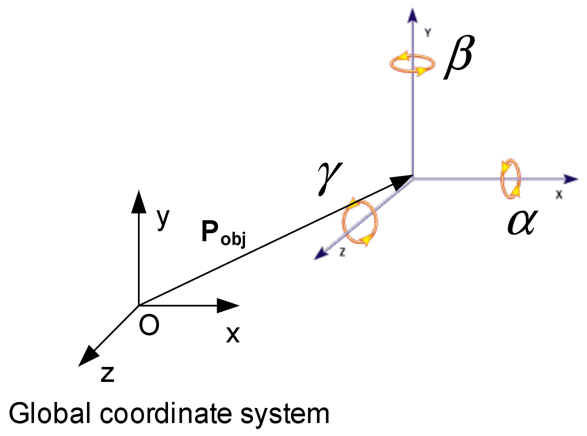


Figure 3.16: Position and orientation of a component

Given a position and orientation, a 4 by 4 transform matrix is built and set to the assembly, not each solid body in CAD system.

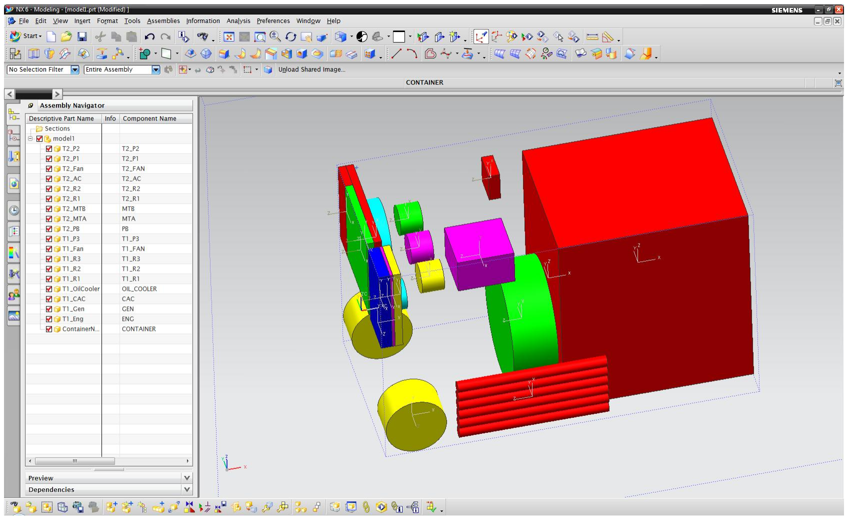
Constraints

The first and most important constraints is a non-overlapping constraint among the components. Also, radiators have specific location constraints due because of the performance model. The different layout for radiators means a totally different problem that requires rebuilding a simulation model.

GA algorithm is used to solve the problem because of the non-overlapping constraint that is non-smooth. As can be seen in g_1 in Equation 3.2, analytical formulation of a non-overlapping constraint is affected by the shapes, location, and orientation of components. However, it is replaced in Equation 3.3 by a more general form that can handle generic shapes and locations.

Selecting initial layout has two possibilities. First one is that finding a feasible layout and changing locations relatively small to keep the initial layout as much as possible. In this case, the initial layout may play important role in both performance and packaging problem, or is fixed based on the performance models. Then, the optimal or better layout is searched. If there are more than one candidate initial layout, optimization is performed for each layout. Second one is that setting an initial layout, even infeasible layout is possible, and searching design space widely to find the optimal solution. Although it can fail to find solutions, the whole new layout can be found, if succeeded. This case can more likely happen when the new product is designed, or the existence of a feasible layout is not known.

The initial layout of the running example is shown in Figure 3.17. Some components such as fans, radiators, and engine are located based on knowledge of the system and assumptions of the performance problem. And, other components such as electrical pumps and CAC are randomly placed while non-overlapping with other components.


NX 6 - Modeling - [model1.prt (Modified)]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
Start
No Selection Filter
Entire Assembly
Upload Shared Image...
CONTAINER
Assembly Navigator
Preview
Dependencies

Descriptive Part Name	Info	Component Name
Sections		
model1		
T2_P2		T2_P2
T2_P1		T2_P1
T2_Fan		T2_FAN
T2_AC		T2_AC
T2_R2		T2_R2
T2_R1		T2_R1
T2_MTB		MTB
T2_MTA		MTA
T2_PB		PB
T1_P3		T1_P3
T1_Fan		T1_FAN
T1_R3		T1_R3
T1_R2		T1_R2
T1_R1		T1_R1
T1_OilCooler		OIL_COOLER
T1_CAC		CAC
T1_Gen		GEN
T1_Eng		ENG
ContainerN...		CONTAINER

Figure 3.17: Initial layout for the running example

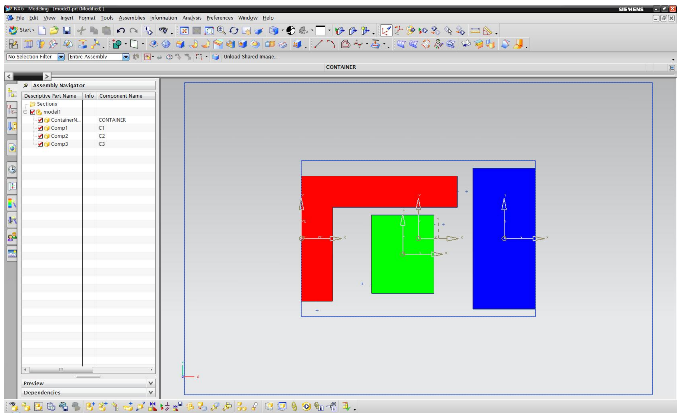
3.4.4 Results and Discussion

First, the elementary example is examined with the following formulation:

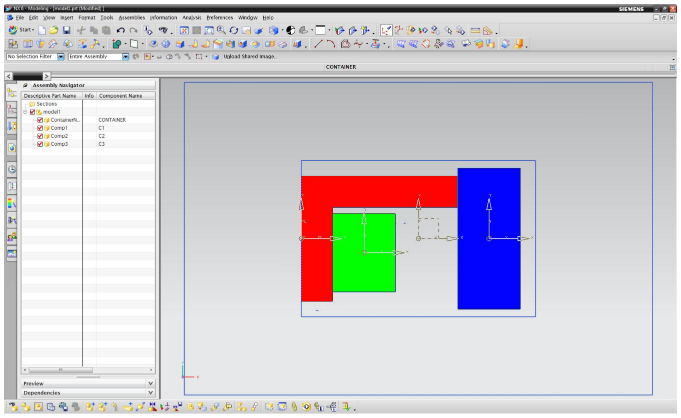
$$\begin{aligned} \underset{\mathbf{x}_g}{min} \quad & f_g(\mathbf{x}_g) \\ \text{subject to} \quad & g_1 = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} Vol(C_i \cap C_j) \leq 0 \\ \text{where} \quad & \mathbf{x}_g = (x_2, y_2, x_3) \\ & f_g(\mathbf{x}_g) = \|\mathbf{P}_{C_1} - \mathbf{P}_{C_2}\| + \mathbf{BB}_x \end{aligned}$$

$\mathbf{BB}_x$ is the length of the bounding box in X-axis direction, which represents the compactness of the system. Figure 3.18 shows the result with GA, and also verifys the developed computational environment.

An optimization result of the running example is shown in Figure 3.19. The population size is 170, and GA stops at 26th generation. The number of function evaluation is 4951. The used termination criterion is that the relative change in percentage with respect to the previous best objective function value is less than 0.001. The compactness is improved by 5% while satisfying the non-overlapping



(a) Initial layout



(b) Result

Figure 3.18: Optimization results of packaging problem for the elementary example constraint.

3.5 Integrated Problem

3.5.1 Formulation

First, All-in-one formulation with the sum of the objective functions is examined:

$$\begin{aligned} \min_{\mathbf{x}=[\mathbf{x}_p,\mathbf{x}_g]} \quad & f_p(\mathbf{x}) + f_g(\mathbf{x}) \\ \text{subject to} \quad & g_1 = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} Vol(C_i \cap C_j) \leq 0 \\ & \mathbf{g}_2 = [\mathbf{g}_{p2}, ..., \mathbf{g}_{p7}] \end{aligned} \tag{3.4}$$

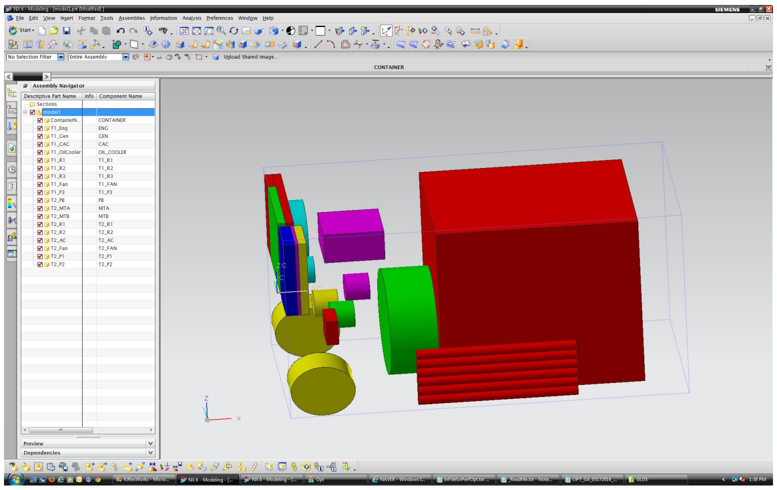
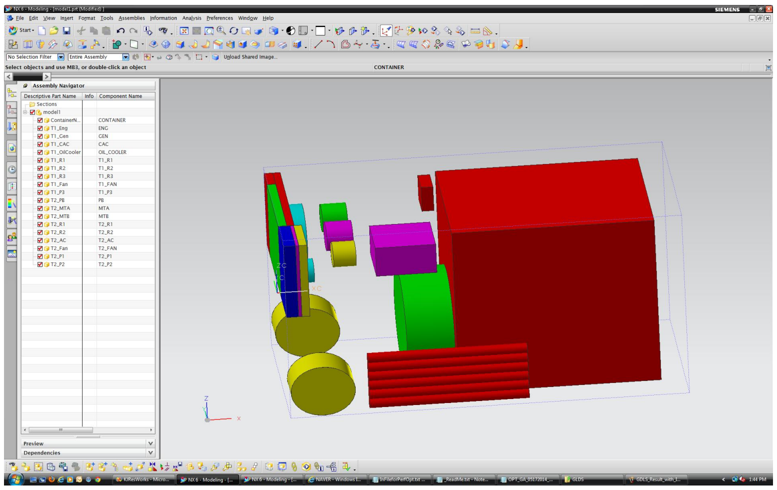


Figure 3.19: An optimization result of the running example

In addition to the design variables in Equation 3.2, the positions of the components that have more than one of DOF are added as the design variables. $\mathbf{g}_p$ are constraints in Equation 3.2 except g_1, which is not necessary because the non-overlapping constraint is computed more rigorously with real CAD model.

This All-in-one approach can be applied to the elementary example. However, for the running example, the total number of design variables is increased from 15 for performance and 17 for packaging to 32. Moreover, the used optimization algorithms for performance and packaging optimization are different, which are CFSQP and GA, respectively. Also, as can be seen in 3.20, after applying the optimal size of performance optimization, the compactness of the system does not change dramatically. So, All-in-one approach is not suitable for the running example.

Sequential strategy is applied to the running example. In the sequential strategy each design is optimized once. The solution obtained the previous problem becomes the parameters for the following problem. In the running problem, the optimal sizes from the performance is applied to the shapes of components in the packaging problem as parameters.


NX 6 - Modeling - [model1.prt (Modified)]
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
No Selection Filter
Entire Assembly
Select objects and use MB3, or double-click an object
CONTAINER
Assembly Navigator
Preview
Dependencies

Descriptive Part Name	Info	Component Name
Sections		
model1		
ContainerN...		CONTAINER
T1_Eng		ENG
T1_Gen		GEN
T1_CAC		CAC
T1_OilCooler		OIL_COOLER
T1_R1		T1_R1
T1_R2		T1_R2
T1_R3		T1_R3
T1_Fan		T1_FAN
T1_P3		T1_P3
T2_PB		PB
T2_MTA		MTA
T2_MTB		MTB
T2_R1		T2_R1
T2_R2		T2_R2
T2_AC		T2_AC
T2_Fan		T2_FAN
T2_P1		T2_P1
T2_P2		T2_P2

Figure 3.20: Initial layout with the optimal sizes from performance optimization

3.5.2 Results and Discussion

First, the optimization results of the elementary problem are shown in Figure 3.21. All-in-one formulation is used. Figure 3.21(c) would be different if the weight for the packaging objective function is higher. The area of C_2 is more important than the length of the bounding box of the components.

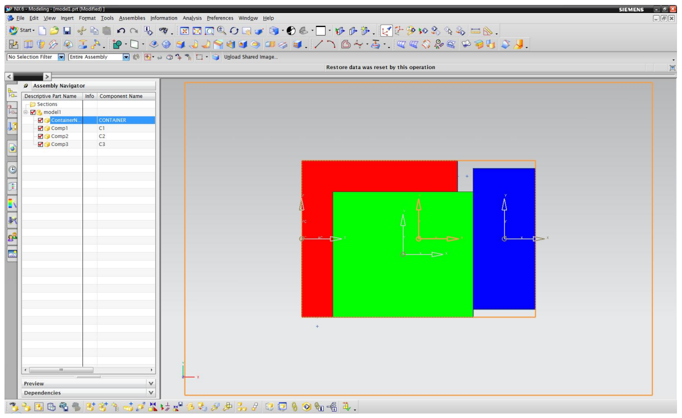
The optimization for the running example is sequentially solved with GA, which means the optimization for functionality is solved first, then the optimal design is passed to the optimization for packaging because the size changes in radiators, fans, and electric pumps are not significant in the packaging problem, as explained in the previous chapter. The population size is 170, and GA stops at 49th generation. The number of function evaluation is 8541. The same termination criterion used in the packaging problem is applied. The compactness is improved by 10% while satisfying the non-overlapping constraint. The result is illustrated in 3.22.

3.6 Concluding Comments

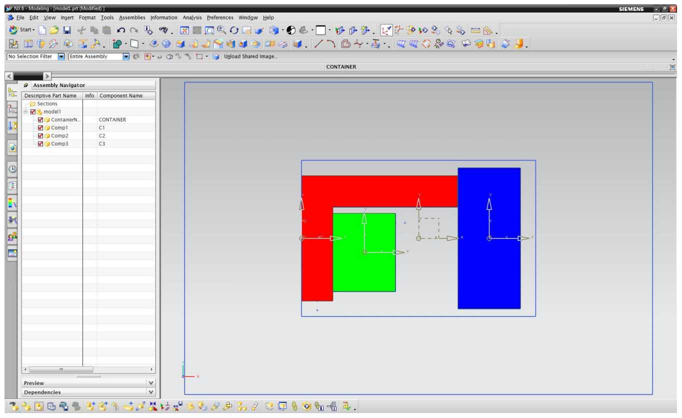
The software implementation issues and solutions are explained, then the interface for MATLAB is proposed. With the elementary example, the proposed interface is proven to work well. Then, the optimization problems for functionality, packaging, and integrated are formulated and the results are presented.

Using the proposed framework, optimization study of both functionality and geometric realization can be done on the same computation environment. Since commercial CAD modeler and its API are used, most geometric requirements can be programmed and computed. By providing or being able to develop interfaces, integrations with other analysis tools are possible. Also, feedback from the result can easily be given to engineers because the software that they use for designing and optimizing products are the same. Although the integrated optimization problem can be run on the same environment, it increases the number of design variables. If we deal with 20 components with 6 DOFs for each, total 120 variables are added to the design variables of a performance problem. This might take very long time to converge. Also, implementing both problems requires knowledge of both domains, which means engineers should understand the performance model, and know geometry and CAD modeler programming.

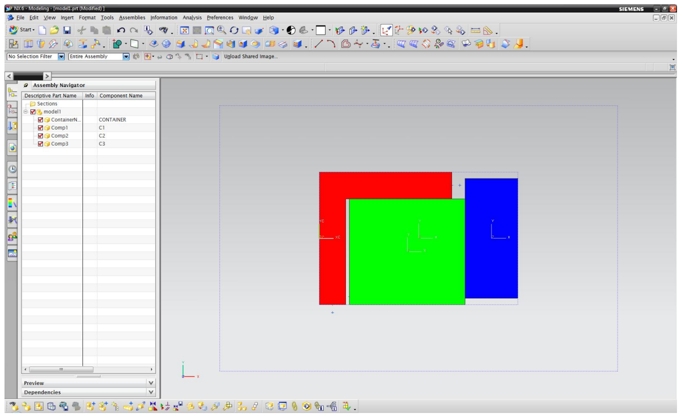
The packaging problem is extended with pipe routing consideration in the next chapter.



(a) Functionality

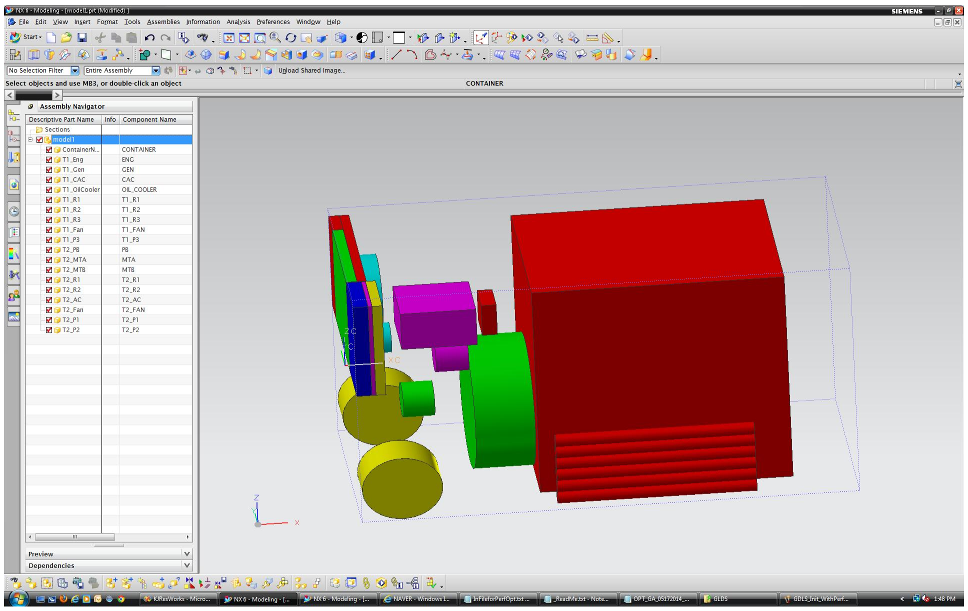


(b) Packaging



(c) Functionality with packaging

Figure 3.21: Optimization results of the elementary example



NX 6 - Modeling - [model1.prt (Modified)]

Select objects and use MB3, or double-click an object

Descriptive Part Name	Info	Component Name
Sections		
model1		
ContainerN...		CONTAINER
T1_Eng		ENG
T1_Gen		GEN
T1_CAC		CAC
T1_OilCooler		OIL_COOLER
T1_R1		T1_R1
T1_R2		T1_R2
T1_R3		T1_R3
T1_Fan		T1_FAN
T1_P3		T1_P3
T2_PB		PB
T2_MTA		MTA
T2_MTB		MTB
T2_R1		T2_R1
T2_R2		T2_R2
T2_AC		T2_AC
T2_Fan		T2_FAN
T2_P1		T2_P1
T2_P2		T2_P2

Figure 3.22: Optimization result with the optimal sizes from the performance optimization

CHAPTER IV

Pipe Routing

This chapter describes a pipe routing problem in mechanical system design and its implementation using robot motion planning algorithm.

4.1 Introduction

In this dissertation, finding the shortest path avoiding interference with the components and other pipes and cables is referred to as pipe routing. This problem has been widely studied in many fields such as cable routing in electrical engineering, chemical and power plant design, building design, and automobile and aerospace industry. Pipe routing is also known as an NP-hard problem, and heavily depends on human experience [81].

Pipes, hoses, and electrical cables in a system performance model are generally represented by lines that do not have volume and mass. In a real world, however, without considering these components, a packaging problem in a mechanical system cannot be completed because the space that is necessary for their own shapes and the additional space for assembly are not negligible. In addition to no overlapping among the components, there should be feasible paths for pipe to be a feasible layout for a system.

To integrate the pipe routing problem into an optimization routine, fast generation of pipes is preferred. The role of piping routing in the problem is to find the good estimation of pipe length, and to see if a feasible layout exists, so pipe generation is formulated as an analysis function, not as an optimization problem.

The order of pipe generation is important because the pipes that is previously generated are used as obstacles for the next pipe generation. Human experience, such as from inside to outside, thick pipe to thin pipe, and short pipe to long pipe, are used to determine the piping order [84]. In this dissertation, the piping order are predetermined by human, and given to the problem as a parameter.

4.2 Previous Work

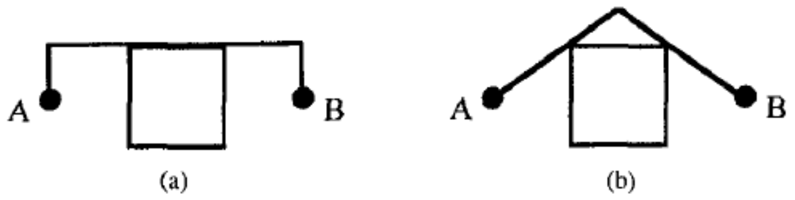


Figure 4.1: Manhattan (a) vs. nonorthogonal (b) routing with obstacles reproduced from [69]

As shown in Figure 4.1, pipe shapes can generally divided into two shapes: orthogonal and non-orthogonal. Orthogonal route bends 90 degree, which is widely used in chemical plant design [33]. In mechanical system design, assuming pipe shape as orthogonal route looks impractical. Szykman and Cagan focus on developing a pipe routing algorithm to create non-orthogonal routes [69]. SA is used as the optimization algorithm to obtain the optimal routes.

Some researches formulates a pipe routing problem as an optimization problem. the number of bends and the locations of the bend are design variables [64, 71, 78], and GA, SA, and extended search methods are applied.

On the other hand, as pointed out in [85], pipe routing problems can be seen as robot path planning with many constraints. When a set of obstacles, the initial position of a robot, and the final position of a robot are given, robot path planning algorithms are to find a path that moves the robot from the initial to final position avoiding the obstacles at all times. When the sphere shape robot with the three degree of freedom for translation in x, y, and z directions finds a path, it is the same as pipe generation.

4.3 Proposed Method

As explained in the previous section, because of the similarity between pipe routing and robot path planning, this dissertation uses a robot path planning algorithm to generate the pipes in a system. At the early stage of design, the approximate evaluation of pipe length and pipe locations would be enough.

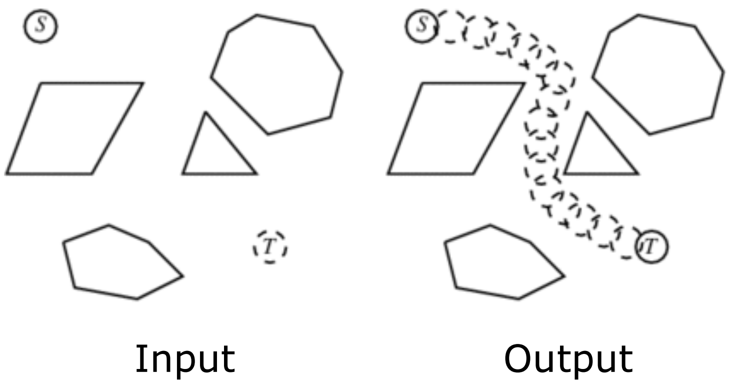


Figure 4.2: Motion planning reproduced from [65]

Robot motion planning is a large research field and also a difficult problem. This problem seeks to the path from the initial configuration to the final configuration, as appeared in Figure 4.2. Many approaches are proposed to address this problem, such as potential function, roadmaps, cell decomposition, and sampling-based algorithms [14].

Among the algorithms, sample-based motion planning have been successfully used to solve high degree-of-freedom motion planning problems arising in different applications. Many sampling strategies have been proposed to improve the performance. However, finding a more efficient method to generate the pipes is not the goal of this research. So, instead of implementing the algorithm, adopting and integrating the existing algorithm approach is chosen.

4.4 Software Implementation

This section describes the implementation issues when the library for a robot motion planning algorithm is integrated with the developed computational environment.

4.4.1 Motion Planning Kit

Motion Planning Kit (MPK) is a C++ library and toolkit for developing single- and multi-robot motion planning applications, which is developed in Stanford AI Laboratory [49]. In addition to MPK, two more separate libraries are required to link MPK, which are Coin3D/SoWin and Proximity Query Package (PQP). MPK uses Coin3D/SoWin for GUI and graphic library, and POP for collision detection.

Coin3D/SoWin

Open Inventor is an object-oriented, cross-platform 3D graphics toolkit, and its free windows port is Coin3D. SoWin library is a C++ GUI toolkit that binds together the Coin with the user interface parts of the Windows Win32 API. This is why MPK uses the Open Inventor file format for its triangulated geometry models. Furthermore, MPK extends Open Inventor's scene file format for defining planning scenarios, and uses Open Inventor classes for graphics output and a simple keyboard command interface.

Proximity Query Package

PQP is a library for static collision checking and distance computation that uses oriented bounding boxes (OBBs) and rectangle swept spheres (RSSs), which can perform three types of proximity queries on a pair of geometric models composed of triangles:

- Collision detection - detecting whether the two models overlap, and optionally, all of the triangles that overlap
- Distance computation - computing the minimum distance between a pair of models, i.e., the distance between the closest pair of points
- Tolerance verification - determining whether two models are closer or farther than a tolerance distance

PQP is developed by GAMMA research group at the University of North Carolina at Chapel Hill [59].

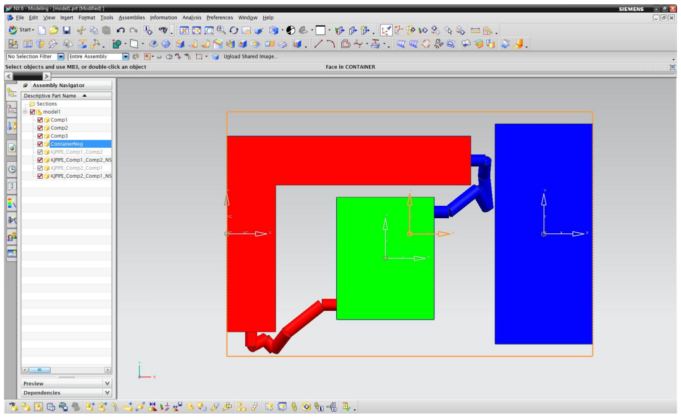
Path smoothing

MPK has smoothing functionality, which is very useful. As can be seen in Figure 4.3, the better pipe path can be obtained after applying the smoothing functionality of MPK. Because MPK samples the space, and connects the collision free points, the path can be not smooth.

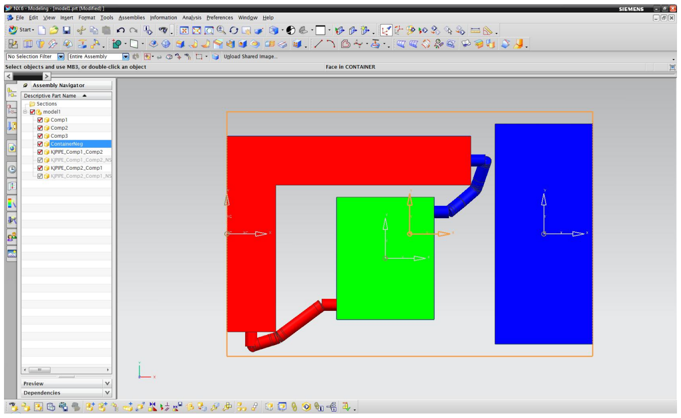
4.4.2 Implementation Issues

Definition of Ports

Figure 4.4 shows the information that is needed to define the input and output ports for piping. The port names and diameter for a pipe can be defined a priori, but the location of the ports should be evaluated during optimization. The reference



(a) Before smoothing



(b) After smoothing

Figure 4.3: Path smoothing

points are generated in a CAD model and named, as shown in Figure 4.5. When the component is moved or resized, the reference point can be relocated because it is created based on the relations with the existing geometry entities such as vertex, edges, and faces.

Once the reference points are created on the CAD files, the information for the pipe generation is added to the input file for optimization, as illustrated in Figure 4.6. In the figure two pipes are generated during optimization. The first one connects the point of which name is ‘P_OUT_C2’ in ‘comp1’ and the ‘P_IN_C1’ in ‘Comp2’ with green color, and its diameter is 5 (mm). Then, the second pipe is generated in red with 5 (mm) diameter. Note that if the first pipe is failed to find a path, the

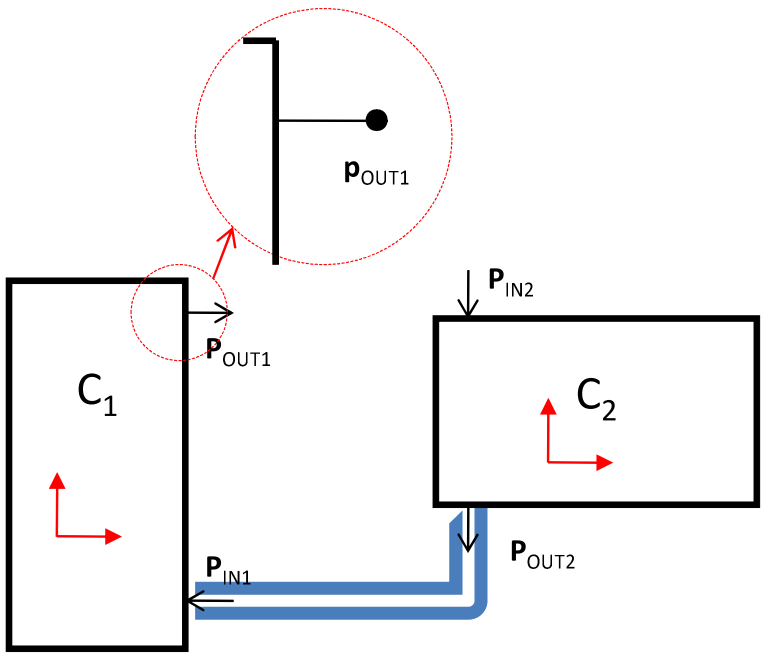


Figure 4.4: Input and output ports for a pipe

second pipe does not try to find a path. Also, note that the order of pipe generation is predetermined, as discussed, but it can be easily changed by switching the line in the input text file.

File Format

Several files are required to run MPK, which are for robot shape definition, obstacles, and scenes. As explained in Section 4.4.1, Open Inventor file format is used, so the geometry data of the components must be exported as the Open Inventor file format to compute the pipe path. Unfortunately, NX CAD modeler dose not support the export to the Open Inventor file, so the Open Inventor file format are studied, and the codes to export the geometry to Open Inventor file is developed. Interface module is developed with triangular mesh data. It is relatively easy to develop an exporting code for primitives such as box, cylinder, and sphere, but general shapes should also be able to be exported in the form of triangular mesh data. Transformation matrix that is a 4X4 matrix represents the rotation and translation of a component must

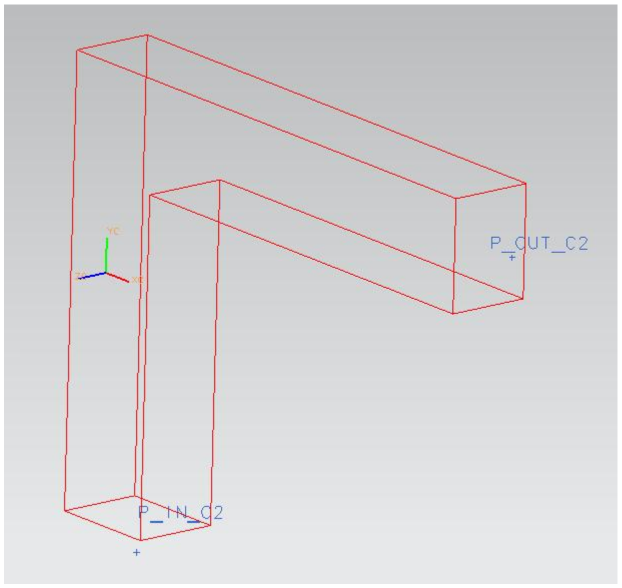
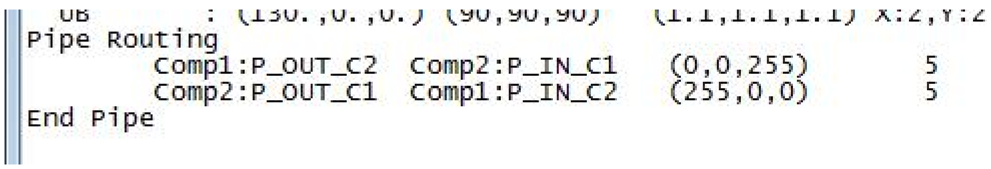


Figure 4.5: Defining the port locations in CAD systems



```
  OB        : (130.,0.,0.) (90,90,90)    (1.1,1.1,1.1) X:2,Y:2
Pipe Routing
        Comp1:P_OUT_C2  Comp2:P_IN_C1   (0,0,255)      5
        Comp2:P_OUT_C1  Comp1:P_IN_C2   (255,0,0)      5
End Pipe
```

Figure 4.6: Information for pipe generation in the input file

be exported, too. This transformation matrix is extracted from the transformation matrix from an assembly in a CAD modeler.

Creating Pipe Part in CAD system

MPK generates the list of points for the pipe path, from which the pipe length can be calculated. This pipe length, however, is not enough to consider the pipes in packaging problems. For example, When the path for the first pipe is calculated, the path for the second pipe must avoid the first pipe as well as other obstacles in a system. When MPK produces the list of points, the geometry of the pipe must be created in CAD system, and exported to MPK too. Multiple solid bodies of cylinder shapes are currently created in a part file.

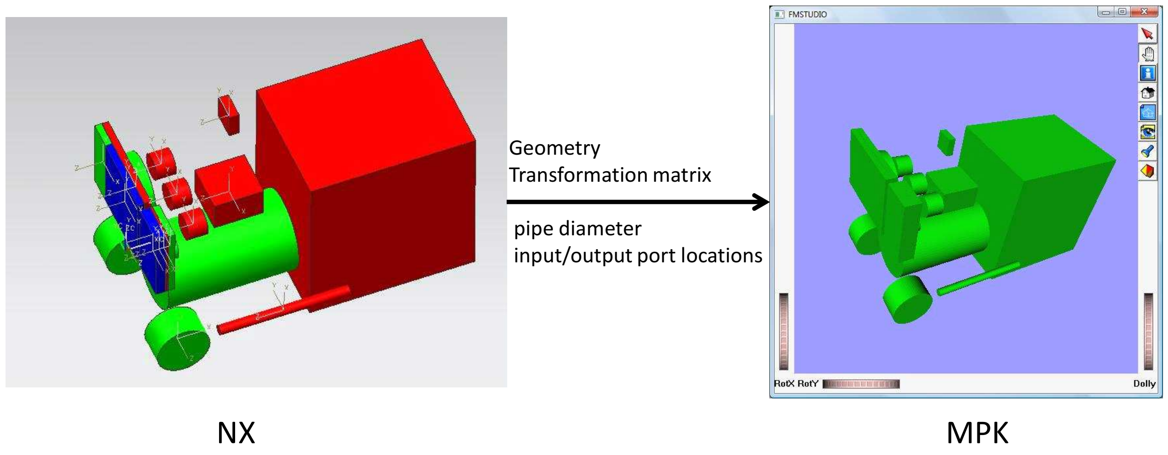


Figure 4.7: Export of geometry from NX to MPK

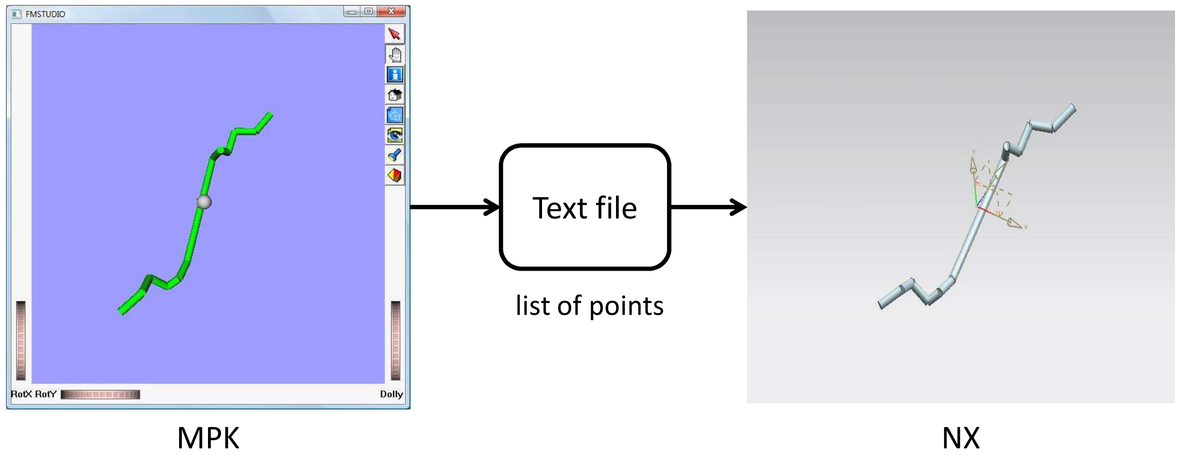


Figure 4.8: Creation of pipe geometry

Execution Issues

After MPK library is successfully integrated and tested, the code is called during optimization. At the first run, the codes works fine, but at the second run the program crashes. Although many attempts including changing various compile options are made to find a solution to the problem, the exact reasons of the crash cannot be found. Therefore, as proposed to MATLAB interface, another standalone program is developed, and called form the main computational environment in batch mode, as illustrated in Figure 4.9.

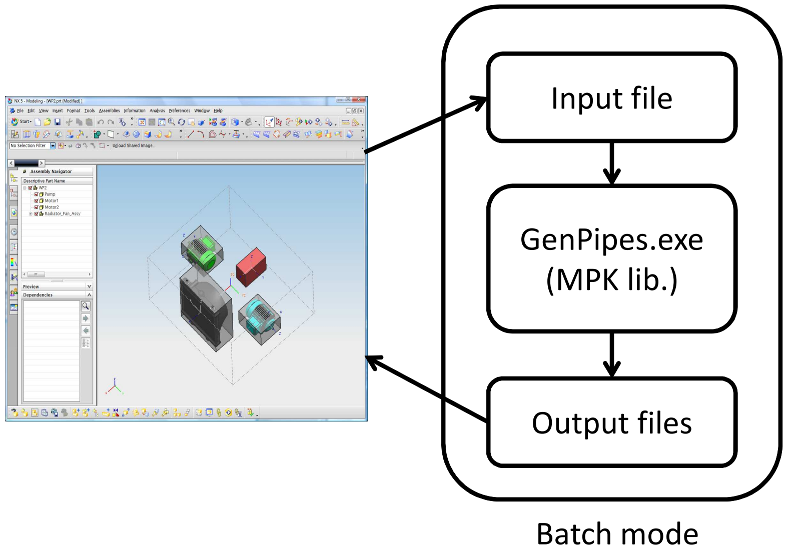


Figure 4.9: Integration of MPK library

4.5 Problem Formulation

Although the performance model for pipes is not created in the running problem, it is believed that the shorter is the better in the pipe design problem. Pressure drop models for pipes can be added to the performance later based on the pipe length created in the geometry problem. Pipe generation is only executed when there is no interference.

$$\begin{aligned} \underset{\mathbf{x}_g}{min} \quad & f_g(\mathbf{x}_g) + L_{pipe} \\ \text{subject to} \quad & g_1 = \sum_{i=0}^{N-1} \sum_{j=i+1}^{N} Vol(C_i \cap C_j) \leq 0 \\ \text{where} \quad & L_{pipe} = \begin{cases} \infty & \text{if } g_1 \neq 0 \\ \sum_{i=1}^{N_{pipe}} L_{pipe_i} & \text{if } g_1 = 0 \end{cases} \end{aligned} \tag{4.1}$$

Objective function

The summation of all pipes' length is added to the objective function of the packaging optimization problem in Equation 3.3 to minimize the total pipe length.

Design Variables

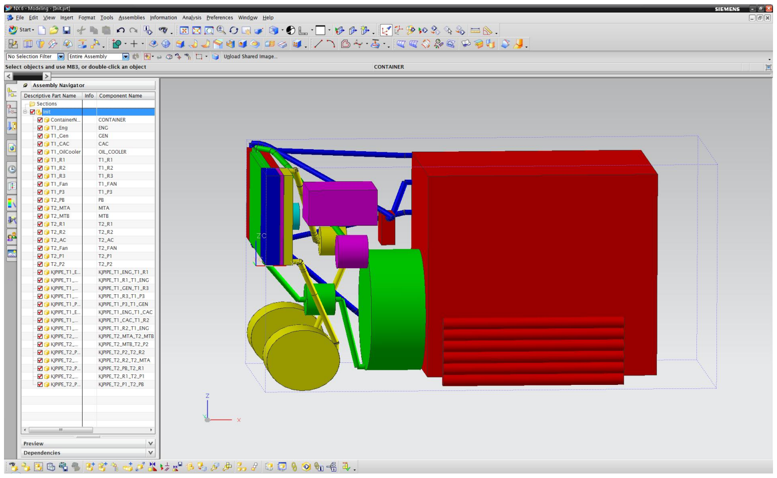
The design variables are the same as those of the packaging problem. Since, given the location input and output ports, the pipes are automatically generated while avoiding collision with the components and the previously generated pipes, design variables such as the number and/or location of bends are not included. Also, the locations of engine and oil cooler are fixed with the optimization result of packaging problem.

Constraints

g_1 is a non-overlapping constraint among the components in the system.

As explained earlier, the predetermined order of pipe generation based on heuristics is given to the optimization problem. Once the location of components is fixed, the pipes are automatically generated by the given order. So, the initial pipe path is determined by the initial positions of components. If the specific location must be on the path for a certain pipe as an additional constraint, the location can be inserted between input and output ports so that the two pipes are internally generated. This functionality is not implemented in this research, but can easily be enhanced in the computational environment.

Pipes are generated based on the sample-based algorithm, which means that the generated pipe shapes can be different for each run. However, the shapes show that there is a feasible solution at least, which can answer the question of whether the pipes can be placed in a system given locations of the components. Also, their length after smoothing can be good estimation in an early design stage.

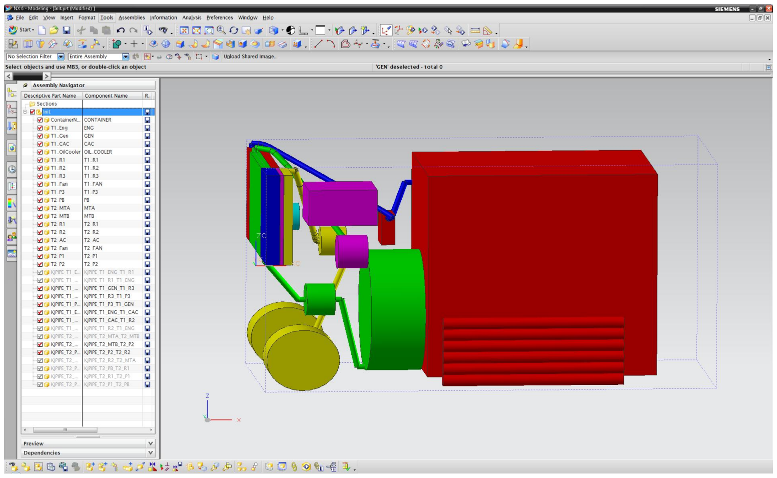

NX 6 - Modeling - [Init.prt]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
Start
No Selection Filter
Entire Assembly
Upload Shared Image...
Assembly Navigator
Preview
Dependencies

Select objects and use MB3, or double-click an object

CONTAINER

Descriptive Part Name	Info	Component Name
Sections		
Init		
ContainerN...		CONTAINER
T1_Eng		ENG
T1_Gen		GEN
T1_CAC		CAC
T1_OilCooler		OIL_COOLER
T1_R1		T1_R1
T1_R2		T1_R2
T1_R3		T1_R3
T1_Fan		T1_FAN
T1_P3		T1_P3
T2_PB		PB
T2_MTA		MTA
T2_MTB		MTB
T2_R1		T2_R1
T2_R2		T2_R2
T2_AC		T2_AC
T2_Fan		T2_FAN
T2_P1		T2_P1
T2_P2		T2_P2
KJPIPE_T1_E...		KJPIPE_T1_ENG_T1_R1
KJPIPE_T1_...		KJPIPE_T1_R1_T1_ENG
KJPIPE_T1_...		KJPIPE_T1_GEN_T1_R3
KJPIPE_T1_...		KJPIPE_T1_R3_T1_P3
KJPIPE_T1_P...		KJPIPE_T1_P3_T1_GEN
KJPIPE_T1_E...		KJPIPE_T1_ENG_T1_CAC
KJPIPE_T1_...		KJPIPE_T1_CAC_T1_R2
KJPIPE_T1_...		KJPIPE_T1_R2_T1_ENG
KJPIPE_T2_...		KJPIPE_T2_MTA_T2_MTB
KJPIPE_T2_...		KJPIPE_T2_MTB_T2_P2
KJPIPE_T2_P...		KJPIPE_T2_P2_T2_R2
KJPIPE_T2_...		KJPIPE_T2_R2_T2_MTA
KJPIPE_T2_P...		KJPIPE_T2_PB_T2_R1
KJPIPE_T2_...		KJPIPE_T2_R1_T2_P1
KJPIPE_T2_P...		KJPIPE_T2_P1_T2_PB

(a) Initial pipe routing


NX 6 - Modeling - [Init.prt (Modified)]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
Start
No Selection Filter
Entire Assembly
Upload Shared Image...
Assembly Navigator
Preview
Dependencies

Select objects and use MB3, or double-click an object

'GEN' deselected - total 0

Descriptive Part Name	Component Name	R.
Sections		
Init		
ContainerN...	CONTAINER	
T1_Eng	ENG	
T1_Gen	GEN	
T1_CAC	CAC	
T1_OilCooler	OIL_COOLER	
T1_R1	T1_R1	
T1_R2	T1_R2	
T1_R3	T1_R3	
T1_Fan	T1_FAN	
T1_P3	T1_P3	
T2_PB	PB	
T2_MTA	MTA	
T2_MTB	MTB	
T2_R1	T2_R1	
T2_R2	T2_R2	
T2_AC	T2_AC	
T2_Fan	T2_FAN	
T2_P1	T2_P1	
T2_P2	T2_P2	
KJPIPE_T1_E...	KJPIPE_T1_ENG_T1_R1	
KJPIPE_T1_...	KJPIPE_T1_R1_T1_ENG	
KJPIPE_T1_...	KJPIPE_T1_GEN_T1_R3	
KJPIPE_T1_...	KJPIPE_T1_R3_T1_P3	
KJPIPE_T1_P...	KJPIPE_T1_P3_T1_GEN	
KJPIPE_T1_E...	KJPIPE_T1_ENG_T1_CAC	
KJPIPE_T1_...	KJPIPE_T1_CAC_T1_R2	
KJPIPE_T1_...	KJPIPE_T1_R2_T1_ENG	
KJPIPE_T2_...	KJPIPE_T2_MTA_T2_MTB	
KJPIPE_T2_...	KJPIPE_T2_MTB_T2_P2	
KJPIPE_T2_P...	KJPIPE_T2_P2_T2_R2	
KJPIPE_T2_...	KJPIPE_T2_R2_T2_MTA	
KJPIPE_T2_P...	KJPIPE_T2_PB_T2_R1	
KJPIPE_T2_...	KJPIPE_T2_R1_T2_P1	
KJPIPE_T2_P...	KJPIPE_T2_P1_T2_PB	

(b) Reduced pipe routing

Figure 4.10: Initial layout for the running example with pipe routing

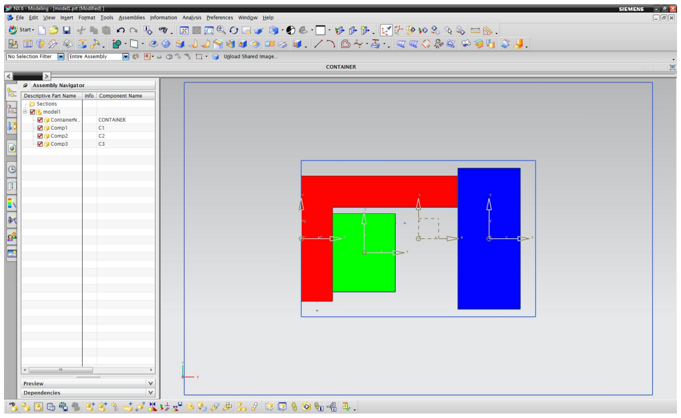
Figure 4.10(a) shows the initial layout with pipe routing. Among total 15 pipe routings, some connections are excluded for optimization study because either diameter of pipes are smaller than others or locations of the connected two components are fixed. 7 pipe routings are generated during optimization, as shown in Figure 4.10(b). Also, the locations of components are given from the packaging problem. Therefore, sequential approach is also used for the packaging problem with pipe routing of the running example.

4.6 Result and Discussion

The revised optimization problem for packaging with pipe routing is solved with GA because the length of the pipes are not smooth and continuous. MPK is the sampling-based algorithm, so the generated path is different at each run. Path is created only when the layout has no overlapping.

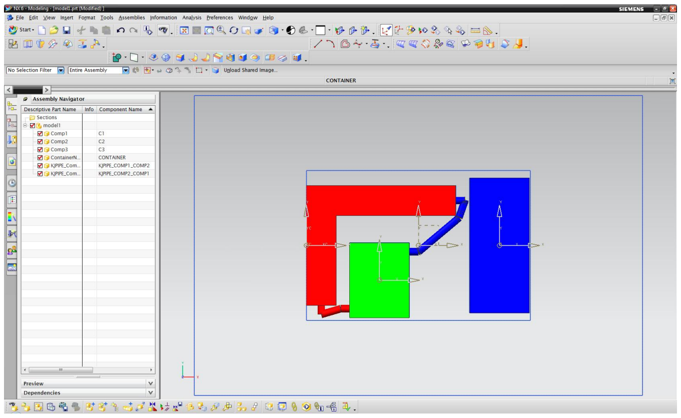
Figure 4.11 shows the results of optimization problem for packaging problem for the elementary example. This result shows that pipe routing should be considered and formulated in optimization problem for packaging in mechanical system. When the pipe routing is considered, the layout compactness decreases to make space for the pipes.

Figure 4.12(a) shows an optimization result of the running example. The population size is 150, and GA stops at 20th generation, which is the number set as the maximum generation. The number of function evaluation is 3151. The used termination criterion is that the relative change in percentage with respect to the previous best objective function value is less than 0.1. The compactness is decreased by 2%, but the pipe length is improved by 38% while satisfying the non-overlapping constraint. This result is consistent with that of the elementary example. Although the better solution in terms of overall compactness and total pipe length is found, the algorithm is terminated at the maximum generation. This is because the pipe generation algorithm is based on sampling, which is not deterministic. This characteristic makes GA hard to converge. All 15 pipe routings are generated after optimization to see whether the excluded pips can be created in the system, as shown in Figure 4.12(b).


NX 6 - Modeling - [model1.prt (Modified)]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
No Selection Filter
Entire Assembly
Upload Shared Image...
CONTAINER
Assembly Navigator
Preview
Dependencies

Descriptive Part Name	Info	Component Name
Sections		
model1		
ContainerN...		CONTAINER
Comp1		C1
Comp2		C2
Comp3		C3

(a) Original packaging problem


NX 6 - Modeling - [model1.prt (Modified)]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
No Selection Filter
Entire Assembly
Upload Shared Image...
CONTAINER
Assembly Navigator
Preview
Dependencies

Descriptive Part Name	Info	Component Name
Sections		
model1		
Comp1		C1
Comp2		C2
Comp3		C3
ContainerN...		CONTAINER
KJPIPE_Com...		KJPIPE_COMP1_COMP2
KJPIPE_Com...		KJPIPE_COMP2_COMP1

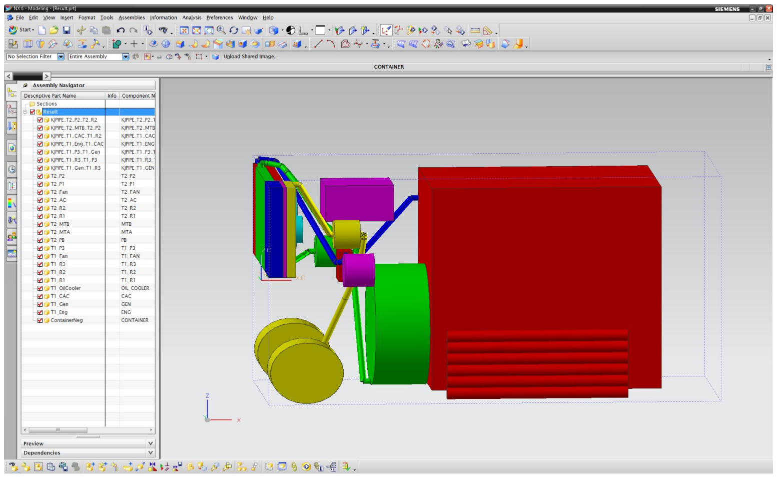
(b) Packaging problem with pipe routing consideration

Figure 4.11: Optimization results of revised packaging problem for the elementary example

4.7 Concluding Comments

A pipe routing problem in mechanical system design is explained in this chapter. Pipe routing is addressed vis robot motion planning library. The implementation issues are discussed, and their solutions are presented to help researchers who works on the same or similar problems. An optimization problem for packaging problem with pipe routing consideration is formulated by adding the pipe length to the objective function. The optimization results shows the usefulness of the proposed method and formulation.

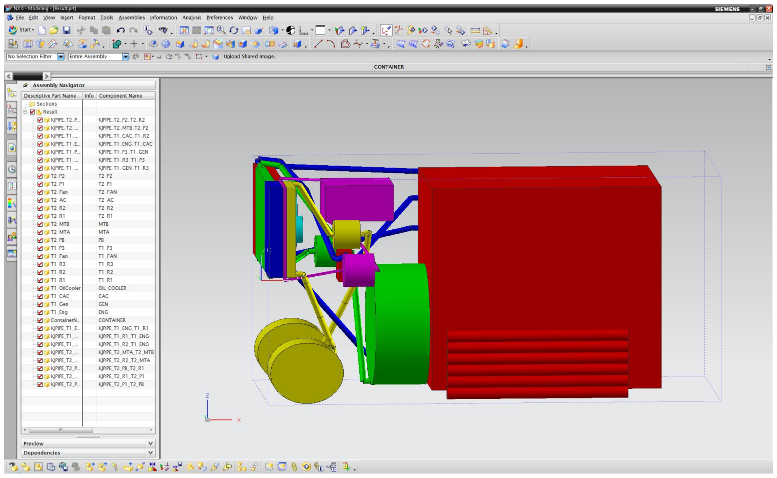
Performance, packaging, and packaging with pipe routing problems are solved in



NX 6 - Modeling - [Result.prt]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
Start
No Selection Filter
Entire Assembly
Upload Shared Image...
CONTAINER
Assembly Navigator
Preview
Dependencies

Descriptive Part Name	Info	Component N
Sections		
Result		
KJPIPE_T2_P2_T2_R2		KJPIPE_T2_P2_T
KJPIPE_T2_MTB_T2_P2		KJPIPE_T2_MTB
KJPIPE_T1_CAC_T1_R2		KJPIPE_T1_CAC
KJPIPE_T1_Eng_T1_CAC		KJPIPE_T1_ENG
KJPIPE_T1_P3_T1_Gen		KJPIPE_T1_P3_T
KJPIPE_T1_R3_T1_P3		KJPIPE_T1_R3_
KJPIPE_T1_Gen_T1_R3		KJPIPE_T1_GEN
T2_P2		T2_P2
T2_P1		T2_P1
T2_Fan		T2_FAN
T2_AC		T2_AC
T2_R2		T2_R2
T2_R1		T2_R1
T2_MTB		MTB
T2_MTA		MTA
T2_PB		PB
T1_P3		T1_P3
T1_Fan		T1_FAN
T1_R3		T1_R3
T1_R2		T1_R2
T1_R1		T1_R1
T1_OilCooler		OIL_COOLER
T1_CAC		CAC
T1_Gen		GEN
T1_Eng		ENG
ContainerNeg		CONTAINER

(a) Result



NX 6 - Modeling - [Result.prt]
SIEMENS
File Edit View Insert Format Tools Assemblies Information Analysis Preferences Window Help
Start
No Selection Filter
Entire Assembly
Upload Shared Image...
CONTAINER
Assembly Navigator
Preview
Dependencies

Descriptive Part Name	Info	Component Name
Sections		
Result		
KJPIPE_T2_P...		KJPIPE_T2_P2_T2_R2
KJPIPE_T2_...		KJPIPE_T2_MTB_T2_P2
KJPIPE_T1_...		KJPIPE_T1_CAC_T1_R2
KJPIPE_T1_E...		KJPIPE_T1_ENG_T1_CAC
KJPIPE_T1_P...		KJPIPE_T1_P3_T1_GEN
KJPIPE_T1_...		KJPIPE_T1_R3_T1_P3
KJPIPE_T1_...		KJPIPE_T1_GEN_T1_R3
T2_P2		T2_P2
T2_P1		T2_P1
T2_Fan		T2_FAN
T2_AC		T2_AC
T2_R2		T2_R2
T2_R1		T2_R1
T2_MTB		MTB
T2_MTA		MTA
T2_PB		PB
T1_P3		T1_P3
T1_Fan		T1_FAN
T1_R3		T1_R3
T1_R2		T1_R2
T1_R1		T1_R1
T1_OilCooler		OIL_COOLER
T1_CAC		CAC
T1_Gen		GEN
T1_Eng		ENG
ContainerN...		CONTAINER
KJPIPE_T1_E...		KJPIPE_T1_ENG_T1_R1
KJPIPE_T1_...		KJPIPE_T1_R1_T1_ENG
KJPIPE_T1_...		KJPIPE_T1_R2_T1_ENG
KJPIPE_T2_...		KJPIPE_T2_MTA_T2_MTB
KJPIPE_T2_...		KJPIPE_T2_R2_T2_MTA
KJPIPE_T2_P...		KJPIPE_T2_PB_T2_R1
KJPIPE_T2_...		KJPIPE_T2_R1_T2_P1
KJPIPE_T2_P...		KJPIPE_T2_P1_T2_PB

(b) Result with all pipe routing

Figure 4.12: Optimization result for revised packaging problem for the running example

sequence. Starting locations of components for pipe routing problem are given from packaging problem to find a better solution in terms of compactness and pipe lengths.

However, since a sample-based algorithm is used, maintaining the good path from the previous generation is not easy during iterations. To achieve this, in addition to the design variables, which are locations and orientations, each position of all bends needs to be stored. This is not implemented in this research, but would be good enhancement for future.

CHAPTER V

Conclusion

5.1 Summary

This dissertation proposed the framework and computational environment that can address system optimization problems with geometry consideration.

Chapter II explained geometric realization at both component-level and system-level. First, the issues related to component-level realization are explained. Various issues related computational geometry such as abstract geometry, bounding volumes, and convex and concave shapes are discussed because these issues affects the problem complexity, problem formulation, and computational performance of the optimization. Also, various options to represent the geometry are explained. Then, the solution methods to address 3D packaging problems are reviewed. Because of the combinatorial characteristic of the packaging problem, heuristic methods, GA, SA, and extended pattern search methods are applied with no guarantee that it finds the best one.

Based on the developed computational environment, Chapter III shows the process to solve the running example. Implementation issues when integrating other software, especially MATLAB and Simulink, are addressed, which is practically important. The optimization problems for functionality and packaging are formulated and solved respectively, then the integrated problem is formulated and solved using sequential approach.

Chapter IV explains how to integrate pipe routing problems into the packaging problem. Pipe routing problems are also NP-hard problems as well as packing problem, so instead of formulation another optimization problem for pipe routing, robot motion path planning library that uses a sample-based algorithm is integrated in the computational environment to evaluate the feasibility of a layout considering pipe routing and the length of the generated pipes that is used to another objective function in a packaging problem. Implementation issues to integrate the path planning library are explained and addressed.

5.2 Contributions

The main contributions of this dissertation are summarized as follows:

- Integration of a system's abstract representation suitable for simulation with the system's actual embodiment in its geometric physical instantiation; and the use of such integration for overall performance and geometric layout optimization.

- A computational environment on a commercial CAD system is developed, which enables the optimization study that needs geometric computation with CAD models. Three optimization algorithms are integrated in the computational environment: one gradient based algorithm, CFSQP, two gradient-free algorithm, GA and SA. For more convenient use of those algorithms and easy integration, the wrapper C++ classes are developed for three algorithms. Also, because a commercial CAD system used as the foundation for the computational environment, the developed codes can be integrated more easily with the design process in a company.

- Pipe generation and evaluating the pipe length are addressed via integrating the motion planning algorithm in robot path planning research field. The pipe shapes are included in a packaging problem because the shapes of pipes are not negligible in mechanical system design. The library of motion planning algorithm is successfully integrated with the developed computational environment.

- The developed framework is applied to the thermal management system for a heavy duty tracked SHEV. With the assumed shape and size of the components, the packaging problem is formulated, and integrated with the optimization problem for functionality.

5.3 Future Work

The following research issues require future investigation.

Convex decomposition of the geometry

Although the automatic partitioning the convex shapes into the pieces of convex shapes are not possible, semi-automatic and meaningful methods for a specific application might be found. Although it increases the number of geometry in the problem, dealing with convex shapes has many advantages. Further research that examines the tradeoff between the number of convex geometry and computation time for both interference check and optimization would be interesting. Only convex shapes would be beneficial in formulating non-overlapping constraints.

Distributed formulation of the integrated problem

Because the research problem deals with the components in a system, this problem might be divided into subproblems that design each component. Parameters passed to the subproblems can be the location of a component and available space determined by surrounding components in a system. This available space can guarantee a non-interference layout. The group of some components can be treated as one big component in a higher level layout problem. Also, geometry itself can be targets and responses. Distributed formulation, such as ATC (Analytical Target Cascading), would be a good methodology to address the problem in this dissertation.

Interactive with engineers

With more sophisticated user interface, engineers can move the component wherever they want to locate, then run the optimization model with or without pipe routing routine. This enables engineers to visualize the layout and the results of optimization. Because of combinatorial characteristic of layout problems, all possible layouts cannot be explored. Engineers, however, would want to see different layouts and to progress the design more with some layouts to see if which one in the best for them. This is partly because objective functions for layout are difficult to be mathematically formulated in optimization model.

Other robot motion planning algorithms

More survey on robot motion planning algorithms and available libraries cam make more tight integration possible without making another stand alone problem, as implemented in this dissertation, which can reduce the efforts to compile and integrate.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] http://www.spatial.com/products/3d-acis-modeling.

[2] Chandankumar Aladahalli, Jonathan Cagan, and Kenji Shimada, "A Sensitivity-Based Pattern Search Algorithm for 3D Component Layout," *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Montreal, Quebec, Canada, September 29–October 2, 2002.

[3] Chandankumar Aladahalli, Jonathan Cagan, and Kenji Shimada, "Minimum Height Packing for Layered Manufacturing using an Extended Pattern Search Algorithm," *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Chicago, Illinois, September 2–6, 2003.

[4] Chandankumar Aladahalli, Jonathan Cagan, and Kenji Shimada, "Objective Function Based Pattern Search - A Computationally Efficient Algorithm for 3D Component Layout," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, September 28–October 2, 2004.

[5] Chandankumar Aladahalli, Jonathan Cagan, and Kenji Shimada, "Objective Function Effect Based Pattern Search - Theoretical Framework Inspired by 3D Component Layout," *Journal of Mechanical Design, Transactions of the ASME*, vol. 129, no. 3, pp. 243–254, 2007.

[6] Chandankumar Aladahalli, Jonathan Cagan, and Kenji Shimada, "Objective Function Effect Based Pattern Search - an Implementation for 3D Component Layout," *Journal of Mechanical Design, Transactions of the ASME*, vol. 129, no. 3, pp. 255–265, 2007.

[7] Vincent Y. Blouin, Yi Miao, Xun Zhou, and Georges M. Fadel, "An Assessment of Configuration Design Methodologies," *10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Albany, New York, August 30–September 1, 2004.

[8] Vincent Y. Blouin, Georges M. Fadel, Joshua D. Summers, and Peter A. Fenyes, "Three-Dimensional Packing by a Heuristic-Based Sequential Genetic Algorithm," *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, Virginia, September 6–8, 2006.

[9] Craig Lawrence, Jian L. Zhou, and Andre L. Tits, *User's Guide for CFSQP Version 2.5*. 1997.

[10] Jonathan Cagan, K. Shimada, and S. Yin, "A Survey of Computational Approaches to Three-Dimensional Layout Problems," *Computer-Aided Design*, vol. 34, pp. 597–611, 07, 2002.

[11] Jonathan Cagan, Drew Degentesh, and Su Yin, "A Simulated Annealing Based Algorithm using Hierarchical Models for General Three Dimensional Component Layout," *Computer-Aided Design*, vol. 30, no. 10, pp. 781–790, 1998.

[12] M. I. Campbell, C. H. Amon, and J. Cagan, "Optimal Three-Dimensional Placement of Heat Generating Electronic Components," *Journal of Electronic Packaging, Transactions of the ASME*, vol. 119, no. 2, pp. 106–113, 1997.

[13] Bernard Chazelle, "Convex Partitions of Polyhedra: A Lower Bound and Worst-Case Optimal Algorithm," *SIAM Journal on Computing*, vol. 13, no. 3, pp. 488–507, 1984.

[14] Howie Choset, Kevin M. Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia E. Kavraki, and Sebastian Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. 2005.

[15] Hoon Cho, Dohoy Jung, Zoran S. Filipi, Dennis N. Assanis, John Vanderslice, and Walter Bryzik, "Application of Controllable Electric Coolant Pump for Fuel Economy and Cooling Performance Improvement," *Journal of Engineering for Gas Turbines and Power, Transactions of the ASME*, vol. 129, no. 1, pp. 239–244, 2007.

[16] http://www.atlanticdda.com/vehiclesystems_engines_DD13.htm.

[17] John K. Dickinson and George K. Knopf, "Moment Based Metric for 2-D and 3-D Packing," *European Journal of Operational Research*, vol. 122, no. 1, pp. 133–144, 2000.

[18] Quan Ding and Jonathan Cagan, "Automated Trunk Packing with Extended Pattern Search," *SAE Technical Paper Series*, 2003-01-0671.

[19] Hong Dong, Georges M. Fadel, and Vincent Y. Blouin, "Packing Optimization by Enhanced Rubber Band Analogy," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Long Beach, California, September 24–28, 2005.

[20] Hong Dong, Georges M. Fadel, and Vincent Y. Blouin, "Vehicle Component Layout with Shape Morphing - an Initial Study," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Philadelphia, Pennsylvania, September 10–13, 2006.

[21] Friedrich Eisenbrand, Stefan Funke, Andreas Karrenbauer, Joachim Reichel, and Elmar Schomer, "Packing a Trunk - Now with a Twist!," *SPM 2005 - ACM Symposium on Solid and Physical Modeling*, Cambridge, Massachusetts, June 13–15, 2005.

[22] Christer Ericson, *Real-Time Collision Detection.* 2005.

[23] http://www.netlib.org/f2c.

[24] Georges M. Fadel, Avijit Sinha, and Todd McKee, "Packing Optimization using a Rubberband Analogy," *ASME Design Engineering Technical Conference and Computers and Information in Engineering Conference*, Pittsburgh, Pennsylvania, September 9–12, 2001.

[25] Kikuo Fujita, Masanori Kuriyama, and Takashi Suyama, "A Perspective of Hierarchical Layout Design Optimization for Highly Packaged Equipments," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Las Vegas, Nevada, September 4–7, 2007.

[26] Kunihiro Fujiyoshi, Hidenori Kawai, and Keisuke Ishihara, "A Tree Based Novel Representation for 3D-Block Packing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 5, pp. 759–764, 2009.

[27] http://www.iitk.ac.in/kangal/codes.shtml.

[28] Sungjin Park, Andreas Malikopoulos, Dohoy Jung, and Michael Kokkolaras, "Modeling and Optimization of Thermal Systems," Annual Report, University of Michigan, Ann Arbor, Michigan, 2007.

[29] Vladimir B. Gantovnik, Santosh Tiwari, Georges M. Fadel, and Yi Miao, "Multi-Objective Vehicle Layout Optimization," *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, Virginia, September 6–8, 2006.

[30] Pierre M. Grignon and Georges M. Fadel, "A GA Based Configuration Design Optimization Method," *Journal of Mechanical Design, Transactions of the ASME*, vol. 126, no. 1, pp. 6–15, 2004.

[31] Pierre M. Grignon, John R. Wodziak, and Georges M. Fadel, "Bi-Objective Optimization of Components Packing using a Genetic Algorithm," *AIAA, NASA, and ISSMO, Symposium on Multidisciplinary Analysis and Optimization*, Bellevue, Washington, September 4–6, 1996.

[32] Pierre M. Grignon and Georges M. Fadel, "Configuration Design Optimization Method," *ASME Design Engineering Technical Conferences*, Las Vegas, Nevada, September 12–15, 1999.

[33] Reginaldo Guirardello and Ross E. Swaney, "Optimization of Process Plant Layout with Pipe Routing," *Computers and Chemical Engineering*, vol. 30, no. 1, pp. 99–114, 2005.

[34] Pei-Ning Guo, Toshihiko Takahashi, Chung-Kuan Cheng, and Takeshi Yoshimura, "Floorplanning using a Tree Representation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 2, pp. 281–289, 2001.

[35] http://www.spatial.com/products/3d-visualization.

[36] W. Hills and N. Smith, "A New Approach to Spatial Layout Design in Complex Engineered Products," *International Conference on Engineering Design*, Tampere, Finland, August 19–21, 1997.

[37] J. Hur, K. Lee, J. Ahn, and H. C. Lee, "A Three-Dimensional Algorithm using Two-Dimensional Slice Data for Building Multiple Parts in Layered Manufacturing," *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture*, vol. 214, no. 5, pp. 365–378, 2000.

[38] Ilkka Ikonen, William E. Biles, Anup Kumar, and Rammohan K. Ragade, "Concept for a Genetic Algorithm for Packing Three Dimensional Objects of Complex Shape," *Proceedings of the First Online Workshop on Soft Computing*, Nagoya University, August 19–30, 1996.

[39] Ilkka Ikonen, William E. Biles, Anup Kumar, Rammohan K. Ragade, and John C. Wissel, "A Genetic Algorithm for Packing Three-Dimensional Non-Convex Objects having Cavities and Holes," *Proceedings of the 7th International Conference on Genetic Algorithms*, East Lansing, Michigan, July 19–23, 1997.

[40] Ilkka Ikonen, William E. Biles, James E. Lewis, Anup Kumar, and Rammohan K. Ragade, "GARP: Genetic Algorithm for Part Packing in a Rapid Prototyping Machine," *Intelligent Systems in Design and Manufacturing*, Boston, Massachusetts, November 2–4, 1998.

[41] Frank P. Incropera and David P. DeWitt, *Fundamentals of Heat and Mass Transfer.* 5th ed., 2002.

[42] Seung Ho Jang and Kyong Yop Rhee, "An Application of Annealing Algorithm to the Layout Design of Ocean Space Submergible Boat," *JSME International Journal, Series C: Mechanical Systems, Machine Elements and Manufacturing*, vol. 47, no. 2, pp. 770–775, 2004.

[43] Dohoy Jung and Dennis N. Assanis, "Numerical Modeling of Cross Flow Compact Heat Exchanger with Louvered Fins using Thermal Resistance Concept," *SAE Technical Paper Series*, 2006-01-0726.

[44] Masakazu Kobayashi, Yuya Suzuki, and Masatake Higashi, "Integrated Optimization for Supporting Functional and Layout Designs during Conceptual Design Phase," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, San Diego, California, August 30–September 2, 2009.

[45] Ashish Kolli, Jonathan Cagan, and Rob Rutenbar, "Packing of Generic, Three-Dimensional Components Based on Multi-Resolution Modeling," *ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Irvine, California, August 18–22, 1996.

[46] M. D. Landon and R. J. Balling, "Optimal Packaging of Complex Parametric Solids According to Mass Property Criteria," *Journal of Mechanical Design, Transactions of the ASME*, vol. 116, no. 2, pp. 375–381, 1994.

[47] F. Paul Lomangino, "Grammar- and Optimization-Based Mechanical Packaging," Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, Georgia, 1995.

[48] http://www.mathworks.com/help/nnet/ug/radial-basis-neural-networks.html.

[49] http://ai.stanford.edu/ mitul/mpk/.

[50] Martiti Mäntylä, *An Introduction to Solid Modeling.* 1988.

[51] Yi Miao, Vincent Y. Blouin, and Georges M. Fadel, "Multi-Objective Configuration Optimization with Vehicle Dynamics Applied to Midsize Truck Design," *ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Chicago, Illinois, September 2–6, 2003.

[52] Yi Miao, Vincent Y. Blouin, and Georges M. Fadel, "Design Configuration of Vehicle with Domain Knowledge Enhancement," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Salt Lake City, Utah, September 28–October 2, 2004.

[53] Yi Miao and Georges M. Fadel, "A Packing Genetic Algorithm for Configuration Design," *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Long Beach, California, September 24–28, 2005.

[54] Yi Miao, Georges M. Fadel, and Vladimir B. Gantovnik, "Vehicle Configuration Design with a Packing Genetic Algorithm," *International Journal of Heavy Vehicle Systems*, vol. 15, no. 2/3/4, pp. 433–448, 2008.

[55] Jeremy J. Michalek, "Interactive Layout Design Optimization," Master's Thesis, University of Michigan, Ann Arbor, Michigan, 2001.

[56] http://www.plm.automation.siemens.com/en_us/products/nx.

[57] http://www.noesissolutions.com/index.php?col=/products&doc=optimus.

[58] http://www.opengl.org.

[59] http://gamma.cs.unc.edu/SSV.

[60] Sungjin Park and Dohoy Jung, "Numerical Modeling and Simulation of the Vehicle Cooling System for a Heavy Duty Series Hybrid Electric Vehicle," *SAE Technical Paper Series*, 2008-01-2421.

[61] Sungjin Park and Dohoy Jung, "Design of Vehicle Cooling System Architecture for a Heavy Duty Series-Hybrid Electric Vehicle using Numerical System Simulations," *Journal of Engineering for Gas Turbines and Power, Transactions of the ASME*, vol. 132, no. 9, p. 092802 (11 pp.), 2010.

[62] http://www.plm.automation.siemens.com/en_us/products/open/parasolid/index.shtml.

[63] Sanjay Sachdev, Christiaan J. J. Paredis, Satyandra K. Gupta, and Sarosh N. Talukdar, "3D Spatial Layouts using A-Teams," *ASME Design Engineering Technical Conferences*, Atlanta, Georgia, September 13–16, 1998.

[64] Sunand Sandurkar, Wei Chen, and Georges M. Fadel, "GAPRUS: Three-Dimensional Pipe Routing using Genetic Algorithms and Tessellated Objects," *ASME Design Engineering Technical Conferences*, Sacramento, California, September 14–17, 1997.

[65] Steven S. Skiena, *The Algorithm Design Manual.* 2008.

[66] N. Smith, W. Hills, and G. Cleland, "A Layout Design System for Complex made-to-Order Products," *Journal of Engineering Design*, vol. 7, no. 4, pp. 363–375, 1996.

[67] Zhi-Guo Sun and Hong-Fei Teng, "Optimal Layout Design of a Satellite Module," *Engineering Optimization*, vol. 35, no. 5, pp. 513–529, 2003.

[68] S. Szykman and Jonathan Cagan, "Simulated Annealing-Based Approach to Three-Dimensional Component Packing," *Journal of Mechanical Design, Transactions of the ASME*, vol. 117, no. 2, pp. 308–314, 1995.

[69] S. Szykman and J. Cagan, "Synthesis of Optimal Nonorthogonal Routes," *Journal of Mechanical Design, Transactions of the ASME*, vol. 118, no. 3, pp. 419–424, 1996.

[70] S. Szykman and Jonathan Cagan, "Constrained Three-Dimensional Component Layout using Simulated Annealing," *Journal of Mechanical Design, Transactions of the ASME*, vol. 119, no. 1, pp. 28–35, 1997.

[71] S. Szykman, Jonathan Cagan, and P. Weisser, "An Integrated Approach to Optimal Three Dimensional Layout and Routing," *Journal of Mechanical Design, Transactions of the ASME*, vol. 120, no. 3, pp. 510–512, 1998.

[72] Hong fei Teng, Shou lin Sun, De quan Liu, and Yan zhao Li, “Layout Optimization for the Objects Located within a Rotating Vessel - a Three-Dimensional Packing Problem with Behavioral Constraints,” *Computers and Operations Research*, vol. 28, no. 6, pp. 521–535, 2001.

[73] Atul Thakur, Ashis Gopal Banerjee, and Satyandra K. Gupta, “A Survey of CAD Model Simplification Techniques for Physics-Based Simulation Applications,” *Computer-Aided Design*, vol. 41, no. 2, pp. 65–80, 2009.

[74] Santosh Tiwari, Georges Fadel, and Peter Fenyes, “A Fast and Efficient Compact Packing Algorithm for Free-Form Objects,” *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Brooklyn, New Work, August 3–6, 2008.

[75] John W. Whitehead, “Design and Performance of Derivative-Free Optimization Algorithms used with Hybrid Electric Vehicle Simulations,” Master’s Thesis, University of Michigan, Ann Abor, Michigan, 2001.

[76] Frank M. White, *Fluid Mechanics.* 3 ed., 1994.

[77] Hiroyuki Yamazaki, Keishi Sakanushi, Shigetoshi Nakatake, and Yoji Kajitani, “3D-Packing by Meta Data Structure and Packing Heuristics,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E83-A, no. 4, pp. 639–645, 2000.

[78] Su Yin and Jonathan Cagan, “Extended Pattern Search Algorithm for Three-Dimensional Component Layout,” *Journal of Mechanical Design, Transactions of the ASME*, vol. 122, no. 1, pp. 102–108, 2000.

[79] Su Yin and Jonathan Cagan, “Exploring the Effectiveness of various Patterns in an Extended Pattern Search Layout Algorithm,” *Journal of Mechanical Design, Transactions of the ASME*, vol. 126, no. 1, pp. 22–28, 2004.

[80] Su Yin, Jonathan Cagan, and Peter Hodges, “Layout Optimization of Shapeable Components with Extended Pattern Search Applied to Transmission Design,” *Journal of Mechanical Design, Transactions of the ASME*, vol. 126, no. 1, pp. 188–191, 2004.

[81] Y. H. Yin, C. Zhou, and J. Y. Zhu, “A Pipe Route Design Methodology by Imitating Human Imaginal Thinking,” *CIRP Annals - Manufacturing Technology*, vol. 59, no. 1, pp. 167–170, 2010.

[82] Su Yin and Jonathan Cagan, “A Pattern Search-Based Algorithm for Three-Dimensional Component Layout,” *ASME Design Engineering Technical Conferences*, Atlanta, Georgia, September 13–16, 1998.

[83] Su Yin, Jonathan Cagan, Peter Hodges, and Xianren Li, “Layout of an Automobile Transmission using Three-Dimensional Shapeable Components,” *ASME*

Design Engineering Technical Conferences, Las Vegas, Nevada, September 12–15, 1999.

[84] Chen Zhou and Yuehong Yin, “Pipe Assembly Planning Algorithm by Imitating Human Imaginal Thinking,” *Assembly Automation*, vol. 30, no. 1, pp. 66–74, 2010.

[85] David Zhu and Jean-Claude Latombe, “Pipe Routing = Path Planning (with Many Constraints),” *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, California, April 9–11, 1991.

[86] http://www.engineous.com/products/isight.html.