ARTIFICIAL NEURAL NETWORK-ASSISTED REPAIR TECHNIQUE FOR HANDLING CONSTRAINTS IN STRUCTURAL OPTIMIZATION

by

YUECHENG CAI

M.Eng., University of British Columbia, 2019

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Mechanical Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

January 2022

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Artificial Neural Network-Assisted Repair Technique for Handling Constraints in Structural Optimization

submitted by Yuecheng Cai in partial fulfilment of the requirements for the degree of Master of Applied Science in Mechanical Engineering

Examining Committee:

Dr. Jasmin Jelovica, Assistant Professor, Mechanical Engineering, UBC
Supervisor

Dr. Clarence De Silva, Professor, Mechanical Engineering, UBC
Supervisory Committee Member

Dr. Carl Ollivier-Gooch, Professor, Mechanical Engineering, UBC
Supervisory Committee Member

Abstract

Structural design typically involves nonconvex criteria that need effective optimization algorithms which can find the global optimum or Pareto optima. Constraints create complex hyperspaces that are difficult to navigate, and traditional constraint handling techniques (CHTs) might not be capable of steering the search. Repair techniques are one type of CHTs that can be very effective but have a few limitations that restrict their use. We here present a new repair-based CHT that addresses these issues by being: (i) adaptive to the share of infeasible solutions in a population and (ii) free of problem-specific heuristic for repair that a user typically needs to provide. Only the best performing infeasible solutions are repaired, to balance the normal operating procedure of the optimization algorithm with CHT, i.e., minimizing objectives and satisfying constraints. A procedure is proposed to apply artificial neural network (ANN) to automate the definition of problem-specific knowledge by identifying and ranking the most significant variables that influence each constraint. The proposed CHT approach is implemented in single-objective swarm algorithm PSO and multi-objective evolutionary algorithms NSGA-II and MOEA/D. The following test cases are considered: mathematical benchmark problem, truss optimization and structural optimization of a chemical tanker's main frame. Trained ANN is used as surrogate model in the latter case. In comparison to the original algorithms, a few state-of-the-art algorithms and CHTs, all modified algorithms show significantly better performance.

Lay Summary

Engineering problems and structures can be optimized using optimization algorithms. Failure criteria and other limitations of the problem often prevent the algorithms from finding the best solution. Constraint handling techniques e.g., repair techniques, can help optimization algorithms deal with complex design constraints. However, repair methods cannot be widely applied due to a few limitations. This thesis develops a new repair-based constraint handling technique that addresses these issues by using artificial neural networks to automate the repair process. The proposed approach has been embedded into three prominent optimization algorithms. Test problems involve one mathematical benchmark problem and two engineering problems. The results demonstrate the improved performance of the modified algorithms in comparison to the original algorithms, state-of-the-art algorithms, and a few other prominent constraint handling techniques.

Preface

The idea of this project was developed by Dr. Jasmin Jelovica and M.A.Sc student Yuecheng Cai. Yuecheng Cai performed all the coding, analysis, and writing of the thesis draft. Part of the computational experiments was conducted at UBC ARC Sockeye. This thesis was supervised by Dr. Jasmin Jelovica. The validation of the research work was conducted by Dr. Jasmin Jelovica and Yuecheng Cai.

Part of the thesis has been presented at the following conferences:

- Cai, Y., Jelovica, J. "Adaptive Constraint Handling in Optimization of Complex Structures by Using Machine Learning". Proc. 40th International Conference on Ocean, Offshore and Arctic Engineering, June 2021.

- Cai, Y., Jelovica, J. "Improved multi-objective structural optimization using artificial neural networks for repair-based constraint handling". 1st International Conference on Mechanistic Machine Learning and Digital Twins for Computational Science, Engineering & Technology, San Diego, USA, September 2021.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ABC	Artificial Bee Colony algorithm
ANN	Artificial Neural Network
AI	Artificial Intelligence
CHT	Constraint Handling Techniques
CNN	Convolutional Neural Network
DBN	Deep Belief Networks
DCGAN	Deep Convolutional Generative Adversarial Network
DLH	Dimension Learning-based Hunting
EA	Evolutionary Algorithm
EBCH	Evolutionary Bound Constraint Handling
EP	Evolutionary Programming
ES	Evolutionary Strategy
FA	Firefly Algorithm
FEA	Finite Element Analysis
GA	Genetic Algorithm

GWO	Grey Wolf Optimizer
GSA	Gravitational Search Algorithm
HiDeNN	Hierarchical Deep Learning Neural Network
HV	Hypervolume Indicator
IGD	Inverted Generational Distance
IGWO	Inverted Grey Wolf Algorithm
KP	Knapsack Problem
LSTM	Long Short Term Memory network
ML	Machine Learning
MOBA	Multi-objective Bat Algorithm
MOEA/D	Multi-objective Evolutional Algorithm based on Decomposition
MOGWO	Multi-objective Grey Wolf Optimizer
MOMVO	Multi-objective Multi-Verse Optimizer
MVO	Multi-Verse Optimizer
NSGA-II	Non-dominated Sorting Genetic Algorithm–II
PEBCH	Probabilistic Evolutionary Bound Constraint Handling

PF	Pareto Front
PMX	Partially Mapped crossover
PSM	Pattern Search Method
PSO	Particle Swarm Optimizer
RBM	Restricted Boltzmann Machines
RNN	Recurrent Neural Network
SCA	Sine-Cosine Algorithm
SR	Stochastic Ranking
SVM	Support Vector Machine
TBX	Tie-breaking crossover
TDR	Travelling Distance Rate
TSP	Traveling Salesman Problem
WER	Wormhole Existence Rate

List of Symbols

a	Activation of a neuron
acc	Accuracy of an ANN
$alpha$	Grid inflation parameter
$\boldsymbol{b}$	Bias vector
$beta$	Leader selection pressure parameter
c_1	Personal learning coefficient
c_2	Global learning coefficient
C_i	Penalty coefficient
$\boldsymbol{f}(\boldsymbol{x})$	Vector of objective functions
F	Feasible region
$F(\boldsymbol{x})$	Fitness function
F_t	Feasible solutions set
$\boldsymbol{g}(\boldsymbol{x})$	Vector of inequality constraint functions
G	Activation function
G_{best}^{t}	Global best position of at generation *t*

$\boldsymbol{h}(\boldsymbol{x})$	Vector of equality constraint functions
$H(\boldsymbol{x})$	Hessian matrix
l	Number of constraint functions
L	Depth of ANN
L_{fea}	Number of feasible solutions in P_t
m	Number of objective functions
n	Number of variables
$nGrid$	Number of grids per each dimension
nt	ANN testing data size
N	Number of neurons in the hidden layer
$P(\boldsymbol{x})$	Penalty function
P_f	Probability factor in Stochastic ranking
$\boldsymbol{P}_{best}^{t}$	Particle's history best position of the particle at generation t
$\boldsymbol{P}^{*}$	Set of solutions in the true Pareto Front
p_c	Probability of crossover
p_m	Probability of mutation

P_t	Parent population
Q_t	Children population
S	Search space
T	ANN accuracy training threshold
v_x	Displacement of node in x-direction
v_y	Displacement of node in y-direction
$\boldsymbol{V}^t$	Particle's moving velocity
w	Inertia weight
$\boldsymbol{W}$	Neuron's weight matrix
$W(A)$	Weight function
$\boldsymbol{x}$	Vector of variables of a solution
$\hat{\boldsymbol{x}}$	Pareto optimal alternative
$\boldsymbol{x}^*$	Optima point
$\boldsymbol{X}$	All solution alternatives formed in an optimization algorithm
X_i^0	i^{th} input of ANN
Y	Objective space
Y_K	ANN output for test example K

z^*	Reference point
Z	Activation input
μ_k	Step coefficient
Ω	Feasible solution set
$\nabla f(\boldsymbol{x})$	Gradient of $f(\boldsymbol{x})$
$\widehat{\boldsymbol{\Omega}}$	Pareto front
$\varphi(\boldsymbol{x})$	Penalized objective function
λ^i	i^{th} weight vector
τ	Allowable violation threshold
η	Variable influencing threshold
ε	Dynamic expansion factor
μ	Expectation of the gaussian distribution
σ	Standard deviation
σ_i	The stress of bar element
γ	Infeasible front
ω	Feasible front

α	Repair candidate collection in NSGA-II

Acknowledgements

Here, I would like to show my greatest gratitude to my parents for everything done for me. Your love and care make me a better person, the courage given by you always motivate me to move forward.

Also, I am truly grateful to my girlfriend Ying Yang, who has accompanied me all the time. I appreciate your understanding and confidence in me, and I wouldn't make this without your love and support. Additional thanks for helping me with the formatting.

Major thanks to my supervisor Dr. Jasmin Jelovica, who helped me a lot throughout the whole master program. You have shown your great patience and taught me many things not limited to doing research but also my life. Your continuous encouragement helps me a lot in finishing my research work.

I also want to thank all my friends who helped me prepare the master's thesis. Also, I appreciate the UBC NAME group and UBC sockeye for providing me with a lot of computational resources.

Lastly, I want to express my thanks to the Natural Sciences and Engineering Research Council of Canada [grant number IRCPJ 550069-19] for the financial assistance.

1 Introduction

1.1 Optimization formulations and definitions

For natural problems with more than one potential solution, the procedure of selecting the best solution under certain criteria/criterion is called optimization. Optimization problems of sorts arise in all quantitative disciplines from computer science and engineering to operations research and economics. Mathematically, an optimization problem could be formulated as maximization or minimization of one or more functions, which are called objective functions. Typically, the input variables could be chosen from an allowed set and be required to satisfy one or more restricting functions called constraints. An optimization algorithm aims to reach the optimal design while satisfying all constraints. A constrained optimization problem can be mathematically formulated as:

$$\begin{aligned} &\text{Minimize} && \boldsymbol{f}(\boldsymbol{x}) = \left(f_1(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})\right)^T \\ &\text{Subject to} && \boldsymbol{g}(\boldsymbol{x}) = \left(g_1(\boldsymbol{x}), \cdots, g_l(\boldsymbol{x})\right)^T \geq 0 \\ & && \boldsymbol{h}(\boldsymbol{x}) = \left(h_1(\boldsymbol{x}), \cdots, h_p(\boldsymbol{x})\right)^T = 0 \\ &\text{while} && \boldsymbol{x} = \left\{(x_i, \cdots, x_n)^T \middle| x_{min,i} \leq x_i \leq x_{max,i}\right\} \end{aligned} \tag{1-1}$$

where $\boldsymbol{f}(\boldsymbol{x})$ is the set of *m* objective functions, $\boldsymbol{g}(\boldsymbol{x})$ is the set of inequality constraint functions, $\boldsymbol{h}(\boldsymbol{x})$ is the set of equality constraint functions, and $\boldsymbol{x}$ is the set vector of solutions or designs. The aim is to optimize vector $\boldsymbol{x}$, while minimizing objective functions $\boldsymbol{f}(\boldsymbol{x})$ and satisfying constraints $\boldsymbol{g}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$. As equality constraints are not that frequent in engineering optimization problems, they are not considered later in this thesis. Maximization of an objective function can be treated the same way by multiplying the value with -1. If we give

weights to different objectives, we can transform the problem in Eq. (1-1) to a single objective problem. Often it is difficult to define weights in optimization beforehand, thus we need multi-objective approaches, which are introduced below. Furthermore, optimization problems can contain local optima, which is the optimal solution within a neighboring set of candidate solutions. This contrasts with global optimum, which is the optimal solution among all possible solutions, not just those in a particular neighborhood of design values. One purpose in optimization is to avoid being trapped into the local optima and reach the global optima. Constraint functions can be linear or nonlinear, depending on the optimization problem. Typically, each variable x_i can be bounded between a lower limit $x_{min,i}$ and upper limit $x_{max,i}$. A vector of variables $\boldsymbol{x}$ is called a solution or a design. If all constraints are satisfied, we call this solution a feasible solution. Otherwise, if one or more constraints are violated, we call it an infeasible solution. Feasible solutions are members of the feasible set $\boldsymbol{\Omega}$.

$$\boldsymbol{\Omega} = \{\boldsymbol{x} \in \boldsymbol{X} | \boldsymbol{g}(\boldsymbol{x}) \geq 0\} \tag{1-2}$$

where $\boldsymbol{X}$ is the set of all possible solutions between lower x_{min} and upper x_{max} variable bounds. The solution of the optimization problem in the feasible space (Eq.(1-2)) is a Pareto optimal alternative $\hat{\boldsymbol{x}}$ which is non-dominated by other feasible alternatives in the objective space. For two solutions $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, if $\boldsymbol{x}_1$ is to dominate $\boldsymbol{x}_2$, the following requirements must be met [1]:

1. $\boldsymbol{x}_1$ is no worse than $\boldsymbol{x}_2$ in all the objectives;

2. $\boldsymbol{x_1}$ is strictly better than $\boldsymbol{x}_2$ in at least one objective.

In other words, there is no feasible solution better than $\hat{\boldsymbol{x}}$ for all objectives. Such non-dominated solutions $\hat{\boldsymbol{x}}$ belong to a set of feasible Pareto optima $\widehat{\boldsymbol{\Omega}}$, also called the Pareto Front (PF):

$$\widehat{\boldsymbol{\Omega}} = \left\{\widehat{\boldsymbol{x}} \in \boldsymbol{\Omega} \middle| \nexists \boldsymbol{x}^k, f_i(\boldsymbol{x}^k) < f_i(\boldsymbol{x}), \forall i \in [1, m], \forall \boldsymbol{x}^k \in \boldsymbol{\Omega}, \forall \boldsymbol{x} \in \widehat{\boldsymbol{X}} \backslash \widehat{\boldsymbol{x}}\right\}, \quad (1\text{-}3)$$

Thus, the Pareto Front can be considered as a set of nondominated solutions among all permutation solutions [1], and there isn't one solution that is equally good or better than solutions in PF with respect to all objectives. In single-objective optimization problems, there's no PF since only one objective is being optimized. However, in most multi-objective optimization problems, objective functions are in conflict, which means that there is no single solution with the best objective value for all objective functions. The best trade-offs between each objective form the PF. One aim of the multi-objective optimization algorithms is to find the PF accurately and effectively.

1.2 Introduction to optimization algorithms

Optimization algorithms are designed for solving optimization problems. Based on mathematical and physical knowledge, or inspired by natural phenomena, various types of optimization algorithms have been proposed in recent decades. In the following subsections, different types of optimization algorithms, together with the constraint handling techniques (CHTs), and their development for dealing with the engineering problems are described.

1.2.1 General overview

Based on the *No Free Lunch Theorem* [2], no optimization algorithm is suitable for all types of optimization problems. To determine which algorithm is the best for solving a given problem, it is necessary to understand the underlying principles and purpose of different optimization methods. For multivariate problems, optimization algorithms can be categorized into direct search methods, gradient-based methods, and nature- and physics-inspired methods.

Direct search methods

As one of the most popular optimization methods, direct search methods could be easily implemented into non-linear programs [3]. The direct search method starts from a random location and iteratively improves a solution. Direct search methods can be the method of first recourse, even among well-informed users [4].

Pattern Search Method (PSM) is one of the earliest and still most common direct search methods. This method is characterized by a series of exploratory moves and patterns that consider the behavior of the objective function as a pattern of points. PSM tends to improve the objective value through pattern moves, which are considered successful if an improvement in the objective value is observed [4]. The exploratory move can be written as:

$$x_{k+1}^{i} = x_{k}^{i} + d_{k}^{i} \tag{1-4}$$

where k indicates the iteration, i is the i^{th} variable and d_k^i is the step size in i^{th} coordinate. The variable value is kept if a better objective value is observed. The procedure is performed in all variable coordinates and the final vector of variables is called the base point. The vector difference between the current and previous base point indicates the direction of the pattern move [4]:

$$\acute{\boldsymbol{x}} = \boldsymbol{x}_k + \Delta \boldsymbol{k} \tag{1-5}$$

$$\Delta \boldsymbol{k} = \boldsymbol{x}_k - \boldsymbol{x}_{k-1} \tag{1-6}$$

where $\acute{\boldsymbol{x}}$ is a new vector of variables created based on the previous pattern direction $\Delta \boldsymbol{k}$. The updating of variables will continue if a better objective value is found. Otherwise, the algorithm restarts the exploratory moves. The procedure will stop if no further improvement is achieved even with small step sizes [4]. The direct search method is suitable for simpler problems, but hard to converge for complex problems with a huge design space or large discontinuities.

Moreover, this type of method is computationally costly due to the low efficiency of pattern moves.

Gradient-based methods

Different from direct search methods, gradient-based methods utilize the gradient information of the objective functions. As one of the most well-known mathematical optimization approaches, the Lagrange multiplier is a method for finding the optimum under equality constraints [5]. One advantage of Lagrange multipliers is that no additional parameters need to be adjusted during optimization, which leads to a broad application of the method.

Instead of directly finding the optimum, such as with the Lagrange multiplier, many algorithms tend to iteratively find an optimum, which can be either local or global. Gradient descent, also called the steepest descent, utilizes the gradient of the objective function at a point to define the direction of the search, and does so repeatedly until convergence [6]:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \mu_k \nabla f(\boldsymbol{x}_k) \tag{1-7}$$

where $\boldsymbol{x}_{k+1}$ is the vector of variables in iteration $k+1$, which is improved from $\boldsymbol{x}_k$ by adding a term $\mu_k \nabla f(\boldsymbol{x}_k)$. μ_k is the step coefficient for iteration k and $\nabla f(\boldsymbol{x}_k)$ is the gradient of the objective function at the current point:

$$\nabla f(\boldsymbol{x}) = [\frac{\partial f}{\partial x_1}(\boldsymbol{x}) \ \frac{\partial f}{\partial x_2}(\boldsymbol{x}) \ \cdots \ \frac{\partial f}{\partial x_n}(\boldsymbol{x})] \tag{1-8}$$

Other famous methods based on derivatives are the Newton's method [7], Sequential quadratic programming [8], etc. The idea is similar as with Gradient descent: start from an initial guess $\boldsymbol{x}_0$, and converge towards the optima point $\boldsymbol{x}^*$ using the information of Hessian matrix ***H(x)***:

$$\mathrm{H}(\boldsymbol{x}) = \nabla^2 f(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \tag{1-9}$$

where $\nabla f(\boldsymbol{x})$ is the gradient of $f(\boldsymbol{x})$, which is defined beforehand. We consider a solution x^* as optimum if it satisfies both necessary and sufficient optimality conditions. The necessary condition for $\boldsymbol{x}^*$ to be the optimum is that the function's gradient at the point is equal to zero, e.g., $\nabla f(\boldsymbol{x}^*) = 0$. The sufficient condition is that the corresponding Hessian matrix of the point is positive definite [9] if dealing with a minimization problem.

Fast convergence to the (local) optima makes these optimization algorithms very popular in many fields, e.g., cost function optimization of an Artificial Neural Network (ANN), shape and topology optimization in mechanical and civil engineering [10], etc. However, these methods require the objective function and the constraints to be twice continuously differentiable, which is not applicable for more engineering problems.

Nature- and physics-inspired methods

Inspired by natural phenomena and biological systems, metaheuristic optimization algorithms have emerged as one of the most studied branches of Artificial Intelligence (AI) during the last decades [11]. Many researchers are trying to mimic the evolutional or hunting behavior of various species to address complex optimization problems. They rely solely on function values and not derivatives, which is beneficial when dealing with non-continuous functions and/or discrete design domains. However, this causes a low convergence rate in comparison to the gradient-based algorithms. Metaheuristic algorithms improve a set of solutions (population) through iterations (generations) using various operators. A detailed explanation of their control parameters can be found in Section 2.5.1. Modifying solutions includes some randomness,

which keeps the diversity of the whole population and prevents convergence to a local optimum [12].

Over the last few decades, a wide range of optimization algorithms has been published. Nature-inspired algorithms show great success for various problems, from solving the famous Traveling Salesman Problem (TSP) and Knapsack Problem (KP) to structural engineering design. The idea of Evolutionary Programming (EP) and Evolutionary Strategy (ES) was firstly proposed by Fogel, Rechenberg and Schwefel in the late 60s and 70s [11,13]. After that, as one sub-class of the Evolutionary algorithms (EA), the emergence of Genetic Algorithm (GA) deeply inspired the community and derived various versions of optimizers [14]. With the development of the science and engineering field, problems with multi-objective caught the attention of many researchers. Increasing computational resources enables researchers to deal with more constraints, more objectives or even optimization problems with black-box functions. This gives advantages to the development of metaheuristic algorithms. Non-dominated Sorting Genetic Algorithm-II (NSGA-II), as one of the most famous multi-objective optimization algorithms, shows great capability in dealing with various types of constrained optimization problems [15]. Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) was proposed in 2007 as the most famous decomposition-type algorithm [16]. MOEA/D can generate a uniformly distributed PF, which makes this algorithm popular for decision-makers. In this thesis, NSGA-II and MOEA/D have been improved because of their efficiency and efficacy in dealing with different problems. The details of the modified algorithms can be found in Section 3.

Another category of nature-inspired algorithms is swarm algorithms. Particle Swarm Optimizer (PSO) is one of their most popular algorithms, well-known for its high rate of convergence [17]. As mentioned above, swarm-based algorithms mimic the behavior of animals like bird flocks or fish schools. Following similar mechanisms, various swarm-based

algorithms have been proposed; examples include Grey Wolf Optimizer (GWO), Multi-objective Bat Algorithm (MOBA), Firefly Algorithm (FA), Artificial Bee Colony algorithm (ABC), etc. [18–21].

Besides species' natural behavior, physical laws and mathematical principles can also be the inspiration for optimization algorithms. Motivated by the interaction of multiple universes via white holes, black holes, and worm holes, Multi-Verse Optimizer (MVO) has great performance in dealing with mathematical optimization problems [22]. Based on the mathematical properties of trigonometric functions, Sine-Cosine Algorithm (SCA) has become popular in recent years due to its efficiency in solving constrained problems with unknown spaces [23]. Utilizing the theory of Newtonian physics, Gravitational Search Algorithm (GSA) was developed and can provide superior results for various standard benchmark optimization problems [24].

1.2.2 Constraint handling techniques

For constrained problems, in addition to optimizing the objective functions, optimization algorithms also need to ensure that constraints are not violated. Especially when dealing with complex constraints, an effective constraint handling technique (CHT) can increase optimization algorithms' efficiency, meaning that the required number of function evaluations to reach the optima is reduced. Moreover, without a proper CHT, optima might not be found even with an excessive number of generations (algorithm's iterations). In the following subsections, different categories of CHTs are introduced.

Penalty function approaches

Penalty functions are one of the oldest approaches used to consider constraints in optimization [25]. Penalty function is initially proposed by Courant [26], where the penalty term is added to

the objective function to penalize the designs with any violation of the constraints. Thus, constraint optimization problems can be transformed into unconstrained problems:

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + P(\boldsymbol{x}) \tag{1-10}$$

where $F(\boldsymbol{x})$ is the fitness function, $f(\boldsymbol{x})$ is the original objective function, and $P(\boldsymbol{x})$ is the penalty function. When $P(\boldsymbol{x})$ equals zero, the solution is feasible. Instead of optimizing the objective function, algorithms optimize the function given with Eq. (1-10). Similar to Courant's method, Joines [27] proposed the non-stationary penalty function, where the penalty term is the function of generations. As optimization proceeds, the extent of penalization increases. A higher penalty will lead to higher pressure on searching for feasible solutions. However, the penalization for each infeasible solution is the same regardless of the extent of constraint violation, which is inefficient and inappropriate if all solutions in a population are infeasible. To address this issue, Abdollah [28] proposed the systematic multi-level penalty handling method, where penalization is a stepwise function of constraints. After that, various penalty approaches were proposed, such as the Death penalty, Dynamic penalty [27], and Self-adaptive penalty [25].

However, most of these methods have significant flaws. Death penalty rejects every infeasible solution, which is inefficient and can lead to premature convergence. Dynamic penalty is more robust than the death penalty, but the parameter tuning requires precise control. Adaptive penalty requires fewer user-defined parameters, but the computational time of this method is higher.

Special constraint handling operators

Beside penalty functions, there are several other prominent CHTs. Proposed by Koziel [29], decoders showed a competitive performance in the 1990s. This method is based on the idea of

mapping the feasible region Ω onto an easier-to-sample space where a nature-inspired algorithm can provide a better performance. However, the high computational cost is the main drawback of this method [30].

Another popular approach that is widely used for constraint handling is Stochastic ranking (SR) initially proposed by Runarsson and Yao [31]. SR aims to deal with the inherent shortcomings of the penalty function. This method ranks the infeasible solutions based on the sum of constraint violation or objective value. However, some solutions may still get a good rank, although they are infeasible.

Limited by the properties of different types of CHTs, some researchers came up with a hybrid method called the Ensemble constraints handling method. Proposed by Qu & Suganthan [32], three constraint handling approaches (self-adaptive penalty, superiority of feasible solution, and ε-constraint) were used simultaneously for evaluating the population. This method is more versatile than other approaches. However, the computational time increases in comparison with the other techniques.

Repair techniques

Different from the approaches above, repair methods aim to transform the solutions from infeasible to feasible. The idea is to benefit from the objective values of the prominent infeasible solutions which have a low constraint violation. A successful transformation of those solutions into feasible form can help optimization algorithms explore the objective space.

P. Poon and J. Carter [33] proposed the partially mapped crossover (PMX) and the tie-breaking crossover (TBX), which apply crossover operator when performing repair process. In the PMX method, two solutions are repaired by changing variables randomly selected in the parents.

TBX method repairs the worst performance design based on the global behavior of the population.

Due to the low repair success rate, other novel techniques are proposed to increase the search speed for feasible solutions. P.Koch, W. Konen, and C. Foussette [34] developed a repair technique based on the gradient information of the constraints functions called RI-2. This technique shows a high successful repair rate of 30% compared to other repair techniques [35]. However, the application of this method is limited to complex problems with discrete variable values or disconnected design domains.

Repairing can also be conducted based on a heuristic, coming from user's understanding of the problem. For example, in a stacking sequence optimization of composite materials (combinatorial problem), constraints limit the maximum number of consecutive plies with the same fiber orientation, and if this is violated the solution is modified by the repair algorithm until it becomes feasible [36].

Recently, a novel repair operator was proposed by Samanipour and Jelovica [37] where a more general framework was proposed allowing prominent infeasible solutions to be repaired based on other solutions in a population using a variable-constraint mapping. The idea is that constraint violations can be traced back to the variables. The result shows that this novel repair method can successfully repair infeasible solutions and use them to converge to PF faster. However, the variable-constraint mapping needs to be provided by the user beforehand, which is not applicable for some complex problems.

As an extension of this novel repair operator, this thesis addresses this issue and proposes a method that can automatically predict the mapping. The details of the proposed method can be found in Section 3.2.

1.2.3 Optimization in engineering

In recent decades, optimization algorithms have been widely used in solving engineering problems in various fields. Complex engineering problems mostly require many variables to define a design. One of such problems is the structural design of marine vessels. The most important decisions are made during the conceptual and preliminary design of ships. Generally, it is impossible for engineers to find the optimal design using engineering experience and intuition for such complex problems. Therefore, optimization algorithms are needed to find global optimum or optima.

In some cases, variables in engineering problems can only adopt discrete values due to market restrictions [1] which would result in inconsistencies and gaps in both design space and objective space. Some of the disjoint parts of objective space contain local optima, restricting further the optimization. Algorithms like steepest descent or Newton's method would likely get trapped into local optimum due to their reliance on the gradient information of the functions. Moreover, gradients are difficult or impossible to estimate when dealing with discrete variable values.

Evolutionary and swarm algorithms have become popular for dealing with engineering problems because they can properly address the aforementioned issues. Examples of engineering problems that are optimized using evolutionary algorithms can be found in [38–44], starting from a simple truss optimization problem to a complex bridge design. One advantage is that evolutionary algorithms do not require gradient information of the functions. This makes EA applicable to most types of problems. Furthermore, most engineering problems have more than one objective. This provides another advantage of using EA: due to the randomness of genetic operators, they can generate better distributed non-dominated fronts.

Even though evolutionary and swarm algorithms show good optimization capabilities, a high number of function evaluations makes them computationally very expensive. Traditionally, this difficulty is tackled by using reduced-order modes based on FEM and CFD, to save time when estimating objectives and constraints. Other than this, people also use surrogate models to predict constraints and objectives. Typical surrogate model techniques include Kriging models [45], response surface method [46], and ANNs [47]. As a subfield of Machine Learning (ML), ANNs have drawn a lot of attention due to advancements in data-driven models and deep neural networks [48]. Neural networks show the extraordinary capability to preserve the accuracy of high-fidelity models for only a fraction of the computational cost, which will be described in Section 1.3. In this thesis, we apply ANNs for two purposes: to generate a problem-specific variable-constraint mapping and to construct the surrogate model for optimization. These could be found in Section 3.2 and Section 5.1, respectively.

1.3 Overview of machine learning

Machine Learning (ML) is used in many fields, such as function approximation, computer vision, speech recognition, natural language processing, machine translation, etc [49,50]. ML has led to revolutionary progress in many subjects.

In the 1940s and 1950s, ANNs were composed of a simple perceptron, which could only deal with simple problems. After that, the proposed backpropagation technique enabled ANNs to track and utilize the gradient information of each hidden layer. This allowed ANNs to deal with more complex problems. Subsequently, developed by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Vapnik et al., 1997), Support Vector Machine (SVM) became one of the most robust prediction methods in the 1990s. Nowadays, other ML techniques appear and show their capability of prediction. Based on whether the training data

is labelled or not, they can be categorized into supervised learning algorithms and unsupervised learning algorithms. The most famous of them are introduced as follows.

Recurrent Neural Network (RNN) is a class of supervised learning algorithms, where the connections between neurons can form a directed graph along a temporal sequence [51]. This allows RNN to capture and predict the temporal dynamic features of the data. RNNs have demonstrated good performance in sequence labelling and predicting tasks, e.g. unsegmented, connected handwriting recognition, speech recognition, etc [52]. An example of the application of RNN can be found in [53], where the Long Short Term Memory network (LSTM) was applied for nonlinear structural seismic response prediction.

Another famous supervised learning algorithm is Convolutional Neural Network (CNN), which has been widely used for image recognition or classification in recent years. Different from the typical feedforward neural network, the architecture of CNN contains lots of convolutional layers, which are designed for capturing various features of the image, such as vertical lines or honeycomb shapes. Furthermore, softmax layers serves as activation functions to output the probability distribution of convolutional layers. Based on CNN, systems such as face recognition and automatic driving can be efficiently constructed. For example, improved design of microstructural materials was presented in Ref. [54] based on deep convolutional generative adversarial network (DCGAN) and CNN.

Autoencoder is one of the most popular types of the unsupervised neural network, used to learn efficient coding of unlabeled data [55]. Automatic learning from data makes autoencoders easily adaptable to a new (but related) task. Furthermore, autoencoders with more layers can significantly decrease the required training data size and shrink training time [56]. Thus, autoencoders could be used in dimensionality reduction, image processing, and anomaly detection [56–58], etc.

Deep Belief Network (DBN) is another type of unsupervised learning algorithm, being a probabilistic generative model that can establish a joint distribution between the observation data and the label. Constructed by multiple unsupervised networks such as Restricted Boltzmann Machines (RBM)s [59], DBN can reach a better accuracy for unsupervised learning than other ML techniques. For example, DBN is used in Ref. [60] to accelerate topology optimization.

ML tools are reaching many engineering and research fields. For example, instead of using FEA, the deep learning model could be applied for modelling both elastic and inelastic responses of buildings. In other fields, ML and cell concatenation could be used in the design of novel acoustic metamaterials [61]. Ref. [62] presented Hierarchical Deep Learning Neural Network (HiDeNN), an AI framework for solving challenging problems in computational science and engineering. It was demonstrated that the network can achieve better accuracy than conventional finite element method by learning the optimal nodal positions and capturing the stress concentration with a coarse mesh.

In many engineering optimization problems, evaluation of the functions may not be possible analytically. In this case, software simulations with FEM and CFD should be linked to the optimizer to perform the function evaluations [1]. Even though these tools are robust and accurate, their significant computational cost makes them inappropriate in some cases. For this purpose, ANNs are applied for function approximation as they can preserve the accuracy of high-fidelity models at a lower computational cost. Many researchers have proved that ANNs could accurately replace the original high-order models [63–65]. Furthermore, the universal approximation theorem states that a feedforward neural network with one hidden layer could accurately predict any continuous function arbitrarily well with respect to the uniform norm provided there are enough hidden units [66]. This gives us the confidence to use ANN for the aims given below.

1.4 Aim of this thesis

The overarching objective of this thesis is to create a generic repair-based constraint handling technique that does not need user-provided information of the problem for repair. The proposed method aims to help different optimization algorithms to efficiently deal with complex constrained engineering problems. The background of the approach is a recent repair technique, which is limited with the need for the problem-specific heuristic. This thesis aims to overcome this limitation and decrease the required number of function evaluations to reach PF. The research is divided into the following objectives:

1. Develop a method that can automatically discover the mapping between variables and constraints;

2. Modify the recent repair-based constraint handling technique and implement the automated mapping;

3. Implement the constraint handling procedure in few prominent algorithms for single- and multi-objective optimization;

4. Construct surrogate model of ship response to decrease CPU time in optimization;

5. Assess the framework's performance on both mathematical and engineering problems.

1.5 Thesis outline

This thesis is organized as follows. **Section 1** provides the background of the related research topics and outlines the thesis objectives. **Section 2** introduces the original optimization algorithms used in this study. Also, several state-of-the-art optimization algorithms and some CHTs used for validation are presented. **Section 3** of this thesis describes the repair-based CHT

and the proposed technique for automatically discovering the mapping. In addition, the implementation of the proposed approach to few types of optimization algorithms is presented. **Section 4** introduces optimization case studies, involving one mathematical benchmark problem and two engineering problems. **Section 5** discusses the parameters that influence the accuracy of the mapping and surrogate model. Further, the results obtained by the modified algorithms are compared with other state-of-the-art algorithms. In **Section 6**, the conclusion of this thesis, the limitation of the work and future work are presented.

2 Basic concepts

This thesis considers both single- and multi-objective algorithms. PSO, NSGA-II and MOEA/D are selected due to their excellent performance for many types of problems. The operating principles of these algorithms and their control parameters are described below.

2.1 Particle Swarm Algorithm

One of the most frequently used single-objective algorithms nowadays in engineering is PSO [67], because of its rapid convergence rate. In nature, each individual in the swarm relies on both its own and swarm's intelligence for finding food. If an individual finds out that another member in the swarm has a better path towards food, it will change its direction to follow that path. PSO follows a similar mechanism, as shown in Figure 2.1. Each solution is not only moving towards the swarm's best position, but also towards the particle's best position.

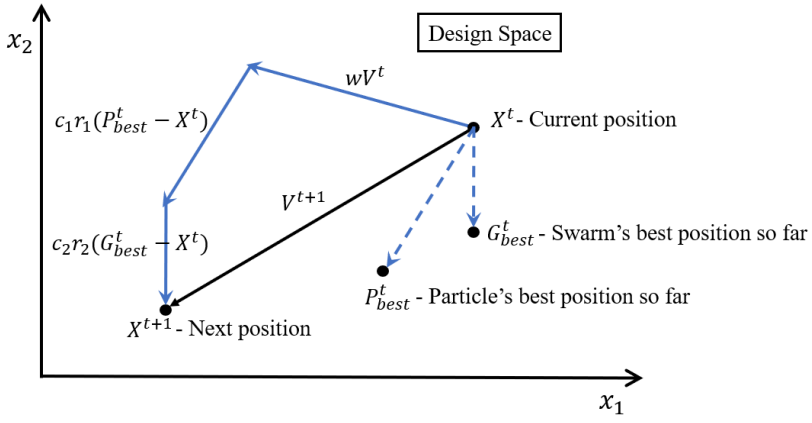


Figure 2.1. PSO algorithm configuration.

Relocation of a particle in the design space is represented through velocity, which is controlled by the swarm's best position, particle's best position, and velocity of the previous generation. The velocity of a particle in generation *t+1* is calculated as [68]:

$$\boldsymbol{V}^{t+1} = w * \boldsymbol{V}^t + c_1 * r_1 * (\boldsymbol{P}_{best}^t - \boldsymbol{X}^t) + c_2 * r_2 * (\boldsymbol{G}_{best}^t - \boldsymbol{X}^t) \quad (2\text{-}1)$$

And the position is updated as:

$$\boldsymbol{X}^{t+1} = \boldsymbol{X}^t + \boldsymbol{V}^{t+1} \quad (2\text{-}2)$$

where w is the inertia weight, c_1 and c_2 are two control parameters defined by a user. r_1 and r_2 are two random numbers from 0 to 1, which are used to ensure the randomness of the velocity vector. This helps the PSO avoid being trapped in the local optima. $\boldsymbol{P}_{best}^t$ and $\boldsymbol{G}_{best}^t$ are the best position of the current particle and the entire swarm, respectively.

2.2 Multi-objective Evolutionary Algorithm based on Decomposition

As one of the most popular evolutionary optimization algorithms, MOEA/D has been applied in many fields due to its capability of generating well-distributed fronts. The essence of MOEA/D is to transform a multi-objective problem into a number of single-objective optimization problems through aggregate functions. The algorithm optimizes them simultaneously, which can be seen from Figure 2.2.

MOEA/D transforms the multi-objective problem to *N* scalar subproblem through a set of evenly spread weight vectors $\lambda^1, \dots, \lambda^N$. Each subproblem is a projection of multi-objective functions to the specific weight vector. In this study, we apply the Tchebycheff approach to present the j^{th} subproblem [16]:

$$\text{minimize } g^{te}(\boldsymbol{x}|\lambda, z^*) = \max_{1 \le i \le m} \{\lambda_j^i | f_i(\boldsymbol{x}) - z_i^*|\} \quad (2\text{-}3)$$

where $\boldsymbol{z}^* = (z_1^*, \dots, z_m^*)$ is the reference point and $z_i^* = \min\{f_i(\boldsymbol{x})\}, i = 1, \dots, m$. Another major feature of MOEA/D is the definition of neighborhood. Each weight vector λ^i is defined as a neighborhood, which is a set of several closest weight vectors $\{\lambda_1, \cdots, \lambda_N\}$. The

neighborhood of the i^{th} subproblem consists of all the subproblems from the neighborhood of λ^i. Thus, the optimization of one subproblem needs the information from its neighboring subproblems.

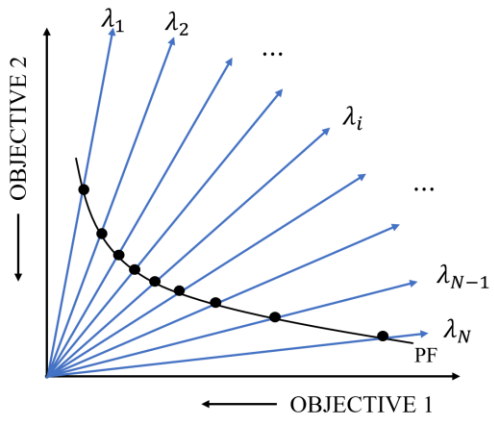


Figure 2.2. Pareto front and weight vectors in MOEA/D.

2.3 Non-dominated Sorting Genetic Algorithm-II

In engineering design, one of the most frequently used multi-objective optimization algorithms is NSGA-II. Because of its adaptiveness, it is popular in many research fields, and shown in a review publication [69].

NSGA-II modifies solutions based on genetic operators crossover and mutation. Crossover and mutation operators are commonly used in many optimization algorithms because they ensure randomness and give chance to population to get away from local optima. Different crossover and mutation operators are presented in the optimization literature for binary and real-parameter encodings. The crossover operator randomly picks two solutions from the population as the parent solutions to create two new children solutions by re-combining them. The mutation operator is intended to create small changes by randomly flipping variables in some solutions. Those two operators are controlled by two parameters p_c and p_m, which are the probability of crossover and the probability of mutation, respectively.

In NSGA-II, population is handled through non-domination sorting, which is performed by dividing feasible solutions into multiple non-dominated fronts in each generation. Once the non-dominated front is found, the solutions in that set will be temporally removed. A second non-dominated front will be found based on the remaining solutions. This procedure will be repeated until all the solutions are assigned to a certain non-dominated front, see Figure 2.3.

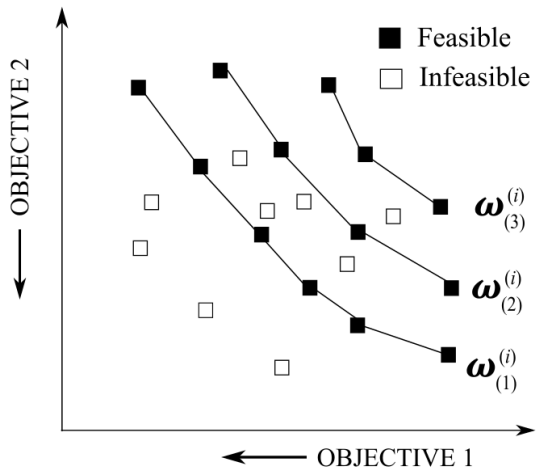


Figure 2.3. Non-dominated sorting in objective space [59].

Based on non-domination sorting, the selection operator is performed to decide which solutions will be used in genetic operators to create offspring for the next generation. Specifically, two solutions are randomly picked and the one with a higher non-domination rank will be preserved. As shown in Figure 2.4, the selection operator will be repeated until the number of the selected solution reaches the population size. Thus, NSGA-II preserves the best solutions for each generation. Infeasible solutions will be granted a lower priority or completely discarded.

Furthermore, the application of crowding distance helps NSGA-II keep a good diversity among solutions in a population. It is calculated as the average distance of a solution to its nearest neighbors in the same front. As such, crowding distance indicates the local density of solutions. Solutions in the same front with a lower crowding distance will be selected for genetic

operations (crossover and mutation). The exploration of the less populated area can help the algorithm to find a better spread of the PF and maintain the diversity of the solutions.

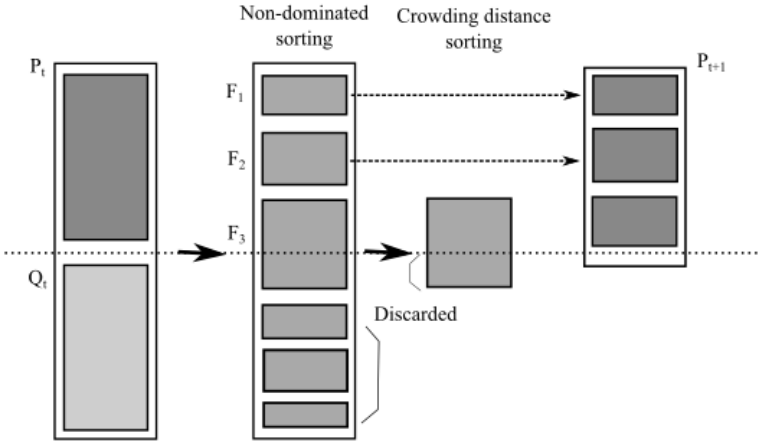


Figure 2.4. NSGA-II schematic procedure [15].

2.4 Novel optimization algorithms for comparison

In addition to PSO, NSGA-II, and MOEA/D, we use a few other optimization algorithms in this research for comparison.

For single-objective algorithms, we apply Gravitational Search Algorithm (GSA) [24], Multi-Verse Optimizer (MVO) [22], and Improved Grey Wolf Optimizer (IGWO) [70]. For multi-objective algorithms, we conducted the comparison using Multi-Objective Grey Wolf Optimizer (MOGWO) [71] and Multi-Objective Multi-Verse Optimizer (MOMVO) [72]. Some description of these algorithms is shown below.

GSA is an optimization algorithm inspired by the law of gravity and mass interactions. The search agent in GSA considers solutions as masses that interact with each other based on Newtonian gravity and the laws of motion. GSA could provide superior results for various standard benchmark problems in terms of other testing algorithms [24].

IGWO is an improved and better performing version of GWO, as shown for engineering problems [70]. IGWO addresses the issues of insufficient population diversity, imbalance between exploitation and exploration, and premature convergence of GWO, by introducing a new movement strategy called Dimension Learning-based Hunting (DLH). Instead of moving toward the leader, solutions under the DLH strategy are learning from their neighbors. Therefore, IGWO selects the leader either from the original GWO or DLH search strategies. No extra parameters are needed for DLH. The control parameters for IGWO are the same as for GWO, which can be found in Section 4.4.

As the multi-objective extension of GWO [18], MOGWO shows a very competitive performance compared to other state-of-the-art algorithms. MOGWO introduced two more strategies to deal with multi-objective problems: (i) prominent solutions are stored in an external archive at each generation, from which leaders (α, β, and δ) could be selected [71]; (ii) using the grid to preserve the diversity of the archive solutions. Therefore, three parameters, grid inflation parameter $alpha$, leader selection pressure parameter $beta$ and number of grids per each dimension $nGrid$ are used to control the leader selection mechanism and the grid mechanism. The detailed parameter setting of MOGWO can be found in Section 4.4.

MVO and MOMVO are recent population-based algorithms that mimic one of the theories in physics on the existence of multiple universes [22,72], for single-objective and multi-objective problems, respectively. The two algorithms have similar mechanisms, except that MOMVO requires an external archive to store the best non-dominated solutions obtained so far. The main inspiration of both algorithms is the interaction of multiple universes via white holes, black holes, and worm holes. Objects (variables) are transferred from a universe (solution) through a tunnel from a white hole to a black hole. Also, worm holes can move objects from one corner of a universe to another without a need for a white or black hole. Two adaptive

parameters are defined to control the exploration and exploitation of optimization design space. They are Wormhole Existence Rate (WER) and Travelling Distance Rate (TDR), respectively.

2.5 Control parameters

The performance of an optimization algorithm largely depends on control parameters. Some of the control parameters are common for all evolutionary algorithms, and others are unique. The following sub-sections introduce those parameters.

2.5.1 General control parameters

In evolutionary algorithms, general control parameters involve population size and the number of generations. Those two parameters are important as they decide the number of function evaluations in a run. Proper setting of population size and the number of generations may help optimization algorithm find the global optimum faster.

Population size

In evolutionary algorithms, population represents a set of current solutions. This set is iteratively updated until the algorithm meets the stopping criterion. Population size refers to the number of solutions in one population. A higher population size means that an algorithm needs to deal with more solutions simultaneously. Furthermore, a greater population size will provide optimization algorithms with a higher possibility of exploring the design space and a lower possibility of becoming trapped in a local optimum. In general, complex problems are defined with large number of variables, thus having a large design space for optimization algorithms to explore. A low population size will be less likely to reach the global optimum within a given amount of computational resources. Therefore, a complex optimization problem

always requires a high population size (50-300) [1]. Furthermore, a high population size for multi-objective algorithms could lead to better convergence and wider distributed final PF.

Number of generations/iterations

Iterations in EAs are called generations. Population is updated iteratively until a pre-defined number of generations is reached, although there could be other types of stopping criteria. In general, higher number of generations allows an algorithm to perform better. As mentioned above, a complex optimization problem generally involves a large design space. Thus, large number of generations is required, typically 500 to 1000 [1]. So that the optimization algorithm can get closer to the global optima or PF.

2.5.2 Specific control parameters

In addition to common control parameters, some optimization algorithms have specific parameters.

PSO includes three inertia coefficients that are used to control particle's velocity. The inertia weight w is used to control the particle's velocity inherited from the previous generation, which is normally set to 0.99, and aims to exploit the design space at the end of the optimization. c_1 and c_2 are used to control the velocity towards the particle's best position and the swarm's best position. In general, a larger c_1 and c_2 will lead the algorithm to converge to an optimum faster, but it could be a local optimum. A lower c_1 and c_2 is recommended if the problem involves a large design space.

Genetic operators in NSGA-II are controlled through probabilities of crossover and mutation. Crossover exchanges variable values between two designs and its probability p_c determines whether the exchange will occur or not. p_c is commonly set in the range of 0.6 to 0.95 [73].

The probability of mutation p_m determines whether a variable will be changed at random, and is generally calculated by $1/n$, where n is the number of variables. For binary problems, n could be the number of bits to the chromosome. p_m is always set as a small number, since large mutation probability might transform many feasible solutions to infeasible.

MOEA/D involves neighborhood size to control the exploration and exploitation process of the algorithm. New solutions of MOEA/D are generated from the neighboring solutions by using crossover and mutation. Literature shows that a large neighborhood size grants MOEA/D a high searchability in the objective space [74]. However, a small neighborhood size is also shown to be beneficial for diversity maintenance in the objective space.

In both MVO and MOMVO, the algorithm's performance is controlled by WER and TDR. WEP was defined as the probability of wormholes' existence in universes, which increases linearly from 0.2 to 1 to emphasize the exploitation process. Furthermore, TDR decreases from 1 to 0, which controls the distance that an objective can be teleported by a wormhole around the current best universe. A lower TDR indicates a more precise search around the best-obtained solution [22].

The control parameters of IGWO are mostly the same as the original GWO. The mathematical form of the encircling behavior is controlled by two coefficient vectors, $\vec{A}$ and $\vec{C}$. Both are influenced by a component $\vec{a}$, which decreased linearly from 2 to 0 [18,70]. Control parameters of MOGWO are different from the other GWO variants. As mentioned above, MOGWO is controlled by $alpha$, $beta$, and $nGrid$. These parameters are used to maintain the diversity of the archive during optimization, which are set to 0.1, 4, and 10 to obtain the best performance in test problems, respectively.

2.6 Prominent constraint handling methods

This thesis considers two constraint handling approaches to validate the proposed approach: penalty function and adaptive threshold.

As one of the most commonly used constraint handling methods which have been applied to many optimization algorithms [75], static penalty has been used for validation in this thesis. The penalized objective function is defined as:

$$\varphi(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} C_i G_i \,, \tag{2-4}$$

$$G_i = max\{0, g_i(\boldsymbol{x})\}^{\beta}, \tag{2-5}$$

where $\varphi(x)$ is the penalized objective function and C_i is the penalty coefficient. We use few penalty coefficients in the numerical experiments, where the higher one (10^7) is found to give the best performance out of several that were tested in preliminary studies.

Another constraint handling approach used for comparison is the adaptive threshold. The adaptive threshold approach was firstly introduced in [76] for MOEA/D, validated in [77] and shown to work the best among several other constraint handling approaches on NSGA-II in [78]. This approach defines a threshold in terms of constraint violation, until which infeasible solutions are accepted in the population. This method solved the issue for algorithms if initialized with total infeasible solutions. Solutions with a low constraint violation are closer to feasible design space and could be utilized to guide the algorithm toward the feasible space. No extra parameters are needed in this method, and the threshold is adaptively updated during the optimization process. The allowable violation τ is calculated as:

$$CV_i = vio * \sum_{j=1}^{l} \min(g_j, 0), \tag{2-6}$$

$$CVmean = \frac{1}{N} \sum_{i=1}^{N} CV_i, \tag{2-7}$$

$$FR = \frac{Number\ of\ the\ feasible\ solutions}{Population\ size}, \tag{2-8}$$

$$Allowable\ violation\ threshold\ (\tau) = CVmean \cdot FR, \tag{2-9}$$

where vio is denoted as the number of violated constraints for i^{th} solution, g_j is the value of the j^{th} constraint, considered as violating if g_j is negative. Eq. (2-6) to Eq. (2-9) shows that the allowable violation threshold τ is calculated based on the average level of the total population. *FR* is increasing during the optimization, while *CVmean* is severely decreasing, thus τ is decreasing as well. In the optimization process, the solutions with violations less than the threshold τ are treated as feasible. Hence, some infeasible solutions with relatively small constraint violation would remain in the population and transferred to the next generation.

2.7 Performance measures

For single-objective problems, algorithm's performance could be analyzed by tracking the best objective value found during optimization. An algorithm is considered more efficient if it needs a lower number of function evaluations.

However, in multi-objective optimization, the performance should be quantified based on the non-dominated front of the current generation. The purpose of algorithms is to find a solution set with high diversity and high proximity to the true Pareto front. Therefore, both the convergency and diversity of the solution set indicate the performance of the current non-

dominated front. This thesis uses two performance matrices, namely Hypervolume (HV) and Inverted Generational Distance (IGD). Configuration for both performance indicators is shown in Figure 2.5. HV [79] calculates the dominated hypervolume (or area in 2D space) with respect to a reference point. IGD [80] measures the average of the Euclidean distances of each solution in the true Pareto front to the nearest solution in the population's non-dominated front. Only feasible solutions are considered when calculating both HV and IGD. The IGD evaluation requires that the true PF be known. IGD is calculated as:

$$IGD(P, P^*) = \frac{\sum_{A \epsilon P^*} min_{B \in P} dis(A, B)}{|P^*|}, \tag{2-10}$$

where P represents the solutions of the current non-dominated front, P^* is a set of uniformly distributed solutions in the true Pareto front. $dis(A, B)$ is the Euclidean distance in objective space between solutions A and B. In this thesis, both reference point and P^* are defined by the best combination of objective values encountered in all history optimization runs, thus being set after all runs have been performed. Higher HV values and lower IGD values are preferred. Both performance matrices are calculated in normalized objective space.

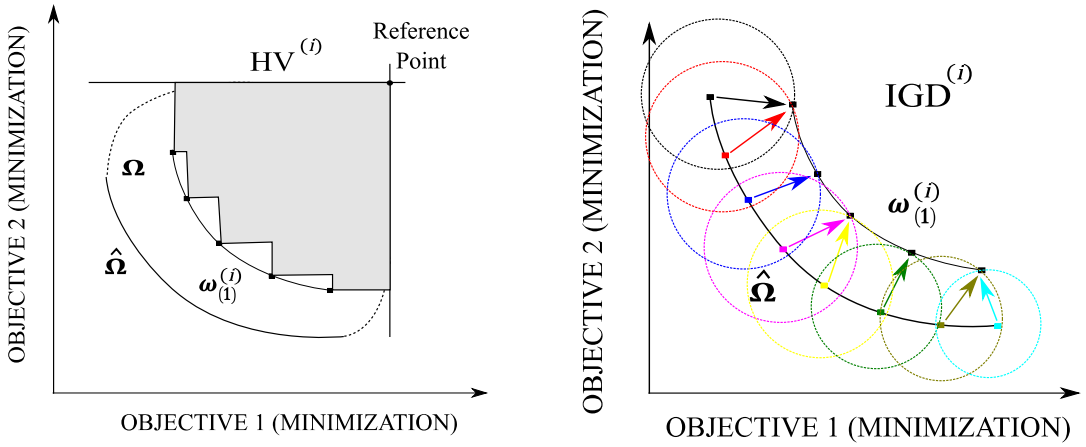


Figure 2.5. Visualization of the performance measures (HV and IGD).

3 Proposed constraint-handling framework and implementation

3.1 Repair heuristic

In this thesis, a new CHT is developed as an improvement of a recently published repair-based constraint handling method. The recent repair technique can handle complex constraints and has shown excellent performance in dealing with completely infeasible populations [37]. This is enabled by variable-constraint mapping which allows constraint violations to be traced back to variables.

In this thesis, we extend this repair technique, and modify its operating principle, to make it more logical and adaptive to both single- and multi-objective algorithms. Details of the modified repair technique are shown below.

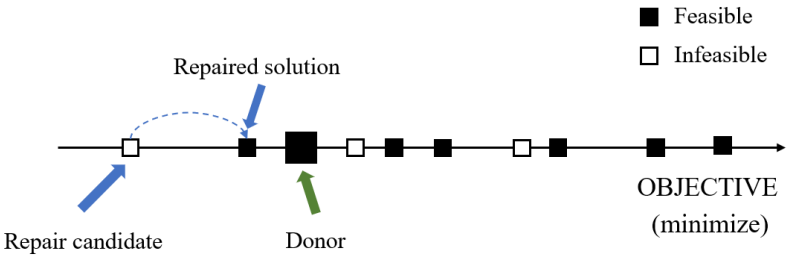


Figure 3.1. Repairing an infeasible solution in single-objective space.

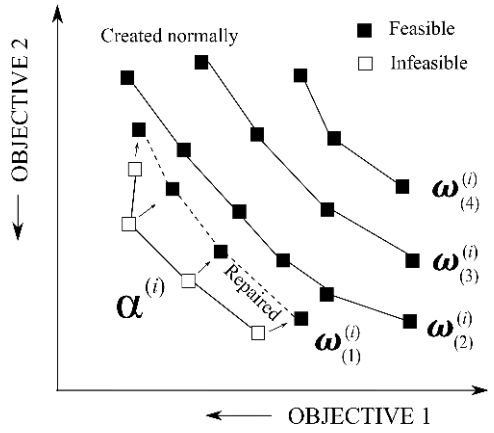


Figure 3.2. Situation where infeasible solutions after successful repairing dominate the current feasible solutions in multi-objective space [37].

In the optimization process, some infeasible solutions might have a better objective value (show dominance in multi-objective problems) than the existing feasible solutions. Those solutions are closer to the PF and can improve the optimization if we transform them to be feasible, see Figure 3.1 and Figure 3.2. The repair technique performs this by replacing the variables that caused infeasibility (constraint violation) using variable-constraint mapping. Examples of the mapping can be found in Figure 3.3 and Figure 3.4. It is modelled as a matrix of size $l \times n$, where l is the number of constraints and n is the number of variables. A non-zero value in each row indicates that the variable significantly affects the constraint. According to this mapping, infeasible solutions (repair candidates) can be repaired based on the donor solutions.

Repair candidates are selected based on the objective value and constraint violations of the solutions. Generally, infeasible solutions with prominent objective values are considered as repair candidates. One could also add some limitations on candidate selection, such as controlling the maximum number of repair candidates to preserve the diversity of the population. Different from the repair candidates' selection, the strategy of donor selection depends on the current population's situation, partially infeasible or completely infeasible. Following the different donor selection strategies, the repair strategy also changes. Therefore, repair strategies for two different population cases are described below.

Repair Case 1 (Population partially infeasible)

If the current population is partially infeasible, only feasible solutions will be considered as donors since that raises the probability of successful repair. For each repair candidate, the closest feasible solution will be selected as the donor based on their Euclidean distance in the normalized objective space. Therefore, the donor might differ for different repair candidates. Figure 3.3 shows how the repair operator works in this case. For a repair candidate that violates

one or more constraints, all the variables that need to be replaced can be found using the variable-constraint mapping. After that, the repair technique is performed by replacing all marked variables using variables from the donor solution.

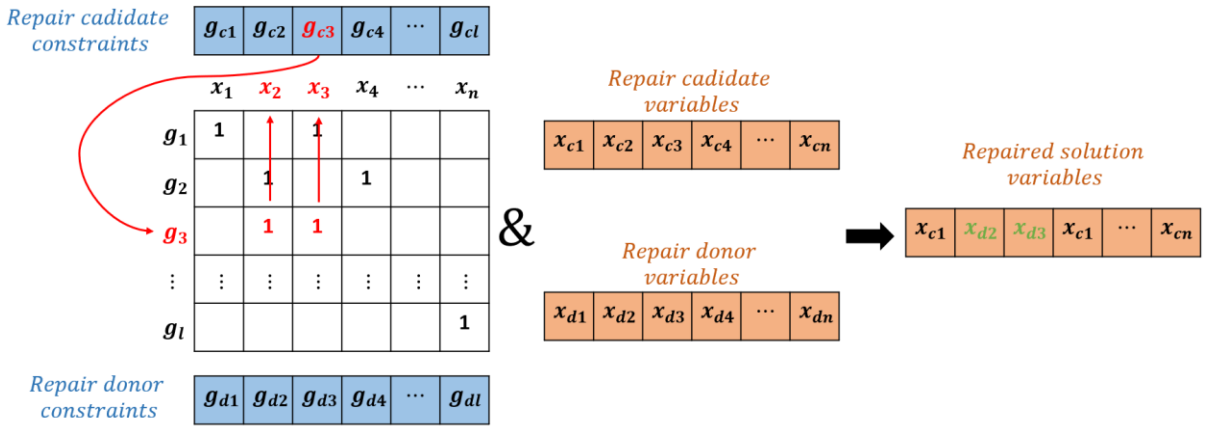


Figure 3.3. Illustration of the repair case 1.

Repair Case 2 (Entire population infeasible)

If the current population doesn't have any feasible solution, prominent infeasible solutions will be selected as repair donors. As Figure 3.4 shows, similar to the repair case 1, if a solution violates a constraint, corresponding variables can be found based on the mapping. Afterwards, the repair technique will mark all the constraints influenced by those variables. Solutions that satisfy all those constraints are the potential donors. The advantage of this modified approach is to prevent violation of other constraints while repairing. If all solutions violate the same constraint, the solution with the least amount of violation is selected as the donor. A few donors might be used for each repair candidate, as they might violate different constraints.

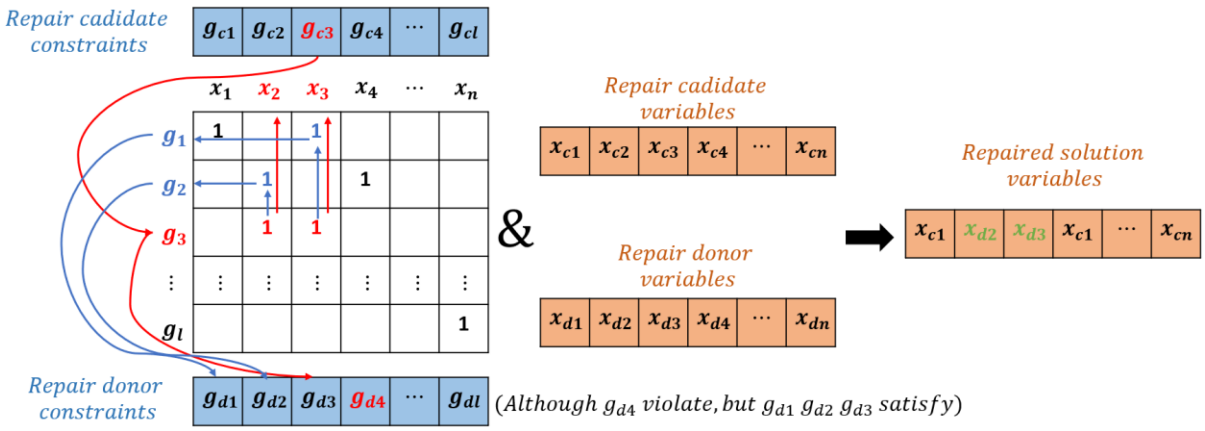


Figure 3.4. Illustration of the repair case 2.

The idea behind repairing is to make small changes to the solution to preserve its position in objective space. For this purpose, the donors are prioritized starting from solutions nearest to the candidate solution while being in the best front (if applicable). Therefore, donors may be different for each repair candidate. Combining the above two cases, the pseudo-code for the modified repair technique is described below. Note that all the solutions are stored in an external archive, to ensure the consistency of the donor selection.

Repair Operator

Input: Input population P_t; Variable-constraint mapping; Number of variables n; Number of constraints l; Generation number t.

Start *Repair Operator*

Compute the number of feasible solutions in P_t as L_{fea}

Collect the feasible solutions as a set F_t

For each solution ($i = 1, \cdots, n$) **do**

If solution is infeasible (repair candidate) **do**

If $L_{fea} \neq 0$ **do**

(i) Sort feasible solutions F_t based on the Euclidean distance to the repair candidate. Select the closest solution as the $donor$.

(ii) Replace the variables that caused infeasibility of the candidate solution from the $donor$ based on the mapping.

Else do

For each violated constraint **do**

(i) Find the significant variables based on the mapping.

(ii) Find the constraints that are affected by those variables.

(iii) Sort all solutions P_t based on the Euclidean distance to the repair candidate. Set the closest solution that satisfies the above constraints as the $donor$.

(iv) Replace the variables that causing the specific constraint violation of the candidate solution from the $donor$ based on the mapping.

End For

End If

End If

End For

End *Repair Operator*

Output: Repaired solutions

3.2 Automated mapping based on ANN

As introduced above, the dependency matrix (the mapping) points out to variables that affect each constraint. It is modelled as a matrix of size $l \times n$, where l is the number of constraints in a problem and n is the number of variables. A non-zero value in each row denotes that the

variable significantly affects the constraint. The mapping for one of the problems in this paper is shown in Figure 5.4. To automatically define the mapping matrix, we use artificial neural networks. Other ML techniques have also been investigated to generate the mapping in a preliminary study, such as Canonical Correlation Analysis (CCA). CCA can find the most significant variable, but could decrease in accuracy for following variables, which is insufficient for this research, thus is not pursued further. Figure 3.5 shows the general architecture of a typical feedforward ANN. A shallow neural network has only one hidden layer. A deep neural network has at least two. It is generally accepted that deep neural networks have higher predicting accuracy than shallow networks. However, deep networks require more training data and more computational resources.

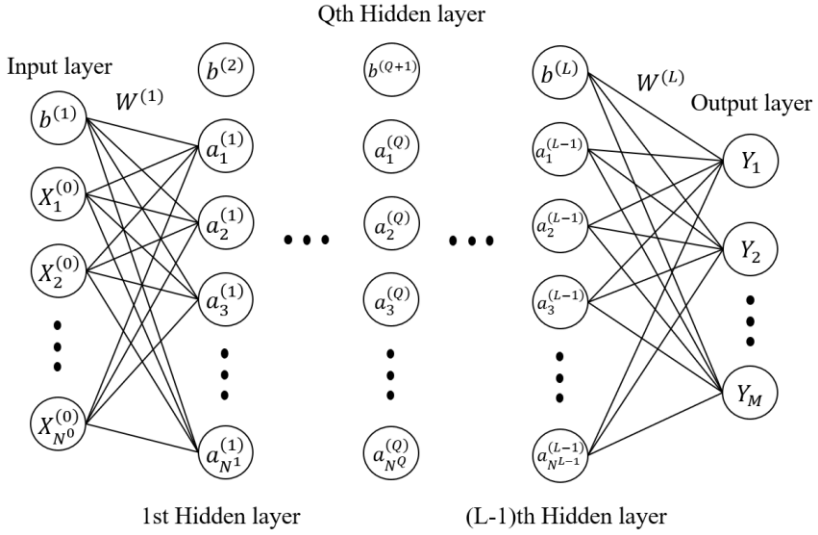


Figure 3.5. Artificial Neural Network architecture.

We define the significance of variables based on their statistical contribution to the optimization criteria. *Significance* was used in [81] to construct the best performing radial basis function neural network. The strength of the connection between neurons in different layers is defined in the current study by the neuron's weight, and if this value is significant, the input neuron has a significant effect on the output neuron. As each input neuron is connected to the output neuron through a network, removing one input neuron will lead to the change of the

output result. The percentage of change indicates the contribution of that input neuron to the output. Optimization variables are input neurons to the ANN, while constraints and objectives (i.e. criteria) are the output neurons. Therefore, the mapping could be generated based on the variables that significantly contribute to the outputs. This method is applicable to artificial neural networks with one or more hidden layers.

For a feedforward neural network, the activation of a neuron $a_I^{(Q)}$ (I^{th} neuron of layer Q) can be represented as:

$$a_I^{(Q)} = G^{(Q)}\left(Z_I^{(Q)}\right), \tag{3-1}$$

where $G^{(Q)}$ is the activation function of the layer Q and $Z_I^{(Q)}$ is the input to the activation function for neuron I. Specifically, the input $Z_I^{(Q)}$ can be written as:

$$Z_I^{(Q)} = \sum_{K=1}^{N^{Q-1}} W_K^{[I](Q)} a_K^{(Q-1)} + \boldsymbol{b}^{(Q)}, \tag{3-2}$$

where $W_K^{[I](Q)}$ is the weight between neuron K in layer $(Q-1)$ to the neuron I in layer Q, $\boldsymbol{b}^{(Q)}$ is the bias vector for hidden layer $(Q-1)$, and N^{Q-1} is the number of neurons in layer $(Q-1)$. $a_K^{(Q-1)}$ is the activation of the previous hidden layer, which is equal to X in the first layer. For example, the final output of the u^{th} output neuron can be expressed as:

$$\begin{aligned} Y_u = g^{(L)}\Big[& w_1^{[u](L)} x_1^{(L-1)} + w_2^{[u](L)} x_2^{(L-1)} \cdots + w_J^{[u](L)} x_J^{(L-1)} \cdots \\ & + w_{N^{L-1}-1}^{[u](L)} x_{N^{L-1}-1}^{[u](L-1)} + w_{N^{L-1}}^{[u](L)} x_{N^{L-1}}^{(L-1)} + \boldsymbol{b}^{(L)} \Big], \end{aligned} \tag{3-3}$$

where L is the depth of the neural network. For a shallow neural network, L is two (hidden and output layer). Eq. (3-2) shows that the input of the activation function could be represented as the linear combination of weights multiplied by activation of the previous layer. For example, if the input value is fixed, the corresponding weight of each input neuron could directly control the input of the activation function.

Neurons with larger weight contribute more to the activation function and vice versa. Based on this, we introduce the *Significance* of an input neuron. For an input neuron I, we can write the activation for the neuron of the first hidden layer in expanded form as:

$$a_I^{(1)} = G^{(1)}\left[W_1^{[I](1)}X_1^{(0)} + W_2^{[I](1)}X_2^{(0)} \cdots + W_J^{[I](1)}X_J^{(0)} \cdots + W_{N^0-1}^{[I](1)}X_{N^0-1}^{(0)} + w_{N^0}^{[I](1)}X_{N^0}^{(0)} + \boldsymbol{b}^{(1)}\right], \tag{3-4}$$

The activation of this neuron when the weight $W_J^{[I](1)}$ is equal to 0 can be written as:

$$a_{I-J}^{(1)} = G^{(1)}\left[W_1^{[I](1)}X_1^{(0)} + W_2^{[I](1)}X_2^{(0)} \cdots + W_{J-1}^{[I](1)}X_{J-1}^{(0)} + W_{J+1}^{[I](1)}X_{J+1}^{(0)} \cdots + W_{N^0-1}^{[I](1)}X_{N^0-1}^{(0)} + W_{N^0}^{[I](1)}X_{N^0}^{(0)} + \boldsymbol{b}^{(1)}\right], \tag{3-5}$$

where $a_I^{(1)}$ is the original output of the neuron I in the first hidden layer and $a_{I-J}^{(1)}$ is the activation of the neuron I without input J. Therefore, the contribution of the neuron J to the activation is controlled by the weight $W_J^{[I](1)}$, which can be set to zero to test the *significance* of the input neuron X_J.

Furthermore, we introduce *accuracy* to evaluate the performance of the neural network. Abbreviated as *acc*, it is calculated as:

$$Acc = 1 - avg\left[\sum_{K=1}^{nt} abs\left\{\frac{Y_K(predict) - Y_K(exact)}{Y_K(exact)}\right\}\right], \quad (3\text{-}6)$$

where *nt* is the testing data size, $Y_K(predict)$ is the predicted output value for test example *K*, and $Y_K(exact)$ is the correct output given by the test data. The value in the square bracket is averaged over all the test data, due to statistical variation. Therefore, for *accuracy* equal to one the network perfectly predicts constraints and/or objectives in optimization. Common loss functions, such as Mean Square Error (MSE) or Root Mean Square Error (RMSE), can vary greatly depending on the type of problem (constraints in our case). To make sure that the accuracy of the ANN is not affected by different ranges of inputs, we use Eq. (3-6) to calculate the accuracy.

Using the same test dataset for ANN with and without W_J, we can define accuracy of both networks. The loss of accuracy is defined as significance, and for variable *J*, the significance can be expressed as:

$$Significance(J) = Acc(ANN) - Acc\big(ANN\ without\ W_J\big) \quad (3\text{-}7)$$

The more significant the neuron is, the larger error will occur if removed. Significance close to $Acc(ANN)$ means that the constraint is completely controlled by this variable, and significance equal to 0 indicates that the constraint is un-correlated to the variable. We introduce a threshold which is used to determine whether the variable is significant or not. In this study, the threshold value of 0.1 is used, and variables whose significance is larger than the threshold (if they exist) will be determined as variables that affect that constraint. For problems in this research only one or two variables are above the threshold, for each constraint, and in some cases the number is zero, if constraint is never violated. This approach allows us to automatically generate the link. In addition, the trained ANN can be used as the surrogate

model in optimization. Surrogate model creation and automated mapping generator are shown in pseudo-code 2.

Pseudo-code 2: Surrogate model and automated mapping generator

Input: Number of variables n; Number of constraints l; *Significance* threshold η

Parameter: ANN accuracy T (T = 99% here); *max number of training iterations* = 20*;*

Start *Surrogate model and automated mapping generator*

Randomly generate training data, evaluate objectives and constraints

While $Acc \leq T$ or *max number of training iterations not reached* **do**

Train ANN based on the training data.

Update ANN if $Acc > Acc(Current\ best)$.

End While

For each constraint ($i = 1, \cdots, l$) **do**

For each input variable ($j = 1, \cdots, n$) **do**

Set $W_j^{[I](1)}(I = 1, \cdots, max) = 0$, and reconstruct the ANN.

Compute *Significance(j)* for each variable.

End For

Rank *Significance* for each variable, select the variable(s) with *Significance* greater than η, generate row vector of *Mapping*.

End For

End *Surrogate model and automated mapping generator*

Output: Automated mapping; Artificial Neural Network surrogate model

3.3 Implementation to PSO

In this thesis, the novel repair-based CHT is implemented into PSO as one of the prominent single-objective algorithms. In the modified PSO, particles will not move toward infeasible solutions. Only feasible solutions are considered for particle's and swarm's best position. To ensure exploration and exploitation of the design space, two more features are introduced for the repair technique in PSO: dynamic selection of repair candidates and selection of repair donor based on Gaussian distribution.

Dynamic selection of repair candidate

Instead of only repairing the infeasible solutions with better objective value than the G_{best}, infeasible solutions whose objective value is slightly worse than G_{best} could also be repaired. An example can be found in Figure 3.6, where the second-best infeasible solution is also repaired. Initially, infeasible solutions could easily surpass G_{best} in terms of the objective as the design space is not deeply explored. Later when the population gets closer to the global optimum, generated infeasible solutions become less likely to outperform G_{best}. In this case, to further explore the objective space, the criteria of selecting the repair candidates becomes less strict as the generation number increases.

In other words, for a minimization problem, infeasible solutions with an objective value smaller than an adaptive threshold can be repaired. This value can be formulated as:

$$Candidate\ selection\ threshold = \varepsilon \times Objective(G_{best}), \tag{3-8}$$

where ε is defined in this thesis as the dynamic expansion factor, which can be expressed as:

$$\varepsilon = 1 + \frac{t}{Max\ Generation}, \tag{3-9}$$

where t is the current generation number. At the beginning of the optimization, the repair technique works normally as ε is equal to one and increases to two in the end.

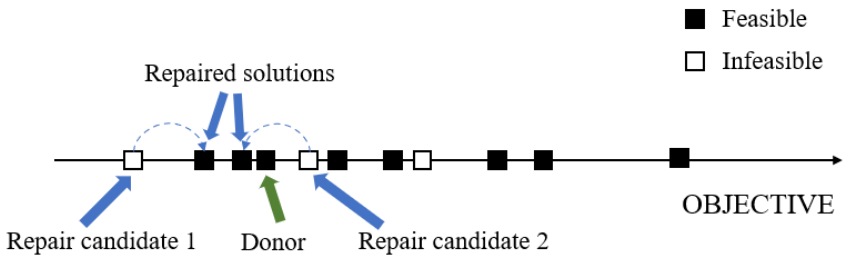


Figure 3.6. Dynamic selection of repair candidate.

Selection of repair donor based on Gaussian distribution

Repair operator allows infeasible solutions to become feasible. However, many solutions might share the same variables after the repair, which might influence the population's diversity. In single-objective problems, objective space is in 1-D, meaning that the closest feasible solution of all prominent infeasible solutions is G_{best}. This will dramatically reduce population diversity if all the candidates are being repaired based on the same donor. For this reason, we give some randomness for donor selection. Gaussian distribution has been used here to decide the donor index as this is one of the most basic probability distributions. Figure 3.7 shows an example of this procedure: instead of selecting the closest feasible solution, the second or the third closest feasible solution can also be selected as repair donors. In this research, two parameters μ and σ are used to control the shape of the probability density function. The index of the repair donor could be written as:

$$index_{donor} = \max\left(1, round\left(abs\left(normrnd(\mu, \sigma)\right)\right)\right), \quad (3\text{-}10)$$

where μ is the mean or expectation of the gaussian distribution and σ is the standard deviation. In this thesis, μ is defined as one and σ is defined as two. Statistically, based on Eq. (3-10), the probability of selecting the i^{th} closest feasible as the repair donor is shown in Table 3.1.

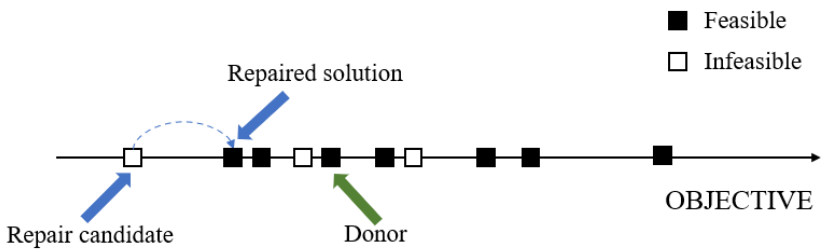


Figure 3.7. Repair donor selection using Gauss probability.

Table 3.1. Donor selection probability.

Donor index	1	2	3	4	5	6	7	8
Probability (%)	49.21	24.48	14.71	7.58	3.10	0.63	0.18	0.1

3.4 Implementation to MOEA/D

In MOEA/D, the proposed CHT has been embedded to update the position of the prominent infeasible solutions at each generation. Another modification of MOEA/D in this thesis is that each solution is re-assigned to the weight vector that gives it the lowest aggregate value. Because the essence of MOEA/D is to optimize multiple single-objective problems, and this modification can position every solution closer to the PF. As shown in Figure 3.8, any infeasible solution that is better than the current best solution of a weight vector will be selected as the repair candidate. Furthermore, the donor is selected from the repair candidate's corresponding neighborhood solutions. If all neighborhood solutions are infeasible, the donor will be selected from the nearby neighborhoods, starting from the nearest one.

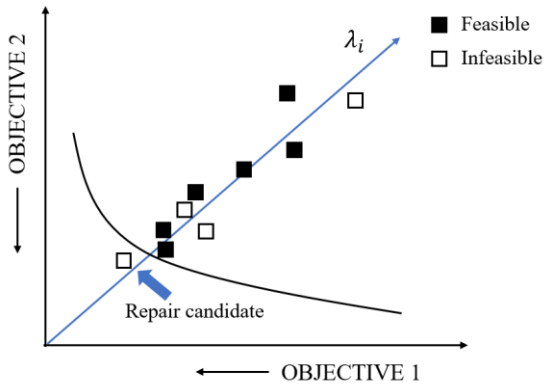


Figure 3.8. Repair candidate selection in MOEA/D.

3.5 Implementation to NSGA-II

The proposed approach is also embedded into NSGA-II for handling multi-objective problems. Original NSGA-II uses constrained dominance principle (CDP) as CHT. CDP says that the solution with a larger constraint violation is always dominated by the one with a lower constraint violation, and an infeasible solution is always dominated by a feasible one.

The operating procedure of the modified NSGA-II is the same as [37], where infeasible solutions are also sorted based on the non-domination principle by ignoring their violation and only considering their objective values. As could be seen from Figure 3.9, $\gamma_{(j)}^{(t)}$ denotes the j^{th} ranking of the infeasible front at generation t, while the symbol ω denotes the feasible front. Prominent infeasible solutions from different ranks are collected as $\alpha^{(t)}$ based on non-domination sorting and repaired.

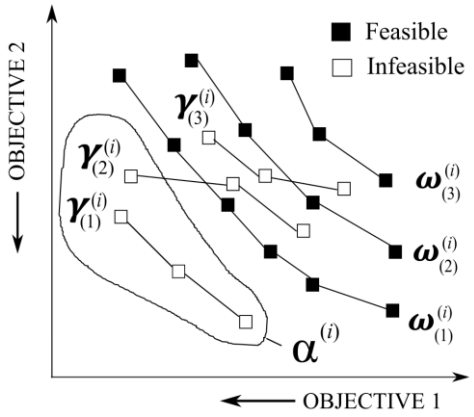


Figure 3.9. Possible location of sorted solutions in the population in NSGA-II [37].

In modified NSGA-II, the repair-based CHT is performed before the genetic operator and non-domination sorting. In each generation, solutions in $\alpha^{(t)}$ from P_t will be selected as repair candidates, and they will be modified based on the donors. The donor selection strategy in modified NSGA-II follows the procedure described in Section 3.1. Afterwards, Q_t is composed of repaired solutions and solutions generated by the genetic operator. Non-dominated sorting is conducted later to select solutions for the next generation.

3.6 Repair-based Optimization Framework

The flowchart of the repair-based optimization framework is presented in Figure 3.10. In this thesis, the three algorithms follow a similar flow except for the selection of repair candidates and donors, which has been explained in the previous sections. In this framework, a set of solutions is randomly generated at the beginning to train the ANN. Afterwards, ANN defines the variable-constraint mapping and is ready to be used as surrogate model in optimization (used in this thesis only for the tanker problem). The detailed setting of ANN training can be found in Section 4.4.2 and Section 5.1. After the complete preparation of the surrogate model and automated mapping, the optimization process starts. In each generation, repair operator is invoked before evaluation. Solutions are updated afterwards using surrogate model.

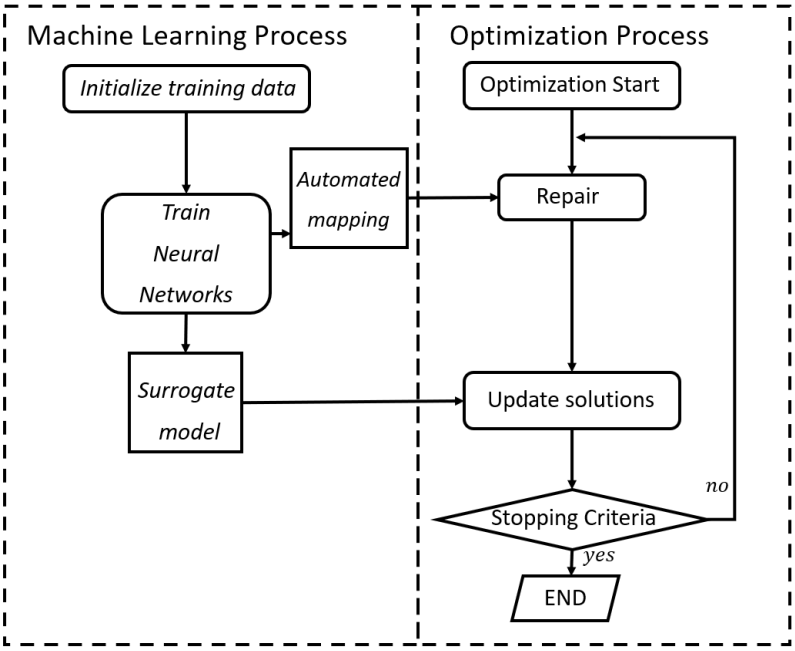


Figure 3.10. Repair based optimization framework.

It is worth mentioning that the surrogate model is used throughout the whole tanker optimization. However, since the surrogate model is trained from randomly generated data, its accuracy inevitably decreases as the exploration of the design space progresses. To validate the influence of the surrogate model's accuracy on the algorithm's performance, we conducted an additional test on NSGA-II with the repair where the algorithm will switch from the surrogate model back to the original structural model at 80% of maximum generations. The results in Appendix A show that NSGA-II using the surrogate model throughout the whole process can reach the same level of accuracy at the end of the optimization. Therefore, limited by the computational resources, we apply the surrogate model for the whole optimization process in the rest of the thesis.

4 Test case introduction

4.1 Mathematical benchmark problem

In this thesis, a constrained benchmark mathematical problem called OSY is used to evaluate the performance of both single- and multi-objective optimization algorithms. The details of the OSY problem are described below, where two objective functions need to be optimized subject to six constraints.

In multi-objective testing, both objectives are considered, while only the first objective is considered for single-objective testing. OSY is defined as:

$$min\begin{Bmatrix} f_1(x) = -25(x_1-2)^2-(x_2-2)^2-(x_3-1)^2-(x_4-4)^2-(x_5-1)^2 \\ f_2(x) = \sum_{i=1}^{6} x_i^2 \end{Bmatrix} \tag{4-1}$$

$$\text{s.}t. = \begin{cases} g_1(x) = x_1 + x_2 - 2 \geq 0 \\ g_2(x) = 6 - x_1 - x_2 \geq 0 \\ g_3(x) = 2 + x_1 - x_2 \geq 0 \\ g_4(x) = 2 - x_1 + 3x_2 \geq 0 \\ g_5(x) = 4 - (x_3-3)^2 - x_4 \geq 0 \\ g_6(x) = (x_5-3)^2 + x_6 - 4 \geq 0 \end{cases} \tag{4-2}$$

$$0 \leq x_1, x_2, x_6 \leq 10,$$

$$1 \leq x_3, x_5 \leq 5.$$

4.2 Truss problem optimization

In this thesis, the common 10-bar truss design problem is used for single-objective algorithms' performance testing. The configuration of this problem is shown in Figure 4.1, where the cross-sectional area of each bar needs to be optimized to minimize the total weight [82].

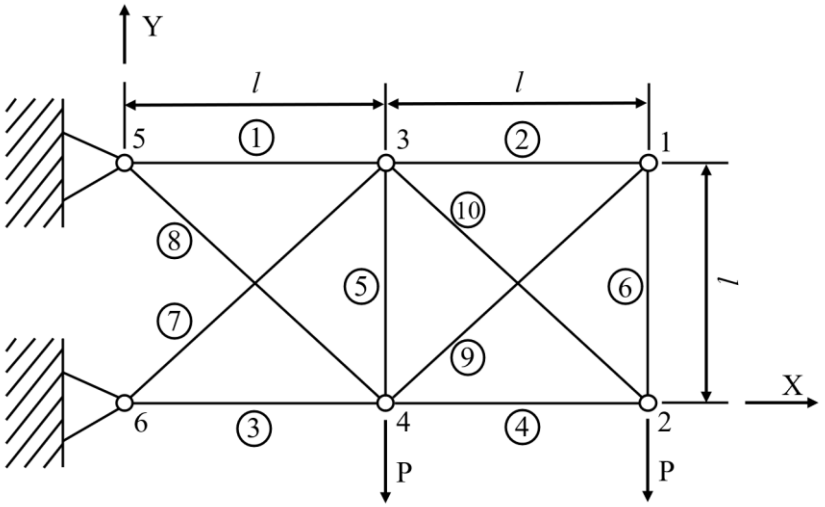


Figure 4.1. Configuration of the ten-bar truss.

In Figure 4.1, the length of the members l_{7-10} is 509 in, while the length of the rest members is 360 in. The member's cross-sectional areas are denoted as A_i, ranges from 0.1 in^2 to 33.5 in^2. The node 5 and node 6 are fixed by the hinges [83]. Thus, the displacement of node 5 and node 6 are zero. Two external loads P of 100 kips are applied in node 2 and node 4, respectively. Furthermore, the maximum allowable stress in any member of the truss is ± 25 ksi, and the maximum nodal deflection in both vertical and horizontal directions is ± 2 inch [84]. The density of the truss material is 0.1 lb/in^3 and the modulus of elasticity is 10^7 psi. Based on this, this problem can be formulated as:

$$\min \qquad W(A) = 0.1 \sum_{i=1}^{10} l_i A_i, \tag{4-3}$$

$$\begin{aligned} s.t. \quad & \sigma_i \leq 25ksi, (i = 1,2,\cdots,10), \\ & v_{kx}, v_{ky} \leq 2in, (k = 1,2,3,4), \end{aligned} \tag{4-4}$$

where $W(A)$ is the weight function representing the objective, l_i is the length of the i^{th} bar element, σ_i is stress in each truss member, and v_{kx} and v_{yk} are the displacements of the node

k in *x* and *y* direction. The response of the structure has been analyzed using FEA code in Matlab.

4.3 Structural optimization of a tanker

For real-life engineering optimization problem, we consider chemical tanker which is 180 m long, 32 m wide and 18 m deep, with a draught of 11.5 m. It is operating under normal service condition. The layout of this tanker is shown in Figure 4.2. Optimization is performed on longitudinal structural members. Problem modelling, optimization and result analysis is conducted using Matlab and Fortran. The problem was introduced in [42] and optimized later in [85] and [37]. Variables, structural constraints and objectives are described below.

4.3.1 Design variables and parameters

The tanker is subjected to lateral pressure from sea and cargo. Because of the local difference between buoyancy force and ship's weight, vertical shear force and bending moment arise along the ship, under sagging and hogging conditions [86]. Figure 4.2 (a) shows the distribution of vertical bending moment and shear force along the ship. The maximum bending moment is $2.93 \cdot 10^6$ kNm and $2.41 \cdot 10^6$ kNm for sagging and hogging conditions, respectively, and the maximum vertical shear force is $48 \cdot 10^3$ kN for both conditions.

Figure 4.2 (b) shows the detailed drawing of the half cross-section of the tanker. Due to symmetry, one half of the midship section is considered in optimization, nonetheless mass is presented as the total mass of the structure in the results. The structure consists of 47 strakes, each of them defined by five parameters: plate thickness, stiffener size, stiffener type, number of stiffeners, and panel's material type. Plate thickness and stiffener size of each strake are considered as the variables of the problem (total of 94 variables). The other three are assumed as fixed, together with the transverse structures. The choice of decision variables will influence

the final values of the objectives and consideration of all five parameters above could lead to more optimized structure, however, the aim of the study is to compare different optimization algorithms, and this comparison remains qualitatively the same once optimization problem is set up. Plate thickness is considered from 5 mm to 26 mm with a step of one millimeter. Stiffeners are selected from a standard table of profiles given by the steel producers [87], ranging from HP 100x5 to HP 430x17.

4.3.2 Design constraints

In this problem, the yield strength of the cargo tanks' plating is 460 MPa and for the rest of the structure it is 355 MPa. 8 constraints are considered for each strake, namely stiffener yielding, plate yielding, plate buckling, stiffener web buckling, flange buckling, lateral buckling, tripping and crossover. To prevent uncontrolled panel collapse, crossover constraint ensures that the global buckling strength is larger than the plate or stiffener buckling strength [88]. Stresses from global loads (shear force and bending moments) are calculated using Coupled Beam (CB) method [89] at each part of the cross-section under both hogging and sagging conditions. Local stresses (plate and stiffener under pressure loading) are calculated using classic plate and beam theory. Local stresses are superimposed to the ones arising from global loads and the worst possible combination is taken at each strake. Finally, constraints are normalized based on non-linear normalization function [90] :

$$g_j(\boldsymbol{x}) = \frac{A_j(\boldsymbol{x}) - \left|B_j(\boldsymbol{x})\right|}{A_j(\boldsymbol{x}) + \left|B_j(\boldsymbol{x})\right|}, \tag{4-5}$$

where $A_j(\boldsymbol{x})$ is the capacity of the structural element j, and $B_j(\boldsymbol{x})$ is the stress acting on it. Normalized constraints range from -1 to 1, where negative value indicates violation and zero represents the boundary. Large majority of computational time in the tanker problem goes towards the assessment of constraints.

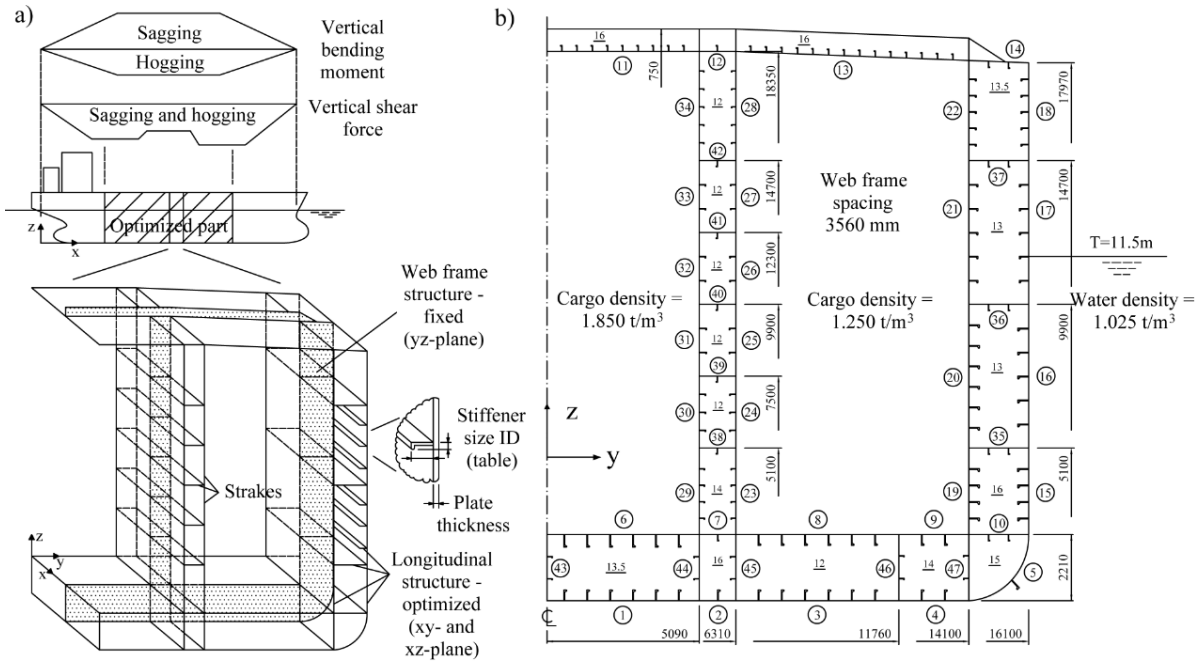


Figure 4.2. Cross-section of the tanker [37]. Strake numbers are shown in circles. Dimensions are in mm.

4.3.3 Design objectives

Two objectives are considered: (a) minimization of the structural mass and (b) maximization of deck's adequacy. Mass is calculated by multiplying steel density of 8 t/m^3 with cross sectional area of longitudinal members, extending them for the whole ship's length (and breadth), effectively considering the ship to be prismatic. Moreover, 21.44 t is added as the mass of the transverse structure every 3.56 m. The second objective aims to increase structural safety by reducing stresses in the deck in order to decrease occurrence of significant fatigue cracks. The deck is selected as the critical part of the structure since other parts of the structure have higher redundancy due to the double-plated construction. Therefore, the second objective is formulated as the sum of the normalized deck constraints, which reach the maximum value of one when stress approaches zero; see Eq. (4-5). The deck's adequacy was taken as a simplified measure of safety for the purpose of comparing the optimization algorithms. More refined models should be used for engineering purposes; see e.g. Ref. [91] .We can anticipate

a conflict between the two objectives: lighter solutions have higher stresses in the deck, thus lower deck adequacy. The second objective is leading the solutions away from the constraint boundary and deeper into the feasible space, which makes the structure heavier. It is worth mentioning that since ANN is used as the surrogate model to predict constraints, and constraints are used to calculate the second objective (adequacy), the surrogate model is predicting the second objective for the tanker problem, until we revert back to the original structural code.

4.4 Control parameters

4.4.1 Parameter settings in optimization algorithms

Since certain steps in metaheuristic optimization algorithms are carried out at random, 100 independent runs are performed for the OSY problem and the 10-bar truss problem. Limited by the computation resources, the tanker problem is optimized for 30 times. An initial population can be manually provided by the user beforehand so that the algorithm could get better results from a good starting point. To prevent bias and allow fair comparisons, initial populations can be generated randomly using the uniform distribution of the variables between their upper and lower bounds. For complex engineering problems, the randomly generated initial population could be completely infeasible which is challenging for optimization algorithms to start from. Nonetheless, this is a suitable way to evaluate the algorithms' performance. Furthermore, all simulations for the tanker are performed based on the same initial populations to alleviate the influence of the optimization algorithms' starting position.

The following general control parameters are used for all test optimization algorithms for consistency. Population size is set as 100 for all test problems. It is worth mentioning that the size of the population is fixed even when using the repair, in which case the number of "normal" solutions is reduced. The number of generations is 1000 for tanker optimization, while this

number is 50 for OSY and truss problem since these are easier to optimize. We use continuous variables for OSY and truss problem. However, the variables for the tanker problem are discrete, thus a solution is defined with a 400-bit-long chromosome.

Algorithm-specific parameters are listed in Table 4.1. Most of the parameters are taken from the suggested value in their original papers as they all show the best performance in the preliminary tests [1,16,18,37,71,72]. An exception is MOGWO, whose control parameters are optimized to obtain a better result.

Table 4.1. Parameter settings for testing algorithms.

Algorithms	Settings
NSGA-II	$p_c = 0.9$; $p_m = 1/n$
MOEA/D	Neighborhood size $T = 20$
PSO	$w = 0.99$; $c_1 = 1.5$; $c_2 = 2$
PSO repair	$\sigma = 2$; $\mu = 1$
MVO & MOMVO	WEP was linearly increased from 0.2 to 1; $p = 6$ in TDR
IGWO	a was linearly decreased from 2 to 0
MOGWO	$alpha = 0.5$; $beta = 10$; $nGrid = 10$

Note that *n* is the number of variables in continuous problems and the number of bits in discrete problems.

4.4.2 Hyperparameter settings in ANN

Other than the control parameters of the optimization algorithms, hyperparameter settings of the ANN are also listed here. It is well known that the hyperparameters profoundly affect its accuracy and the time required for training. Hyperparameters considered here are the training size, learning rate, activation functions, loss function optimizer, regularization method and architecture. Except for the ANN architecture, the other hyperparameters could be easily optimized. Several preliminary tests have been done and the settings of those hyperparameters are shown in Table 4.2.

Table 4.2. ANN hyperparameter settings.

Hyperparameters	Settings
Initial learning rate	0.1
Activation functions	Sigmoid and Purelin
Optimizer	Levenberg-Marquardt backpropagation
Regularization method	Early stopping
Maximum number of epochs	2000

It is worth mentioning that the early stopping is applied to prevent overfitting. To avoid underfitting, maximum of 2000 epochs has been set for the optimizer to fully reach the optimal weight matrices. Sigmoid and Purelin activation functions are used in the hidden layers and the output layer, respectively. Levenberg-Marquardt backpropagation results in the best training among other options in Matlab.

5 Results

5.1 ANN accuracy investigation

As mentioned in Section 4.4.2, ANN's accuracy largely depends on hyperparameter settings. Some hyperparameters have been optimized as shown in Table 4.2.

In this thesis, the purpose of applying ANN is to help improve optimization algorithm's efficiency. Thus, minimizing ANN training time is equally important as maximizing ANN accuracy. The hyperparameters that influence both factors the most are training data size and ANN architecture. The influence of data size will be discussed in the end as it is widely known that more data improves the accuracy. In contrast, ANN architecture needs to be adjusted properly, as a wider and deeper network will slow down the training process and may lead to overfitting. Further, a narrow and shallow ANN cannot meet the desired accuracy. In this thesis, we present the performance for different ANN architectures with the other hyperparameters fixed. We focus on the tanker optimization case, since OSY and truss problem are much simpler. For the tanker, ANN is used as surrogate model, besides defining the variable-constraint mapping. The term "total time" is used to denote time it takes to train a network up to 20 times to predict all constraints in a problem. We focus on constraints since their assessment takes the most of computational resources; objectives are simple explicit functions. Since large number of ANNs need to be trained, in order to save computational time, they are not be trained after reaching accuracy of more than 99% (Eq. (3-6)). For come constraints that are hard to predict, ANN needs to be trained multiple times. The maximum number of training attempts is set to 20; if accuracy is less than 99% for all, the best network is selected. The number of independent training runs is set to 20 because of the influence of training data sequence on the ANN accuracy. Hence, for each independent training, we randomly shuffle

the sequence of training data. The error is reported for the most accurate network. The training set consists of 3,000 randomly generated solutions.

Figure 5.1 shows the prediction error and training time when using shallow ANN with different numbers of outputs (constraints) for the tanker. We would normally want to predict a larger number of constraints with a single network. However, we can observe from Figure 5.1 that both the error and training time increase with a larger number of outputs, thus a network with only one output is selected as the most appropriate. We can see that ANN shows the lowest error of 3.57% with five neurons in the hidden layer. The network takes 1.3 hours to train, which is acceptable for the optimization process, given that the optimization of the tanker takes about 100 hours using the original structural model with 1000 generations. Further increasing the neural network capacity decreases the accuracy as more neurons may cause overfitting.

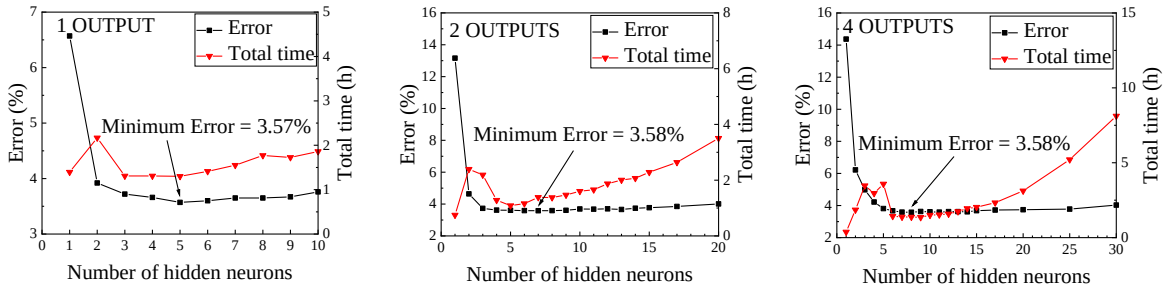


Figure 5.1. Accuracy and training time for shallow ANN used on tanker optimization problem.

Figure 5.2 shows the performance of a neural network with two hidden layers and a single output. The lowest error is 3.55% when the ANN contains five neurons in the first hidden layer and three neurons in the second. This is very close to the performance with a single hidden layer, but it requires more time to train. The training time for the best performing deep neural network (two hidden layers) is 47% higher than the time for the best performing shallow network. Similar results are found for deep neural networks with two and four outputs, where the smallest error is 3.55% and 3.57%, respectively, but it requires longer training. The same trend for accuracy and training time continues with higher number of hidden layers, thus is not

pursued. Therefore, a shallow ANN is used in continuation, where the network with one to seven neurons is trained, tested and the best one is selected for each constraint (376 constraints in the tanker case). Generally, if a more complex problem is optimized, deep network might be beneficial to reduce the error.

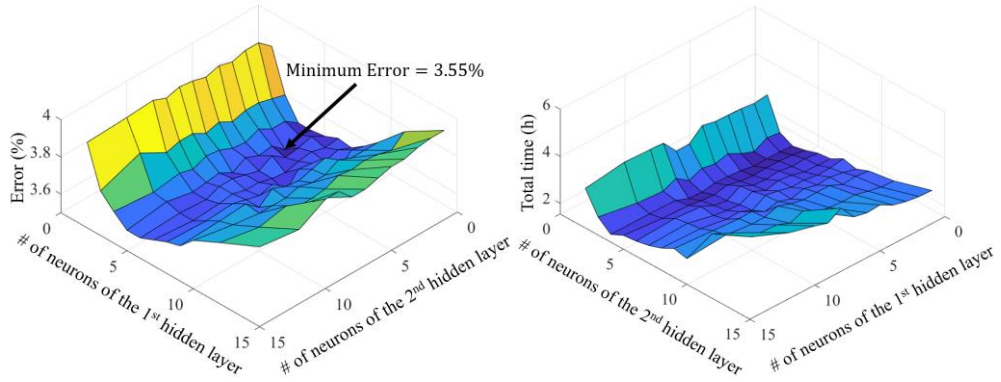


Figure 5.2. Accuracy and training time for deep ANN with two hidden layers and one output used on tanker optimization problem.

The size of the training data is also an important factor that influences the accuracy of ANNs. As shown in Figure 5.3, more data causes the error to decrease quite fast initially and afterwards much slower. Training time increases gradually from 2.25 hours to 57.8 hours as the data size increases from 500 to 10,000. The error of ANN goes down from 6.56% to 3.13%. Considering both ANN training time and accuracy, optimization results in Section 5.3 and Section 5.4 involve ANN trained with 10,000 solutions.

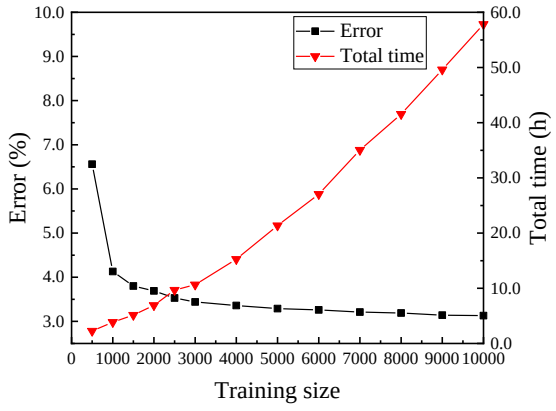


Figure 5.3. Influence of the training data size on the accuracy of a shallow neural network for tanker optimization problem.

5.2 ANN-based variable-constraint mapping

As discussed in Section 3.2 the variable-constraint mapping outlines the most significant variables that affect constraints. More accurate variable-constraint mapping could improve the efficacy of the proposed repair technique. Section 3.2 describes the procedure to automate the mapping using a trained ANN. In this subsection, the effect of training data size on the accuracy of the mapping is discussed.

Figure 5.4 (a) shows the mapping for the OSY problem, which could be defined manually based on the closed-form functions. For the truss problem or tanker that is much harder (even impossible) to define, thus we use ANN as outlined in Section 3.2. The training data are randomly generated set of variable values and their resulting constraints. We can see that even a relatively small data size of 60 points leads to the correct definition of the mapping matrix for the OSY.

On the other hand, the constraint formulas for the truss problem are more complex. Both stress and displacement constraints are influenced by several structural members. A user can hardly define the variables that significantly influence each constraint. We assume that ANN trained with high amount of data leads to the correct mapping. We can see from Figure 5.4 the convergence of the mapping for the truss problem as the data size increases.

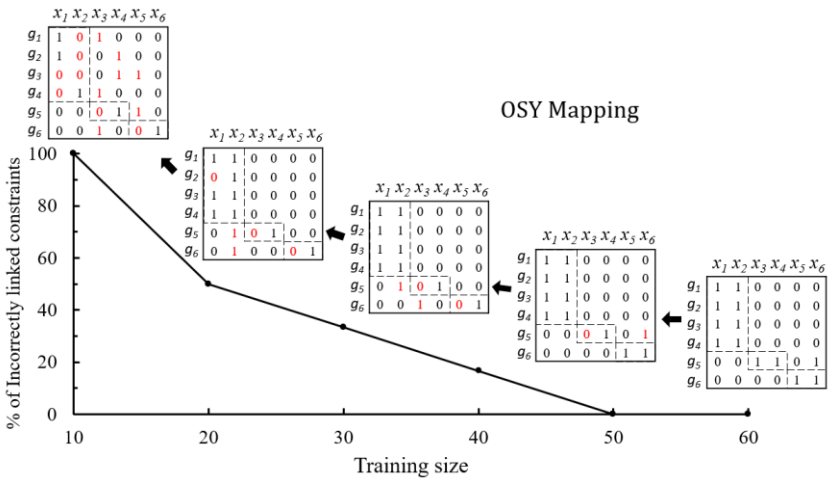
The difference is only 5.6% (1 constraint incorrectly predicted) when going from 4000 to 2000 data samples. It is worth mentioning that due to the randomness of ANN, the *significance* value generated for each variable fluctuates. Thus, those variables with *significance* around the threshold η might not be selected sometimes. This explains the fluctuation of difference when the ANN contains more training data.

Table 5.1 shows the details of the automated mapping for the truss problem, including all constraints and significant variables. We can notice that some members can be categorized into groups, and the members of the group will affect each other's stress. Two groups are found: bar #1, #3, #7, #8, and bar #2, #4, #6, #9, #10. Furthermore, the variables that influence the displacement are mostly bars #1, #3, #7, and #8.

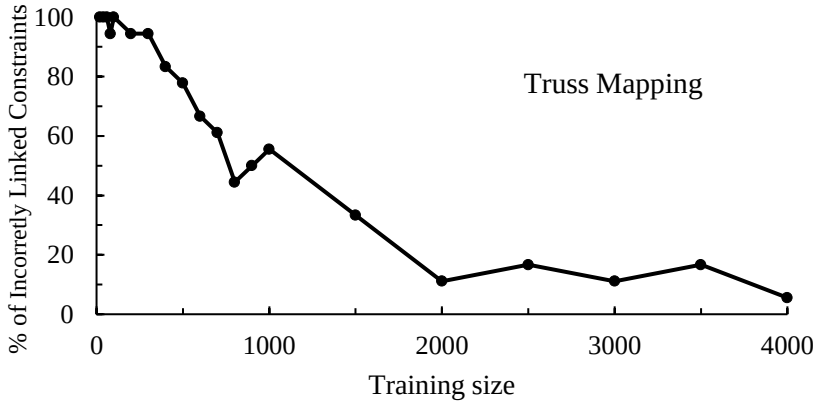
Table 5.1. Most significant variables which influence each constraint for the truss problem, defined by the automated mapping algorithm.

Constraint	Member(s) that affect the constraint:
Stress in bar #1	1,3,7,8
Stress in bar #2	2,4,6,9,10
Stress in bar #3	1,3,7,8
Stress in bar #4	2,4,9,10
Stress in bar #5	3,5,7,8,9
Stress in bar #6	4,6,9,10
Stress in bar #7	1,3,7,8
Stress in bar #8	1,3,7,8
Stress in bar #9	2,6,9,10
Stress in bar #10	2,6,9,10
Node 1 displacement in x-direction	1,3,7,8
Node 1 displacement in y-direction	1,3,7,8
Node 2 displacement in x-direction	1,3,7,8
Node 2 displacement in y-direction	1,3,7,8,9
Node 3 displacement in x-direction	1,2,3,7,8
Node 3 displacement in y-direction	1,3,7,8,9,10
Node 4 displacement in x-direction	1,3,7,8
Node 4 displacement in y-direction	1,3,7,8

a)



b)



c)

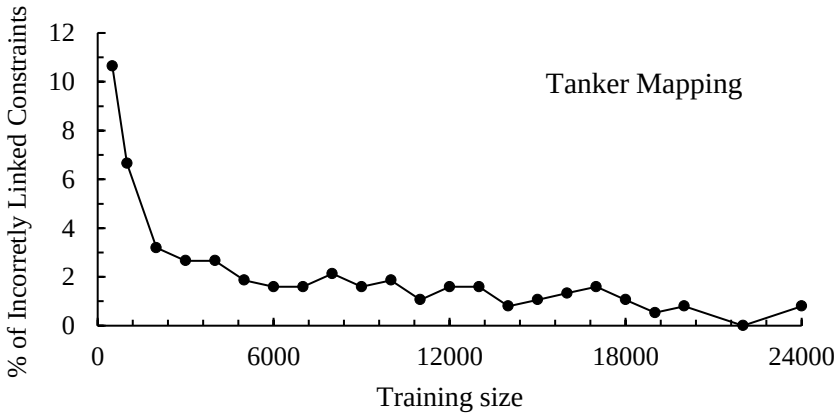


Figure 5.4. Accuracy of automated mapping a) OSY problem b) Truss problem c) Tanker problem.

The tanker problem is even more complex and requires a much larger training data. Similar to the truss problem, we don't know the correct mapping matrix for such a complex problem, as constraints could be affected by any number of variables. Therefore, we also assume that the prediction is correct for a large data size. We can see from Figure 5.4 that the deviation from the mapping decreases as we increase the training data size. Initially with 500 data samples, the deviation is 10.6%, which decreased rapidly to 3.19% with 2000 points. The deviation can be further decreased to 0.79% when the data size increases to 20,000, even though a large training time is required. Thus, a training size of 3,000 data points (with 2.66% mapping deviation) is used to be consistent with ANN training for the purpose of surrogate model definition. It is worth mentioning that the automated mapping for the tanker is generally sparse, having only 220 non-zero members out of 376×94 matrix ($l \times n$). This is even more sparse than assumed in Ref. [37].

Table 5.2 shows the most significant variable which affects each type of constraint for the tanker problem, defined by the automated mapping. In some cases, the constraints are always satisfied, which can be explained by the choice of variable bounds. This shows one advantage of using ANN to define the mapping: only 'active' constraints are linked to their variables, which is hard for a user to assume beforehand, for such complex problems. Preliminary investigation revealed similar success of the repair technique in creating feasible solutions using automated mapping and the mapping assumed in Ref. [37]. Since the training data shows that some constraints are never violated, training could be omitted in such cases, but since some of them are used for the second objective, that is not pursued for the ease of implementation. Moreover, the total training time is relatively short (10 hours) in comparison to the normal optimization run with the original structural model (100 hours).

Another phenomenon observed in automated mapping is that ANN considers all variables as 'insignificant' for some constraints. This validates what we assumed in Section 3.2: multiple

variables are equally important, and no variable significantly influences that constraint. This shows the second advantage of automated mapping: even if no variable is selected, insignificant variables will not be selected to define the mapping.

Table 5.2. The most significant variable that influences each constraint for the tanker problem, defined by the automated mapping algorithm.

Constraint	Automated mapping defined it as the function of [*]:
Plate yielding	Plate thickness in 70% of strakes[*]
Plate buckling	Plate thickness in 95% of strakes[*]
Stiffener yielding	Stiffener size in 57% of strakes[*]
Stiff. web buckling	Constraint is always satisfied[*]
Stiff. flange buckling	Constraint is always satisfied[*]
Stiff. lateral buckling	Stiffener size in 70% of strakes[*]
Stiff. torsional buckling	Plate thickness in 10% of strakes, stiffener size in 15% of strakes[*]
Crossover	Plate thickness in 8.5% of strakes, stiffener size in 17% of strakes[*]

[*] In remaining strakes the constraint is always satisfied.

5.3 Optimization comparisons for single-objective problems

In this subsection, the results of single-objective optimization algorithms are compared. We apply static penalty in PSO, IGWO, GSA, and MVO to handle constraints. Together with PSO with repair, the minimum objective value found by those algorithms at each generation for each test problem is shown in Figure 5.5 to Figure 5.7. Note that the initial study tested several penalty factors for the static penalty approach; the best performance for all problems was achieved with a high penalty factor. Thus, we apply the penalty factor as 10^7 for most comparisons. Moreover, all the tests are conducted for 100 runs except the tanker problem which is limited by the computational resources, and the median value is selected for plotting the results.

Figure 5.5 shows the history of the minimum value of the first objective value for OSY. Since OSY is relatively easy to optimize, the algorithms quickly reach the global minimum. PSO

with repair converges to the global optimum at the 32nd generation. PSO and IGWO with static penalty take a longer time, 80 and 200 generations, respectively. However, the median of GSA and MVO fails to reach the global optimum within 1000 generations. Median objective values can be found in Table 5.3. Static penalty shows a good ability to handle constraints, which leads to a competitive performance of PSO, IGWO, and MVO. GSA does not perform so well on this problem.

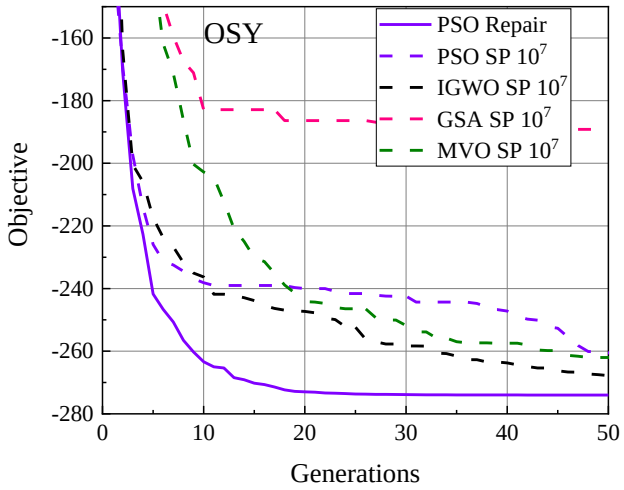


Figure 5.5. The lowest value of the first objective of OSY throughout the optimization.

In Figure 5.6, we can observe a similar trend as in Figure 5.5, where PSO with the repair achieves the best result. The optimum has already been obtained in [83], which is shown as a horizontal line in Figure 5.6. Only PSO with repair fully converges to the global optimum within 50 generations. PSO and IGWO with static penalty require about 560 and 980 generations to reach the same point. Similar to Figure 5.5, median GSA and MVO with static penalty cannot find the global optimum solution in 1000 generations. Further, PSO with static penalty shows the second-best performance. IGWO and MVO are relatively worse, while GSA fails to improve the solutions after the 6th generation.

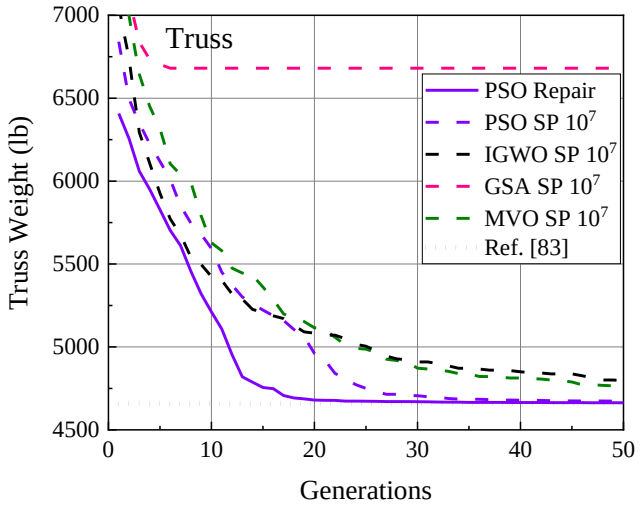


Figure 5.6. Minimum truss weight found by different algorithms.

Figure 5.7 shows the history of the minimum weight design for the tanker. The starting point of each curve indicates when the algorithm finds feasible solutions. Due to high complexity of the tanker problem, solutions randomly generated at the initial stage are all infeasible. Algorithms need to properly handle constraints to reach feasible design space. PSO with static penalty finds the feasible space at generation 22, while it takes 102 generations for GSA. IGWO and MVO take relatively longer to find their first feasible solution, at generation 337 and 384, respectively. Even though IGWO is slow to find feasible space at the initial stage, it converges fast and finds similar result as PSO with the repair at the end, see Table 5.3.

In contrast, PSO and GSA with static penalty show a slow convergence. PSO with repair outperforms other algorithms, and converges to the global optimum at generation 107, which is nine times faster than IGWO with static penalty which obtains a similar result at the end. The excellent convergence of PSO with the repair indicates that this approach could be potentially applied to the early design of engineering problems.

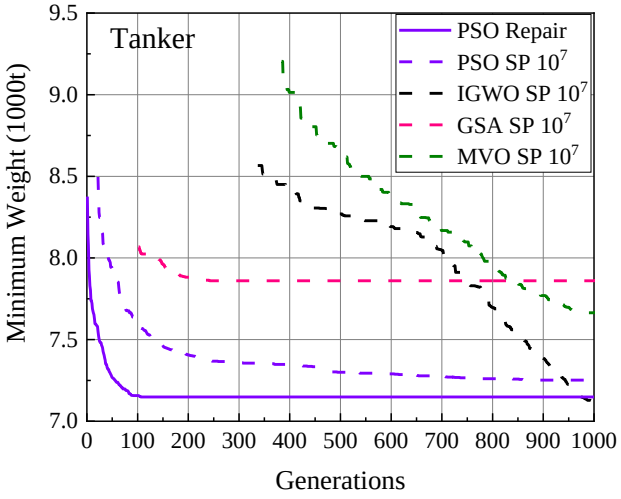


Figure 5.7. Minimum tanker weight found by different algorithms.

Table 5.3. Statistics of the algorithms' performance at the end of optimization for single-objective problems.

		Median	Average	St.Dev.	Best
PSO	PSO Repair	-274.0	-274.0	0.00026	-274.0
	PSO SP 10^7	-262.2	-260.6	7.693	-272.3
	IGWO SP 10^7	-267.8	-267.1	2.546	-271.2
	GSA SP 10^7	-189.2	-184.5	26.34	-228.9
	MVO SP 10^7	-262.0	-261.6	5.240	-270.2
Truss	PSO Repair	4657	4657	1.777	4656
	PSO SP 10^7	4695	4702	20.48	4676
	IGWO SP 10^7	4797	4798	24.50	4751
	GSA SP 10^7	6681	6644	258.0	5692
	MVO SP 10^7	4765	4771	30.56	4723
Tanker	PSO Repair	7148	7153	68.71	6984
	PSO SP 10^7	7252	7254	106.1	7020
	IGWO SP 10^7	7120	7125	51.34	7024
	GSA SP 10^7	7860	7960	134.9	7696
	MVO SP 10^7	7664	7664	156.0	7264

Analysis of the repair-based CHT in PSO

Figure 5.8 shows the median performance of the repair operator during optimization for the tanker problem. Only the first 300 generations are shown since PSO with repair is fully converged after that. At the beginning of the optimization, the algorithm generates many infeasible solutions to explore the design space. Later the algorithm focuses on the exploitation where more prominent infeasible solutions are selected as repair candidates (see Section 3.3 dynamic selection of repair candidate). This could be observed from the red dots, where up to 60% of the solutions are repaired at generation 80. Most of the solutions are successfully repaired, while the unsuccessful could be explained with inaccuracies in the automated mapping. Afterwards, the number of generated infeasible solutions and repaired solutions is decreased. This indicates that the algorithm has converged to the global optima.

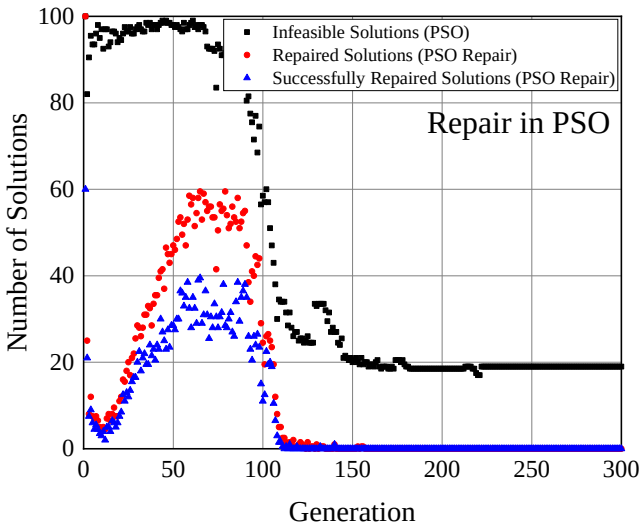


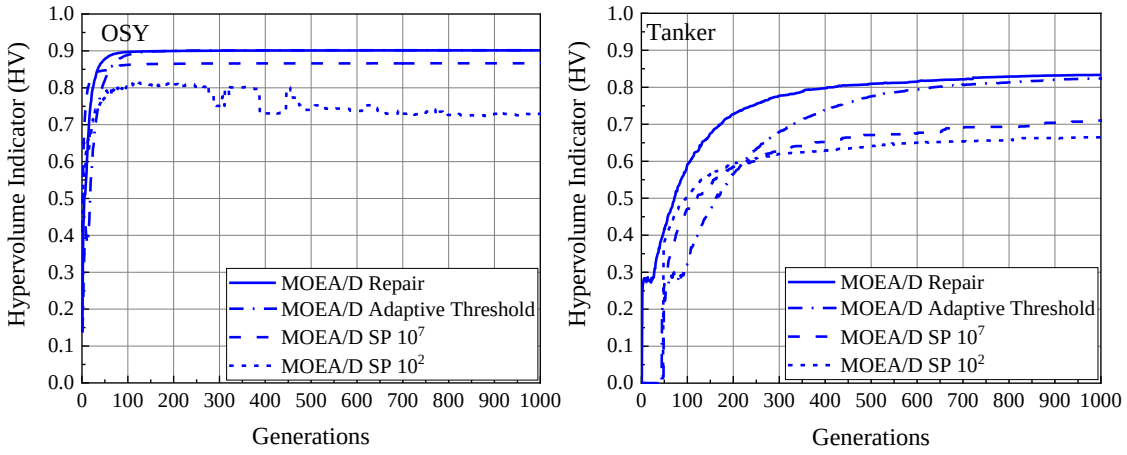
Figure 5.8. Performance of the proposed repair CHT in OSY for tanker optimization.

5.4 Optimization comparisons for multi-objective problems

In this subsection, we discuss the influence of repair-based CHT on multi-objective optimization algorithms, in comparison to other CHTs. Figure 5.9 presents median values of the performance indicators throughout the optimization for MOEA/D with different constraint

handling approaches. We can see that constraint handling approach can significantly alter the performance of the optimization algorithm. For the OSY problem, essentially the same performance is achieved with repair and adaptive threshold, having the same final HV value and almost the same IGD, see Table 5.4. This is significantly better than with static penalty. Here, we show MOEA/D with penalty factors 10^2 and 10^7 for comparison. On the tanker problem, repair outperforms adaptive threshold approach, not just in the end but also at every generation, which is important for practical purposes when optimization needs to be interrupted prematurely due to e.g., lack of time or resources.

Similar to PSO with repair, the novel repair-based CHT helps MOEA/D find feasible solutions at the initial optimization stage (2nd generation). It is worth mentioning that the first generation is entirely infeasible with solutions violating 25 to 35 constraints for the tanker problem, which is the reason that other approaches take about 50 generations to find the feasible space, thus having initial HV values of zero. This could be observed in all approaches from Figure 5.9 to Figure 5.11 except for the algorithms with the repair technique. 30 runs that were made with each approach start with the same 30 randomly generated populations to eliminate the influence of algorithms' starting position. Table 5.4 gives median, average, standard deviation, and the best values of performance indicators at the end of optimization.



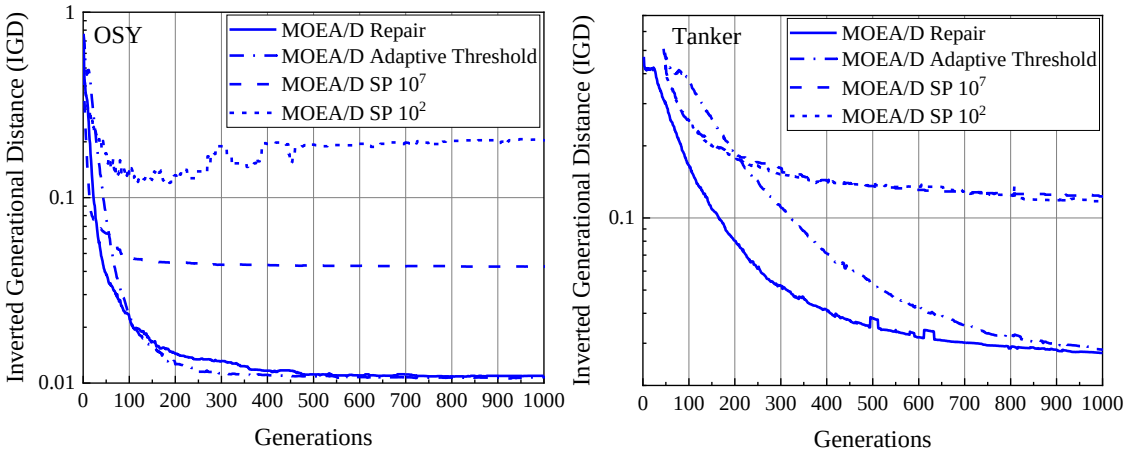


Figure 5.9. Median performance indicators during optimization for MOEA/D with different constraint handling approaches for OSY and tanker problem.

Table 5.4. Statistical values of the performance indicators at the end of optimization.

		HV				IGD			
		Med.	Avg.	St.Dev.	Best	Med.	Avg.	St.Dev.	Best
OSY	MOEA/D Repair	0.9020	0.9003	0.0064	0.9020	0.0109	0.0119	0.0046	0.0094
	MOEA/D Ad. Th.	0.9018	0.9003	0.0127	0.9021	0.0108	0.0128	0.0171	0.0094
	MOEA/D SP 10^7	0.8665	0.8349	0.1041	0.9017	0.0424	0.0684	0.0745	0.0116
	MOEA/D SP 10^2	0.7294	0.7587	0.0550	0.8640	0.2038	0.1875	0.0559	0.0621
	NSGA-II Repair	0.8662	0.8336	0.1137	0.9016	0.0386	0.0694	0.0860	0.0063
	NSGA-II Ad. Th.	0.8629	0.8131	0.1166	0.8994	0.0462	0.1009	0.0995	0.0078
	NSGA-II SP 10^7	0.8635	0.8079	0.1241	0.9004	0.0439	0.0990	0.1003	0.0068
	NSGA-II Original	0.8645	0.8238	0.1128	0.9003	0.0457	0.0927	0.0925	0.0074
	MOMVO SP 10^7	0.8886	0.8518	0.0900	0.8963	0.0244	0.0634	0.0798	0.0175
	MOGWO SP 10^7	0.6972	0.6820	0.1086	0.8113	0.0183	0.0195	0.0041	0.0155
Tanker	MOEA/D Repair	0.8238	0.8218	0.0121	0.8378	0.0282	0.0288	0.0052	0.0193
	MOEA/D Ad. Th.	0.8145	0.8105	0.0234	0.8505	0.0363	0.0380	0.0126	0.0143
	MOEA/D SP 10^7	0.7100	0.6974	0.0378	0.7600	0.1230	0.1207	0.0176	0.0930
	MOEA/D SP 10^2	0.6648	0.6661	0.0252	0.7148	0.1174	0.1180	0.0241	0.0688
	NSGA-II Repair	0.8376	0.8366	0.0115	0.8676	0.0234	0.0229	0.0062	0.0060
	NSGA-II Ad. Th.	0.7532	0.7246	0.0511	0.7864	0.0672	0.0816	0.0264	0.0527
	NSGA-II SP 10^7	0.7113	0.7147	0.0429	0.7948	0.0899	0.0884	0.0205	0.0496
	NSGA-II Original	0.6908	0.6953	0.0491	0.7981	0.0974	0.0961	0.0266	0.0449
	MOMVO SP 10^7	0.5410	0.5381	0.0325	0.5980	0.2125	0.2173	0.0238	0.1800
	MOGWO SP 10^7	0.2745	0.2757	0.0108	0.2980	0.3870	0.3861	0.0209	0.3440

Similar to MOEA/D, different constraint handling approaches are embedded into NSGA-II. Together with its original version, the performance of those algorithms for both multi-objective test problems is shown in Figure 5.10. Since static penalty with low penalty factor shows a poor performance as mentioned above, only the high penalty factor is used. We can see that the repair approach significantly improves the performance of NSGA-II. It should be noted that the original NSGA-II uses constrained non-domination for CHT, however, MOEA/D is proposed for unconstrained problems, thus the original MOEA/D is not considered in this research since all test problems are constrained. We can see from Figure 5.10 that static penalty improves the performance of NSGA-II, while further improvement is achieved with adaptive threshold. However, the most significant improvement of NSGA-II is achieved through the repair technique.

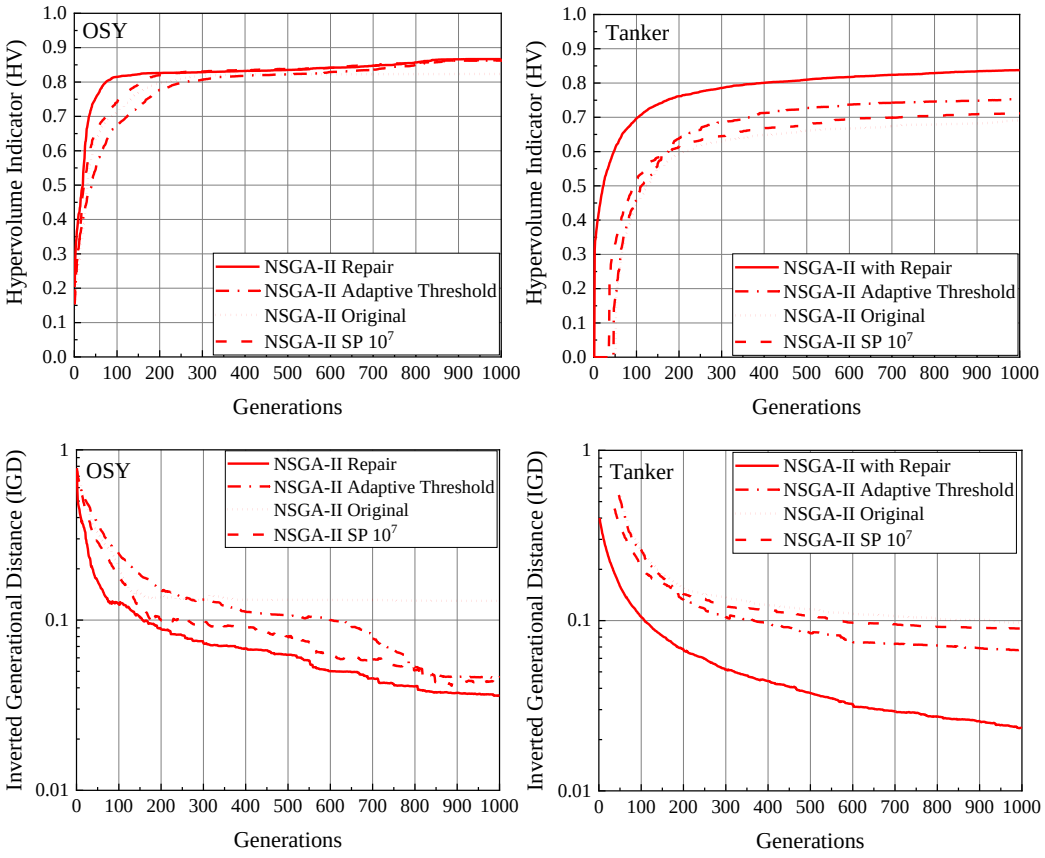


Figure 5.10. Median performance indicators during optimization for NSGA-II with different constraint handling approaches for OSY and tanker problem.

Since the field of metaheuristic optimization is constantly evolving, we need to compare the performance of MOEA/D and NSGA-II with recent swarm algorithms. Figure 5.11 brings to perspective MOMVO and MOGWO. Static penalty is used on all four algorithms. Performance with the repair-based constraint handling in MOEA/D and NSGA-II is shown for comparison. We can see that on OSY problem MOMVO with static penalty is initially making rather slow progress, but in the end, it outperforms NSGA-II with repair. However, MOEA/D with repair still achieves better results. MOGWO initially progresses fast but does not yield desirable performance in the end. MOMVO and MOGWO did not perform that well on the tanker problem. Final HV and IGD values are given in Table 5.4. According to the *no free lunch theorem* [2], an optimization algorithm is not suited for all types of problems. We can see here that MOEA/D and NSGA-II work better on tanker problem than those recent swarm algorithms, despite them being superior on typical optimization benchmark problems in literature.

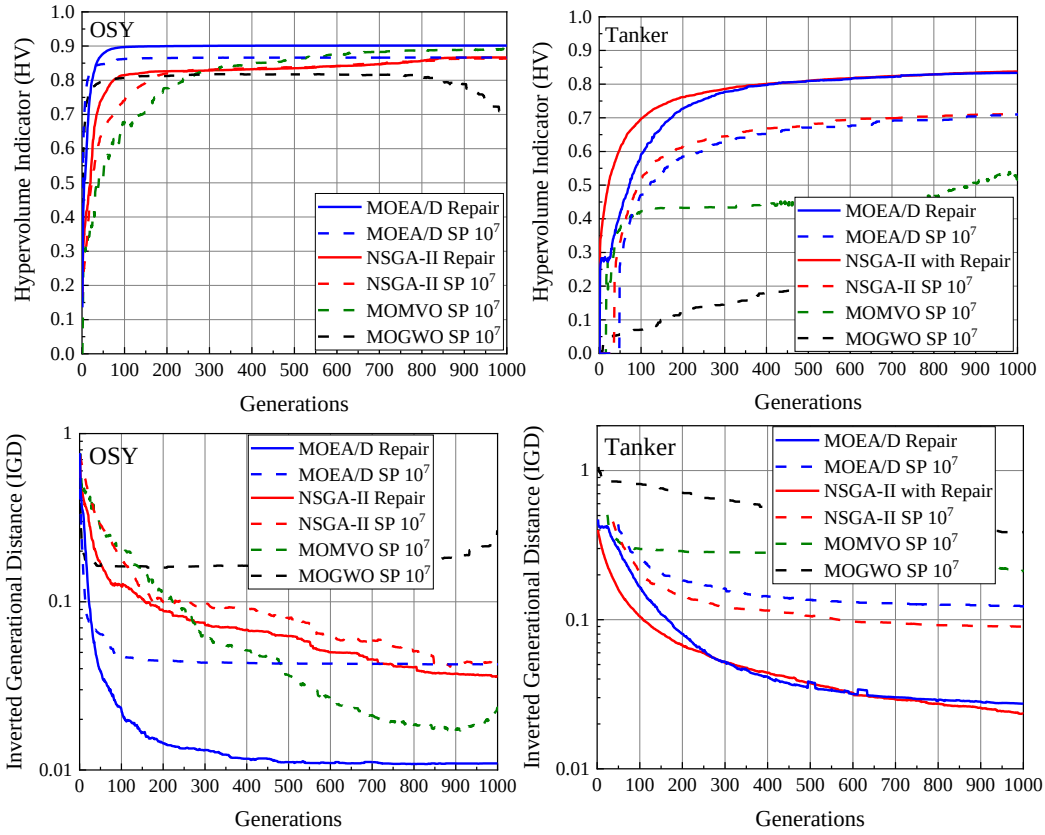


Figure 5.11. Performance indicators for all considered multi-objective optimization algorithms having repair or static penalty as constraint handling approach.

Analysis of the repair-based CHT in MOEA/D and NSGA-II

Similar to Section 5.3, we track the progress of the repair when dealing with the tanker problem for both MOEA/D and NSGA-II. As shown in Figure 5.12, in MOEA/D the median share of infeasible solutions is large. Thus, many infeasible solutions are repaired, about 35% of the whole population. However, the percentage of the successfully repaired solutions decreases significantly. A lower number of solutions still dominates the existing feasible solutions after being repaired.

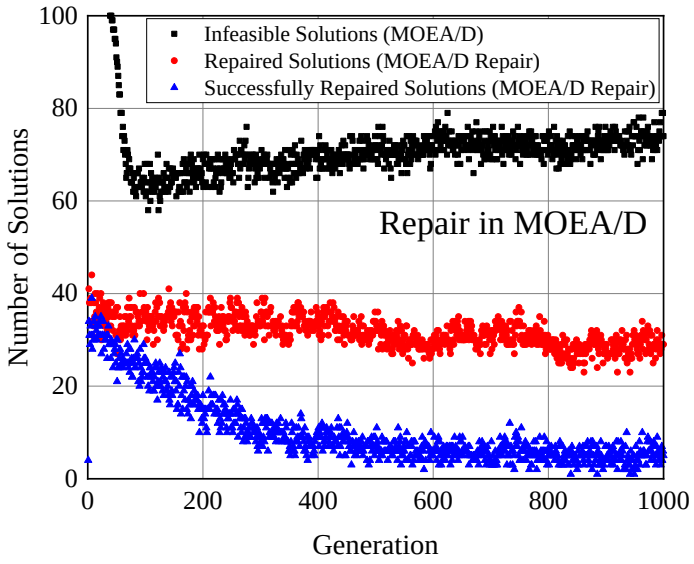


Figure 5.12. Performance of the proposed repair CHT in MOEA/D for tanker optimization.

In contrast, the repair in NSGA-II shows a different trend (Figure 5.13). Because of the genetic operators and non-domination sorting, the algorithm generates sufficient number of prominent infeasible solutions at each generation. The number of repaired solutions is even gradually increasing as the optimization progresses. Different from the repair processes in PSO and MOEA/D, the repair operator shows a steady capacity of providing new non-dominated solutions. This explains why the NSGA-II with repair shows a good performance in tanker optimization.

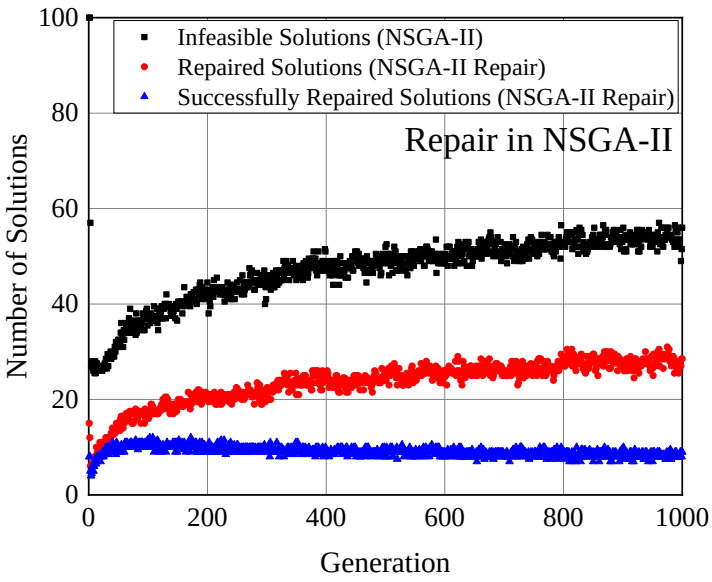


Figure 5.13. Performance of the proposed repair CHT in NSGA-II for tanker optimization.

Non-dominated fronts and optimized tanker with minimum weight

Considered performance indicators quantify the spread and distribution of non-dominated fronts. Figure 5.14 shows the actual non-dominated fronts at the end of optimization. Out of 100 runs for OSY and 30 runs for the tanker problem, the run with the best (lowest) IGD value is presented for each algorithm. We can see that NSGA-II and MOEA/D find the PF of the OSY case with fairly even distribution, except for the bottom right corner which is somewhat sparse. MOMVO and MOGWO clearly lack diversity which is the reason for their somewhat worse performance indices. The differences are larger for the tanker problem. In this case, MOMVO and MOGWO are significantly worse. NSGA-II and MOEA/D with repair find designs with low mass that are very difficult to reach because the design space is highly constrained in that region. The lowest-weight solutions found by both algorithms are 7191 tons and 7132 tons, respectively. Considered constraints prevent failures which generally occur for structural members with low thickness and thus low weight. Therefore, it is comparably easier to find designs with high adequacy than low weight.

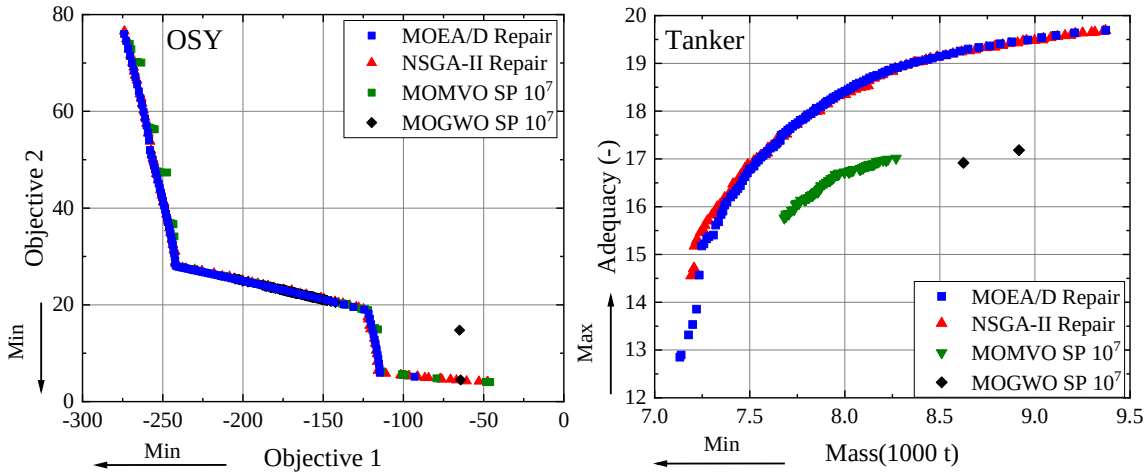


Figure 5.14. Final non-dominated fronts of the best performing runs for both multi-objective test problems.

Optimized tanker cross-section structures

Figure 5.15 (b) shows scantlings of the two designs on the extreme sides of the Pareto front: one with minimum mass and one with maximum adequacy. Structure was not standardized before presented here nor was it given any corrosion addition. Designs obtained in this research are the result of considered constraints and objectives and as such are simplified. This can explain some differences between neighbouring strakes, which could be improved by considering production constraints. However, the focus in this research is on conceptual design where production constraints might be omitted. Structural elements in double bottom, side shell and deck structure from the minimum mass design follow vertically the beam distribution of weight, in order to satisfy the area moment. Side has the lowest plate thicknesses and stiffener sizes, while the bottom elements are additionally increased to resist the water pressure. Inner bottom elements in the cargo tanks are larger than in the bottom because of the increased cargo density, and the same is valid for the inner side. For the outer cargo tank, reduction in scantlings can be seen in the longitudinal bulkhead and inner side when going upwards in the direction of decreasing cargo pressure. The same happens in the central tank but with generally

higher plate thicknesses and stiffener sizes due to higher density. Comparing to this design, structure with the maximum adequacy differs in the deck and nearby strakes, and double bottom. As can be expected, plates are thicker and stiffeners are larger in the deck, but also in the strakes immediately below since they contribute to stress reduction in the deck. Likewise, double bottom is reinforced to decrease global stress component. Other minor differences between the two design alternatives (for example close to the neutral axis) can be explained by the randomness of the algorithm's working principle.

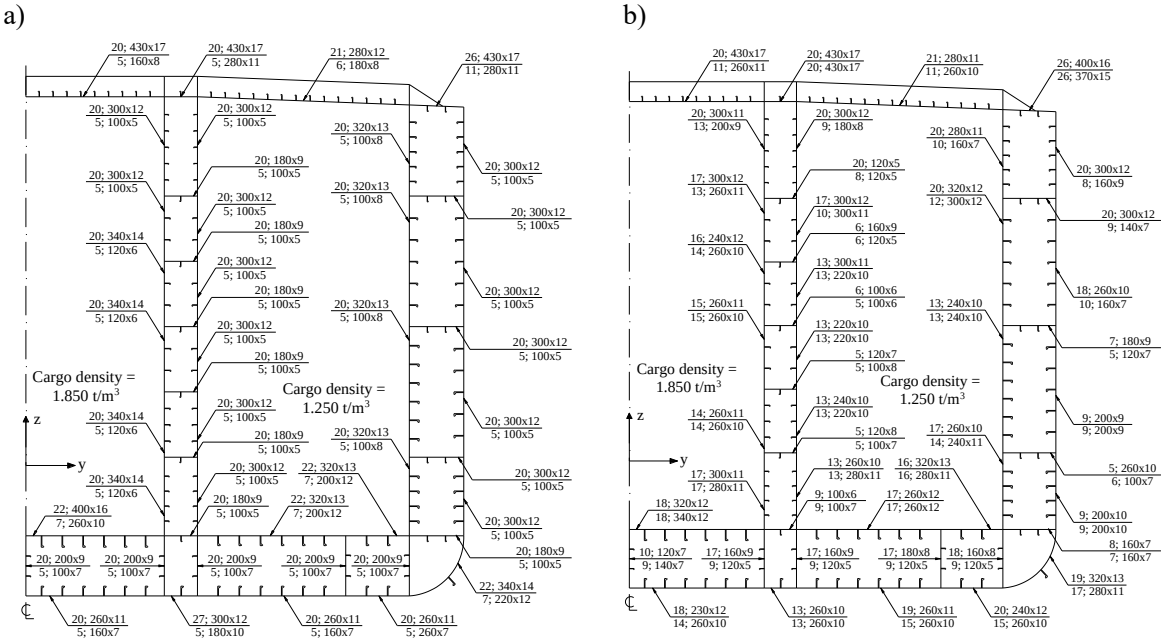


Figure 5.15. a) Lower and upper bounds of each strake; b) scantlings of minimum mass and maximum adequacy design.

6 Conclusion

6.1 Overall conclusions

This thesis presents an automated repair-based technique to handle complex constraints with optimization algorithms. This is achieved by automatically identifying and ranking the most significant variables that influence constraints in optimization, which is useful when constraint functions are not given in closed-form. Deep and shallow artificial neural networks (ANN) are used for this purpose. This information replaces a heuristic typically provided by designer for constraint handling with repair algorithm. We have modified the recent adaptive repair algorithm and tailored it into both single- and multi-objective algorithms, PSO, MOEA/D and NSGA-II, due to their wide adoption in the optimization field. Modified algorithms have been compared with few other population-based algorithms. GSA, IGWO and MVO are used for single-objective algorithms' performance comparison, while MOMVO and MOGWO are applied for multi-objective cases. Two constraint handling techniques (CHT) are embedded into those algorithms for evaluation: static penalty and adaptive threshold. Both mathematical and engineering problems are studied in this thesis to present the effectiveness of this approach, including a common benchmark problem from literature, truss weight optimization and real-life structural design of a chemical tanker. Proposed repair based CHT significantly improves the original algorithms' performance for all test problems. Several conclusions can be made as follows:

Automatic mapping: ANN can be used to automate the variable-constraint mapping, where the significance of each variable is quantified from 0 to 1 to represent whether the variable is significantly influencing a constraint. The procedure essentially replaces a user-provided knowledge of a problem, which in many cases cannot even be devised by human. The mapping is embedded into the novel repair based CHT and validated against few other CHTs.

Generality: The proposed approach has been tailored for three optimization algorithms: PSO, NSGA-II and MOEA/D, the first being single-objective and the latter two multi-objective. Only repair candidate and donor selection strategies require modification. This indicates that the approach could be broadly implemented into other algorithms.

Time saving: Following the automatic mapping, the trained ANNs can be used as surrogate model for predicting constraints in optimization. Hyperparameters of ANN have been optimized to show the best performance. The proposed approach can save up to 90% of the total CPU time in tanker optimization. This is very useful to designers who need to get the optimal solution in a relatively short time.

Convergence and diversity: Modified algorithms are assessed using both mathematical and engineering problems. Hypervolume and inverted generational distance are selected as the performance indicators to evaluate multi-objective optimization algorithms. The proposed repair-based CHT can provide a significant improvement for the optimization algorithms:

(i) For an equal number of function evaluations, modified algorithms can reach a better front compared to their original version and other constraint handling techniques.

(ii) Modified multi-objective algorithms can lead to wider spread of the final front than other state-of-the-art optimization algorithms.

6.2 Limitation of the work

The performance of the proposed approach largely depends on the ANN accuracy, which is important for both variable-constraint mapping and surrogate model. For a different problem ANN accuracy might decrease.

The proposed CHT is not suitable for solving optimization problems involving only a few variables. For instance, if a problem contains only one variable, repairing a solution is equivalent to replacing the solution. This could significantly reduce the diversity of the population.

Modified PSO shows a limited capability of exploring the design space as the algorithm converges quickly at the initial optimization stage in tanker optimization. Even though the modified algorithm can rapidly approach the global optimum, premature convergence might be a problem in some cases.

The threshold η is defined as 0.1 in this thesis, which potentially needs to be adjusted for different problems. Change of η will directly influence the automated mapping matrix.

6.3 Future work

Future work could be focused on applying more advanced ML techniques to generate more accurate mapping and surrogate model. The operating principle of the repair-based technique needs to be improved to adapt to more general problems. In addition, the way to embed repair CHT to PSO could be modified, as fixing the issue of premature convergence might further improve the overall performance of the algorithm. Moreover, the definition of the threshold η should be investigated, for example, using an adaptive parameter instead of a fixed value.

References

[1] Deb K. Multi-Objective Optimization Using Evolutionary Algorithms. Chichester, England ; Toronto, ON : John Wiley & Sons; 2001.

[2] Wolpert DH, Macready WG. No Free Lunch Theorems for Optimization 1997;1:67–82.

[3] Lobato FS, Steffen V. Multi-Objective Optimization Problems: Concepts and Self-Adaptive Parameters with Mathematical and Engineering Applications. 2017.

[4] Lewis RM, Torczon V, Trosset MW, William C. Direct Search Methods : Then and Now Operated by Universities Space Research Association. Science (80-) 2000;124:191–207.

[5] Hoffmann, Laurence D.; Bradley GL. Calculus for Business, Economics, and the Social and Life Sciences (8th ed.) 2004:575–588.

[6] Lemaréchal C. Cauchy and the Gradient Method. Doc Math 2012;ISMP:251–4.

[7] Nocedal J, Wright SJ. Numerical optimization / {Jorge} {Nocedal}, {Stephen} {J}. {Wright}. 2006.

[8] Boggs PT, Tolle JW. Sequential quadratic programming for large-scale nonlinear optimization. J Comput Appl Math 2000;124:123–37. https://doi.org/10.1016/S0377-0427(00)00429-5.

[9] Arora JS. Numerical Methods for Unconstrained Optimum Design. Introd to Optim Des 2004:277–304. https://doi.org/10.1016/b978-012064155-0/50008-2.

[10] Svanberg K. MMA and GCMMA – two methods for nonlinear optimization 2007;1:1–15.

[11] Del Ser J, Osaba E, Molina D, Yang XS, Salcedo-Sanz S, Camacho D, et al. Bio-inspired computation: Where we stand and what's next. Swarm Evol Comput 2019;48:220–50. https://doi.org/10.1016/j.swevo.2019.04.008.

[12] Aguiar H, Junior O. Evolutionary Global Optimization, Manifolds and Applications. 2016.

[13] Lawrence J. Fogel, Alvin J. Owens MJW. Artificial intelligence through simulated evolution. 1966.

[14] Holland JH. Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology. Control Artif Intell 1975.

[15] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6:182–97. https://doi.org/10.1109/4235.996017.

[16] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 2007;11:712–31. https://doi.org/10.1109/TEVC.2007.892759.

[17] Clerc M. Particle Swarm Optimization. Part Swarm Optim 2010:1942–8. https://doi.org/10.1002/9780470612163.

[18] Mirjalili S, Mirjalili SM, Lewis A. Grey Wolf Optimizer. Adv Eng Softw 2014;69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.

[19] Laudis LL, Shyam S, Jemila C, Suresh V. MOBA: Multi Objective Bat Algorithm for Combinatorial Optimization in VLSI. Procedia Comput Sci 2018;125:840–6. https://doi.org/10.1016/j.procs.2017.12.107.

[20] Yang XS. Firefly algorithm, Lévy flights and global optimization. Res Dev Intell Syst XXVI Inc Appl Innov Intell Syst XVII 2010:209–18. https://doi.org/10.1007/978-1-84882-983-1_15.

[21] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm. J Glob Optim 2007;39:459–71. https://doi.org/10.1007/s10898-007-9149-x.

[22] Mirjalili S, Mirjalili SM, Hatamlou A. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 2016;27:495–513. https://doi.org/10.1007/s00521-015-1870-7.

[23] Mirjalili S. SCA: A Sine Cosine Algorithm for solving optimization problems. Knowledge-Based Syst 2016;96:120–33. https://doi.org/10.1016/j.knosys.2015.12.022.

[24] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: A Gravitational Search Algorithm. Inf Sci (Ny) 2009;179:2232–48. https://doi.org/10.1016/j.ins.2009.03.004.

[25] Coello Coello CA. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput Methods Appl Mech Eng 2002;191:1245–87. https://doi.org/10.1016/S0045-7825(01)00323-1.

[26] Courant R. Variational methods for the solution of elliptic equations 1974:157–202. https://doi.org/10.1090/mmono/042/04.

[27] Joines JA, Houck CR. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. IEEE Conf Evol Comput - Proc 1994:579–84. https://doi.org/10.1109/icec.1994.349995.

[28] Homaifar A, Qi CX, Lai SH. Constrained optimization via genetic algorithms. Simulation 1994;62:242–54. https://doi.org/10.1177/003754979406200405.

[29] Koziel S, Michalewicz Z. A decoder-based evolutionary algorithm for constrained parameter optimization problems. Lect Notes Comput Sci (Including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 1998;1498 LNCS:231–40. https://doi.org/10.1007/bfb0056866.

[30] Mezura-Montes E, Coello Coello CA. Constraint-handling in nature-inspired numerical optimization: Past, present and future. Swarm Evol Comput 2011;1:173–94. https://doi.org/10.1016/j.swevo.2011.10.001.

[31] Runarsson TP, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput 2000;4:284–94. https://doi.org/10.1109/4235.873238.

[32] Qu BY, Suganthan PN. Constrained multi-objective optimization algorithm with an ensemble of constraint handling methods. Eng Optim 2011;43:403–16. https://doi.org/10.1080/0305215X.2010.493937.

[33] Poon PW, Carter JN. Genetic algorithm crossover operators for ordering applications. Comput Oper Res 1995;22:135–47. https://doi.org/10.1016/0305-0548(93)E0024-N.

[34] Koch P, Konen W, Foussette C, Krause P, Bäck T. A New Repair Method For Constrained Optimization 2015:273–80.

[35] Chootinan P, Chen A. Constraint handling in genetic algorithms using a gradient-based repair method. Comput Oper Res 2006;33:2263–81. https://doi.org/10.1016/j.cor.2005.02.002.

[36] Todoroki A, Haftka RT. Stacking sequence optimization by a genetic algorithm with a new recessive gene like repair strategy 1998;8368:277–85.

[37] Samanipour F, Jelovica J. Adaptive repair method for constraint handling in multi-objective genetic algorithm based on relationship between constraints and variables. Appl Soft Comput J 2020;90:106143. https://doi.org/10.1016/j.asoc.2020.106143.

[38] Mirjalili S, Lewis A. The Whale Optimization Algorithm. Adv Eng Softw 2016;95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008.

[39] Mirjalili S, Jangir P, Saremi S. Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems. Appl Intell 2017;46:79–95. https://doi.org/10.1007/s10489-016-0825-8.

[40] Zavala G, Nebro AJ, Luna F, Coello Coello CA. Structural design using multi-objective metaheuristics. Comparative study and application to a real-world problem. Struct Multidiscip Optim 2016;53:545–66. https://doi.org/10.1007/s00158-015-1291-3.

[41] Garcia R de P, de Lima BSLP, Lemonge AC de C, Jacob BP. A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. Comput Struct 2017;187:77–87. https://doi.org/10.1016/j.compstruc.2017.03.023.

[42] Klanac A, Jelovica J. A concept of omni-optimization for ship structural design. Adv Mar Struct - Proc MARSTRUCT 2007, 1st Int Conf Mar Struct 2007:473–81.

[43] Kumar R, Parida PP, Gupta M. Topological design of communication networks using multiobjective genetic optimization. Proc 2002 Congr Evol Comput CEC 2002 2002;1:425–30. https://doi.org/10.1109/CEC.2002.1006272.

[44] Brownlee AEI, Wright JA. Constrained, mixed-integer and multi-objective optimisation of building designs by NSGA-II with fitness approximation. Appl Soft Comput J 2015;33:114–26. https://doi.org/10.1016/j.asoc.2015.04.010.

[45] Liu Y, Collette M. Improving surrogate-assisted variable fidelity multi-objective

optimization using a clustering algorithm. Appl Soft Comput J 2014;24:482–93. https://doi.org/10.1016/j.asoc.2014.07.022.

[46] Myers RH, Montgomery DC, Geoffrey Vining G, Borror CM, Kowalski SM. Response Surface Methodology: A Retrospective and Literature Survey. J Qual Technol 2004;36:53–78. https://doi.org/10.1080/00224065.2004.11980252.

[47] Smith M. Neural Networks for Statistical Modeling. 1st editio. Van Nostrand Reinhold; 1993.

[48] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521:436–44. https://doi.org/10.1038/nature14539.

[49] Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2012:3642–9. https://doi.org/10.1109/CVPR.2012.6248110.

[50] Hu J, Niu H, Carrasco J, Lennox B, Arvin F. Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning. IEEE Trans Veh Technol 2020;69:14413–23. https://doi.org/10.1109/TVT.2020.3034800.

[51] Graves A, Liwicki M, Fernández S, Bertolami R, Bunke H, Schmidhuber J. A novel connectionist system for unconstrained handwriting recognition. IEEE Trans Pattern Anal Mach Intell 2009;31:855–68. https://doi.org/10.1109/TPAMI.2008.137.

[52] Senior A. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling Has n.d.

[53] Zhang R, Chen Z, Chen S, Zheng J, Büyüköztürk O, Sun H. Deep long short-term memory networks for nonlinear structural seismic response prediction. Comput Struct 2019;220:55–68. https://doi.org/10.1016/j.compstruc.2019.05.006.

[54] Tan RK, Zhang NL, Ye W. A deep learning–based method for the design of microstructural materials. Struct Multidiscip Optim 2020;61:1417–38. https://doi.org/10.1007/s00158-019-02424-2.

[55] Kramer MA. Nonlinear principal component analysis using autoassociative neural networks. AIChE J 1991;37:233–43. https://doi.org/10.1002/aic.690370209.

[56] Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. Genet Program Evolvable Mach 2018;19:305–7. https://doi.org/10.1007/s10710-017-9314-z.

[57] Morales-Forero A, Bassetto S. Case Study: A Semi-Supervised Methodology for Anomaly Detection and Diagnosis. IEEE Int Conf Ind Eng Eng Manag 2019:1031–7. https://doi.org/10.1109/IEEM44572.2019.8978509.

[58] Theis L, Shi W, Cunningham A, Huszár F. Lossy image compression with compressive autoencoders. 5th Int Conf Learn Represent ICLR 2017 - Conf Track Proc 2017:1–19. https://doi.org/10.17863/CAM.51995.

[59] Wang S, Zhang X, Ye P, Du M. Deep Belief Networks based toponym recognition for Chinese text. ISPRS Int J Geo-Information 2018;7. https://doi.org/10.3390/ijgi7060217.

[60] Kallioras NA, Kazakis G, Lagaros ND. Accelerated topology optimization by means of deep learning. Struct Multidiscip Optim 2020;62:1185–212. https://doi.org/10.1007/s00158-020-02545-z.

[61] Wu RT, Liu TW, Jahanshahi MR, Semperlotti F. Design of one-dimensional acoustic metamaterials using machine learning and cell concatenation. Struct Multidiscip Optim 2021;63:2399–423. https://doi.org/10.1007/s00158-020-02819-6.

[62] Saha S. Hierarchical Deep Learning Neural Network (HiDeNN): An artificial intelligence (AI) framework for computational science and engineering. Comput Methods Appl Mech Eng 2021;373:113452. https://doi.org/10.1016/j.cma.2020.113452.

[63] Liu PX, Zuo MJ, Meng MQH. Using neural network function approximation for optimal design of continuous-state parallel-series systems. Comput Oper Res 2003;30:339–52. https://doi.org/10.1016/S0305-0548(01)00100-9.

[64] Ait Gougam L, Tribeche M, Mekideche-Chafa F. A systematic investigation of a neural network for function approximation. Neural Networks 2008;21:1311–7. https://doi.org/10.1016/j.neunet.2008.06.015.

[65] Li FJ. Constructive function approximation by neural networks with optimized activation functions and fixed weights. Neural Comput Appl 2019;31:4613–28. https://doi.org/10.1007/s00521-018-3573-3.

[66] Hornik K, Stinchcombe M, White H. Multilayer feedforward networks are universal

approximators. Neural Networks 1989;2:359–66. https://doi.org/10.1016/0893-6080(89)90020-8.

[67] Kennedy J, Eberhart R. Particle swarm optimization. Proc. ICNN'95 - Int. Conf. Neural Networks, vol. 4, 1995, p. 1942–8 vol.4. https://doi.org/10.1109/ICNN.1995.488968.

[68] Pulido GT, Coello Coello CA. A constraint-handling mechanism for particle swarm optimization. Proc 2004 Congr Evol Comput CEC2004 2004;2:1396–403. https://doi.org/10.1109/cec.2004.1331060.

[69] Nguyen AT, Reiter S, Rigo P. A review on simulation-based optimization methods applied to building performance analysis. Appl Energy 2014;113:1043–58. https://doi.org/10.1016/j.apenergy.2013.08.061.

[70] Nadimi-shahraki MH, Taghian S, Mirjalili S. An improved grey wolf optimizer for solving engineering problems. Expert Syst Appl 2021;166:113917. https://doi.org/10.1016/j.eswa.2020.113917.

[71] Mirjalili S, Saremi S, Mirjalili SM, Coelho LDS. Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. Expert Syst Appl 2016;47:106–19. https://doi.org/10.1016/j.eswa.2015.10.039.

[72] Mirjalili S, Jangir P, Mirjalili SZ, Saremi S, Trivedi IN. Optimization of problems with multiple objectives using the multi-verse optimization algorithm. Knowledge-Based Syst 2017;134:50–71. https://doi.org/10.1016/j.knosys.2017.07.018.

[73] Problem TS, Algorithm E, Algorithms INO, Networking IB, Algorithms INO. Cross-over Mutation Advances in Geophysics Analysis of Algorithms 2015.

[74] Purshouse RC, Fleming PJ, Fonseca CM, Greco S, Eds JS, Hutchison D. LNCS 7811Evolutionary Multi-Criterion Optimization. 2013.

[75] Hussain SF, Iqbal S. Genetic ACCGA: Co-similarity based Co-clustering using genetic algorithm. Appl Soft Comput J 2018;72:30–42. https://doi.org/10.1016/j.asoc.2018.07.045.

[76] Asafuddoula M, Ray T, Sarker R, Alam K. An adaptive constraint handling approach embedded MOEA/D. 2012 IEEE Congr Evol Comput CEC 2012 2012:1–8. https://doi.org/10.1109/CEC.2012.6252868.

[77] Asafuddoula M, Ray T, Sarker R. A decomposition-based evolutionary algorithm for many objective optimization. IEEE Trans Evol Comput 2015;19:445–60. https://doi.org/10.1109/TEVC.2014.2339823.

[78] Hobbie JG, Gandomi AH, Rahimi I. A Comparison of Constraint Handling Techniques on NSGA-II. Arch Comput Methods Eng 2021;1. https://doi.org/10.1007/s11831-020-09525-y.

[79] Thiele L, Zitzler E. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans Evol Comput 1999;3:257–71.

[80] Coello Coello CA, Reyes Sierra M. A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm. In: Monroy R, Arroyo-Figueroa G, Sucar LE, Sossa H, editors. MICAI 2004 Adv. Artif. Intell., Berlin, Heidelberg: Springer Berlin Heidelberg; 2004, p. 688–97.

[81] Huang G Bin, Saratchandran P, Sundararajan N. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. IEEE Trans Neural Networks 2005;16:57–67. https://doi.org/10.1109/TNN.2004.836241.

[82] Yokota T, Taguchi T, Gen M. A Solution Method for Optimal Weight Design Problem of 10 Bar Truss Using Genetic Algorithms Department of Industrial and Systems Engineering 1998;35:367–72.

[83] Cagnina LC, Esquivel SC, Coello CAC. Solving constrained optimization problems with a hybrid particle swarm optimization algorithm. Eng Optim 2011;43:843–66. https://doi.org/10.1080/0305215X.2010.522707.

[84] Camp C V, Farshchin M. Design of space trusses using modified teaching – learning based optimization. Eng Struct 2014;62–63:87–97. https://doi.org/10.1016/j.engstruct.2014.01.020.

[85] Jelovica J, Klanac A. Multi-objective optimization of ship structures: Using guided search vs. conventional concurrent optimization. Anal Des Mar Struct Incl CD-ROM 2009:447–56. https://doi.org/10.1201/9780203874981-61.

[86] Ship structural design: A rationally based, computer aided, optimization approach: O. F. Hughes John Wiley, Chichester, November 1983, 600 pp., $86.45/£61.75, ISBN 0471 032417. Appl Ocean Res 1984;6:177. https://doi.org/https://doi.org/10.1016/0141-

1187(84)90014-2.

[87] British Steel. Bulb flats n.d. https://britishsteel.co.uk/media/40438/bulb-flats-brochure.pdf (accessed March 6, 2021). n.d.

[88] Hughes OF, Ghosh B, Chen Y. Improved prediction of simultaneous local and overall buckling of stiffened panels. Thin-Walled Struct 2004;42:827–56. https://doi.org/https://doi.org/10.1016/j.tws.2004.01.003.

[89] Naar H, Varsta P, Kujala P. A theory of coupled beams for strength assessment of passenger ships. Mar Struct 2004;17:590–611. https://doi.org/10.1016/j.marstruc.2005.03.004.

[90] Hughes OF, Mistree F, Zanic V. Practical Method for the Rational Design of Ship Structures. J Sh Res 1980;24:101–13.

[91] Zanic V, Andric J, Prebeg P. Design synthesis of complex ship structures 2013;5302. https://doi.org/10.1080/17445302.2013.783455.

Appendices

Appendix A: Validation of the surrogate model

Figure A.1 shows the performance of NSGA-II with repair for the tanker optimization. The red curve represents the algorithm using the surrogate model throughout the whole optimization process. To validate the surrogate model in optimization, we switched the algorithm from the surrogate model back to the original structural code at the 800th generation, whose performance metrics are shown as the black curve in the figure. Both cases start from the same initial population. The median value is presented in each case out of 30 runs. Control parameters are the same.

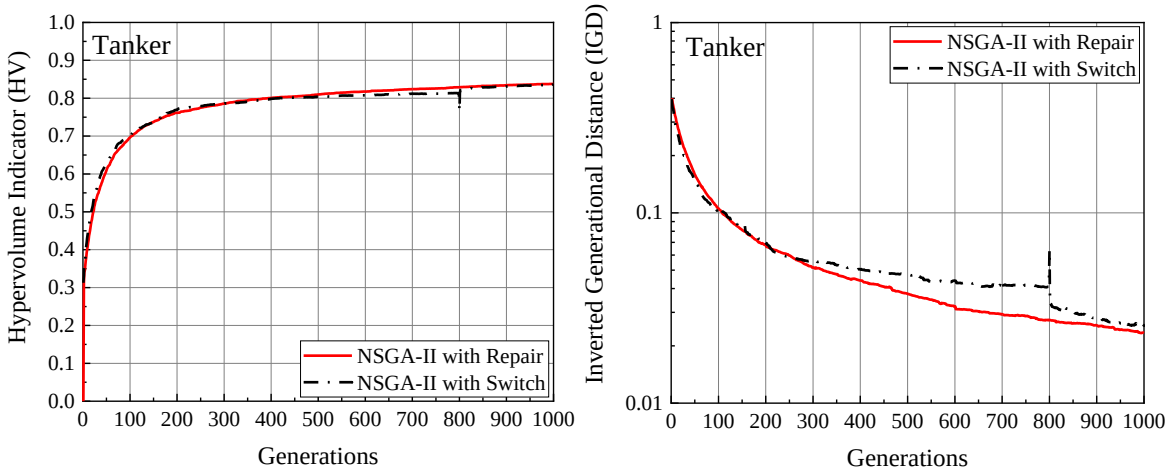


Figure A.1. Performance metrics for NSGA-II with and without the full surrogate model in tanker problem.

Both performance measures noticeably deteriorate when the algorithm switches to the original structural model. ANN bears an error compared to the original structural model used for training, whose error becomes even larger with deeper exploration of the design space. At this stage, both the number of infeasible solutions and non-dominated infeasible solutions increases significantly, see Figure A.2. This leads to a performance reduction for both HV and IGD. Again, the repair operator aids in this situation: both the number of repaired solutions and the

number of successfully repaired solutions are increased significantly in a few generations after the switch. After that, the performance improves and reaches a similar level as the algorithm with original structural model.

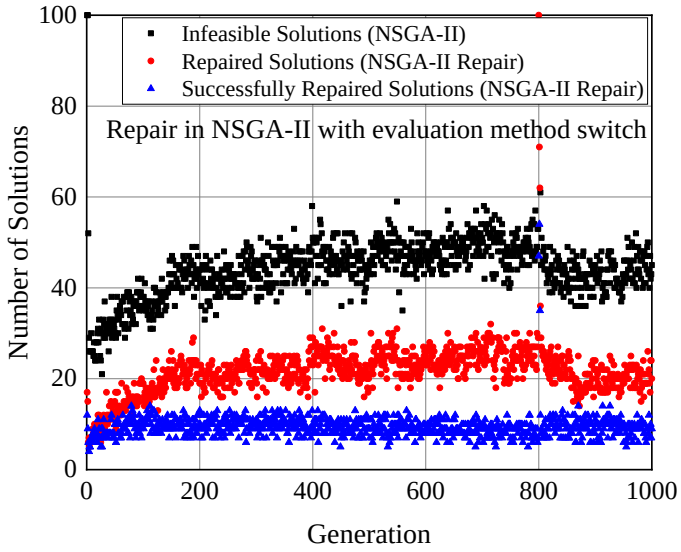


Figure A.2. Performance of the repair operator in NSGA-II including the switch from the surrogate to the original structural model.

Appendix B: Variable bounds and stiffener dimensions

For the tanker problem, the variable bounds and their corresponding stiffener scantlings are shown here. In Table B.1, we can see the lower and upper bounds of plate thinness and stiffener ID per strake. The dimensions of the bulb profiles are given in Table B.2.

Table B.1. Variable bounds for the tanker optimization problem.

Strake #	*Plate thickness (mm)*		Strake #	*Stiffener size ID*	
	x_{min}	x_{max}		x_{min}	x_{max}
1	5	20	1	12	27
2	5	20	2	17	32
3	5	20	3	12	27
4	5	20	4	12	27
5	7	22	5	22	37
6	7	22	6	26	41
7	5	20	7	1	16
8	7	22	8	20	35
9	7	20	9	20	16
10	5	20	10	1	16
11	5	20	11	13	44
12	5	20	12	29	44
13	6	21	13	15	30
14	11	26	14	29	44
15	5	20	15	1	32
16	5	20	16	1	32
17	5	20	17	1	32
18	5	20	18	1	32
19	5	20	19	4	35
20	5	20	20	4	35
21	5	20	21	4	35
22	5	20	22	4	35
23	5	20	23	1	32
24	5	20	24	1	32
25	5	20	25	1	32
26	5	20	26	1	32
27	5	20	27	1	32
28	5	20	28	1	32
29	5	20	29	6	37
30	5	20	30	6	37
31	5	20	31	6	37
32	5	20	32	6	37
33	5	20	33	1	32
34	5	20	34	1	32
35	5	20	35	1	32
36	5	20	36	1	32
37	5	20	37	1	32

38	5	20	38	1	16
39	5	20	39	1	16
40	5	20	40	1	16
41	5	20	41	1	16
42	5	20	42	1	16
43	5	20	43	3	18
44	5	20	44	3	18
45	5	20	45	3	18
46	5	20	46	3	18
47	5	20	47	3	18

Table B.2. Allowable stiffener dimensions.

Identification number (ID)	Thickness (mm)	Height (mm)
1	5	100
2	6	100
3	7	100
4	8	100
5	5	120
6	6	120
7	7	120
8	8	120
9	7	140
10	8	140
11	10	140
12	7	160
13	8	160
14	9	160
15	8	180
16	9	180
17	10	180
18	9	200
19	10	200
20	12	200
21	10	220
22	12	220
23	10	240
24	11	240
25	12	240

26	10	260
27	11	260
28	12	260
29	11	280
30	12	280
31	11	300
32	12	300
33	13	300
34	12	320
35	13	320
36	12	340
37	14	340
38	13	370
39	15	370
40	14	400
41	16	400
42	14	430
43	15	430
44	17	430