Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

b-it Bonn-Aachen International Center for Information Technology

R&D Project

Towards smart coupling in multiphysics simulation

Deepan Chakravarthi Padmanabhan

Submitted to Hochschule Bonn-Rhein-Sieg,
Department of Computer Science
in partial fullfilment of the requirements for the degree
of Master of Science in Autonomous Systems

Supervised by

Prof. Dr. Paul G. Plöger
Ing.-Inf. Pascal Bayrasy
M.Sc. Ahmad Delforouzi

January 2020

I, the undersigned below, declare that this work has not previously been submitted to this or any other university and that it is, unless otherwise stated, entirely my own work.

Date	Deepan Chakravarthi Padmanabhan

Abstract

Multiphysics simulation is the study of the interaction between multiple physical domains. An example of multiphysics simulation is Fluid-Structure Interaction (FSI). The systems integrating multiple physical domains are called coupled systems. To understand and design coupled systems, the simulations are carried out in a coupling tool.

Being an integral part of various academic and industrial applications, the coupling tool is highly parameterized. The parameters govern the accuracy and time taken for the simulation. The process of manually tuning the optimal parameter configuration is a tedious task given the number of parameters, types of parameters, domain expertise of the users, and the substantial time taken to solve a single simulation instance.

This research work focuses on improving the robustness of the coupling tool by automatically configuring the parameters with optimal values. The proposed methodology is based on an Automated Algorithm Configuration (AAC) approach incorporating Sequential Model-based Algorithm Configuration (SMAC), a variant of Bayesian Optimization (BO) followed by a machine learning model to predict the optimal parameter configurations. The approach is evaluated on the Mesh-based parallel Code Coupling Interface (MpCCI) developed by the Fraunhofer Institute for Algorithms and Scientific Computing (SCAI).

A preliminary multiphysics simulation dataset for machine learning and feature extraction package is developed. SMAC provides a 24.92% mean reduction in the runtime of a simulation instance compared to the runtime of default configurations by performing per-instance optimization with a repeatability coefficient of 19.34 seconds. Furthermore, this work contributes a 'Smart coupling tool' for suggesting three optimal parameter configurations given a simulation problem. The parameter configurations suggested by Random Forest (RF) and Gradient Boosting Machine (GBM) outperform the default configurations on unseen critical instances.

Acknowledgements

I thank my supervisors Prof. Dr. Paul G. Plöger, Ing.-Inf. Pascal Bayrasy, and M.Sc. Ahmad Delforouzi, for providing the opportunity to work on this Research and Development project. I extend my warm gratitude towards Mr. Klaus Wolf for recruiting me at Fraunhofer SCAI. I gratefully thank M. Sc. Hamid Arjmandi for being the backbone of this project, from guiding throughout the project to motivating me at tough times. I sincerely thank M.Sc. Santosh Thoduka, M.Sc. Deebul Nair, and M.Sc. Alex Mitrevski for their continuous guidance and valuable support throughout the entire duration of this project.

I thank Arumuga Vinayagam, Anitha Raj Lakshmi, Deepthishre Gunashekar, Iswariya Manivannan, Kishaan, Lokesh, Mihir, Mohandass, Naresh, Pradheep, Pritesh, Raghuvir, Rubanraj, Sathiya Ramesh, Santosh Reddy, Senthilkumar, Swaroop, and Umer for their constant support, constructive criticism and motivation.

Finally, I extend my love to my parents, brother, and friends for their enduring support, undying inspiration, and endless encouragement.

Contents

List of Abbreviations

AAC	Automated Algorithm Configuration
ACP	Algorithm Configuration Problem
API	Application Programming Interface
BBO	Black-Box Optimization
BFGS	Broyden–Fletcher–Goldfarb–Shanno algorithm
BO	Bayesian Optimization
BOBYQA	Bound Optimization BY Quadratic Approximation
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
CV	Cross Validation
DACE	Design and Analysis of Computer Experiments
EA	Evolutionary Algorithm
EGO	Efficient Global Optimization
EI	Expected Improvement
ES	Entropy search
FSI	Fluid-Structure Interaction
GA	Genetic Algorithm
GBM	Gradient Boosting Machine
GGA	Gender-based Genetic Algorithm
GP	Gaussian Process
KNN	K-Nearest Neighbors

MADS	Mesh Adaptive Direct Search
ML	Machine Learning
MpCCI	Mesh-based parallel Code Coupling Interface
MPI	Maximum Probability of Improvement
PCA	Principal Component Analysis
QN	Quasi-Newton
QRF	Quantile Random Forest
RBF	Radial Basis Function
REVAC	Relevance Estimation and Value Calibration
RF	Random Forest
RMSE	Root Mean Squared Error
SKO	Sequential Kriging Optimization
SMAC	Sequential Model-based Algorithm Configuration
SMBO	Sequential Model-Based Optimization
SPO	Sequential Parameter Optimization
SPOT	Sequential Parameter Optimization Toolbox
SVM	Support Vector Machine
t-SNE	t- distributed Stochastic Neighbor Embedding
TB-SPO	Time-Bounded Sequential Parameter Optimization
TPE	Tree Parzen Estimator

List of Figures

List of Tables

1

Introduction

Multiphysics is the coexistence of various physical phenomena in a system or process. The systems with multiple physical phenomena are called coupled systems. The study of coupled systems in a simulated environment is called multiphysics simulations. An example of multiphysics simulation is Fluid-Structure Interaction (FSI). For instance, on blowing air at a particular pressure into a balloon, the air exerts a force on the walls of the balloon and inflates the balloon. The process of studying the interaction between the balloon (solid domain) and air (fluid domain) in a simulated environment is called multiphysics simulation.

In this study, Mesh-based parallel Code Coupling Interface (MpCCI), a coupling tool developed by the Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), is used to perform multiphysics simulations [35]. In MpCCI, the coupled domains are treated as black-boxes. To perform a multiphysics simulation, the user provides the solid and fluid domain models (fixed parameters) to MpCCI. A few model parameters are solid elasticity, fluid viscosity, Poisson's ratio, fluid density, solid-fluid density ratio, and mesh size depending on the material and design under study. The respective solvers solve the simulation models. Then, the user provides the coupling properties (fixed parameters) depending on the study conditions. The parameters are coupling region, coupled variables, and tolerance of the coupling process. In addition, the user configures the coupling parameters (changeable parameters) such as relaxation scheme, coupling scheme, and various constant values associated with the coupling process. The changeable parameters largely determine the simulation accuracy and time. The changeable parameters are considered to be the hyperparameters of the coupling tool.

Algorithms in various real-time applications expose numerous configurable parameters. A parameter configuration is a vector of acceptable value assignments to all the parameters of an algorithm [45]. The parameters are specific to the algorithm and determine the design choices of the user depending on the inputs of the algorithm. The parameters help

the user by providing flexibility between cost and performance for a particular instance. In addition, the parameters determine the performance and robustness of the application. This signifies the importance of identifying the optimal parameter configuration. However, manually estimating the optimal parameters by evaluating the algorithm performance for every possible parameter configuration is expensive in terms of resources and time. The challenges of configuring the parameters drive the research field of Automated Algorithm Configuration (AAC) since 1990 [72] [41].

AAC is a branch of machine learning and optimization theory. It is the method of automatically identifying the optimal values for the parameters of an algorithm. AAC approaches configure the optimal parameters of various applications such as neural networks, robot localization, and constraint satisfaction problems [44] [63] [17].

This study focuses on incorporating AAC approaches to develop a robust MpCCI coupling tool by configuring the changeable coupling parameters with optimal values. In addition, the proposed strategy to configure the coupling tool with optimal parameters is developed into a tool called 'smart coupling'.

1.1 Motivation

Most of the Modeling & Simulation (M&S) studies for exploring designs and understanding a physical phenomenon to eliminate costly prototypes require robust and accelerated simulation tools. However, a single simulation is associated with a substantial computational time. For instance, a simple crash simulation of an automobile extends up to 48 hours [56]. A single simulation instance to study the aerodynamics of aircraft extends up to weeks [29].

Multiphysics co-simulation is challenging due to the stability issues, different mesh types, numerous discretization methods, various scales in time, and space [16]. The process of setting up a single co-simulation instance incorporates numerous coupling tool parameters to be set by the user (approximately 15 parameters in the MpCCI tool) [35]. An inappropriate parameter setting leads to divergence of the solution and the simulation crash after a prolonged run.

In an industrial and research environment, developing efficient designs necessitate the need to perform repeated tests involving various configurations of the coupling tool in a robust environment. Therefore, the considerable computation time of single simulation instance and tedious setting up procedure is challenging in the industrial and research activities.

In addition, the MpCCI tool does not provide any assistance for FSI simulations to meet the demands of the industrial and academic users by providing robust simulation

setup. This potentially leads to users configuring the simulation with inappropriate configurations, causing simulation failure.

MpCCI coupling tool is widely used in various research and development laboratories and industries, namely, The European Organization for Nuclear Research (CERN), Switzerland, Cambridge University, United Kingdom, TU München, Germany, Hochschule Bonn-Rhein-Sieg, Germany, Universität Bonn, Germany, Airbus, Eurocopter, Daimler, Volkswagen, Audi, Eaton, and an extended list is available at [34].

The project aims to improve the robustness of the coupling tool to perform co-simulation studies in a stable and reduced cycle time. The coupling environment is utilized in various fields of research ranging from biomedical to automobile applications. An overview of the use-cases is provided in section 2.1.1.1.

1.2 Problem statement

The coupling tool exposes numerous parameters for the user to configure. The parameter configuration of the coupling tool determines simulation accuracy and time. The existing process of manually fine-tuning the coupling tool parameters is a tedious and time-consuming task because of the following reasons:

1. Numerous configurable parameters in the coupling tool, approximately 20 parameters [35]. This complicates the setting up of a single simulation instance in the coupling tool.

2. Different types of configurable parameters, namely, numerical parameters, categorical parameters, and ordinal parameters. In addition, the parameters exhibit forbidden and conditional dependencies [35].

3. Numerous categories of multiphysics simulation studies. For instance, FSI, magnetostatics, and heat transfer. This increases the complexity of configuring a co-simulation process for different study types.

4. An inappropriate configuration might lead to divergence of the simulation and increases the overhead cost of the simulation.

The research addresses the problem of estimating the optimal parameter configuration of a coupling tool using an automated algorithm configuration approach. The optimal parameter configurations intend to increase the robustness of the simulation in MpCCI by reducing time.

1.3 Challenges and difficulties

The challenges of the project are substantial computational cost associated with single simulation, unavailability of datasets, unavailability of benchmark simulation instances, and unavailability of feature extraction methods.

1. Large computational cost: The models of a simulation instance are divided into discrete meshes. Each mesh incorporates iterative methods to solve equations concerning the properties of the simulation problem under study. This iterative solving procedure is expensive, both in terms of time and memory.

2. Dataset unavailability: The parameter configurations of a coupling tool govern the accuracy and time taken to perform a multiphysics simulation. However, no preliminary dataset is available to perform an exploratory data analysis and understand the behavior of different parameters. One of the major challenges of the research work is developing a dataset to perform automated algorithm configuration incorporating machine learning models. The machine learning regression models predict optimal parameter configurations with less runtime for a given simulation instance feature. The dataset creation process consumes a large amount of time due to the intricacies of the feature space and computational time associated with a single simulation instance [56] [29].

3. No benchmark instances: A multiphysics simulation benchmark for parameter configuration should provide information regarding the different MpCCI parameter configurations, types of parameters in MpCCI, domain values of the parameters, the corresponding features of the simulation instances, default values of the simulation instance and the respective performance metrics of a simulation. The performance metrics are simulation runtime or accuracy depending on the simulation problem. The unavailability of MpCCI co-simulation benchmarks for automated parameter configuration makes the comparison of results difficult. The project develops a preliminary benchmark. However, the additional challenges are the lack of expert knowledge and the large time taken to develop a relatively small benchmark set.

4. Feature extraction: The features of a simulation instance are the characteristics of the simulation problem, solver of the simulation instance, coupling scheme, and relaxation methods. There are approximately 100 features. The process of selecting the essential features of a simulation instance is a complex study by experts' opinions. Therefore, this study selects the multiphysics simulation features majorly from FSI literature study [93] [94].

1.4 R&D statement

The primary objectives of this research work are:

1. Improve the robustness of the multiphysics simulation coupling tool- MpCCI.
2. Assist the user by automatically configuring the parameters of MpCCI with optimal values.

> **R&D statement:** The proposed strategy based on Automated Algorithm Configuration (AAC) procedures effectively configures the parameters of MpCCI coupling tool for Fluid-Structure Interaction (FSI) multiphysics simulations. The suggested configurations perform simulation in a relatively lesser runtime than the default configurations in MpCCI.

The research hypothesis is that an AAC based approach can effectively configure the parameters of the coupling tool. The ultimate goal is to scale the research for the entire multiphysics simulations domain and provide an optimal parameter configuration given an input case from the user. To the best of our knowledge, being the first work to test the hypothesis, the research is focused on testing the applicability of AAC methods to estimate the optimal parameters of FSI multiphysics simulations.

FSI simulations are a complex category of multiphysics simulations [16]. In addition, FSI simulations are widely used in various applications ranging from life sciences to the design of microelectronic devices.

1.5 Report outline

Chapter 2 provides an extensive overview of multiphysics simulation, FSI, use-cases of the simulations, algorithm configuration problem, and various metrics used in the study. In addition, the chapter briefly discusses the machine learning models in section 2.4 and Bayesian Optimization (BO) in section 2.3. Chapter 3 provides a broad review of various state-of-the-art AAC methods to solve an Algorithm Configuration Problem (ACP) and provides the reason for choosing Sequential Model-based Algorithm Configuration (SMAC) in this study. The following chapter 4 illustrates the proposed strategy for automatic parameter configuration. It briefly discusses the simulation problem and hyperparameters of the coupling tool in section 4.2. Furthermore, the section 4.4.6 provides the smart coupling algorithm developed by extending SMAC for configuring the coupling tool parameters. The experimentation and results (chapter 5) tests various

sub-tasks of the proposed strategy. In addition, the research claim is tested in the final experiment 5.6 followed by a summary. The conclusion portrays the lessons learned and future research directions. Furthermore, the conclusion provides a clear enumeration of all the research contributions of this work. The appendix includes the steps involved in BO, the workflow of SMAC, and the working of the software tools contributed. In addition, features of the various simulation instances are enumerated.

2

Background

This chapter delineates a general overview of multiphysics simulation, coupling tool, and co-simulation process. This research majorly focuses on FSI interaction studies to test the research hypothesis- automated algorithm configurations can effectively configure coupling tool parameters. Therefore, FSI simulations are introduced, and the use-cases are discussed. In addition, the chapter explains automated algorithm configuration and mathematically formulates the problem in the final section.

2.1 Multiphysics simulation

Multiphysics simulation is the process of simulating the interaction of various physical phenomena involving multiple physical domains. Multiphysics simulation tools aid in modeling the physical processes with a set of mathematical formulas and studying the behavior in a software interface. There are various categories of multiphysics simulations, depending on the physical processes under study. A few categories are FSI, heat transfer, structural mechanics, fluid flow, hydrostatics, chemical reactions, acoustics, and magnetostatics [16].

The examples below are the results of the multiphysics simulation. Figure 2.1 illustrates the distribution of the acoustic pressure field in Pascal(Pa) across a car with the sound source at the speaker. This study helps the user to optimize the speaker location depending on the mirror sources and damping factors [75].

The figure 2.2 is a simulation to study the magnetic signature of a submarine traveling underwater. The study is vital in designing naval ships because the submarine traveling underwater disturbs the Earth's magnetic field and potentially triggers weapons. In addition, the study of magnetic signatures is imperative to design navigational control units, and magnetic grippers [76]. The figure 2.2 shows the distribution of magnetic flux density, Tesla (T) of a submarine underwater with the arrows representing the tangential magnetic flux density [76].

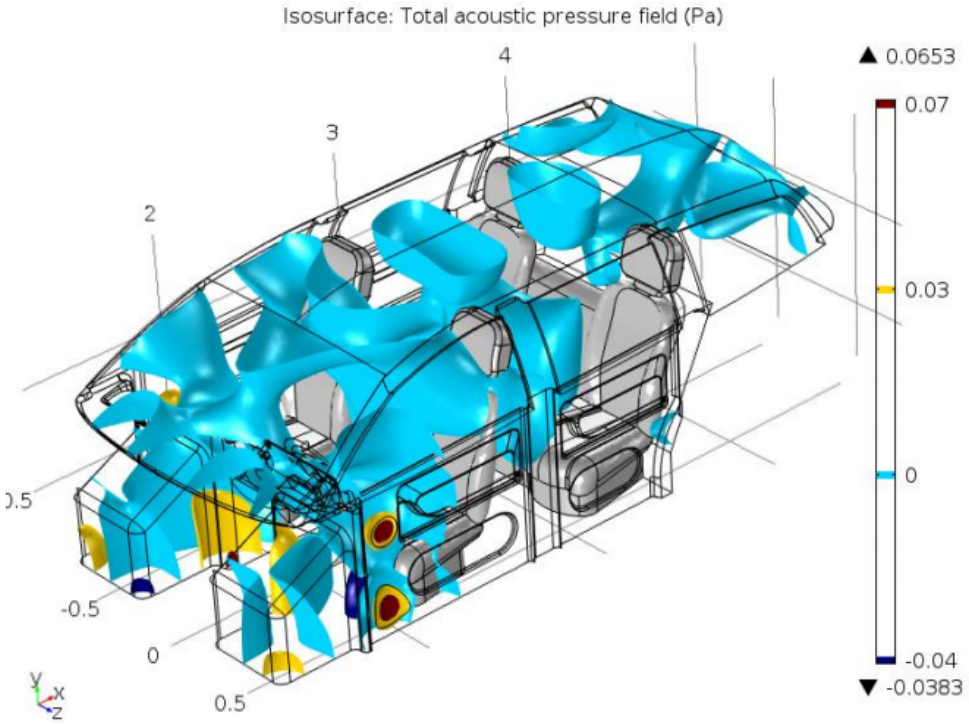


Figure 2.1: Total acoustics pressure field in Pascal (Pa)- Acoustics field study of a car [75].

2.1.1 Fluid-Structure Interaction (FSI)

FSI is one of the multiphysics simulations. In FSI, a deformable solid structure interacts with a compressible or incompressible fluid flow, where pressure and displacement are the coupled quantities. The two domains of an FSI simulation are the fluid and solid domains. The fluid and solid domains are governed by fluid flow principles and solid mechanics principles, respectively. Figure 2.3 illustrates an FSI model. The model aids in understanding the flow of the fluid inside a rigid wall with a flexible flap protruding between the walls. The flap opens and closes depending on the fluid flow direction. In addition, the simulation helps to examine the behavior of a valve with regards to the outlet and inlet fluid pressure. The coupling region is clearly depicted in the left-hand side of the figure 2.3 coupling the flap surface and the fluid.

The other examples of multiphysics simulation are thermomechanical couplings involving solid mechanics and heat conduction [35], and electrothermal couplings involving

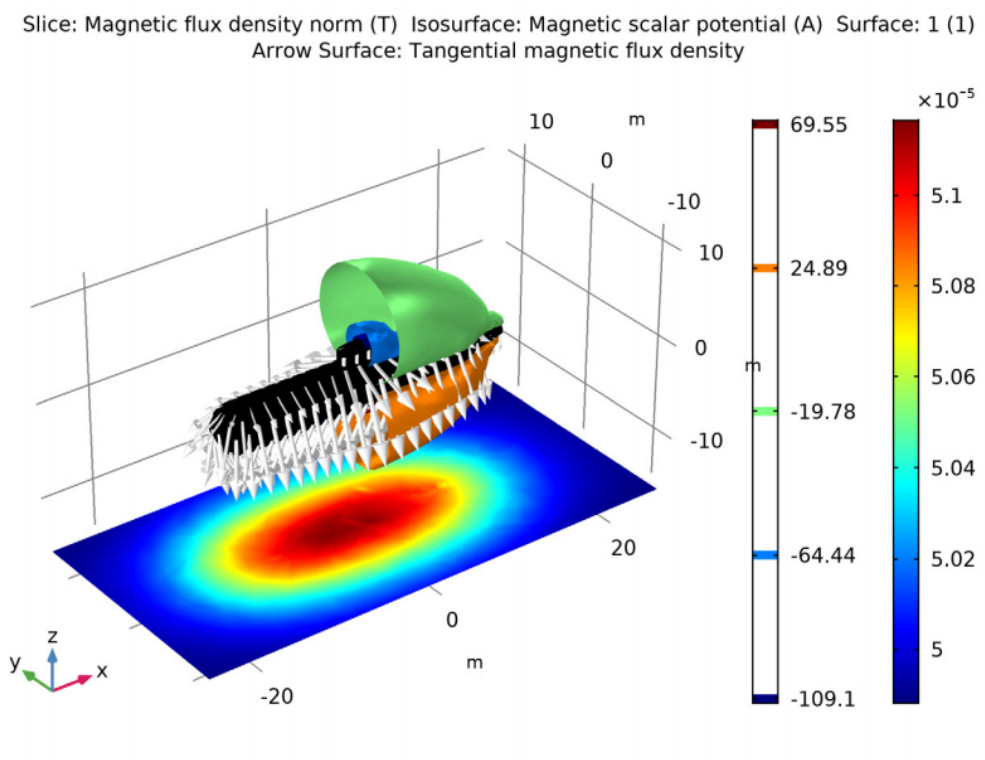


Figure 2.2: Total magnetic flux density in Tesla (T) of a submarine in an ocean [76].

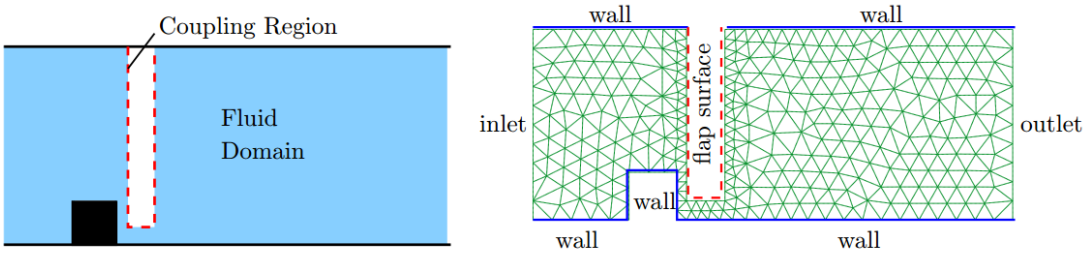


Figure 2.3: An example of Fluid-Structure Interaction model [35].

electrical conduction and heat conduction [35]. This project significantly focuses on the FSI multiphysics simulation.

2.1.1.1 Use-cases of FSI

A few applications of multiphysics simulations can be witnessed in automobiles, aircraft, robotics, and life sciences industries.

- **Automobile:** The figure 2.4 illustrates the motion of the F1 car through the air

in a simulated environment. This is used to design the F1 cars efficiently with optimal fuel consumption and reduce the drag on the car by the airflow.

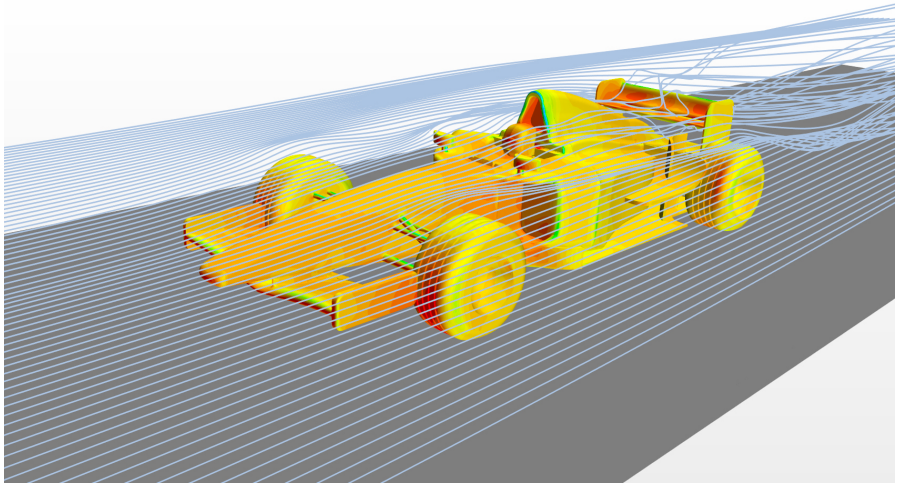


Figure 2.4: Aerodynamics simulation study of F1 cars [35].

- **Aircraft:** Multiphysics simulations are used to study the aerodynamics of aircraft and understand the drag caused on the aircraft during the motion.
- **Robotics:** Robots are widely used in domestic and industrial applications. In all these applications, the solid surfaces of the robots are in contact with air, water, or solid. Multiphysics simulations are carried to study the interaction of the robot surfaces with the environment and efficiently design the body of the robot [54].
- **Life sciences:** In life sciences, simulations play an integral role in studying the interaction of artificial organs with blood and motion of blood in a blood vessel [77]. The figure 2.5 is the output of a multiphysics simulation study. The study investigates the aorta, the blood vessel carrying blood to all parts of the body from the heart. The figure 2.5 illustrates the displacement of the artery walls due to the pressure exerted by the blood flow.

2.1.2 Parameter types of a co-simulation process

Multiphysics simulations are performed by a multiphysics simulation tool. The simulation code and the coupling environment are the integral components of the multiphysics simulation tool. The computer understandable instructions of the physical domain properties, physical processes, and the equations governing the mathematical model of

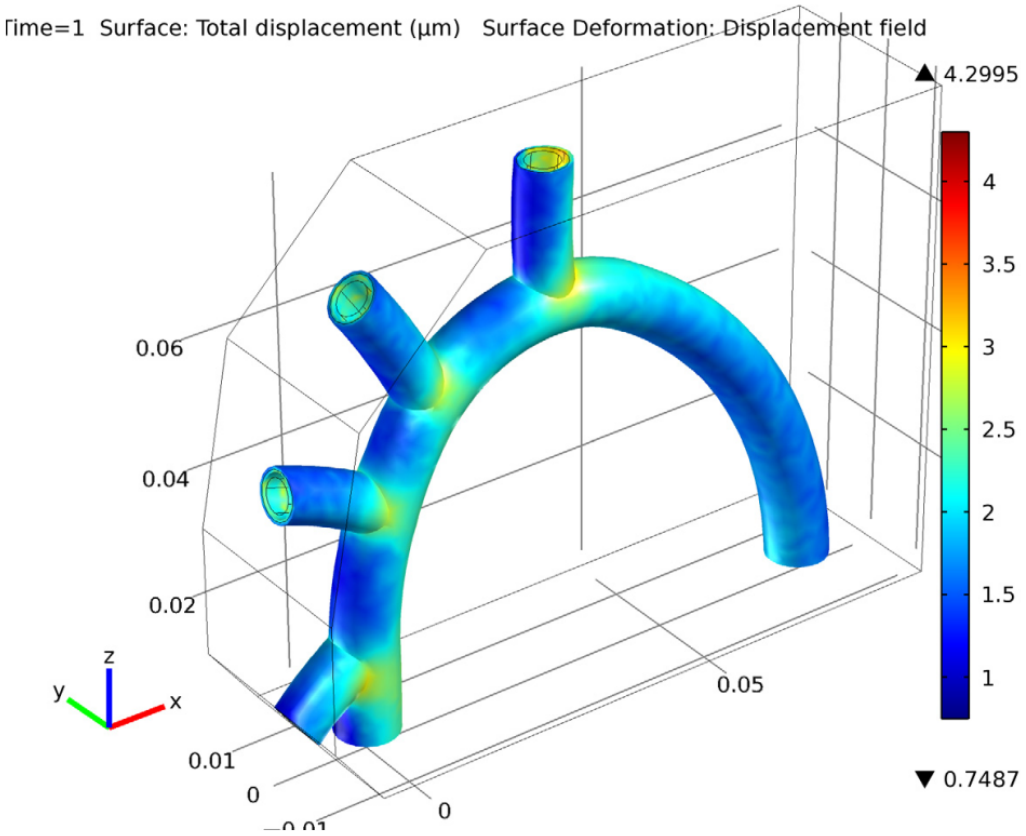


Figure 2.5: Displacement in the blood vessel due to blood flow [77].

the processes are called the simulation code. The physical phenomena are heat transfer, hydrodynamics, aerodynamics, chemical reactions, and solid mechanics. The simulation codes are specific to each domain involved in the study. The examples of simulation codes or solvers are ANSYS Fluent, OpenFoam, and ABAQUS.

The coupling environment or the coupling tool is a computer program performing data communication between the simulation codes, data post-processing, and visualization of the simulation results, as illustrated in figure 2.6. The user sets up the simulation code depending on the study requirements.

Multiphysics simulations incorporate parameters and variables. The parameters define the characteristic of simulation, and on changing the parameters, the simulation behavior changes. The behavioral changes of a simulation impact the accuracy and time taken to complete the simulation. Most of the parameters are fixed over the entire simulation period. However, a few parameters change depending on the complexity and

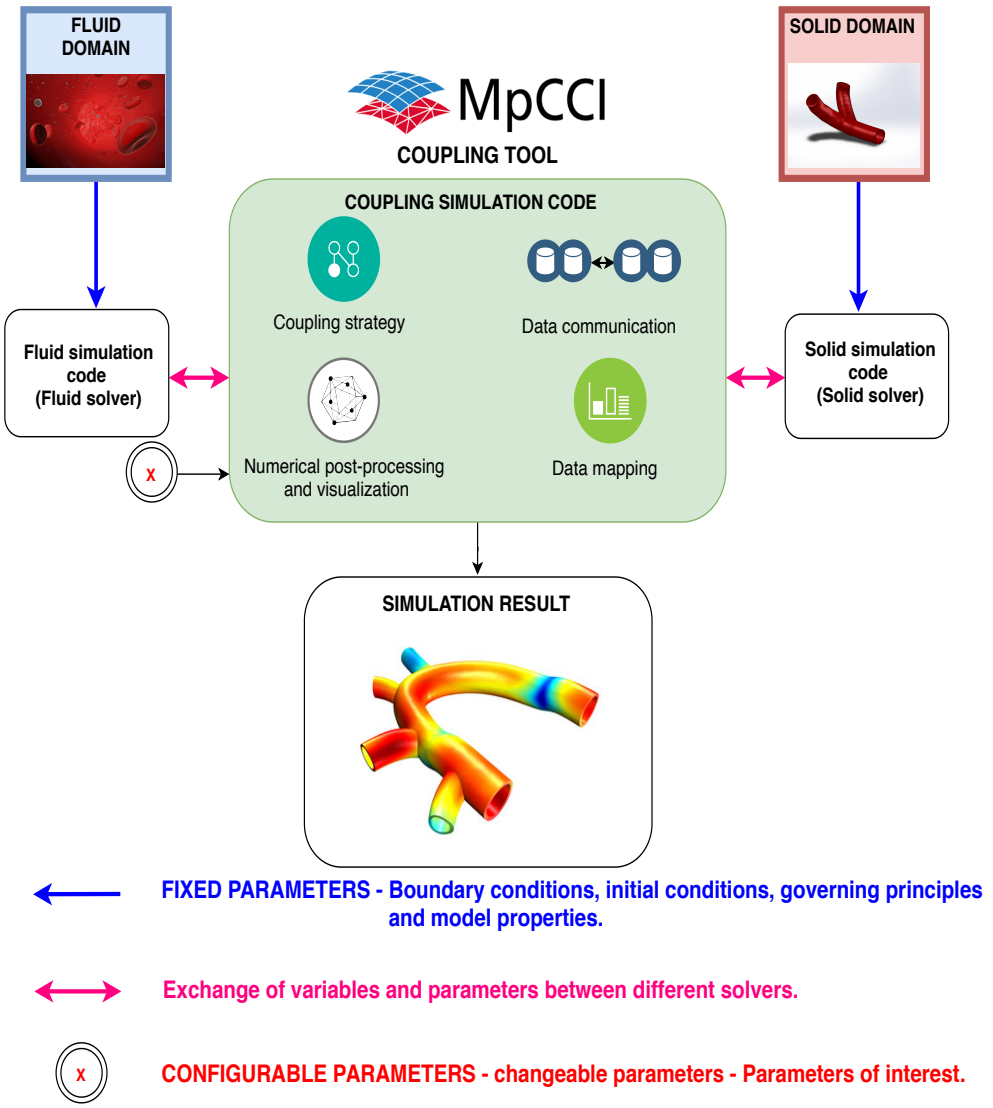


Figure 2.6: Multiphysics simulation of a coupled system. The color gradient in the simulation result represents the displacement profile of the blood vessel due to blood flow[1].

requirements of the study. A variable represents a particular state of a model and serves as a placeholder to store data.

The simulation studies include two types of parameters, namely fixed parameters and configurable parameters.

[1]The image provided inside the rectangular boxes- simulation result, fluid domain, and solid domain are from [77] [58] [20].

- **Fixed parameters:** The fixed parameters are attributed from three sources, namely the solid simulation code, fluid simulation code, and the coupling tool. The fixed parameters of the simulation codes include the simulation models and equations related to the physical process under study. The coupling environment includes a few study condition parameters such as coupled region, coupled variables, tolerance of the study, and the number of iterations. The fixed parameters of a simulation are specific to a particular simulation study performed by the user.

- **Configurable parameters:** The parameters concerning the coupling behaviour are configurable by the user in the coupling tool. The configurable parameters of the coupling tool are the hyperparameters of the coupling tool because the parameters can be tuned to control the simulation time and simulation accuracy. On tuning the configurable parameters manually, the performance of the simulation study largely differ. The examples of configurable parameters are the time step of the simulation, coupling method, relaxation method, and constants of the coupling strategy. A detailed explanation of the parameters is provided in section 4.2.

The figure 2.6 illustrates a multiphysics simulation process of blood flowing in a blood vessel. The blood vessel with blood flow is the coupled system incorporating FSI physics. The coupling tool communicates the data in a particular manner between the solid and fluid solver over the entire course of the simulation. In this study, MpCCI, a coupling tool, is used to perform multiphysics simulations through the partitioned coupling. The configurable parameters set by the user in the coupling tool are the parameters of interest.

2.2 Automated Algorithm Configuration (AAC)

The algorithms solving hard computations are prominently guided by a search strategy and are mostly parameterized. The tree search parameters include bounds at each level, the decision rule for branching, minimum nodes in the child branch, and depth [53]. The search strategy pertaining to IBM ILOG CPLEX, one of the mixed-integer programming solvers, exposes 76 parameters. The parameters largely affect the solver performance, and manually examining the parameter space for best configuration is highly tedious [53]. AAC approaches configure a target algorithm using parameter values that maximize a particular performance metric. The AAC methods are general-purpose configurators. On a training instance set, AAC methods estimate the optimal parameter configuration by optimization of the target algorithm. This process of tuning the parameters on a training set is called offline configuration. The best configuration is used to configure the algorithm in real-time [53]. The applications of AAC includes AI planning [90], mixed

integer programming [48], SAT solvers [46], and answer set programming [39]. In addition, AAC methods are extended to identify the optimum parameter configuration for machine learning frameworks [88] [33], deep architectures and deep neural networks [28] [31]. An extended overview of the various state-of-the-art methods in AAC is provided in chapter 3.

2.2.1 A primer on Algorithm Configuration Problem (ACP)

The problem of configuring the parameter of the coupling tool with optimal values is formulated using an Algorithm Configuration Problem (ACP). The algorithms optimizing the target algorithm performance are called configuration procedures or algorithm configurators. The formal definition of ACP is defined by, "given a parameterized target algorithm $\mathcal{A}$ with a set of instances Π and parameter configuration space Θ with a cost metric 'c': Θ x $\Pi \rightarrow R$, the problem of estimating an optimal configuration $\theta^* \in \Theta$ by minimizing the cost metric over all the instances in Π" [32] [31].

$$\theta^* \in \underset{\theta \in \Theta}{\operatorname{argmin}} \sum_{\pi \in \Pi} c(\theta, \pi) \tag{2.1}$$

The configuration performing best overall the set of instances up to a specified budget of the configuration process is called incumbent configuration, θ_{inc}. The incumbent is the best optimal configuration estimated by the configurator. The incumbent configuration θ^* belongs to the configuration with the minimum cost metric across the configuration space,Θ evaluated on a particular set of instances selected using a random seed. The 'belongs to' symbol is the effect of the cutoff limit κ, restricting the evaluation of the exact cost metric across the set of instances.

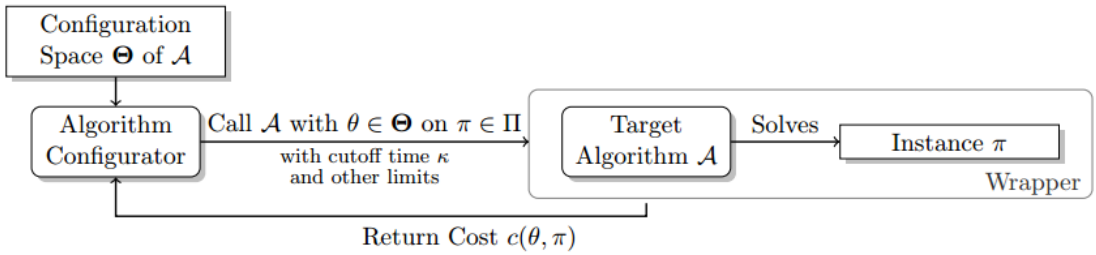


Figure 2.7: Algorithm configuration workflow [32]. The target algorithm $\mathcal{A}$ is evaluated with different parameter configurations θ on various instances π of the problem. The cost metrics of the evaluation $c(\theta, \pi)$ is provided to the configurator.

The workflow of algorithm configuration is illustrated in figure 2.7. Initially, the

target algorithm $\mathcal{A}$ is executed by the configurator to solve a particular problem instance using a valid configuration θ [32]. The configuration space, Θ is used to sample the valid configuration. The target algorithm is optimized for the cutoff time κ and other resource limitations such as memory consumption provided by the user at the time of execution. In order to preserve the generality of algorithm configuration procedures and optimize a wide range of target algorithms, wrappers are utilized. The wrapper runs the target algorithm and returns the cost metric to the configurator. The input to the wrapper is the input configuration and instances for evaluating the algorithm. The possible cost metrics or the performance metrics an algorithm configurator can optimize are runtime, solution quality, memory requirement, and error value. In contrast to runtime and memory, the solution quality optimization is a maximization problem.

With regard to this research work, the variables corresponding to the workflow of the algorithm configuration problem is defined in table 2.1.

Variables	Variable name	Research resemblance
$\mathcal{A}$	Target algorithm	Multiphysics simulation- FSI study
θ	Parameter configuration	Parameters of the coupling tool
Θ	Configuration space	Vector space spanned by the parameters
π	Problem instance	Single simulation instance
Π	Set of problem instances	Set of simulation instances
c	Performance metric	Runtime of the simulation (in seconds)
-	Wrapper	Command-Line Interface

Table 2.1: Research variables.

2.2.2 AAC mindmap

AAC effectively solves a wide range of target algorithms, namely Boolean satisfiability, mixed integer programming, configurable software systems, and machine learning algorithms. In addition, the performance metrics AAC algorithms focus to optimize is solution quality, run-time, memory requirements, approximation error, and plan length. The two major paradigms of algorithm configuration based on the approaches are model-based and model-free. The next chapter 3 provides an extensive overview of the various model-based in section 3.2 and model-free in 3.1 approaches implemented in AAC.

One of the imperative requirements for choosing the suitable AAC approach to a specific problem depends on the parameter types and the number of parameters involved in the study. With developments in the field of AAC, the state-of-the-art algorithms

accommodate four types of parameters, namely categorical, numerical, and conditional. The categorical parameters are variables accepting discrete domain values and subdivided into three types. The three types of categorical parameters are dichotomous, ordinal, and nominal variables accepting binary, ordered, and unordered domain values. The numerical parameters include real or integer values. The discrete and continuous parameters are different types of numerical parameters [30]. The conditional parameters refer to parameters dependent on a parent parameter. The conditional parameters are active in a configuration given the parent parameter is assigned to a specific value. The conditional parameters often denote a sub-process or parameter behaviour dependent on another process or parameter.

The mindmap in figure 2.8 illustrates the semantics of the field AAC.

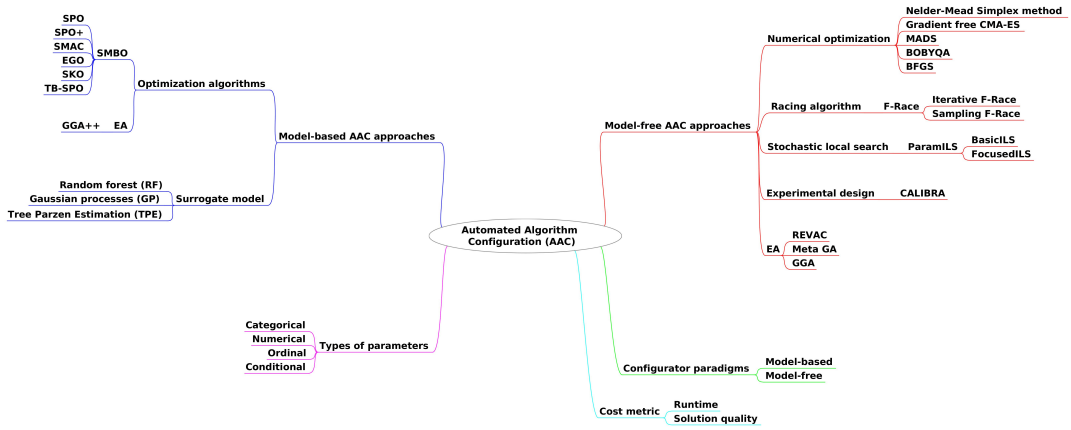


Figure 2.8: Mindmap of Automated Algorithm Configuration.

2.3 Bayesian Optimization (BO)

BO solves the mathematical problem of estimating the global maximum or minimum of an objective function, f.

$$\theta^* = \arg\max_{\theta \in \Theta} f(\theta) \tag{2.2}$$

Θ is the configuration space. θ is a parameter configuration of the black-box function f. The information regarding the function structure such as concave or linear is unknown, and the function lacks the analytical expression to calculate gradient information [36]. However, evaluations can be performed at any valid query point θ from the domain. The

output value of the function, y ∈ R, is stochastic with noise. This serves as the minimum condition for BO [83].

BO is useful in optimizing expensive, multimodal, and non-convex objective functions. For instance, consider an image classification system f with hyperparameters x. A few tunable parameters 'x' of a deep network are learning rate, number of intermediate hidden layers and number of hidden layer neurons. The classification accuracy on a particular dataset is the evaluation result, y = f(x). The process of tuning the hyperparameters to maximize y is expensive because of the high evaluation cost of the function f. In this scenario, BO aids to effectively guide the search of optimal parameters and maximize y by estimating the optimal set of parameters x [83] [36].

The applications of BO includes hyperparameter optimization of neural networks [8], gait optimization for legged robots [66], and mixed-integer programming solvers [51]. In recent extensions of BO, the acceptable input variables are conditional, categorical and numerical [51] [83] [36].

The fundamental principle of BO is Bayes theorem. A probabilistic model (P(cost | hyperparameters)) is constructed using the known prior belief of the objective function evaluations. This model is called the surrogate model. The prior distribution, P(c) is the set of costs for the initial samples/parameters obtained by evaluating the parameters on the objective function R_n= $\{(\theta_1, c_1), (\theta_2, c_2), ...(\theta_n, c_n)\}$. The likelihood estimate of observing the data given the prior is estimated, $P(R_n|c)$. The surrogate is constructed using the below equation.

$$Posterior \propto Likelihood \times Prior \tag{2.3}$$

$$P\left(c|\mathcal{R}_n\right) \propto P\left(\mathcal{R}_n|c\right) P(c) \tag{2.4}$$

The posterior probability is the surrogate model. This conditional probability depicts the objective function performance $c = f(\theta)$, given the samples or parameter configuration is an estimation of the objective function performance. The surrogate model efficiency in summarizing the objective function is proportional with the evaluation count of the objective function with various configurations. In addition, the model aids to select the promising point for the upcoming evaluation. The selection is based on an acquisition function. This guide the search efficiently by decreasing the direct evaluation count of the objective function [83] [36].

The query point is selected using the posterior surrogate model and the acquisition function. The surrogate model exploitation and exploration is mediated by the acquisition

function. The former focus on the query point with the extreme value of the objective function and the latter estimate the query point with maximum uncertainty [83] [36]. The new evidence is collected by evaluating the objective function at the new query point. The evidence is used to refine the model and update the posterior probability again. This is the basic principle behind all variants of Sequential Model-Based Optimization (SMBO) and the algorithm is provided in Algorithm 1 [36]. The initial function evaluations are carried out using arbitrary query points.

The algorithm of BO performs the following steps.

1. **Select initial sample:** Select sample points to query the noisy objective function and evaluate the points on the objective function. The figure B.1 shows the evaluation of two initial sample points.

2. **Fit surrogate:** The surrogate model is fit to the initial samples. The iteration 1 in figure B.2a illustrates the curve fit to the initial samples. The surrogate model performs regression task to predict the performance or cost metric of the objective function given the configurations. The various surrogate models available are Gaussian Process (GP), Random Forest (RF), and Tree Parzen Estimator (TPE).

3. **Optimize acquisition function:** This step includes optimizing the acquisition function to obtain the next query point to evaluate the objective function [36]. The various objective functions available are expected improvement (EI), maximum probability of improvement (MPI), and entropy search (ES). In this example, EI is used to select the upcoming query point. The improvement function, $\mathrm{I}(\theta)$ is the difference in the predictions by the surrogate model for a particular configuration θ and the current best configuration θ^+ [14]. The equation is positive for a particular configuration better than the existing best configuration.

$$\mathrm{I}(\theta) = \max\left\{0, f_{n+1}(\theta) - f\left(\theta^+\right)\right\} \tag{2.5}$$

The next query point is determined by the maximal value of the expected improvement given by the following equation [73].

$$\theta = \underset{\theta}{\operatorname{argmax}} \mathbb{E}\left(\max\left\{0, f_{n+1}(\theta) - f\left(\theta^+\right)\right\} | \mathcal{R}_n\right) \tag{2.6}$$

where, $f(\theta^+)$ is the best performance metric at the particular iteration and $f(\theta)$ is the predictions by the surrogate for a particular configuration in the distribution θ.

Symbol	Meaning
Θ	Configuration space.
θ_i	i^{th} parameter configuration from the configuration space, ($\theta \in \Theta$).
c_i	Cost of the i^{th} parameter configuration.
Θ_{inc}	Best configuration/ Maximum or minimum query point.
R_n	Set of n parameter configurations and respective costs.
f	Objective function or target algorithm under optimization.
m	Surrogate model- Gaussian process, Random forest.
α	Acquisition function.

Table 2.2: Symbols used in the BO algorithm (Algorithm 1).

An analytical closed form of the EI is calculated for different surrogate models [73] [14].

EI quantifies the utility of the configuration to provide better performance on the target algorithm. Therefore, the configuration with larger value of EI is selected for evaluation on the target algorithm. EI is calculated using the empirical mean and variance of the performance metric distribution in the surrogate model [73] [14]. In addition, the score balances between the high uncertainty regions of the distribution and high performance metric. The acquisition function subplots of the figures B.2a, B.2b, B.2c, B.2d, B.3a, B.3b, B.3c, B.3d depicts this step. The minimization and maximization of a particular problem depends on the procedure involved in optimizing the acquisition function. The maximization of the negated acquisition function models the minimization problem.

4. **Evaluate target algorithm:** The objective function is evaluated using the sample or the configuration selected from the acquisition function. The surrogate model is fit again. This is the step of collecting new evidence and updating the posterior.

5. **Repeat:** Perform fit surrogate, optimize acquisition function and evaluate target algorithm step until the consecutive query points converge or a budget criterion is satisfied.

6. **Incumbent configuration:** After exploiting the budget criterion or achieving convergence, the best sample point or the parameter configuration is returned. This sample point is the minimum or the maximum of the objective function depending on the problem formulation.

The BO algorithm (Algorithm 1) is adapted from [36] [83].

Algorithm 1: Pseudo-code for Bayesian Optimization

Input: Surrogate model m, objective function f, acquisition function α, Initial query points $\theta_{initial} = \{\theta_1, \theta_2, ..., \theta_n\}$

Result: Global Maximum query point

$c_{initial} \leftarrow \mathrm{f}(\theta_{initial})$

$R_n = \{(\theta_1, c_1), (\theta_2, c_2), ...(\theta_n, c_n)\}$

m $\leftarrow$ FitModel(R_n).

for *i = 1,2 ..j* **do**

  Select new θ_{n+1} by optimizing acquisition function α.

  $\theta_{n+1} \leftarrow \arg\max_{\theta \in \Theta} \alpha(\theta, R_n)$

  Query objective function to obtain c_{n+1}

  Augment the data, $R_{n+1} \leftarrow \{R_n, (\theta_{n+1}, c_{n+1})\}$

  m $\leftarrow$ FitModel(R_{n+1}).

  return *Query point θ_{inc} with maximum f(θ) from R_{n+1}*

end

2.4 Machine Learning (ML) models

The following ML models are used to perform regression (refer section 4.4.4) in this work.

2.4.1 K-Nearest Neighbors (KNN)

KNN is a non-parametric supervised learning algorithm. The test samples are predicted depends on the K nearest training samples in the feature space provided. For classification, the output prediction is the majority vote of the K nearest samples class membership in the training examples. For regression, the mean of the K nearest samples is the dependent variable value. The algorithm is sensitive to the K value and model is slower on increasing data volume [42]. The majority vote is affected in a skewed distribution of classes. One of the ways to mitigate the issue is to assign a weight value to each data point depending on the distance from the test sample [79].

2.4.2 Support Vector Machine (SVM)

SVM is a supervised learning machine for classification and regression tasks [18]. For classification, SVM estimates the optimal decision boundary or hyperplane with maximum margin of separation between different classes. The distance between the hyperplane and the data samples on both sides of the hyperplane is called the margin. SVM finds an optimal decision boundary for linearly separable data. However, SVM supports linearly non-separable data by using a kernel trick. The kernels are functions projecting the low dimensional input features to high dimensional features. This kernel functions transform a linearly non-separable data in lower dimension into linearly separable data in higher dimension. Polynomial, Radial Basis Functions (RBF) and sigmoid are popular kernel functions. The support vectors are the critical points exactly at the margin of separation on either side. The support vectors affect the position of the hyperplane. The optimal decision boundary is estimated by minimizing an error function with constraints. The decision y for a particular feature x is given by, $y = sign(w.x+b)$. w and b are parameters of the decision boundary. For regression, $y = w.x + b$ provides the target variable value [18].

2.4.3 Random Forest (RF)

RF is an ensemble based classification and regression machine learning model building numerous decision trees. RF utilizes a bootstrap sampling process for selecting the training set by random sampling with replacement called bagging. In addition, RF selects random subset of the available features for splitting the data at each node. Among the subset of features, the features for splitting the samples depends on the information gain (IG) provided by the feature. The IG is a metric providing the amount of information provided by the feature regarding the class in classification tasks or prediction variable in the regression tasks. The feature with maximum information gain is selected for splitting the data at each node. A large value of IG represents less entropy in the resulting data groups after split. In classification task, each tree predicts the label class and the class with maximum vote is assigned to the test data. In regression task, each tree predicts the response dependent variable and the average of the prediction across all trees is the test data value. RF reduces the overfitting issue of decision trees by injecting randomness in the process through bootstrap aggregation, feature bagging and averaging the result of numerous trees [25] [13].

2.4.4 Quantile Random Forest (QRF)

QRF performs quantile regression and the process of building a tree is exactly similar to random forest. The significant difference between QRF and RF is all the information regarding the target variable is preserved after training instead of averaging the target variable value at a particular leaf node. All the target variable values at a particular leaf are used in the prediction phase to predict at a particular quantile value. For instance, the leaf node of any particular tree after training hold the values - 1,2,3,4, and 5, the prediction at 25 percentile is 2. In constrast to RF predicting conditional mean, this type of quantile regression provides the conditional quantiles [69] [79].

2.4.5 Gradient Boosting Machines (GBM)

GBM is an ensemble method of building weak classifiers in a sequential manner. Each of the week classifiers predict the training labels with atleast 50 percentage accuracy. Initially, a weak model is assigned with a constant value (average of the target variable). Secondly, a GBM fits a weak classifier to the training features with the the pseudo-residual error calculated by the negative gradient of the user-defined error function. The weight of the weak classifier depends on the target and the aggregate prediction of the existing weak classifiers solved using an one-dimensional optimization task. The weak classifiers are built in an additive manner until the pseudo-residual reaches a tolerance. GBM is a variant of Adaboost with a differential error function optimized using steepest descent. In contrast, Adaboost increases and decreases the weight of the wrongly and correctly classified data sample respectively in the next weak classifier. The final classifier is an aggregation of the each weak classifier output [37] [11].

2.5 Metrics

The following metrics are utilized in the experiments and results (chapter 5) to evaluate the proposed strategy and the ML models.

2.5.1 Statistical dispersion

The following statistical metrics quantify the spread of a distribution. The repeatability coefficient is estimated using the common dispersion metrics.

2.5.1.1 Variance and Standard deviation

Variance is based on the measure of deviation of individual samples from the mean [86]. It is the mean of the squared difference of each sample from the mean given by the equation 2.7. x_i is the i^{th} sample in the data, n is the total number of samples in the data and μ is the mean of all the samples in the data. The denominator (n-1) provides an unbiased estimate of the population variance and population standard deviation from the respective sample statistics.

$$\sigma^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1} \tag{2.7}$$

$$\text{standard deviation}, \sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1}} \tag{2.8}$$

2.5.1.2 Repeatability coefficient

Repeatability coefficient, S_r is given by the equation 2.9. The maximum deviation with a probability of 95% between two successive measurements of the same subject under same measurement conditions and using the same procedure [68] [6].

$$S_r = 1.96 \times \sigma \tag{2.9}$$

The standard deviation, σ is the within group standard deviation. The reason is repeatability focus on the agreement of run-time on a particular instance and the deviation between different instances are avoided. In certain literature [12], a multiplication factor $\sqrt{2}$ is included in the above equation 2.9 and provides an over estimate of the repeatability coefficient. Therefore, the equation 2.9 is used in this project [12] [43] [55]. The within-group standard deviation is computed using Analysis of Variance (ANOVA).

"ANOVA is a statistical technique to analyze significant differences between mean of different groups. In one-way analysis of variance, the statistically significant differences between the means of three or more independent groups is tested" [81] [23]. The assumptions of ANOVA include parametric and normally distributed data [81]. ANOVA estimates F-statistics or F-ratio. F-ratio signifies the differences in the sample mean. A less F-ratio signifies the samples are more similar. The two hypothesis in ANOVA are null and alternate hypothesis. The F-statistics and p-value from ANOVA are used to accept the null hypothesis or reject the null hypothesis. Null hypothesis and alternate hypothesis is stated in equation 2.10. μ_g and μ_m are the sample means of any two groups

in the observation. μ_i, where i = 1,...N is the sample means of the N groups in the observations [81].

$$\begin{aligned} &\boldsymbol{H}_o : \boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = \cdots = \boldsymbol{\mu}_N && \text{Null hypothesis} \\ &\boldsymbol{H}_1 : \boldsymbol{\mu}_g \neq \boldsymbol{\mu}_m && \text{Alternate hypothesis} \end{aligned} \tag{2.10}$$

$$\text{F} = \text{ Between group variability / Within group variability} \tag{2.11}$$

$$F = \frac{MS_b}{MS_w} \tag{2.12}$$

MS_b and MS_w is the mean of square differences between and within groups of a sample [81].

$$MS_b = \frac{SS_b}{df_b} \tag{2.13}$$

$$MS_w = \frac{SS_w}{df_w} \tag{2.14}$$

The denominators df_b and df_w are the degrees of freedom between groups and within group described in equation 2.15 and 2.16. N is the total samples in the data without considering the groups. K is the total groups [81].

$$df_w = N - K \tag{2.15}$$

$$df_b = K - 1 \tag{2.16}$$

$$SS_b = n_k \sum_k (\bar{x}_k - \bar{x}_G)^2 \tag{2.17}$$

SS_b is the between group sum of squared differences representing the total variability between the average of different groups [81].

$$SS_w = \sum_i (x_i - \bar{x}_k)^2 \tag{2.18}$$

SS_w is the within-group sum of squared differences. $\bar{x}_G$ is the total mean of all the observations in the dataset. The mean of each group is $\bar{x}_k$. x_i is the value of a specific sample, i in the group [81].

2.5.2 Regressor performance

The regressor models are compared using the following error metric in experiment 5.5.

2.5.2.1 Root Mean Square Error (RMSE)

RMSE provides an estimation of the residual standard deviation. RMSE is the square root of the average of squared difference between the predicted and true values [15] [79]. It provides more error weightage to the larger errors. RMSE criticise larger error values more compared to the smaller errors [15] [79]. The error holds the same units of the estimated quantify. In addition, it overcomes the under-estimation issue of MSE for error values lesser than 1 by providing error in the same units of the quantified variable. RMSE is provided in equation 2.19. y_j and $\hat{y}_j$ represents the actual and predicted values of a set of 'n' samples [15] [79].

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{2.19}$$

3

Related work

The pivotal part of an algorithm configuration task is optimization. The major classes of optimization algorithm based on the procedure of solving the optimization tasks are stochastic optimization and deterministic optimization. Stochastic optimization includes a set of algorithms performing maximization or minimization of an objective function with randomness. It is also called as optimization under uncertainty. The objective function or the constraints in the optimization problem are the sources of randomness. The various stochastic optimization methods include random search, simulated annealing, hill climbing, evolutionary computing and genetic algorithm based methods [85].

Deterministic optimization methods utilize direct mathematical approaches on the analytical formulation of the problems to solve the optimization task. A few examples are mixed-integer programming, non-linear and linear programming [64] [22].

The problem of algorithm configuration can be viewed as a stochastic optimization task. The stochastic optimization methods [85] need the gradient explicitly. However, the few algorithms that estimate the gradient information using certain evaluation functions converge at local extremes only. In addition, most of the methods significantly focus on numerical parameters. This restricts the direct application of stochastic optimization algorithms for Automated Algorithm Configuration (AAC) problems [44].

The various methods for automated algorithm configuration discussed in the literature are based on local search, experimental design, genetic algorithm, racing, numerical optimization and Sequential Model-Based Optimization (SMBO).

3.1 Model-free algorithm configuration methods

This segment gives a brief report of various model-free algorithm configuration approaches. The model-free algorithms are comparatively simple. The methods employ local search strategies, racing algorithms or experimental design procedures to configure the target algorithm.

3.1.1 Local search

Local search methods are useful to solve optimization problems. The search initiates from a random candidate solution in the solution space of the problem. A cost function guides the search strategy by evaluating the performance of different candidate solutions. The search is governed by generating candidate solutions. The candidates are neighbors obtained by applying local changes to single value of the current candidate solution. A candidate solution is a state with complete assignment of the configurations called complete-state formulation [80]. The search terminates with the successive neighbors resulting in the same cost metric or zero cost. The former requires restarting the algorithm with a random state and the latter is the global minimum [26]. A few members of the local search family include hill climbing, simulated annealing, local beam search and genetic algorithm [80]. The applications of the local search approaches include Constraint Satisfaction Problems (CSP), planning and scheduling systems, Travelling Salesman Problem and Vehicle Routing Problem (VRP), heuristic procedures, time-table scheduling and SAT solver configuration.

3.1.1.1 Hill climbing

The neighbors pertaining to the current candidate solutions are generated and the neighbor with the best cost metric is selected to be the next state. The algorithm is similar to moving uphill or downhill depending on the optimization problem at hand [80]. The research work by Gratch, et al., [41] in 1992, offers a probabilistic solution to overcome the utility problem in planning systems and enhance the speed of learning. In this scenario, the performance of planning systems is affected by the knowledge learned by the systems. The paper focus on adaptive problem solving and utilizes hill-climbing to search for along the configuration space. The statistical significance is used compare between different configurations. In 1996, Composer system has been used in scheduling the communication. Composer has significantly improved 5 parameters of the scheduling algorithm and aided in the communication between spacecrafts and antennas [40].

The drawbacks of the system are primarily due to the drawbacks of hill-climbing search. The search algorithm often fails to proceed further at the local extremes, ridges and plateaus [80]. In addition, the study has been restricted to only 5 parameters of an algorithm.

3.1.1.2 Beam search

The beam search is a local search method working by promising node exploration. Initially, the search strategy commences with a set of promising 'n' candidates. Following, beam search generates all the successor candidates and utilize a heuristic to select only the 'n' best successors. The value 'n' is called beam width. The algorithm returns the best candidate until a defined search budget or finding a goal [80].

An analytic learning system by Steven Minton focus in 1993 [71] for specializing heuristics concentrates on the Multi-Tactic Analytic Compiler (MULTI-TAC) systems. The MULTI-TAC systems aid solving constraint satisfaction problems by an analytical approach. The inputs are the generic heuristics concerning the problem, problem domain, and the distribution of the various instances. The system develops various LISP programs corresponding to a particular domain and heuristics. The sampled problem instances are used to evaluate the LISP programs and the best program is selected by performing beam search.

The disadvantages of the system are similar to the challenges of beam search. The beam search is non-optimal and incomplete because of the possibility of pruning a goal state or the best configuration [74]. The beam search is affected by diversity limitation among the 'n' states resulting a restricted exploration of the solution space. In addition, to overcome the local optima, beam search is restarted with 'n' random states [80].

3.1.1.3 Experimental design with local search

Experimental design incorporates the process of closely planning the experiment. This includes data collection and analyse the data to draw conclusion. The two steps of experimental design are design of experiment and statistical analysis. The structure of the experiment corresponds to the experimental design. In this study, factorial design is used to design the experiment. The parameter configurations of the problem is analyzed to formulate the possible levels of the parameters. A factorial design performs 2^l or 3^l experiments on the problem. l is the number of parameters. The base, 2 and 3, is the respective number of choices for a parameter. A fractional factorial design provides a reduction in the total experiment count given by 2^{l-p} or 3^{l-p}, where p is the reduction factor. The experiments evaluate all the possible critical factors on the target algorithm. The search strategy is guided by the experimental results. The major disadvantage is the total number of experiments and parameters are exponentially related. This practically restricts the process of configuring target algorithms with numerous parameters [1].

The study [24] by Steven P. Coy, et al., proposes a method to effectively estimate

the parameter settings of different heuristics. The research employs gradient descent and statistical experimental design procedures. The procedure requires a predesigned training instance set. The algorithm estimates the optimal parameter configuration using gradient descent and experimental design for each training instance. The experimental design employs two design strategies, namely, fractional factorial and full factorial design strategies. The parameter configuration of all the instances are averaged to estimate the optimal parameter values. The method is restricted to only numerical parameters because of the final averaging step to estimate the optimal parameter values.

CALIBRA finds the best search parameter for a heuristic procedure. The study is similar to [24] utilizing full factorial design procedure with response surface. However, the follow up research work on CALIBRA, [1] is comparatively effective. The work by Adenso-Diaz, et al., [1] is a combination of local search and full factorial design is employed in the study. The procedure utilizes two values of a parameter for evaluating each configurations. This iterative process estimates the optimal parameter configuration space by testing nine configurations surrounding the good parameter configuration space. The experimental design grid is modified until local optimum is estimated. CALIBRA outperforms the existing configuration for six target algorithms. CALIBRA is applicable to only ordinal and numerical parameters. In addition, the procedure is restricted to estimate only five parameters at maximum.

3.1.1.4 Genetic Algorithm (GA)

Genetic Algorithm is a variant of stochastic beam search incorporating natural selection and genetics. The stochastic beam search selects the 'n' random successors with a probability assigned to each successors. The associated probability value depends on the cost value with the successors. In genetic algorithm, a random population is sampled from the solution space. The offspring (successors) are generated by applying a genetic operator to the population. The genetic operators are crossover and mutation performing a combination of the parents followed by alteration to provide an offspring. The offspring candidates are assigned with a fitness value. The next population set is selected to produce offsprings depending on the fitness value of the candidates [80].

The initial works on incorporating evolutionary algorithm for solving algorithm configuration problems apply Genetic Algorithm concepts to time-tabling challenges by using an indirect chromosome encoding methodology [87]. This method overcomes the limitations posed by direct encoding and solves problems in various related domains such as neural networks, manufacturing and scheduling. The research work optimizes

the scheduling problem for 12 instances using a modified Brelaz algorithm. A total of seven categorical parameters has been configured. However, the optimization is carried out individually for each instance. In addition, the time taken for optimization is not quantified and still under research. The time taken to estimate the solution is under exploration [44] and evaluation of fitness function for each candidate solution is expensive. The methodology is restricted to single instance optimization. The stochastic nature of selection restricts the optimality condition and completeness of the search [80]. In addition, the parameters such as population size, fitness function, mutation rate, cross over and offspring selection method are critical. The assignment of inappropriate values result in extremely unfavourable outcome [92] [70].

Gender based Genetic Algorithm (GGA) configuration is an evolutionary algorithm based algorithm configuration method. GGA is widely used in parameter tuning of various SAT solvers [57], mixed integer programming [57] and machine reassignment [67]. In GGA, the population is divided into two types non-competitive and competitive. The former aids in diversifying the candidates and later population is tested on the target algorithm directly. This is followed by a tournament based selection procedure and a strong intensification procedure. The comparative studies on GGA and FocusedILS, a variant of ParamILS proves GGA is less effective than FocusedILS [53]. GGA has been extended with response surface methods for choosing offsprings effectively and faster in GGA++[3].

3.1.2 Racing algorithm

Algorithm configuration problems have been solved by incorporating the racing algorithms in machine learning. One of the racing procedures is F-Race [10] and successfully aids in configuring stochastic local search algorithms. The target algorithm $\mathcal{A}$, instance distribution D and finite set of configurations are the inputs of the algorithm. The parameter configurations are sampled from a distribution of the parameters. The configurations are evaluated repeatedly on samples drawn from the instance distribution until only one parameter configuration is available or terminates at the completion of user specified time budget. The configurations performing significantly lesser than a particular configuration are eliminated using Friedman or paired t-test.

A promising variant of F-Race is Iterative F-Race (I/F-Race). In I/F-Race, the configurations are chosen from a probabilistic model followed by the conventional F-Race [4]. The probabilistic model is biased to sample towards good configurations on collecting new evidence.

F-Race evaluates the possible configurations at the beginning of the procedure. Therefore, F-Race is restricted to applications possible of enumerating all the candidate configuration. The procedure is restricted to numerical parameters and certain variants are available to accommodate categorical parameters. However, the effectiveness of the algorithm in handling numerous categorical and conditional parameters at complex algorithm applications as discussed in [46] [49] [59] [89] is unexplored. The applications of racing methods in algorithm configuration include combinatorial optimization, and mixed integer programming.

3.1.3 ParamILS

The paper discusses a new AAC framework by Frank Hutter, Holger H. Hoos, et al., [48] called ParamILS. One of the major contributions of this iterated local search procedure is adaptive capping. It provides cutoff time for the evaluation of each configuration. This improve the time efficiency of the algorithm.

Being a model-free procedure, ParamILS selects the initial configuration by combining random and default parameter values with fixed number of perturbation moves. The methodology accepts equally performing configurations or better configurations and restarts at a random values. The procedure can alter only one parameter value at each target algorithm evaluation. BasicILS and FocusedILS are the two instantiations of ParamILS incorporated in the framework [48].

ParamILS supports discrete parameters only. However, the parameters in most of the problems are continuous. The discretisation of continuous parameters and dynamically extending the domains of the parameters is a challenging task. In addition, ParamILS is restricted to single instance optimization. The applications of ParamILS in algorithm configuration include mixed integer programming, time-tabling, and protein folding [48].

3.2 Model-based algorithm configuration methods

The direct search, model-free methods does not take advantage of each evaluations of the target algorithm by explicitly modeling the configuration space. However, model-based methods capture the regularities in the configuration space over time. The model-based methods consistently fits a model to the target algorithm responses over the parameters. In addition, it utilizes the model to select the configurations for evaluating the target algorithm. The model aids in interpolating and extrapolating the target algorithm performance between the observed and unseen parameter settings respectively. This regression model is called response surface model [44]. In addition, the model quantifies

the parameter interaction and significance of each parameter. Therefore, the model guides the search towards efficient parameter configurations.

3.2.1 Sequential Model-based Optimization (SMBO)

A special type of black-box optimization task can be visualized to be an Algorithm Configuration Problem (ACP). Algorithm configuration carried out for a single instance of a deterministic algorithm is called deterministic Black-Box Optimization (BBO) [45]. In contrast, parameter configurations in a distribution of infinite configurations is stochastic BBO [45]. ACP is mostly a discrete problem because the parameter configurations of target algorithms are rarely continuous. The cost metric in ACP is a mixture of the parameter configuration, seed for instance selection and instances. However, in BBO the cost distribution is a function of the parameter configuration of a single instance of the problem. The algorithm configuration runtime of each instance is different depending on the parameter configuration. In addition, ACP employs a cut-off time for each instance to terminate an instance running for a long time. In contrast, BBO assumes a fixed time for evaluation of each problem [45].

SMBO is developed for BBO. However, SMBO has been extended with certain modifications to perform algorithm configuration. SMBO refers to all the common approaches utilizing a stochastic or noise-free Gaussian Process (GP) regression model assuming noisy or noise-free responses for the parameters from the target algorithm. Design and Analysis of Computer Experiments (DACE) is the GP noise-free model constructed using the empirical cost response of the configurations evaluated [45]. The next promising configuration is given by the model. This configuration is evaluated on the target algorithm. The next query point depends on the model distribution and the expectation of the positive improvement of a particular configuration with regard to the present best configuration. This criterion is called expected improvement (EI). The algorithm combining DACE and EI is called Efficient Global Optimization (EGO) [44].

The applications of SMBO include configuring evolutionary algorithms, SAT solvers, mixed integer programming, timetable scheduling, and hyperparameter optimization.

3.2.1.1 Sequential Parameter Optimization (SPO)

Cannot handle complex conditional parameters. SPO is a framework providing fully automated and interactive procedures to understand and optimize a parameter setting problem. In addition, the Sequential Parameter Optimization Toolbox (SPOT) incorporates fully automated SMBO. SPO is a model-based BBO technique based on DACE [7].

The different configurations for estimating the best configuration for a target algorithm is selected from a Latin Hypercube Design (LHD). The hypercube is obtained from several training instances. The cost metric is estimated for every single assessment of the target algorithm over the different configurations. SPO develops a regression model based on noise-free GP. The next configuration is chosen based on the model predictions and uncertainties of the predictions. SPO utilize the metric expected improvement to determine the next configuration. The response surface model is constructed after each evaluation of the target algorithm.

SPO+ [47], a more robust version of SPO performs better on standard benchmarks. The process of accepting the incumbent serves as the primary difference between both methods. The Time-Bounded Sequential Parameter Optimization (TB-SPO) is another variant of SPO with changes in the initial model construction procedure [50].

The time complexity of SPO is $O(n^3)$ with cubic run-time on the number of unique response values. It impose a large computational overhead. The algorithm configuration is limited to continuous parameters and single instance optimization.

3.2.1.2 Sequential Kriging Optimization (SKO)

A variant of EGO method. The noisy measurements from the target algorithm evaluations is modelled directly using a GP model. Similar to SPO, a LHD is utilized to sample the parameters for evaluation. The next query point is obtained by minimizing mean and variance expression of the GP model by Nelder-Mead Simplex algorithm. The time complexity is computationally expensive ranging to cubic with respect to the response value count for each iteration. It is applicable for continuous and single instance. SKO does not terminate poorly performing configurations and supports only numerical parameters.

3.2.2 Sequential Model-based Algorithm Configuration (SMAC)

SMAC is a model-based algorithm configuration framework primarily based on Bayesian Optimization (BO) by Frank Hutter, et al., [51]. SMAC is the current state-of-the-art framework for AAC. It incorporates racing mechanisms and a RF surrogate model to chose between the configurations for next evaluation. The availability of random forests surrogate model aids SMAC to effectively handle conditional and categorical parameters. A detailed explanation about the SMAC framework will be discussed in Chapter 4.4.2.3.

The use of random forests surrogate model results in a non-differentiable and discontinuous response surface for optimization. This necessitates the need for local and random search for optimizing the acquisition function. In addition, the prediction uncertainty is provided by the empirical variance of the ensemble tree predictions [83]. The applications of SMAC includes hyperparameter optimization, and algorithm configuration of mixed integer programming and boolean satisfiability problems.

3.2.3 Gender-based Genetic Algorithm++ (GGA++)

In 2015, the idea of combining the effectiveness of model-based techniques with model-free genetic algorithm framework is explored. GGA++ method significantly [3] improves the effectiveness of genetic algorithms based algorithm configurators. GGA++ employ a surrogate model incorporating genetic engineering and sexual selection. The offspring for evaluation is identified on comparing different surrogate models. A random forest surrogate model performs better for genetic algorithm based methods. It aids in identifying better configurations at a relatively faster and effective manner [19] [3].

3.3 Summary

The algorithm configuration approaches majorly fall into two paradigms- model-based and model-free. The major disadvantages of the model-free methods is the requirement of numerous target algorithm evaluations and provision for only single instance optimization. In addition, the advantages of the model-based approaches are enumerated below.

1. Only promising parameter configurations based on the prior knowledge are evaluated on the target algorithm.

2. Multi-instance optimization is possible with the inclusion of instance features in model construction.

3. Extrapolate and interpolate the performance between the unseen and evaluated configurations respectively [45].

4. Predict the performance of the configurations on different instances of the problem [45].

5. Performs relatively lesser evaluations of the target algorithm [45].

6. Provides a quantification of the parameter importance and interaction [45].

7. Provides a quantification of the parameter configuration and instance features interaction [45].

The disadvantages of model-based approaches are the performance is dependent on the surrograte model hyperparameter setting and complex development cycle [45]. The following table 3.1 provides a comparison of the limitations of various state-of-the-art algorithms. One of the state-of-the-art algorithms ParamILS [48] handles numerical discrete parameters effectively and requires discretization of the continuous parameters. The results largely depend on the effectiveness of generating discrete parameters. Similarly, GGA++ struggles to manage complex conditional parameters.

SMAC outperforms other algorithm configuration procedures in configuration scenarios with categorical inputs. The reason is the ability of random forest to efficiently handle categorical inputs relative to other surrogate models- GP, Tree Parzen Estimator, etc., [45]. In addition, the decision tree based surrogate model is effective in handling conditional parameters of the problem constructing more precise model specific to the problem. A comparison between GGA, ParamILS and SMAC for configuring a complex SAT solver illustrates the effectiveness of SMAC in algorithm configuration [52]. In this research work, there are 6 categorical and 7 conditional parameters. The need to automatically configure the categorical and conditional parameters is the prime reason for selecting SMAC. This research work focuses on implementing SMAC for the task of smart coupling in multiphysics simulations.

Method	**Model paradigm**	**Limitations**
Hill-climbing [41]	Model-free	a. Restricted to configure 5 parameters. b. Search is stuck at local optima, saddle point and plateaus.
Local beam search [71]	Model-free	a. Non-optimal search method. b. Incomplete search.
Experimental design with gradient descent [24]	Model-free	a. Applicable to only numerical parameters. b. Slow convergence. c. Expensive to estimate the gradient.
Experimental design with local search [1]	Model-free	a. Supports only 5 parameters. b. Restricted to ordinal and numerical parameters.
GA [87]	Model-free	a. Numerous evaluations of target algorithm. b. Only single instance optimization. c. Additional effort for fitness function evaluation.
GGA [2]	Model-free	a. Less efficient compared to local search variants. b. Cannot handle complex conditional parameters. c. Additional effort for fitness function evaluation.
F-Race [10] and IF-Race [4]	Model-free	a. Numerous evaluations of target algorithm. b. Cannot handle complex conditional parameters.
ParamILS [48]	Model-free	a. Supports only discrete parameters. b. Requires additional discretization.
SPO and SKO [45]	Model-based	a. Time complexity is high. b. Only for numerical parameters. c. Cannot terminate poor runs. d. Only single instance optimization.
GGA++ [3]	Model-based	a. Cannot handle complex conditional parameters. b. Additional effort for fitness function evaluation. c. Only single instance optimization.
SMAC [51]	Model-based	a. RF does not explore the search space completely for 1D optimization problem [45].

Table 3.1: Limitations of related work.

4

Methodology

This chapter concentrates on the methodology incorporated to automatically configure the coupling tool of a multiphysics simulation with optimal parameters. The study focuses on Fluid-Structure Interaction (FSI), transient study multiphysics simulations. The chapter begins with the motivation for using Automated Algorithm Configuration (AAC) in the approach, an overview of the proposed solution, followed by an extensive description of the various components of AAC. In addition, the chapter encompasses a brief description of Bayesian optimization and a flowchart of Sequential Model-based Algorithm Configuration (SMAC).

4.1 Proposed strategy

The research work aims to automatically configure the coupling tool, MpCCI, with optimal parameters to reduce the run-time of the simulation. The process of configuring the coupling tool is an online prediction of optimal configuration for a particular simulation feature set provided by the user. The intuition is employing a machine learning model to predict the optimal configurations. To train the model (offline task) and predict the optimal configuration for a given feature set, the knowledge of the best configuration of feature set is imperative. The optimal configurations for a particular feature set are identified by black-box optimization of each feature set. The types of parameters involved in the black-box optimization task are categorical, numerical, and conditional parameters. In addition, a forbidden set of parameter combinations are available. The problem of optimizing a black-box function with various types of parameters is the reason for employing SMAC, an AAC procedure. The complete system configuring the parameters of the coupling tool with optimal values is an AAC procedure. This formulates the research work under the branch of AAC.

The proposed solution is the pipeline illustrated in figure 4.1 for configuring the coupling tool with the optimal parameter configuration with relatively lesser run-time

than the current default configurations used for simulations. The pipeline is divided into two parts- offline and online. The offline part is the development phase starting from developing simulation problems to generate dataset followed by training machine learning models. The online part depicts the idea of the smart coupling tool. The smart coupling tool incorporates the machine learning models trained during the offline development phase to suggest the optimal parameter configuration to the user provided simulation problem.

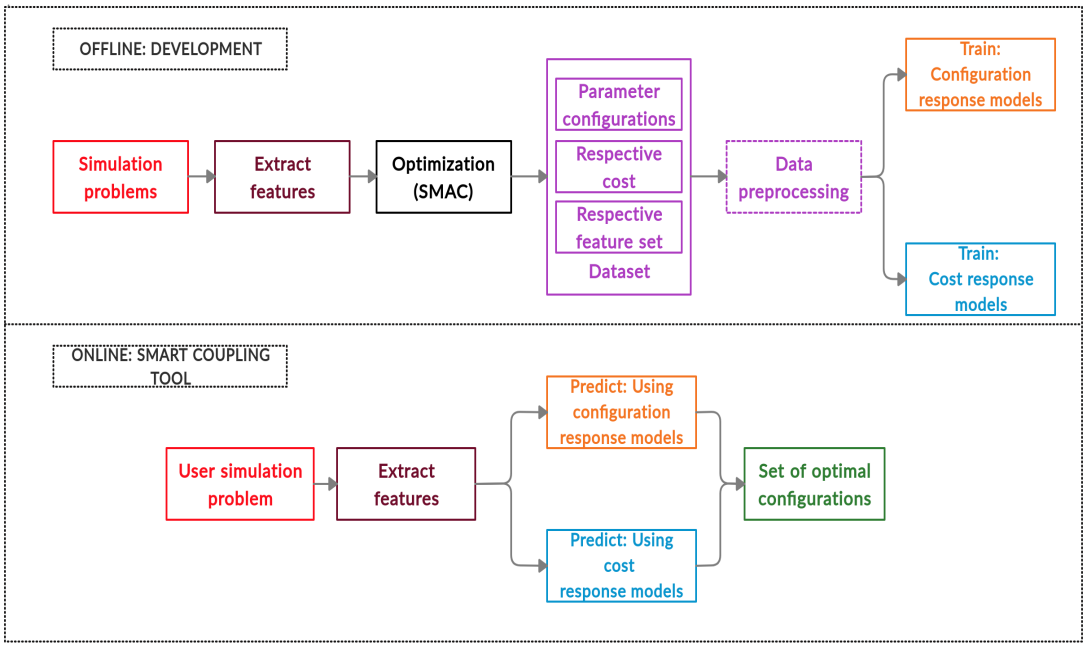


Figure 4.1: Overview of the proposed strategy to predict optimal configuration. Two parts of the pipeline- development and smart coupling tool. The inputs (red) are the FSI simulation problem to the respective parts of the pipeline. The extract features block (brown) extracts the features of the simulation instances. The dataset (purple) is the multiphysics simulation data set generated by single simulation instance optimization. The optimization is performed by SMAC (black). The data pre-processing stage (dashed purple) performs data cleaning and normalization for the training process. Two types of models- configuration response (orange), cost response (blue) are trained. Finally, a set of desirable optimal configurations (green) are predicted. The color resemblances between the online and offline parts of the pipeline illustrate the usage of the same machine learning model, similar feature extraction method and similar simulation problem type.

Firstly, a set of simulation instances across different simulation problems with a different feature set spanning the domain of multiphysics simulation is selected. Owing

to the time constraint and primary need to test the research hypothesis (applicability of AAC approach for automatically configuring the coupling tool), the number of simulation problems is restricted to two.

Secondly, a set of features representing the simulation instance has been extracted using the feature reader module 4.4.1. The features are used to enrich the RF surrogate model of SMAC. A surrogate model approximately mimics the behaviour of the coupling tool using the data collected on evaluating the simulation instances with different configuration settings. The features aid the model in learning about the instance. However, SMAC optimizes the simulation instance by performing the actual simulation with different valid parameter configurations. Therefore, the output of SMAC is different parameter configurations SMAC evaluated on the instances with the respective cost and the respective feature set of the simulation instance. This set of outputs from SMAC serves as the dataset from the next phase of training machine learning models.

The models has been trained in two different approaches- Configuration response and Cost response models to predict the optimal configurations. In the online prediction phase, the trained models are used for inference. The online prediction will be integrated to the coupling tool in future. In this research project, the complete pipeline- optimization phase, training phase and prediction phase has been completed.

4.2 Hyperparameters of the coupling tool

The coupling tool, MpCCI performs a multiphysics simulations by solving the governing equations of the physical domain. Therefore, the entire problem is viewed to be solving system of equations. The governing equations concerning the solid domain are solved by the structural solvers (structural code) such as Abaqus, ANSYS, Calculix, and Marc. The fluid domain governing equations are solved by fluid solvers (fluid code) such as OpenFOAM, FineOpen and ANSYS FLUENT. The fluid flow is described by the Navier-Stokes equations given in [38]. Similarly, the solid mechanics is described by the stress, strain tensors, and Hooke's law given in [38]. The equations are iteratively solved. In a transient problem, a single time step includes several iterations until a particular time step converges for both solvers.

The hyperparameters of the coupling tools performing FSI transient study simulations are coupling scheme, initial exchange, relaxation_0, relaxation_1, relaxation method, relaxation factor, Andersonmix type, number of levels, omega, ramp_0 and ramp_d. The parameters of the coupling tool largely impact the accuracy, stability and run-time of the simulations. This section provides a description of the hyperparameters configured in this research work.

4.2.1 Coupling scheme

There are two coupling schemes available for co-simulation, namely, explicit and implicit. Explicit (loose or weak) coupling involves the computation of the solution directly from the quantities known. At a particular time step, the solid and fluid code exchanges information once in explicit coupling. To preserve the equilibrium between the solid and fluid solvers at every time step and obtain precise results, implicit coupling is used. Implicit coupling includes solving a system of coupled equation in an iterative procedure. Each time step of the solvers is carried out until convergence or maximum iterations. The data exchange between the solvers occur at each iteration of the time step [84].

4.2.2 Initial exchange

The parameter initial exchange in MpCCI determines the coupling algorithm and the data exchange pattern between codes. The possible initial exchange settings are:

- receive:exchange - The simulation code A and B in figure 4.2 is initialized to perform receive and exchange actions, respectively. The code A receive data from the partner code B, shown by encircled 1 (step 1). Once it receives data from B, A performs computation in step 2. The exchange action comprise send and/or receive action. The code B sends data at step 1 and receives data at step 3 after A completes the computation. The code B performs computation in step 4 once it receive the data from A. Finally, step 5 is similar to step 1. This coupling algorithm is serial coupling as illustrated in figure 4.2.

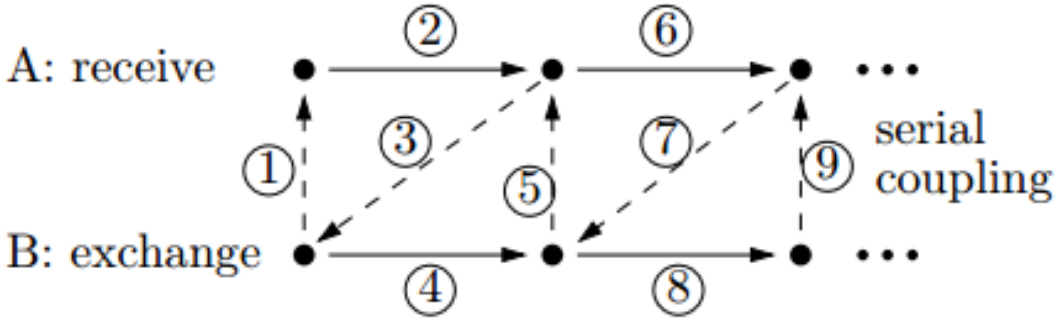


Figure 4.2: Initial exchange setting- receive:exchange [35].

- exchange:receive - This setting is used in serial coupling algorithm. The simulation code A and B is initialized to perform exchange and receive actions, respectively.
- exchange:exchange - It is used in parallel coupling. The simulation code A and B in figure 4.3 is initialized to perform exchange action. At step 1, A and B exchange data simultaneously. Then in step 2, A and B performs computations in a parallel manner followed by simultaneous data exchange (step 3).

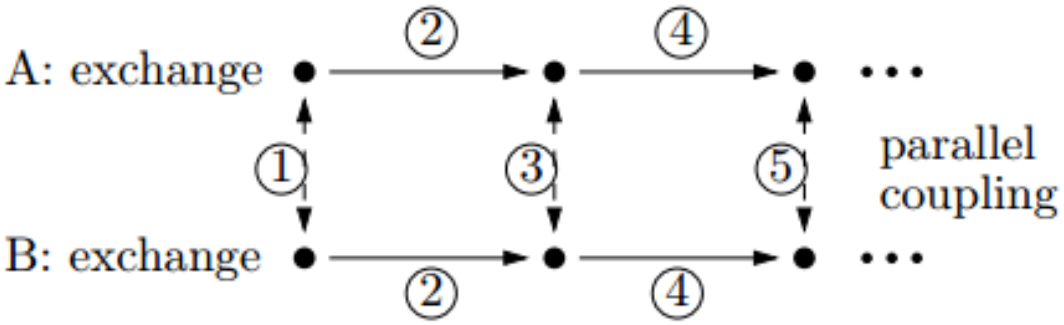


Figure 4.3: Initial exchange setting- exchange:exchange [35].

4.2.3 Relaxation method

Relaxation method is an iterative procedure for solving system of sparse linear equations and non-linear equations [78]. The sparse linear system of equations are obtained from the discretizations of differential equations by finite-differencing. Relaxation methods are used in multiphysics simulations to stabilize and increase the convergence speed of a non-linear coupled process. The 4 relaxation methods available in MpCCI are:

- Fixed
- Ramping
- Aitken
- Quasi-Newton

Anderson mixing, ramp_0, ramp_d, and omega are various child parameters associated with different relaxation methods. A detailed explanation on the relaxation methods is provided in [35].

4.3 Pivotal problems of the simulation instances

The two FSI simulation problems namely '3D Driven cavity' (refer section 4.3.1) and 'Elastic flap in a duct' (refer section 4.3.2) are chosen to generate various simulation instances with different features.

This particular type of simulation problems are chosen because of two reasons. First, the problems are different from each other aiding in generating different set of features for a FSI problem. Secondly, the problems are benchmarks in FSI simulations [62].

4.3.1 Driven cavity

The simulation problem includes a 3D cavity with an interaction between a movable bottom and an incompressible fluid. The driven cavity is a study involving a cube with all edges measuring one meter. A cross section of the driven cavity is shown in 4.4. The top surface of the wall is exposed to an oscillation $u_x(t)$ and the side walls are stationary. The bottom surface is flexible and not fixed. This is solved by an iterative transient coupling with Quasi-Newton method. The software packages ABAQUS and OpenFOAM are used to solve the solid structural and fluid physics of the problem respectively. In this study, MpCCI software package couples the solid and fluid solver [35] [60].

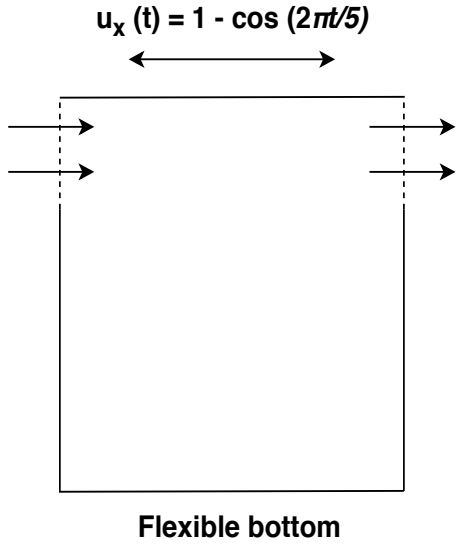


Figure 4.4: Cross section of driven cavity simulation problem with movable bottom wall [60].

A cube with dimensions 1 x 1 x 1 meter is the fluid domain of driven cavity. The top surface of the quadratic fluid domain is vertically oscillating a frequency of 0.2 Hertz. The

cavity has an inlet and outlet on the top left and top right, respectively, each of width 0.2 meter as illustrated in figure 4.4. The study considers the fluid to be an incompressible fluid with constant density and viscosity [35].

The flexible wall is the solid part of the simulation model. The left and right end of the wall is fixed and the wall thickness is 0.002 meters. The results are evaluated at the middle of the bottom flexible wall. The material is considered linearly elastic. The top of the flexible wall aids in exchanging quantities between the solid and fluid domain [35].

4.3.2 Elastic flap in a duct

The simulation model includes a rectangular area for fluid domain and the solid structural domain includes in elastic flap. ABAQUS and OpenFOAM are the solid solver and fluid solver respectively used for this simulation problem. The coupling is mediated by MpCCI software package. The geometry of the simulation problem is provided in figure 4.5.

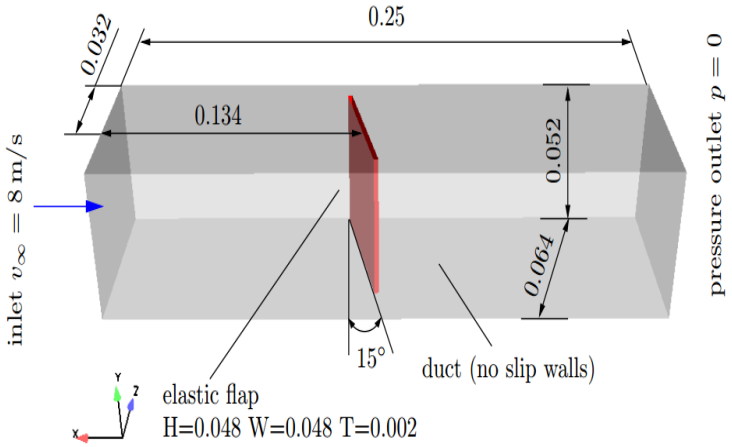


Figure 4.5: Geometry and boundary conditions of elactic flap in a duct simulation model [35]. The dimensions are provided in meters. The innermost bottom corner of the elastic flap is the control node for computing the results

The entire cuboid area excluding the elastic flap corresponds to the fluid domain. An incremental time step of 0.25 millisecond is used by the computational fluid dynamics-fluid solver code.

The red colored rectangle corresponds to the solid surface. The elastic flap is a linearly elastic material. The solid has been divided into mesh grids for computation with 20

nodes for each mesh. The top surface nodes are stationary. An adaptive time step is used with an incremental time step of 0.25 millisecond up to 0.2 seconds. The innermost bottom corner of the elastic flap is the control node for computing the results. In this problem, the quantity time step is coupled in the coupling region.

4.3.3 Significance of hyperparameters

An extensive description on the steps involved in designing and conducting the simulation is provided in the MpCCI manual- chapter Tutorial [35]. In order to explain the importance of the hyperparameter involved in the research project, 3D driven cavity simulation problem is considered. On performing the simulation using MpCCI, the moving surface at the top results in an oscillating bottom wall.

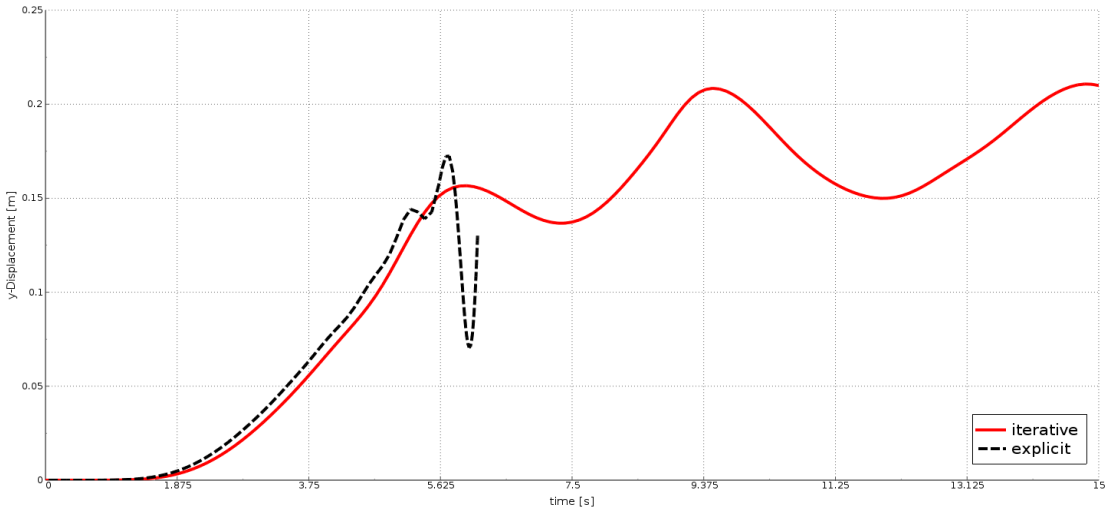


Figure 4.6: Impact of 'Coupling scheme' hyperparameter in driven cavity- Iterative transient coupling versus explicit coupling for 3D driven cavity [35]. The simulation will fail on configuring the coupling scheme with explicit coupling.

The figure 4.6 illustrates on solving the driven cavity problem using explicit coupling scheme diverges the solution. However, the iterative implicit coupling leads to convergence and stable oscillations [35].

The iterative transient coupling scheme solves the simulation in 20.5 hours. On testing the effect of Quasi-Newton relaxation method, the time factor is reduced by seven and the oscillations of each iteration is eliminated as illustrated in figure 4.7 [35].

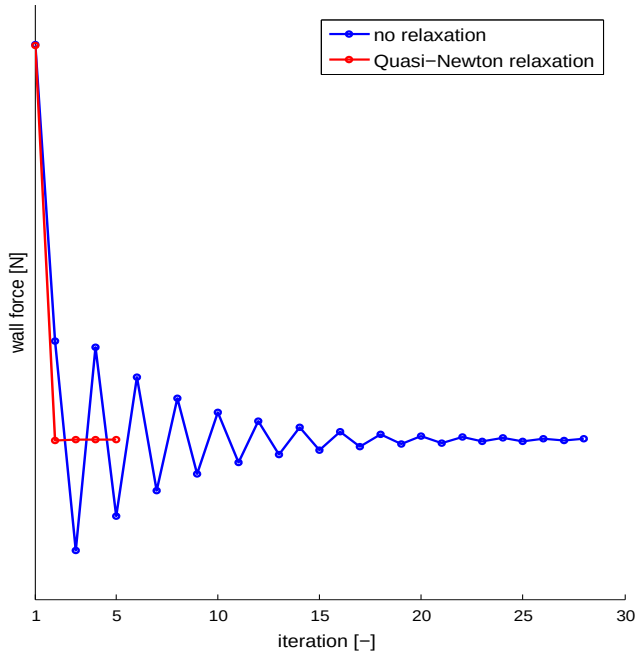


Figure 4.7: Effectiveness of Quasi-Newton relaxation method for 3D-driven cavity [35].

4.4 Smart coupling tool

The research work contributes to the engineers and researchers performing simulation studies by providing two significant implementations- smart coupling tool and feature reader tool.

Smart coupling is the process of automatically configuring the coupling tool using algorithm configuration methods. The development of the smart coupling tool is divided into 4 phases- Optimization 4.4.2, data pre-processing 4.4.3, training 4.4.4 and prediction phase 4.4.5.

The feature reader tool is a separate python-based package. It is used by the smart coupling tool to extract features from the simulation model files, coupling tool input files, and output files. The appendix D provides the step-by-step instructions to use the smart coupling tool and feature reader package.

4.4.1 Feature reader

The features of the simulation problem integrate the characteristics of the following components of the simulation,

- The simulation models- solid and fluid model.
- The coupling method.

The characteristics of the simulation models include – density of the fluid, the density of the solid structure, elasticity of the solid, viscosity of the fluid, Poisson's ratio, and the ratio of solid-fluid density, and type of the simulation problem- stationary or transient study. The characteristics of the coupling method include the coupling type, time step, number of nodes in the coupling region, tolerance of each time step, and the time taken for the co-simulation.

The feature reader package is used to extract approximately more than 100 features covering the entire domain of FSI simulations. This package requires the input simulation code files, coupling code files, and the coupling result-log file of the simulation process. The smart coupling tool utilizes feature reader to enrich the surrogate model by providing the features of the simulation instances.

4.4.2 Optimization phase based on SMAC

The integral part of the optimization phase is Sequential Model-based Algorithm Configuration (SMAC). SMAC is a variant of Bayesian optimization. This section explains the optimization phase elements. The figure 4.8 illustrates the optimization phase. The components of the optimization phase are provided below.

- Inputs to SMAC: The inputs include the simulation instances and features of the instances.
- Configuration space of the MpCCI configurable parameters. The description of the parameters are provided in section 4.2.
- Outputs of SMAC: The various allowable configurations to perform coupling simulations, cost (runtime in seconds) on executing the target algorithm with the respective configuration, and the features of the simulation instances. The outputs of SMAC comprise the dataset for training the machine learning models to predict the optimal configurations.

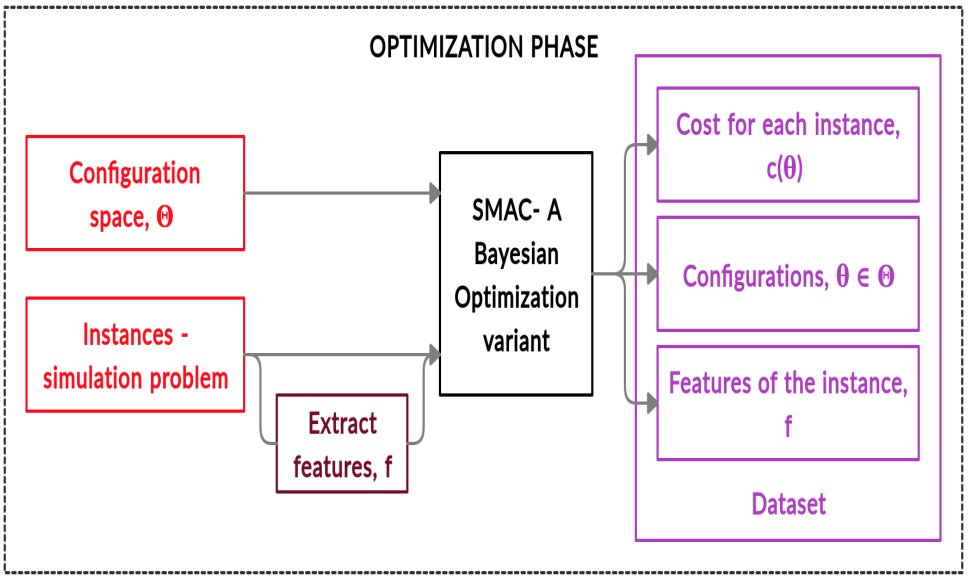


Figure 4.8: Block diagram of optimization phase. The inputs (red) are the configuration space and various simulation instances. The feature is extracted using the feature reader tool (brown). SMAC (black) performs single instance optimization of the simulation problems resulting in a multiphysics simulation dataset (purple).

4.4.2.1 Simulation instances

A simulation instance is a particular occurrence, case, or example of a simulation problem. In this research, various simulation instances are developed by varying the solid and fluid model features of a particular simulation problem. The simulation problems taken into consideration are the 3d driven cavity and elastic flap. The simulation problems are discussed in section 4.3. A total of 19 instances are developed by varying the features, namely- elasticity, solid density, Poisson's ratio, fluid density, and fluid viscosity. The features of the instances are provided in tables A.1 and A.2. The features are used to enhance the RF surrogate model of SMAC. However, SMAC optimizes the simulation instances considering the instances as a black-box.

4.4.2.2 Configuration space of the parameters

The configuration space, Θ is the vector space spanned by the parameters defining the states of a target algorithm, $\mathcal{A}$. The target algorithm in this research work is the coupling tool, MpCCI. A parameter configuration, $\theta \in \Theta$, is evaluated on a particular instance 'i'

of the target algorithm resulting in $\mathcal{A}(\theta)$. Each target algorithm evaluation is associated with a performance metric, 'm'. This metric characterizes the behavior of the target algorithm for a particular pair of parameter configuration and instance.

Parameter, P	**Parameter type**	**Domain of the parameter, D**
Coupling scheme	Categorical	Implicit-Transient:Implicit-Transient, Explicit-Transient:Explicit-Transient
Initial exchange	Categorical	exchange:exchange, receive:exchange, exchange:receive
Relaxation-0	Categorical	True, False
Relaxation-1	Categorical	True, False
Relaxation method	Categorical	Quasi-Newton, Fixed, Aitken, Ramping
Anderson mix	Categorical	Least squares, Standard, Inverse
Number of levels	Integer	[0, 16]
Omega	Float	[1e-07, 2.0]
Ramp-0	Float	[0.0, 2.0]
Ramp-d	Float	[0.001,1.0]
Relaxation factor	Float	[0.0,2.0]

Table 4.1: Hyperparameters of the coupling tool.

This section provides the parameter involved in the study, type of the parameters, domain of the parameter, and the conditional dependencies among the parameters. The following table 4.1 summarizes the various parameters used in the study, the domain of the parameters, and the parameter types. The domain of a parameter is the set of acceptable values for a parameter [45]. An overview of the various data types of parameters is provided in section 2.2.2. In FSI, QN is the most efficient and robust relaxation method for partitioned coupling [82] [27]. The following default parameter values are used in MpCCI for QN based problems.

The conditional parameter clauses define the conditional parameters. The two components of a conditional parameter clause are the child parameter and a logical condition with parent parameters. On the parent parameter satisfying the logical condition, the child parameter is considered for parameter setting [45]. The table 4.3 summarize the conditional parameters in this study.

Furthermore, the forbidden clauses in SMAC allow specifying the restricted parameter configurations. Condition 1 and condition 2 is a pair of invalid parameter configuration. The configurator does not evaluate the target algorithm with the forbidden configurations provided in table 4.4.

The nature of a configuration space of the parameters, Θ is primarily determined by the number of parameters, parameter types, the domain of the parameters, constraints of the parameter configurations, and the various conditional parameters involved in the

Parameter, P	**Default value**
Coupling scheme	Implicit-Transient:Implicit-Transient
Initial exchange	receive:exchange
Relaxation_0	False
Relaxation_1	True
Relaxation method	Quasi-Newton
Andersonmix type	Inverse
Number of levels	1
Omega	0.1
Ramp_0	0.1
Ramp_d	0.1
Relaxation factor	0.1

Table 4.2: Default configuration majorly used in MpCCI.

Conditional Clause	**Child parameter**	**Parent logical condition**
1	Relaxation method	Relaxation_0 == True || Relaxation_1 == True
2	Andersonmix type	Relaxation method == Quasi-Newton
3	Number of levels	Relaxation method == Quasi-Newton
4	Omega	Relaxation method == Quasi-Newton
5	Relaxation factor	Relaxation method == Fixed
6	Ramp_0	Relaxation method == Ramping
7	Ramp_d	Relaxation method == Ramping

Table 4.3: Conditional parameters of the study.

Forbidden Clause	Condition 1	Condition 2
1	Relaxation_0 == True	Coupling scheme == Explicit-Transient: Explicit-Transient
2	Relaxation_1 == True	Coupling scheme == Explicit-Transient: Explicit-Transient
3	Initial exchange == exchange:exchange	Relaxation_0 == True && Relaxation_1 == True
4	Initial exchange == exchange:exchange	Relaxation_0 == False && Relaxation_1 == True
5	Initial exchange == exchange:exchange	Relaxation_0 == True && Relaxation_1 == False

Table 4.4: Forbidden pairs of parameter configuration.

algorithm configuration problem. The algorithm for searching the optimal configurations are decided depending on the configuration space of the parameters. An algorithm configuration procedure, SMAC performs the process of exploring the configuration space for optimal parameter configuration of a given instance [45].

4.4.2.3 Single instance optimization by SMAC

SMAC is a variant of BO with the RF surrogate model. The regression model to fit the sample points is a RF regressor. The major contributions of SMAC are eliminating the two major disadvantages of Sequential Model-Based Optimization (SMBO) discussed in section 3.2, namely the restriction of SMBO for target algorithms with numerical parameters and single instance optimization of the target algorithm [51].

SMAC utilizes RF for modeling the objective function. The objective function or target algorithm is MpCCI in this research. RF handles categorical parameters efficiently [45] by providing an increased performance because of the set of decision trees in the forest [51] [13]. RF incorporates a group of regression trees predicting the performance metric of MpCCI. The regression tree prediction depends on the knowledge of the evaluations at a particular point in the optimization task.

The second significant advantage of SMAC is the ability to handle multiple instances. The set of parameter configurations, the respective performance metric obtained by evaluation of the parameter configuration on instance i, and the respective feature f of the instance i, is used to train the RF model.

AutoML Group, University of Freiburg, provides the algorithm configuration tool-

SMAC3 [65]. An extended algorithm of the smart coupling tool integrating the algorithm of SMAC is provided in algorithm 2.

4.4.3 Data pre-processing

The data pre-processing focus on the transformation of the cost metric, followed by normalization with the minimum cost of the simulation instance.

The cost metric of the optimization task is the runtime (in seconds). The runtime provides large variations in the dataset. The various previous works on runtime prediction recommend logarithmic transformation of the runtime because logarithmic transformation aids in runtime prediction by reducing the dynamic range of the runtime [51] [53]. In addition to logarithmic transformation in equation 4.1, the results are compared using scaled-logarithmic (equation 4.2) and scaled-inverse (equation 4.3) transformations. All the transformations are adapted from [65].

Logarithmic transformation:

$$Tc(\theta_i, f_j) = \ln\left(c(\theta_i, f_j)\right) \tag{4.1}$$

Scaled-logarithmic transformation:

$$Tc(\theta_i, f_j) = \ln\left(\frac{c(\theta_i, f_j) - min[c(\theta, f_j)] + \epsilon}{max[c(\theta, f_j)] - min[c(\theta, f_j)] + \epsilon}\right) \tag{4.2}$$

Scaled-inverse transformation:

$$Tc(\theta_i, f_j) = 1 - \left(\frac{c(\theta_i, f_j) - max[c(\theta, f_j)]}{c(\theta_i, f_j) - min[c(\theta, f_j)] + \epsilon}\right)^{-1} \tag{4.3}$$

c(θ_i, f_j) and Tc(θ_i, f_j) are the cost metric for the configuration θ_i of a particular simulation instance j with feature set f_j before and after transformation, respectively. min[c(θ, f_j)] and max[c(θ, f_j)] is the minimum and maximum cost respectively for a particular simulation instance j with the feature set f_j over all the parameter configurations. The variables i and j ranges up to the total iterations of SMAC and total instances in the training set, respectively. ϵ, a very small constant, is added to the minimum of the cost vector for a particular instance to avoid division by zero issues.

The simulation instances in table A.1 differ by the features. The different features cause variations in the simulation time for different parameter configurations of the coupling tool. The transformed cost of each simulation instance is normalized with

the minimum cost of a particular simulation instance to make the features comparable concerning the cost metric.

$$c_n(\theta_i, f_j) = \frac{Tc(\theta_i, f_j)}{min[Tc(\theta, f_j)]} \tag{4.4}$$

Tc(θ_i, f_j) and $c_n(\theta_i, f_j)$ are the transformed and normalized cost, respectively, for the configuration θ_i of a particular simulation instance j with feature set f_j. min[Tc(θ, f_j)] is the minimum of the transformed cost for a particular simulation instance j with the feature set f_j over all the parameter configurations.

4.4.4 Training phase

At the end of the optimization phase, the dataset for training machine learning models to predict optimal configurations is available. The optimal configurations are predicted using two different models, namely configuration and cost response models. The labels of the configuration and cost response models are the parameter configurations and costs, respectively. The figure 4.9 and figure 4.10 illustrates the pre-processing, training and prediction steps of a configuration and cost response model, respectively.

In configuration response models, the machine learning models- RF, KNN, SVM, GBM, and QRF are trained to predict each of the configurations performing multi-target regression. The method is implemented using the sklearn API [15] employing one regressor per target variable. The cost metric is transformed and normalized with the minimum cost similar to the process described in section 4.4.3.

In cost response models, the machine learning models- RF, KNN, SVM, GBM, and QRF are trained to predict the cost metric of the simulation instance. The configurations and features of the simulation instance in the dataset are the features to train the model. The transformed and normalized cost is the label. In addition to the above-specified models, RF is trained to predict the expected improvement metric of a particular configuration. It is included in the cost response models because EI indicates the utility of a configuration. RF-EI indicates the RF model predicting expected improvement in the upcoming sections.

4.4.5 Prediction phase

The prediction procedure of configuration and cost response models are illustrated in figures 4.9 and 4.10. The prediction phase of both response models incorporates the feature reader tool to extract features from the user-provided simulation problem.

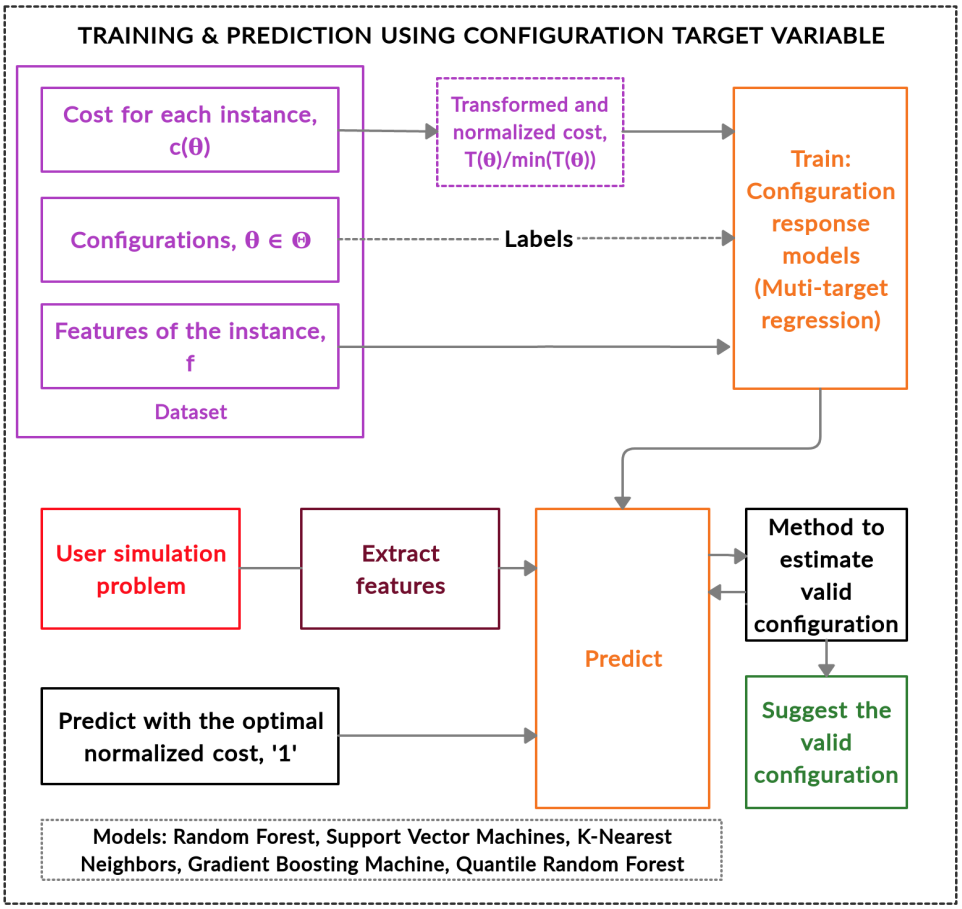


Figure 4.9: Illustration of the configuration response models (orange) training and prediction phase. The labels (dashed line) of configuration response models are the parameter configurations of the dataset (purple). The cost is transformed and normalized in the data pre-processing step (dashed purple) with the respective transformations discussed in 4.4.3. The prediction for the user given FSI simulation problem (red) occurs at optimal normalized cost value 1 because the optimal cost value for a particular feature set is 1 during training phase. The method after prediction check for valid configuration and predict with an incremented normalized cost value up to a threshold searching for valid predictions. The feature reader tool is used to extract features (brown) during prediction.

In configuration response models, the prediction begins with cost value 1 because the optimal cost pertaining to a particular feature set is 1. This is primarily due to the normalization with the minimum cost of a particular instance. The configuration is predicted as a vector of floating-point values for the user given feature set. The

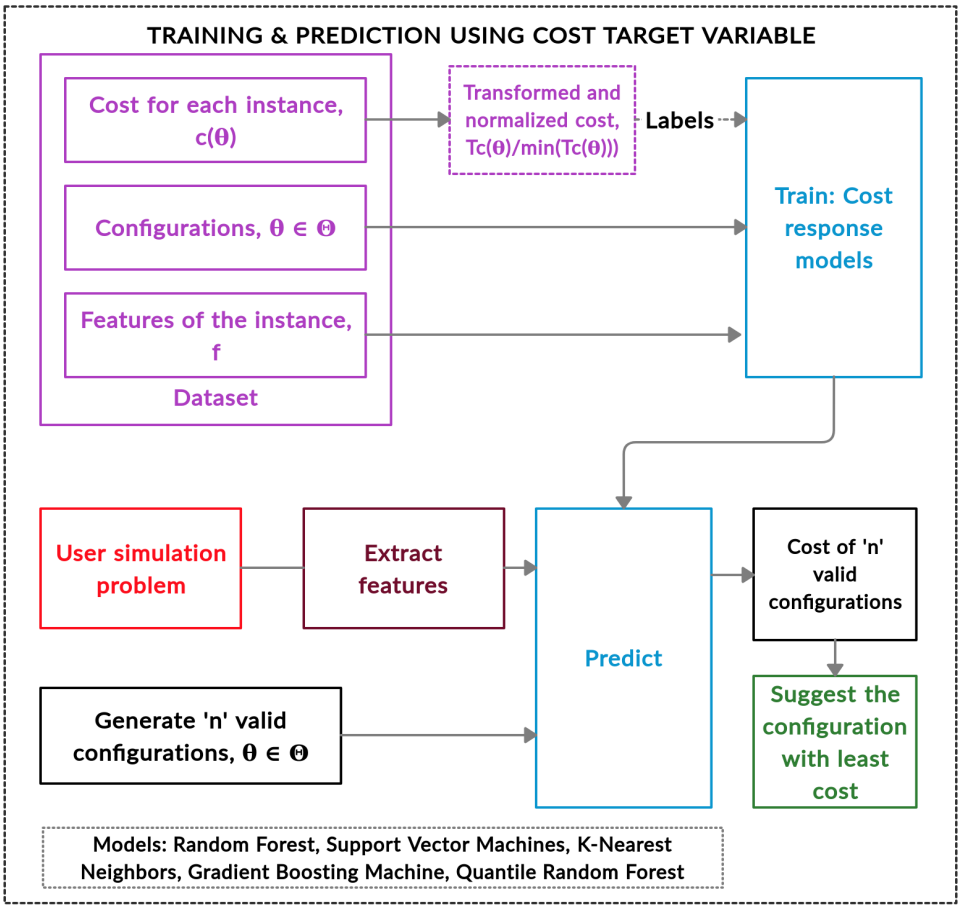


Figure 4.10: Illustration of the cost response models (blue) training and prediction phase. The labels (dashed line) of the cost response models are cost. The dataset (purple) is the multiphysics dataset developed by optimization. The cost is transformed and normalized in the data pre-processing step (dashed purple) with the respective transformations discussed in 4.4.3. The prediction for the user given FSI simulation problem (red) involves the generation of numerous valid configurations followed by predicting the configuration with desirable least cost (green) for a particular simulation instance. The feature reader tool is used to extract features (brown) during prediction.

floating-point predictions for categorical parameters are rounded to the nearest integer to map the predictions to the closest category. This configuration is not assured to be a valid configuration because of the conflict with the forbidden and conditional parameters. Therefore, a method predicts the configurations with an incremented cost value until the predictor estimates a valid configuration. This new incremented cost value is a sub-optimal cost. This is implemented in the method after the prediction block. Finally,

the valid configurations are suggested to the user.

In cost response models, the prediction phase incorporates the generation of numerous random valid configurations. The predictor predicts the cost metric for a particular feature provided by the user using all the random configurations generated. Finally, the parameter configuration with the least cost is proposed to the user.

The performance of each model is analyzed in the experiments and results section 5. Finally, the user is suggested with the top 3 configurations predicted by 3 models out of the 11 configuration and cost response models.

4.4.6 Workflow of the smart coupling tool developed

The workflow of the smart coupling tool revolves majorly around the working of SMAC. The working of SMAC is similar to the BO procedure with the gaussian surrogate model provided in the algorithm 1 with changes in the surrogate model training followed by configurations selection and intensification. The outputs from SMAC by single instance optimization is used for training and predicting the optimal configuration. The workflow of smart coupling is enumerated below.

1. **Define the inputs:** SMAC requires a set of training instances i, configuration space Θ, and features of the training instances x_i. Furthermore, the default configuration $\theta_{default}$ for all instances, the forbidden parameter configuration cases, conditional dependencies of the hyperparameters, the seed for generation of random configurations are provided.

2. **Evaluate the default configuration:** The optimization of each instance begins with the target algorithm evaluation using the default configuration. This can be altered depending on user requirements. The wrapper performing the evaluation estimates the performance metric associated with the evaluation, R_i. The default configuration is the current best configuration for the first iteration with the best cost $\theta_{i,inc}$. The best configuration at any point in the program is called incumbent configuration. At each iteration, $\theta_{i,inc}$, and the associated best cost, $c_{i,inc}$ is modified depending on the performance of the new configuration.

3. **Fit surrogate model:** The training set of the surrogate model includes the parameter configurations and the features specific of a particular instance to predict the cost. This joint model aids in learning the optimal configurations for a particular feature set of the problem. The mean across all the trees is the predicted

performance metric for a particular instance and parameter configuration. The variance across all the trees is the uncertainty measure associated with a particular prediction. The cost metric relating to the runtime of the simulation is transformed using logarithm transformation. This reduces the variance in the runtime and improves model performance in the regression task [51] [91].

4. **Select configurations:** The expected improvement score (EI) is estimated for randomly generated 5000 configurations depending on the surrogate model distribution. The top 50 configurations concerning EI scores are selected to be the challengers of the current best configuration. EI score is high for configurations with low cost metric and high uncertainty [51] [53].

   From the second iteration of SMAC, the algorithm performs local search initialized at each of the top 10 promising configurations depending on the surrogate model. It calculates EI for the neighbors of the 10 configurations until the EI score of all the neighbors is not increasing. The algorithm selects the top 10 among the neighbors. Furthermore, it generates 5000 configurations using the surrogate model distribution over the configurations. The EI score for all the configurations is screened to select top 50 challengers for intensification [51] [53].

5. **Intensification:** The method compares the best configuration performance from the newly selected configurations list, depending on the EI score against the existing best configuration. The intensification method performs a certain number of target algorithm evaluations for each configuration in the new configurations selected. The comparison continues until the exhaustion of a time budget called the intensification budget. The incumbent configuration is evaluated on a new instance randomly selected depending on a seed value. The (instance, seed) value is fixed for a particular intensification step allowing the evaluation of each new configuration in the same instance. The best configuration is replaced by the challenger configuration, depending on the performance on the randomly selected instance for a particular intensification step.

6. **Repeat:** The fit surrogate, select configuration, and intensification steps are continued for a specific time budget. The default scenario performs intensification for 10% of the total time budget. The budget is the wall-clock time or the iteration limit. Each iteration of SMAC performs a single direct evaluation of the algorithm.

7. **Normalization:** After optimizing each instance for the provided time budget, all the parameter configurations θ_i, the features x_i, and the cost c_i of each instance is

Symbol	Meaning
$\Theta_{default}$	Default configuration for the simulation instances.
x_i	Features of the simulation instance i.
R_i	Set of parameter configurations and respective costs for instance i.
$\theta_{i,inc}$	Incumbent configuration for instance i.
Θ_i	Configuration space for the simulation instance i.
$\Theta_{i,new}$	Challengers for the incumbent for instance i.
$\hat{c}_i$	Cost metrics of the simulation instance i.
m_i	RF surrogate model built using the R_i prior.
t_{fit}	Runtime to fit model.
t_{select}	Runtime to select challenger configurations.
θ_i	Set of all parameter configuration for simulation instance i.
c_{inc}	Incumbent Cost specific to a simulation instance.

Table 4.5: Symbols used in the smart coupling algorithm.

combined into a tuple. The tuple is denoted by 'R'. The tuple R across all instances is combined to form a dataset. The costs corresponding to the different parameter configurations of a simulation instance are normalized with the best cost c_{inc} of the simulation instance to compare the cost across different simulation instances.

8. **Train regressors:** The dataset is used to train regression models to predict the best configuration for the features of the simulation problem provided by the user.

The below pseudocode is the algorithm of SMAC for smart coupling. The modifications carried out in the SMAC algorithm for automatically configuring the coupling tool is highlighted in red color text. The non-highlighted text is completely adapted from SMAC [51] to illustrate the modification to SMAC for the smart coupling application. The table 4.5 provides the meaning of the symbols used in the smart coupling algorithm.

Algorithm 2: Smart coupling algorithm- An extension from SMAC

Input: Parameters, instances features (x) obtained from the solid domain models, fluid domain models and coupling properties.

Result: Regression model for prediction.

foreach *Instance* i **do**
  $[R_i, \theta_{i,inc}] \leftarrow$ Initialize($\theta_{default}$, x_i)
  repeat
    $[m_i, t_{fit}] \leftarrow$ FitModel(R_i)
    $\Theta_{i,new}$, $t_{select} \leftarrow$ SelectConfigurations(m_i, $\theta_{i,inc}$, Θ_i)
    R_i, $\theta_{i,inc} \leftarrow$ Intensify($\Theta_{i,new}$, $\theta_{i,inc}$, m_i, R_i, $t_{fit} + t_{select}$, x_i, $\hat{c}_i$)
  until *total time budget for configuration exhausted*
  return R_i
end
$R \leftarrow$ Collect $R_i(\theta_i, x_i, \hat{c}_i)$ of all instances
$R \leftarrow$ **Normalize_cost**($R(\theta, x, \hat{c})$, c_{inc})
Train regression model, m using R and respective features x.

The regression model, m, is used to predict the optimal parameter configuration given a simulation instance. The cost and configuration response models in section 4.4.4 are the regression models trained. The following chapter conducts experiments to assess the machine learning models and recommend the user with the top 3 configurations.

5

Experiments and Results

The experiments focus on testing the different parts of the proposed strategy to configure a simulation instance provided by the user with optimal parameter configuration. The final experiment, in section 5.6, evaluates the cost and configuration response models in section 4.4.4 on unseen simulation instances. The models are compared using the runtime of the configurations predicted by the machine learning models against the best cost from SMAC. Owing to the unavailability of benchmarks for comparison, the comparison against the best cost from SMAC is used. In addition, the comparison against default configurations is avoided because default configurations practically result in crashed simulation for a few unseen simulation instances. The top three models are suggested to the user among the cost and configuration response models. In addition, the configuration predicted by the models is evaluated in terms of runtime against the default configurations on a critical simulation instance to illustrate the performance of the proposed strategy in figure 5.6. The experiments are performed assuming the coupling tool MpCCI provides repeatable results for a particular configuration on a simulation instance. The experiments test the following aspects,

1. Experiment 1: Performance comparison of SMAC best configuration and default configuration using the simulation runtime (in seconds).

2. Experiment 2: Repeatability of SMAC to determine the optimal configuration for a particular simulation instance.

3. Experiment 3: Number of iterations required to achieve reliable results by SMAC.

4. Experiment 4: Ability of SMAC to find many good configurations for a particular simulation instance.

5. Experiment 5: Performance comparison of the cost and configuration response models using RMSE across different data transformations to the cost metric discussed

in section 4.4.3.

6. Experiment 6: Evaluate the cost and configuration response models using the metrics in section 5.6.2 on unseen simulation instances to suggest the user with optimal configurations.

5.1 SMAC best configuration versus default configuration

The experimental setup section provides the simulation instances used in the experiment, solid and fluid solver coupled in MpCCI, the reason for experimenting, and the metrics used in this experiment. The results and discussion section compare the performance of SMAC best and MpCCI default configurations.

5.1.1 Experimental setup

This experiment illustrates the ability of SMAC to determine optimal configurations performing better than the default configuration. The experiment compares the runtime of the best configuration[1] from SMAC against the current default configuration provided in table 4.2 to perform simulation by experts.

The experiment is conducted on 16 different simulation instances. The features of the simulation instances are shown in table A.1. SMAC performs optimization for 100 iterations on each instance. Each iteration execute the target algorithm once. Calculix and OpenFOAM are the solid and fluid solvers respectively used for solving 3D driven cavity simulation problem. ABAQUS and OpenFOAM are the solid and fluid solvers respectively used for solving the elastic flap simulation problem. The best configuration for a particular simulation instance is the configuration with the least runtime among the 100 iterations. In this experiment, the mean percentage decrease in runtime and win percentage of SMAC are useful metrics to estimate the performance of SMAC best configuration against the default configuration. The mean percentage decrease in runtime is given by,

$$\text{Mean \% decrease in runtime } = \frac{\text{Default cost} - \text{SMAC best cost}}{\text{Default cost}} \times 100 \tag{5.1}$$

The SMAC best cost and default cost is the simulation runtime on using the best configuration from SMAC and default configuration, respectively, expressed in seconds.

[1]Best configuration – This term is used with respect to SMAC. It is the configuration with the least runtime provided by SMAC for a particular instance.

The win percentage of SMAC is the number of times SMAC best configuration perform better than the default configuration to the total simulation instances under optimization expressed as a fraction of 100. The performance is compared using the simulation runtime of a particular configuration. The matches is the 16 simulation instances under comparison.

$$\text{Win percentage} = \frac{\text{Number of wins} + (0.5 \times \text{Number of draws})}{\text{Total number of matches}} \times 100 \tag{5.2}$$

The win percentage in the above equation 5.2 provides the win percentage of SMAC.

5.1.2 Results and discussion

The runtime of the best configuration from SMAC is compared against the runtime of the default configuration, as illustrated in table 5.1. The win percentage and mean percentage decrease in runtime is estimated.

Instance	**Default configuration cost (seconds)**	**SMAC best configuration cost (seconds)**	**Decrease in runtime (percent)**
1	447.84	396.41	11.48
2	505.66	435.94	13.78
3	609.80	520.60	14.62
4	435.58	393.68	9.61
5	673.68	452.49	32.83
6	2234.88	2026.78	9.31
7	2398.23	1701.10	29.06
8	Crashed	657.57	NA
9	Crashed	569.00	NA
10	1544.50	1166.05	24.50
11	623.16	513.217	17.64
12	1184.54	851.19	28.14
13	3164.28	2221.32	29.80
14	1581.56	344.37	78.22
15	Crashed	1799.95	NA
16	Crashed	1426.90	NA

Table 5.1: Comparison of SMAC best configuration versus default configuration. The winner is highlighted in blue color text. The crashed simulations are highlighted in red and SMAC has optimized the instances.

Metric	Value
Win percentage of SMAC	100.00%
Mean % decrease in run-time	24.92%

Table 5.2: Performance comparison of SMAC best parameter configuration and default parameter configuration

Owing to a very large cost value of the crashed simulations, the crashed default configurations are avoided for the mean decrease percentage calculation. The mean percentage decrease in runtime after optimization with SMAC is 24.92% for a specific simulation instance. This metric provides an empirical estimate of SMAC performance in optimization tasks.

SMAC has optimized all the crashed simulation instances efficiently as shown in table 5.1. This illustrates the efficiency of SMAC to find a better configuration even with bad initial configurations.

5.2 Repeatability of SMAC

The experimental setup section provides the simulation instances used in the experiment, solid and fluid solver coupled in MpCCI, the reason for experimenting, and the metrics used in this experiment. The results and discussion section discuss the results of the experiment.

5.2.1 Experimental setup

Repeatability is the measure of variation between numerous measurements performed successively by an identical measurement procedure and conditions on the same subject within a short period of time [68]. This experiment is performed to estimate the precision of SMAC in optimizing a particular simulation instance. In addition, this is a measure of the variance associated with the performance of a particular configuration provided by SMAC. For instance, a very high value of the repeatability coefficient indicates a large uncertainty associated with a particular configuration from SMAC.

The experiment is conducted on 6 randomly selected simulation instances, namely, 1, 3, 4, 5, 6, and 12, provided in table A.1. The instances are optimized for 3 trials. SMAC performs optimization for 100 iterations for each trial. Calculix and OpenFOAM are the solid and fluid solvers respectively used for solving the simulation problem. The runtime of the best configurations from each trial is noted. This list of runtimes across 3 trials for a particular simulation instance is used to estimate the variance in the best runtime

provided by SMAC. This variance is used to calculate the repeatability of SMAC.

5.2.2 Results and discussion

According to the repeatability co-efficient metrics provided in section 2.5.1.2, the groups are the different simulation instances. The total mean ($\bar{x}_G$) is the average runtime of the best configuration from SMAC for all the instances under study. The mean run-time of each instance over 3 trials is $\bar{x}_k$ [81].

The sum of squared differences within each group (SS_w), represents the variability in the runtime of the best configurations from SMAC on a particular instance across the trials. x_i is the value of a specific sample, i in the group [81]. The within-group standard deviation is computed using LibreOffice Calc. The table 5.3 provides the metrics computed.

Instance	**Best cost from SMAC (seconds)**			**Average cost (seconds)**	**Standard deviation (seconds)**
	Trial 1	**Trial 2**	**Trial 3**		
1	399.84	397.82	396.41	398.02	1.72
2	545.76	520.97	520.60	529.11	14.42
3	403.16	393.68	416.12	404.32	11.26
4	465.79	452.49	467.15	461.81	8.09
5	2019.46	2026.78	2011.49	2019.24	7.64
6	724.22	726.98	744.65	731.95	11.08

Table 5.3: Average cost and standard deviation for the simulation instances involved in repeatability test of SMAC.

Metric	**Value**
Mean variation in run-time within simulation instances (MS_w)	97.46 seconds
Repeatability coefficient (S_r)	19.34 seconds

Table 5.4: SMAC repeatability metrics

Therefore, with a probability of 95%, the maximum difference in runtime between two successive best configurations from SMAC on the same simulation instance under the same conditions is 19.34 seconds.

5.3 Behavior of SMAC with respect to iteration count

The simulation instances used in the experiment and the reason for performing the experiment is provided in the experimental setup. In addition, the experimental procedure is breifly provided. The results and discussion section discuss the results of the experiment.

5.3.1 Experimental setup

The reason for performing the experiment is to identify the optimal number of iterations required to estimate relatively lesser runtime by SMAC on a particular simulation instance. In addition, the experiment provides an estimation of the number of target algorithm runs SMAC performs to identify optimal configurations.

The experiment is conducted only on 4 different, randomly selected instances. In addition, the random selection is restricted to simulations performing more iterations in a lesser amount of time. The simulation instances 3, 4, 5, 8, and 12 are selected. The features of the instances are provided in table A.1. Each simulation instance is optimized by SMAC for an iteration count of 300. The best configuration is estimated at 6 checkpoints of iterations, namely 25, 50, 100, 150, 200, 250, and 300.

5.3.2 Results and discussion

The graphs in figure 5.1 illustrates the performance of SMAC to find better configurations with reduced runtime. SMAC successfully finds better configurations on increasing iterations. However, the problem in executing SMAC for substantial iterations is the target algorithm is executed for each iteration. Therefore, the many iterations of SMAC are practically challenging to execute. In addition, the figure 5.1 illustrates a saturation behavior of SMAC to find better configuration compared to the initial iterations.

Instance	**Best cost (seconds) for different SMAC iteration (itr) count**						
	25 itr	**50 itr**	**100 itr**	**150 itr**	**200 itr**	**250 itr**	**300 itr**
1	528.41	528.41	520.97	520.97	520.97	520.97	520.97
2	410.08	397.13	397.13	393.68	393.68	393.68	393.68
3	733.68	521.01	479.41	470.28	466.56	456.33	456.33
4	905.96	768.97	735.53	735.53	726.98	726.98	726.98

Table 5.5: Best cost (run-time in seconds) from SMAC for different iteration counts of 4 randomly selected simulation instances.

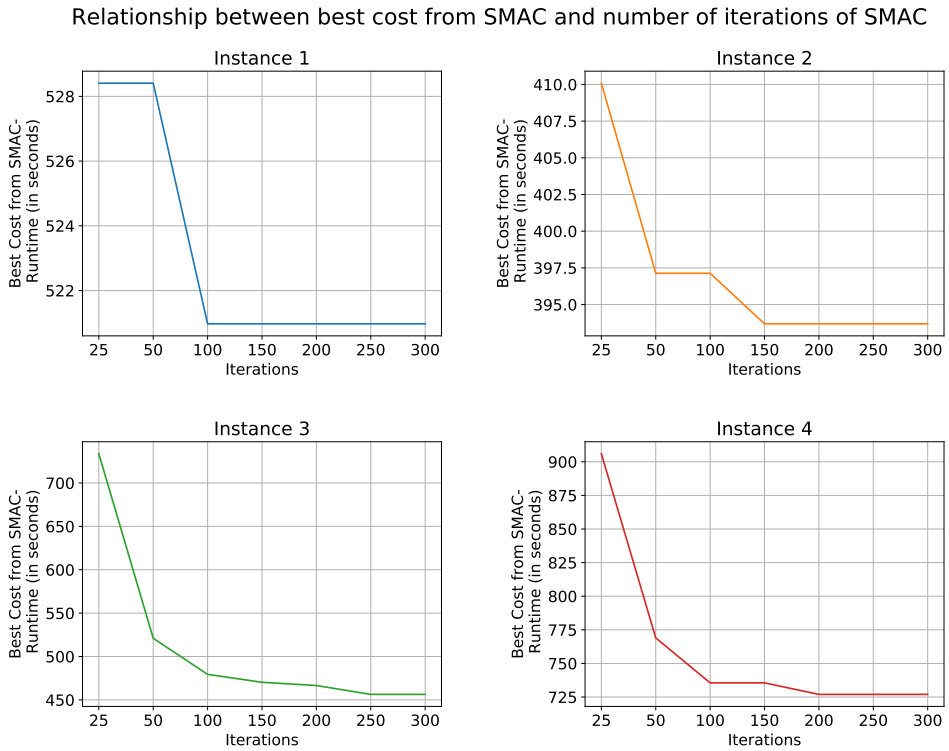


Figure 5.1: SMAC best cost versus number of iterations. The ability to find good configurations increases with iteration count.

5.4 SMAC Good-Bad-Ugly configurations

The previous experiments focus on the quality of the good parameter configurations obtained by SMAC on optimizing a single instance. This section analyses the search trajectory of SMAC and aid in selecting the training data from the dataset generated.

5.4.1 Experimental setup

This experiment provides a deeper insight into the run history of SMAC and analyzes the search path of SMAC. SMAC keeps track of all the parameters configurations θ_i and the respective cost metric c_i on evaluating a particular simulation instance π_i [9]. This set of parameters configurations, simulation instances, and respective cost metric is called run history, r = $\{\theta_i, \pi_i, c_i, \}_{i=1}^{N}$, N is the number of iterations of SMAC for optimizing a particular instance. In this research, the instances are optimized for 100 iterations. The run history of the 16 instances provided in table A.1 is utilized.

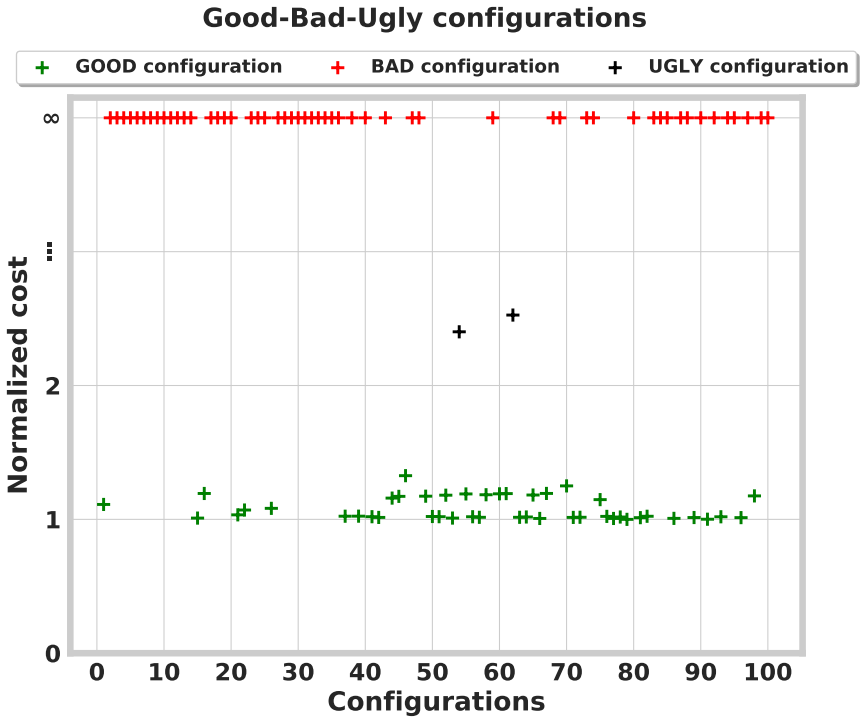


Figure 5.2: An illustration of good-bad-ugly configurations for single instance optimization over 100 iteration. Each iteration is performed with a different parameter configuration. Good and ugly configurations result in successful simulation. Bad configurations result in crashed simulation. The cost metric (runtime in seconds) has been normalized with the best cost from SMAC for a particular simulation instance.

The configurations resulting in crashed simulations are bad configurations (red markings in figure 5.2). The configurations resulting in a successful simulation (green and black markings in figure 5.2) for a particular instance are sub-divided into good (green) and ugly (black) configurations depending on the cost metric associated with the configuration. The good configurations are the sub-optimal configurations performing close to the best cost of SMAC. The ugly configurations are the configurations performing below a particular threshold relative to the sub-optimal configurations. For instance, in figure 5.2 for illustration purposes, the threshold is set to be the normalized cost values above the upper inner fence (Q3 + 1.5 × IQR) [5] normalized cost value of the distribution of the successful simulation. Interquartile Range (IQR) is the difference between third (Q3, 75^{th} percentile) and first quartile (Q1, 25^{th} percentile), (IQR = Q3-Q1).

However, the exact value of the threshold of cost for configurations performing lesser

than sub-optimal configurations for FSI multiphysics simulations should be explored in the future. The plot in figure 5.2 represents the distribution of normalized cost over 100 iterations of SMAC for a single simulation instance.

Good configurations percentage, bad configuration percentage, and ugly configuration percentage are the number of good, bad, and ugly configurations present in 100 iterations of the SMAC for a particular instance.

5.4.2 Results and discussion

On calculating the average good, bad, and ugly configurations percentage across all the 16 instances, the good and bad percentage is approximately close. The table 5.6 provides the percentage of good-bad and ugly configurations.

Configurations	**% of total configurations**
The Good (Optimal and sub-optimal)	49.2
The Bad (Crashed)	47.8
The Ugly (Below sub-optimal)	3.0

Table 5.6: Percentage of good-bad-ugly configurations from SMAC for per-instance optimization (average over 16 instances). The ugly configurations are considered to be the configurations with normalized cost value above the upper inner fence (Q3 + 1.5 $\times$ IQR) of the successful simulations distribution.

The interest of this research work is to predict configurations resulting in successful simulation. The bad configurations bias the predictions by the regressor towards crashed simulation configurations. Therefore, the bad configurations are removed in the data pre-processing step. However, the threshold for defining ugly configurations is a future work of the research.

5.5 Comparison of model performances and runtime transformations

The following experimental setup section provides an overview of the instances and machine learning models used in the experiment. In addition, procedure to train the models is explained. The results and discussion section provides the comparison of model performance using RMSE. The models are trained using data with different transformations applied to the cost metric (runtime in seconds) feature.

5.5.1 Experimental setup

This experiment compares the performance of the cost and configuration response models in predicting the optimal configuration for a given simulation instance. The models are trained on the dataset created by optimizing 16 instances provided in the table A.1. Each training instance is optimized by SMAC with an iteration limit of 100. This results in a dataset with 1600 observations provided in table 5.7. Each observation includes 11 hyperparameters provided in table 4.3, 1 cost metric (run-time in seconds) and 6 features specific to the instance. Therefore, a total of 18 features is available in the dataset.

Number of simulation instances available to generate dataset	16
Number of iterations SMAC optimize each instance	100
Observations (hyperparameters, cost, instance features)	1600

Table 5.7: Number of observations in the training set

The labels of the 6 cost and 7 configuration response models are the cost and configurations, respectively. The response models are explained in section 4.4.4 and section 4.4.5. The training is performed on the dataset with 4 different variants of cost followed by normalization. The variants are the normal cost (runtime in seconds) and 3 functional transformations to the normal cost. The functional transformations are logarithmic transformation, logarithmic-scaled and inverse-scaled provided in 4.4.3. The cost is normalized with optimal cost, 1, following the transformation.

The dataset available is a relatively smaller dataset comparing the various multiphysics simulation problems. Therefore, to estimate the model performance in the best possible method on the small dataset, k-fold cross-validation is performed. The imperative aspect of k-fold cross-validation is every sample in the data forms the training and validation. The samples are used only once for validation and 'k-1' times for training. This aids the model to understand the characteristics of the data better. In addition, the model performance is tested on various test sets across the dataset, providing a better estimation of the generalization error on the unseen data. The dataset is randomly divided into 'k' folds of the same size, and 'k-1' folds are used for training. The model trained on 'k-1' folds is tested on the remaining hold-out set. This procedure is repeated until each fold is used for validation. The model is discarded at the end of the procedure.

The final model for deployment is the model trained on the entire dataset available. The performance measure is calculated on the validation set for each split, and the mean

of the performance measure is used for comparing the models.

RMSE discussed in section 2.5.2.1 is used to compare the performance of different machine learning models. The reason for choosing RMSE is the categorical parameters are encoded to integer values in the regression task, and the error in categorical parameters are more important in configuring the coupling tool. For instance, configuring the coupling scheme parameter 'Explicit' instead of 'Implicit' has a substantial possibility of failure relative to the small change in the omega value. In addition, it is comparable to the normalized distance measure between the vector of different configurations for evaluating configuration response models. RMSE provides the error on the same unit of the variable used in the regression task.

5.5.2 Results and discussion

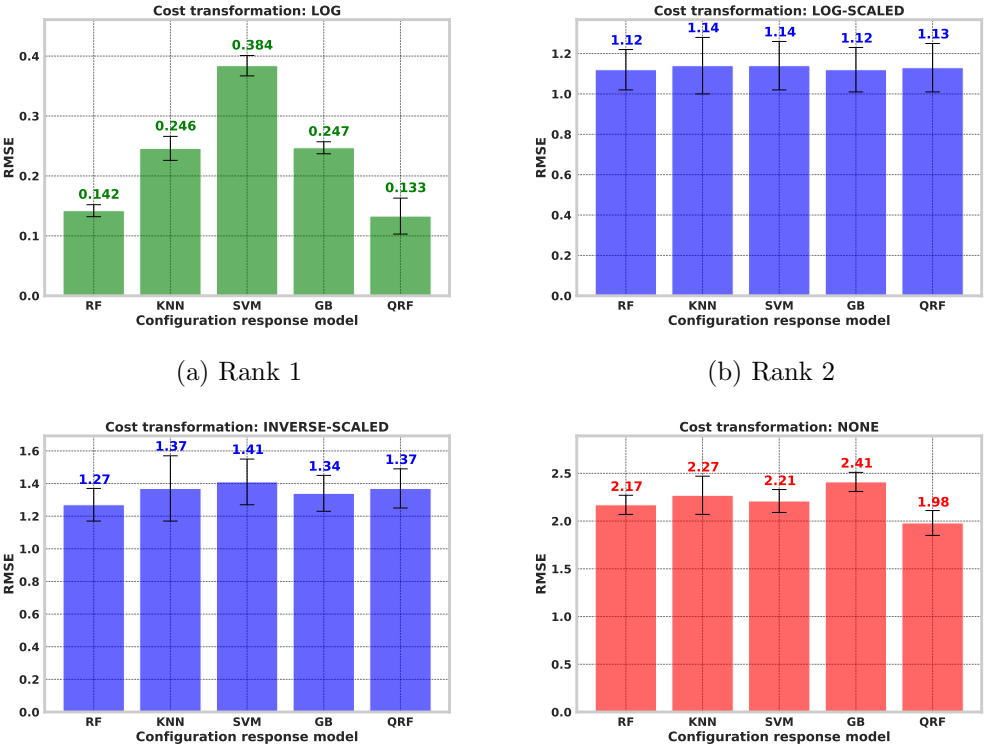
The below figures 5.3 and 5.4 provides the comparison of RMSE for different models provided in table 5.8. In addition, the notations of the models used in the graph is provided in the table 5.8.

Model notation	**Model name**
RF	Random Forest
GB	Gradient Boosting
SVM	Support Vector Machine
QRF	Quantile Random Forest
KNN	K- Nearest Neighbors
RF_EI	Random Forest with the cost metric-Expected Improvement

Table 5.8: Notation of different configuration and cost response models used for training.

The log transformation of the runtime performs better than other transformations because it reduces the skewness and variances in the data. The skewness and standard deviation for all different types of transformed data are provided in table 5.9. The less spread in the predictor variable aids in better prediction and less RMSE in the model evaluation. The figures 5.3 and 5.4 illustrate the importance of logarithmic transformation with less error model evaluation. The various previous works on runtime prediction recommend logarithmic transformation of the runtime because logarithmic transformation aids in runtime prediction by reducing the dynamic range of the runtime [51] [53].

The RMSE error of cost and configuration response models are incomparable. The cost response model errors are a dimensionless quantity (normalized cost), representing only the error in the cost variable. In contrast, the configuration response models illustrate



(a) Rank 1

(b) Rank 2

(c) Rank 3

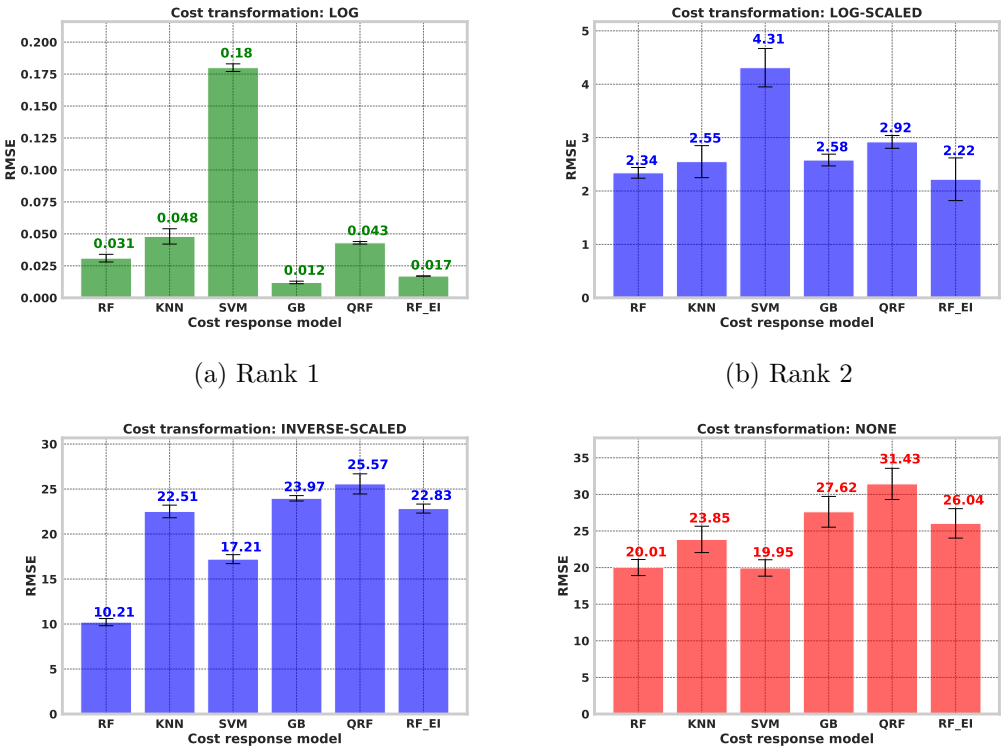
(d) Rank 4

Figure 5.3: RMSE for different configuration response models using 5-fold CV.The green and red bars indicate the least and largest error, respectively, among the models for different transformations. The number on top of each bar is the mean RMSE of the model provided by 5-fold CV[2]. The error bars (black lines with an upper and lower bound) is the deviation across 5-folds.

the distance between the predicted response vector (vector of configurations) and the actual response vector. Furthermore, a small magnitude of error is tolerable in the cost response models. However, a small error in the configuration response model at worst case result in switching the category of the categorical parameters. In order to avoid such worst-case errors, models trained by logarithmic transformation is selected for evaluation on unseen simulation instances.

[2]The RMSE values in each individual graph is rounded to same significant figures in a particular graph.

[3]The RMSE values in each individual graph is rounded to same significant figures in a particular graph.



(a) Rank 1

(b) Rank 2

(c) Rank 3

(d) Rank 4

Figure 5.4: RMSE for different cost response models using 5-fold CV. The green and red bars indicate the least and largest error, respectively, among the models for different transformations. The number on top of each bar is the mean RMSE of the model provided by 5-fold CV[3]. The error bars (black lines with an upper and lower bound) is the deviation across 5-folds.

Transformation	**Pre-processed cost metric**	
	Skewness	**Standard deviation**
LOG	0.2152	0.6718
LOG-SCALED	-1.036	7.417
INVERSE	-1.532	2501
NONE	**6.331**	**3304**

Table 5.9: Statistical measures of the pre-processed cost metrics using different transformations. The green and red highlights indicate the first and last ranked statistical values, respectively, across different transformations.

5.6 Evaluation of smart-coupling on an unseen simulation instance

The experimental setup provides the overview of the simulation instances used in evaluation. The evaluation metrics are defined in the following section 5.6.2. In addition,

the results illustrate the performance of the models.

5.6.1 Experimental setup

The evaluation is performed on two unseen simulation instances of 3D driven cavity. The machine learning models and the training dataset from SMAC have not seen the simulation instance features. In addition, the evaluation simulation instances cover critical cases with density ratio 1. The density ratio value 1 is critical from the computation point of view in the coupling tool [21]. The features of the evaluation simulation instances are provided in table A.2.

The two simulation instances are configured by the parameters predicted by the models trained using the logarithmically transformed cost values. A total of 11 configurations are predicted by the 6 cost response models and 5 configuration response models. This leads to 11 simulations of each instance with different parameters and a total of 22 simulations. OpenFOAM and Calculix are the fluid and solid solvers used in the simulation.

The model performance in the simulation instances is evaluated using the metric provided in the next section.

Number of machine learning models	11
Number of evaluation simulation instances	2
Number of configurations predicted by each model	1
Transformations used for cost variable	LOG
Total evaluation simulation instances	11 x 2 x 1 = 22

Table 5.10: Summary of the number of evaluation simulation instances.

5.6.2 Evaluation metrics

5.6.2.1 Normalized Runtime (NR)

NR of a configuration obtained from 'X' is the ratio of the simulation runtime with the particular configuration obtained from 'X' to the simulation runtime of the best configuration for a particular simulation. 'X' can be a machine learning model or SMAC. NR is calculated for each configuration predicted by the model for a particular instance. NR less than 1 illustrates a better performance of the configuration predicted by the model relative to the best configuration runtime. The best configuration is the configuration obtained from SMAC. Therefore, NR for the best configuration from SMAC is 1, i.e., 'X' is SMAC.

$$\text{NR of a configuration from 'X'} = \frac{\text{Runtime of the configuration from 'X'}}{\text{Runtime of the best configuration from SMAC}} \tag{5.3}$$

5.6.2.2 Average Normalized Runtime (ANR)

ANR of configurations obtained from 'X' provides a measure of the performance of 'X' over a set of simulation instances. Similar to NR, 'X' can be a machine learning or SMAC. The ANR of the configuration obtained from model 'X' is the average of the NR obtained from the model for all the simulation instances in the evaluation set. ANR for the best configurations from SMAC is 1.

$$\text{ANR of configurations from 'X'} = \frac{\Sigma_{i=1}^{N}\text{NR of a configuration from 'X' on instane i}}{N} \tag{5.4}$$

NR of a configuration from a particular model is given in equation 5.3. In equation 5.4, model 'X' is the cost or configuration response model, i is a particular simulation instance, and N is the total number of simulation instances in the evaluation set.

5.6.3 Results and discussion

The figure 5.5 illustrates the performance of models selected from the previous experiment on unseen simulation instances. The RF model predicting each parameter of the configuration by regression is performing better compared to the other configuration response models because RF handles categorical parameters better [45]. In contrast, the random forest predicting the expected improvement metric is performing better in the cost response model. This illustrates the performance of the models in two simulation instances.

On looking closer into the ANR metric, the runtime of each configuration predicted by a model is normalized using the runtime of the best configuration from the optimization of a particular instance using SMAC. The runtime of the best configuration from SMAC is the benchmark runtime for each instance. Therefore, the average normalized runtime of the best configuration from SMAC[4] for a particular instance is 1.

[4]NR of the best configuration from SMAC for a particular simulation instance is 1. Therefore, the ANR across the set of evaluation instances is 1.

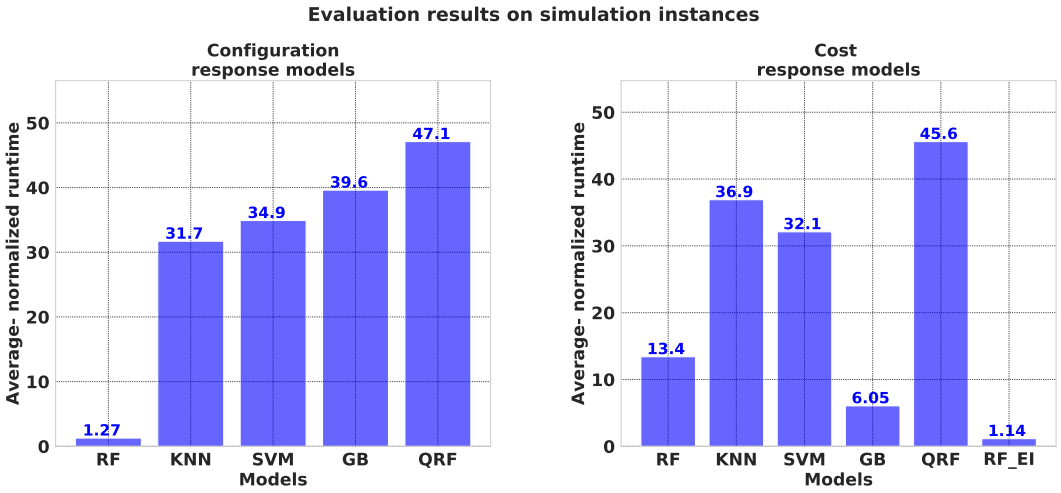


Figure 5.5: Configuration and cost response model evaluation results on simulation. The lower value is the better model.

The model with an ANR less than 1, perform better than the best configuration of SMAC. However, the models cannot predict configurations better than the best configuration from SMAC. The reason is the configurations from SMAC are obtained by performing per-instance optimization of the simulation instance over 100 evaluations of the target algorithm, MpCCI.

The top three models are suggested to the user. The selection criteria is the models with an order of magnitude, 1, from the average normalized runtime of the best cost. In addition, all the models are performing better than the default configuration.

In figure 5.6, the default configuration results in a crashed simulation. However, the performance of the top 3 models is extremely helpful in meeting the desired result of the research work.

5.7 Summary

The chapter addresses the different experiments conducted to evaluate the smart coupling tool developed. The summary of the main observations from the experiments is provided below.

1. A single simulation instance with a particular feature set is optimized using SMAC. The SMAC best configuration provides a lesser runtime compared to the default configurations of MpCCI. SMAC successfully estimates a configuration better than the default configuration with a win-percentage of 100%. In addition, SMAC provides a mean decrease in runtime of approximately 25%. This aids in developing

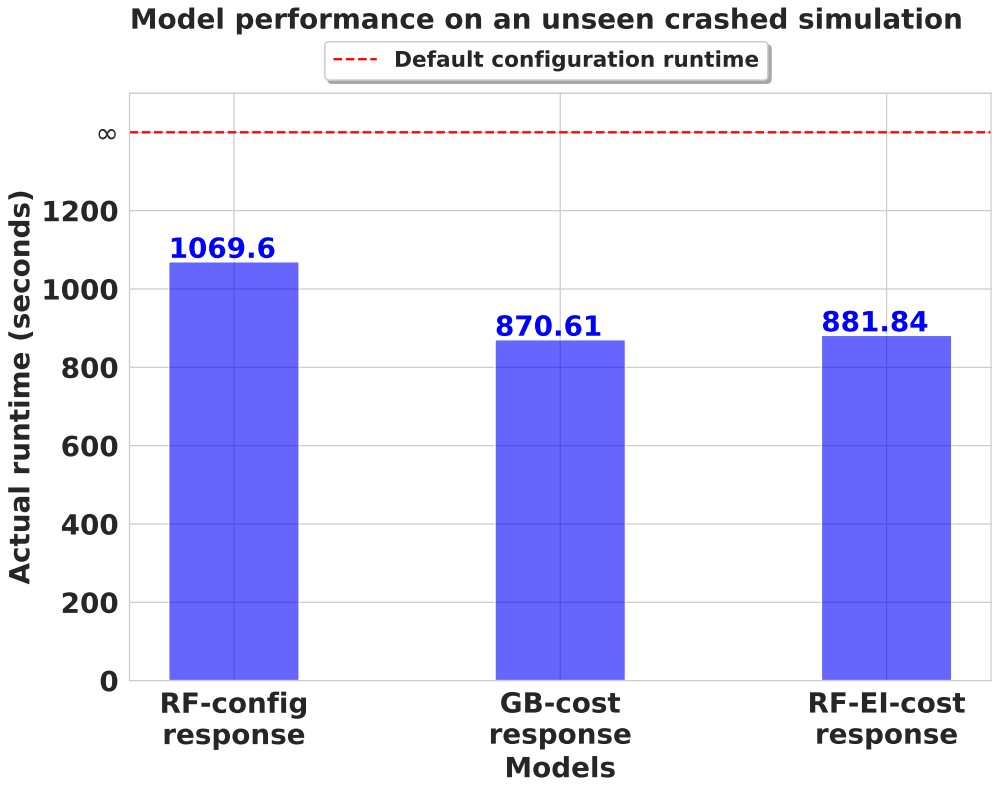


Figure 5.6: Model performance on a critical simulation instance (solid-to-fluid density ratio is 1).

a dataset with configurations better than the default configuration. The dataset is used for training models to predict a configuration with lesser runtime provided an unseen simulation instance.

2. One of the imperative aspects of SMAC is the ability to find a better configuration on a crashed simulation instance for the default configuration. On all 4 crashed simulation instances with default configurations, SMAC is able to estimate a better configuration successfully.

3. Currently, all the configurations resulting in a successful simulation are considered for building a machine learning model to predict better configurations. However, the choice between selecting all the good configurations and only the best configuration for a simulation instance should be explored in the future.

4. The repeatability of SMAC in optimizing a multiphysics simulation is evaluated in section 5.2. This aids in understanding the ability of SMAC to find good

configurations with similar performance repeatedly. In addition, the variance associated with the performance of the configurations generated from SMAC is illustrated. The repeatability coefficient of SMAC is 19.34 seconds.

5. The relationship between the number of times SMAC evaluates the objective function, and the performance of the best configuration from SMAC is directly proportional. The larger the number of times a target algorithm is evaluated, SMAC finds configurations with better performance. However, to effectively manage the resources and reduce the number of target algorithm evaluations, SMAC runs the target algorithm for 100 iterations.

6. On analyzing the run history of SMAC, the number of bad configurations resulting in crashed simulation is approximately equal to the good configurations resulting in successful simulation within the 100 iterations of SMAC. The good configurations are important for the training of the model, and bad configurations are unimportant, leading to wastage of resources.

7. The regression models to predict the best configuration given a simulation instance is trained in two methods, namely cost and configuration response models. In addition, the cost feature (runtime in seconds) of a particular simulation instance is transformed in 4 different ways, namely logarithmic, scaled-logarithmic, scaled-inverse, and none. The model selection incorporates the 5-fold cross-validation technique. The models illustrate relatively better performance on logarithm transformed cost metric with less RMSE. All the models are trained on the complete data available with logarithm transformed cost features for real-time evaluation on unseen simulation instances.

8. To evaluate the model performance on the unseen instance, an Average Normalized Runtime (ANR) metric is devised. After evaluating all the models trained with logarithm transformed cost, the top three models are selected for integration to the smart coupling tool. In figure 5.5, the predicted configurations for each evaluation instance is compared against the best cost obtained by optimizing each evaluation instance. This best cost is treated as a benchmark. The predicted configurations are not performing better than the best cost because the best cost is obtained by optimizing each instance for 100 iterations using SMAC. Therefore, the top three models performing close to the best cost are suggested to the user by the smart coupling tool.

9. On comparing the actual runtime of the suggested parameter configuration against the default configurations runtime in figure 5.6, the model performance is effective than the default. This illustrates the effectiveness of the smart coupling tool to estimate an optimal configuration for a simulation that crashed with the default configurations.

10. The configurations from RF- configuration response model, GB- cost response model, and RF predicting expected improvement are reliable, providing relatively less runtime than other models. The time taken by the smart coupling tool to suggest the three configurations is 0.621 seconds.

6

Conclusions

Multiphysics co-simulation is widely used in many industrial and academic research and development purposes. MpCCI, developed by Fraunhofer SCAI, aids in co-simulation. The current procedure of setting up a coupling simulation in MpCCI is tedious, with numerous parameters to configure. In addition, the process of tuning the parameters to achieve stable and fast simulation results is highly challenging, given the substantial simulation time and computational cost associated with each simulation. The study focuses on reducing the time taken to perform a coupling simulation. The research work suggests optimal parameter configurations for a user-defined simulation problem. This research work proposes a methodology based on automated algorithm configuration and evaluates the ability of the proposed methodology to configure the coupling tool with the optimal parameters.

A set of simulation instances with varying features are optimized using a model-based optimization procedure, SMAC. The parameter configurations resulting in a successful simulation for a particular simulation instance is utilized for training. This machine learning model is used in real-time to predict optimal configurations given a simulation problem. The optimal configurations signify the ability to perform simulation at a relatively lesser time compared to the current default configuration. The models are evaluated on unseen simulation instances to suggest the best performing models to the user. Random forest and gradient boosting outperform the other machine learning models.

6.1 Contributions

The contributions of the research work are:

1. A methodology based on automated algorithm configuration to predict optimal parameter configurations of a coupling tool resulting in robust co-simulation given a simulation problem.

2. **Multiphysics simulation dataset for ML:** A preliminary Fluid-Structure Interaction (FSI) dataset for machine learning tasks with 1600 observations is developed. The 19 features of the dataset include the solid domain features, fluid domain features, coupling tool parameters, the corresponding runtime, and status of the simulation- successful or crashed.

3. **Smart coupling tool:** A package including the implementation of the proposed strategy to reduce the coupling tool runtime by suggesting parameter configurations given a simulation problem (refer section D).in the dataset generated

4. **Feature reader:** A package to extract all the possible features of an FSI simulation instance given the coupling tool configuration files and log files. A set of 100 plus features are extracted from the solid and fluid domain solvers (refer section 4.4.1).

5. SMAC performance and repeatability coefficient to optimize multiphysics simulation instances are estimated.

6. The comparison of machine learning models- KNN, SVM, RF, and GBM to predict the optimal configuration of the coupling tool. The multiphysics simulation data is used for training and validation. In addition, the model performance on unseen simulation instance is evaluated. RF and GBM outperform other models.

6.2 Lessons learned

The following lessons are learned during the research work.

1. The dataset creation is a challenging task because of the substantial simulation time and stability issues in multiphysics simulations. In this work, the dataset is created in 2 steps. Firstly, enumerating the various features of the simulation instance. Then, identifying the parameter configurations and respective cost by running SMAC. The initial plan is to utilize 30 simulation instance and generate 3000 observations of the dataset. However, on continuously kick-starting failed simulations, the final dataset includes only 1600 observations.

2. The second major lesson is data transformation. Logarithmic transformation is a pivotal data pre-processing step for skewed distribution. This transformation convert the features to log-normal scale. The fat-tail distribution of the cost feature results in relatively large RMSE on the training set. However, the error is reduced drastically by transforming to a log-normal distribution. Therefore, the performing transformation depending on the data distribution aids in improving the learning capability of the model.

6.3 Future work

Future research and development prospects are enumerated below.

1. Multiphysics simulation includes different types of simulation problems encompassing FSI, magnetostatics, hydrodynamics, chemical reactions, and acoustics in all fields of engineering and scientific research. In this research work, the dataset and feature extraction focus on the FSI transient simulation problems. However, the dataset spanning the entire multiphysics simulation problems should be focussed in the future to scale the suggested AAC approach to the entire co-simulation domain.

2. The current features of the simulation instances are selected based on the properties of an FSI suggested in various research works [93] [94] and experts opinion. The simulation instances include various additional features. The exploration of this features of multiphysics simulation is an intricate study.

3. SMAC is the current state-of-the-art automated algorithm configuration approach employing a random forest surrogate model. SMAC is well-suited for this research work because of the numerous categorical parameters in the coupling tool. However, the current prospective researches in the field of automated algorithm configuration investigate the usage of quantile random forests surrogate model. A comparison of SMAC performance with RF and QRF surrogate models is a promising examination.

4. In this research work, SMAC has been incorporated to optimize a particular instance and estimate the optimal parameter configurations treating MpCCI as a black-box. However, the performance of SMAC in finding optimal configurations can be compared against other black-box optimization methods such as Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and algorithm configuration approaches such as Gender-based Genetic Algorithm and ParamILS.

5. The machine learning models are trained using the configurations (refer figure 5.2) resulting in successful simulation from the dataset generated by the per-instance optimization of all the instances. However, the optimal count of good configurations for training the models is a possible future work because taking all the good and ugly configurations have the possibility to bias the model.

6. The current smart coupling tool is available as a python package D. In the future, the package should be integrated into the MpCCI software after enriching the machine learning with an enriched dataset covering other coupling simulation problems.

7. The performance of various algorithm configuration approaches are evaluated experimentally against standard benchmarks in problems such as mixed-integer programming and boolean satisfiability problems [45]. In contrast, being a new application domain, no standard benchmarks are available. One of the future prospects can focus on developing benchmarks for testing the algorithm configuration performance on multiphysics simulations.

A

Features of simulation instances

The following sections provide the two set of simulation instance features used in the study.

A.1 Dataset creation and experimental setup instance

The table A.1 provides the features of the simulation instances used for training and experiments. This set of 16 instances are used to generate the dataset for training the cost and configuration response models. Each instance is optimized using SMAC for 100 iterations and results in 1600 (16×100) observations comprising the dataset. The experiments include the repeatability test of SMAC, the ability of SMAC to outperform the current default configurations and the behavior of SMAC for different iterations utilize the instances with the features provided in the table A.1. The table A.1 is referred in the previous chapter 5.

A.2 Unseen simulation instances

The features of the three simulation instances are provided in table A.2. This feature set is used in section 5.6 to evaluate the predictions from the smart coupling tool.

Instance	Simulation problem	Solid-Fluid density ratio	Fluid – density (kg/m^3)	Solid – elasticity (Pascal)	Poisson's ratio	Fluid – viscosity (Pascal second)	Time step (second)
1	3D driven cavity	2.50E+02	1.00E+00	2.50E+02	0.00E+00	1.00E-03	1.00E-01
2	3D driven cavity	1.00E+02	1.00E+00	2.50E+02	0.00E+00	1.00E-03	1.00E-01
3	3D driven cavity	1.00E+02	1.00E+00	1.00E+03	0.00E+00	1.00E-03	1.00E-01
4	3D driven cavity	1.00E+02	1.00E+00	1.00E+04	0.00E+00	1.00E-03	1.00E-01
5	3D driven cavity	1.00E+01	1.00E+01	2.50E+02	0.00E+00	2.00E-04	1.00E-01
6	3D driven cavity	1.00E+02	1.00E+01	1.00E+03	0.00E+00	1.00E-03	1.00E-02
7	3D driven cavity	2.50E+01	1.00E+01	2.50E+02	0.00E+00	1.00E-03	2.00E-02
8	3D driven cavity	1.00E+01	1.00E+01	1.00E+03	0.00E+00	1.00E-03	2.00E-02
9	3D driven cavity	1.00E+01	1.00E+01	1.00E+04	0.00E+00	1.00E-03	2.00E-02
10	3D driven cavity	1.00E+01	1.00E+00	1.00E+04	0.00E+00	1.00E-03	2.00E-02
11	3D driven cavity	1.00E+01	1.00E+01	1.00E+04	0.00E+00	1.00E-03	2.00E-01
12	3D driven cavity	1.00E+01	1.00E+02	1.00E+04	0.00E+00	1.00E-03	5.00E-02
13	3D driven cavity	1.00E+01	1.00E+02	1.00E+05	0.00E+00	1.00E-03	1.00E-02
14	Elastic flap	1.00E+03	1.00E+00	1.00E+08	4.90E-01	2.00E-05	1.00E-04
15	Elastic flap	1.00E+01	1.00E+02	1.00E+06	4.90E-01	2.00E-05	1.00E-04
16	Elastic flap	1.00E+02	1.00E+01	1.00E+04	4.90E-01	2.00E-05	1.00E-04

Table A.1: Features of 16 simulation instances used for dataset collection and experiments.

Evaluation instance	**Simulation problem**	**Solid-Fluid density ratio**	**Fluid – density (kg/m^3)**	**Solid – elasticity (Pascal)**	**Poisson's ratio**	**Fluid – viscosity (Pascal second)**	**Time step (second)**
1	3D driven cavity	1.00E+00	1.00E+03	1.00E+07	0.00E+00	1.00E-03	5.00E-02
2	3D driven cavity	1.00E+03	1.00E+00	1.00E+04	1.00E-01	1.00E-02	1.00E-02

Table A.2: Features of simulation instances used for smart-coupling evaluation.

B

Illustration of BO

The goal of the task is finding the global maximum point of the objective function in the figure B.1 using BO with Gaussian Process (GP) surrogate model. The noisy samples depict the stochastic noisy output in real-time. The objective function in yellow is the noiseless distribution used to sample the noisy points for illustration purpose [61].

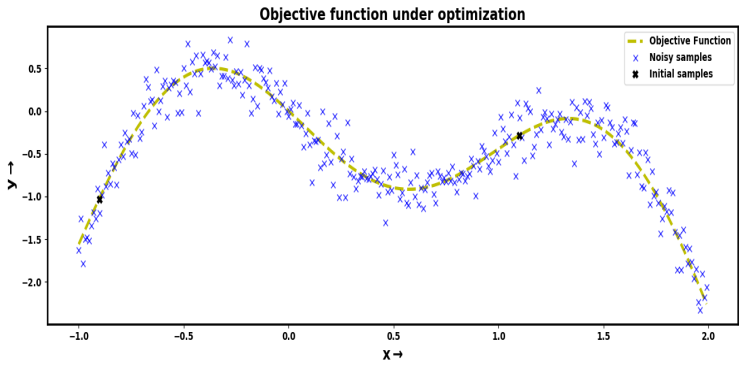
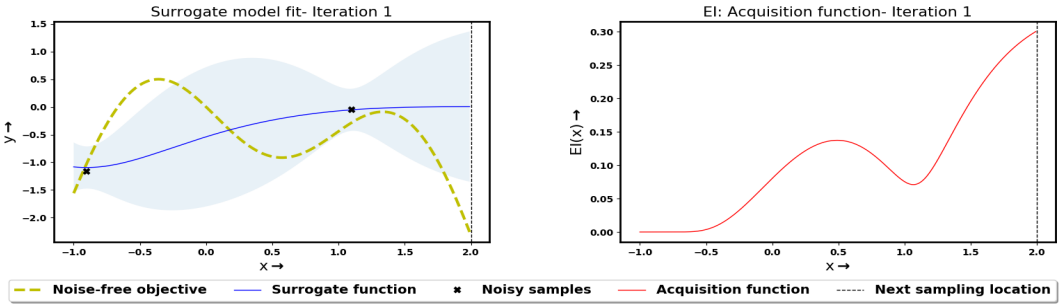
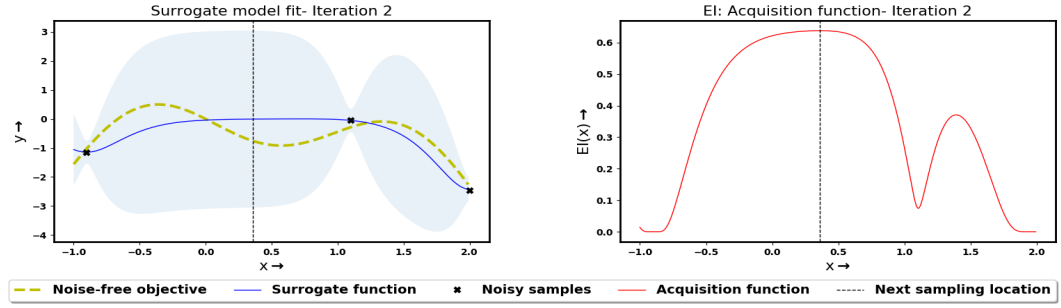


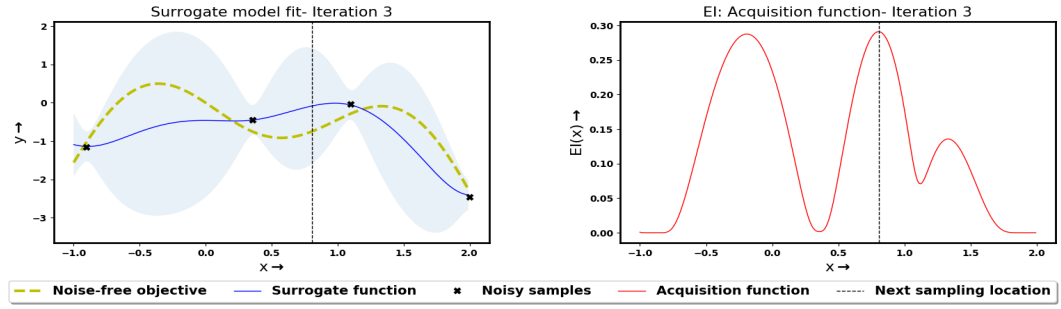
Figure B.1: An example objective function to illustrate the working of Bayesian optimization.



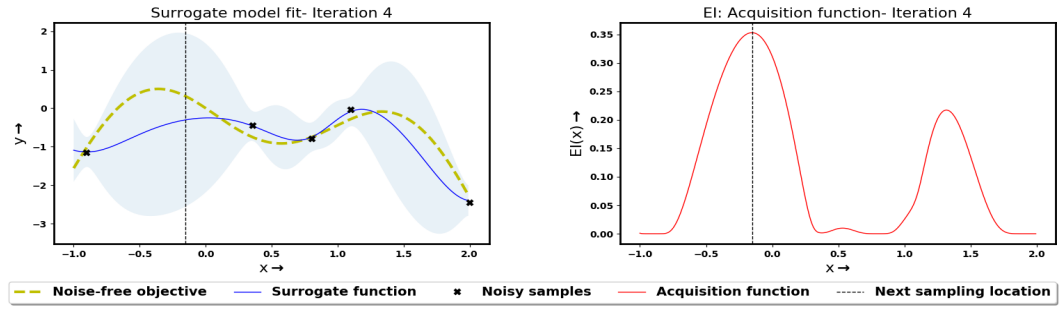
(a) Evaluate the query points (EVALUATE) →Fit surrogate to initial evaluation (FIT)→ Maximize EI to find the next query point (MAXIMIZE EI).



(b) EVALUATE → FIT → MAXIMIZE EI.

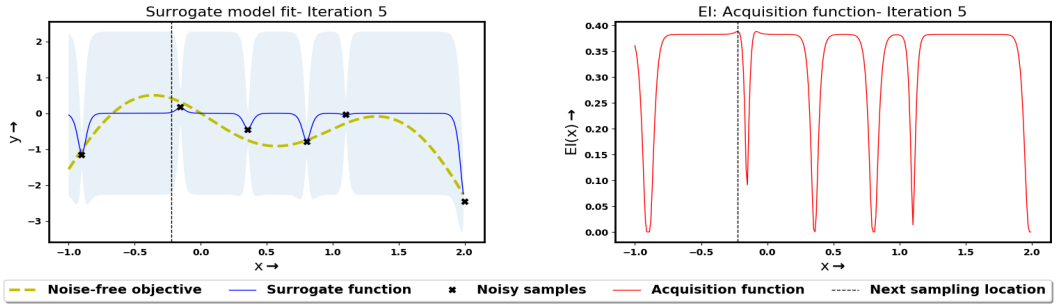


(c) EVALUATE → FIT → MAXIMIZE EI.

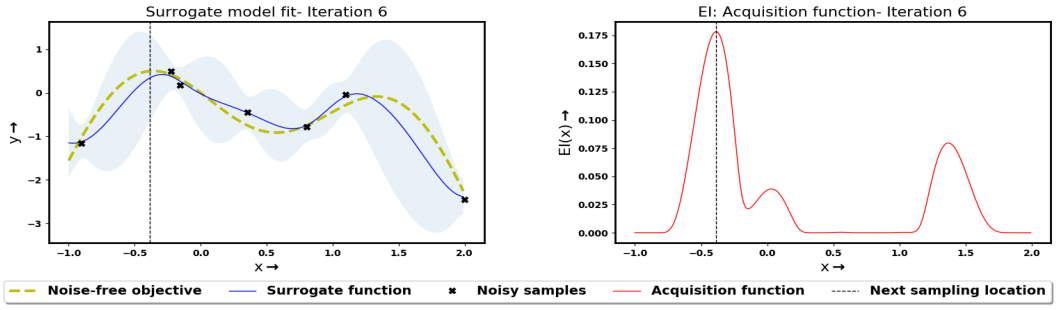


(d) EVALUATE → FIT → MAXIMIZE EI.

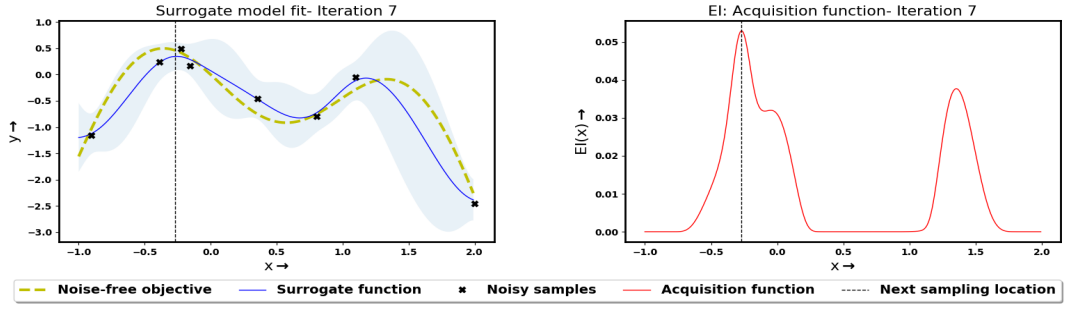
Figure B.2: Illustration of BO- Iterations 1 to 4.



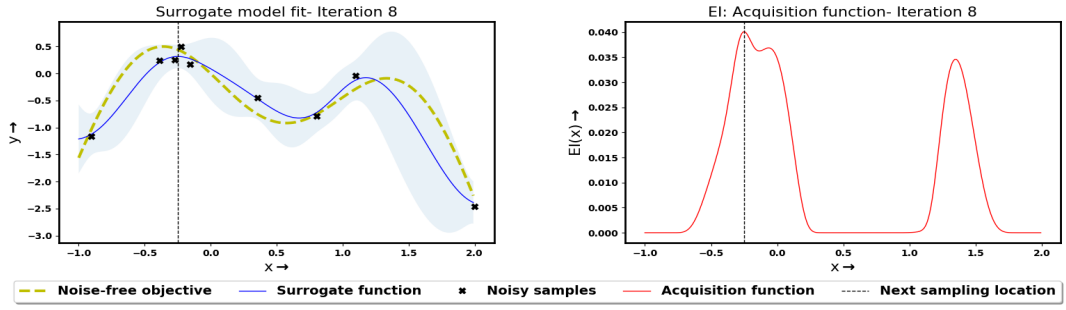
(a) EVALUATE → FIT → MAXIMIZE EI.



(b) EVALUATE → FIT → MAXIMIZE EI.



(c) EVALUATE → FIT → MAXIMIZE EI.



(d) EVALUATE → FIT → MAXIMIZE EI. BO terminates returning the best sample point for 8 iterations.

Figure B.3: Illustration of BO- Iterations 5 to 8.

C

Flowchart of SMAC

This section explains the flowchart of SMAC. In this research work, SMAC is used to optimize the training and evaluation simulation instances by per-instance black box optimization. The blue marks indicate the file names performing the task in the smart coupling tool package D provided for user convenience. The working of the SMAC is explained in detail at 4.4.6. In 4.4.6, the optimization is explained for all the training instance. In contrast, this section covers single instance optimization by running the smart coupling tool with single MpCCI input file.

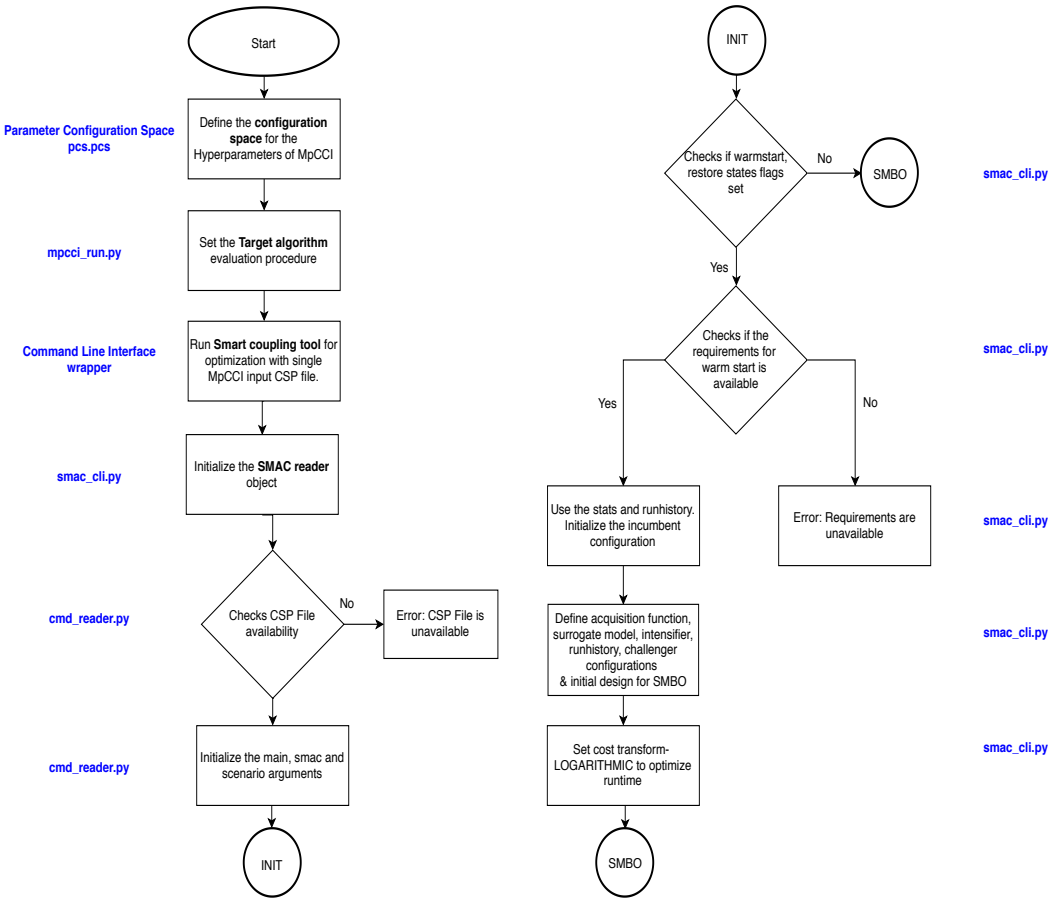


Figure C.1: Flowchart of SMAC performing single instance optimization- 1.

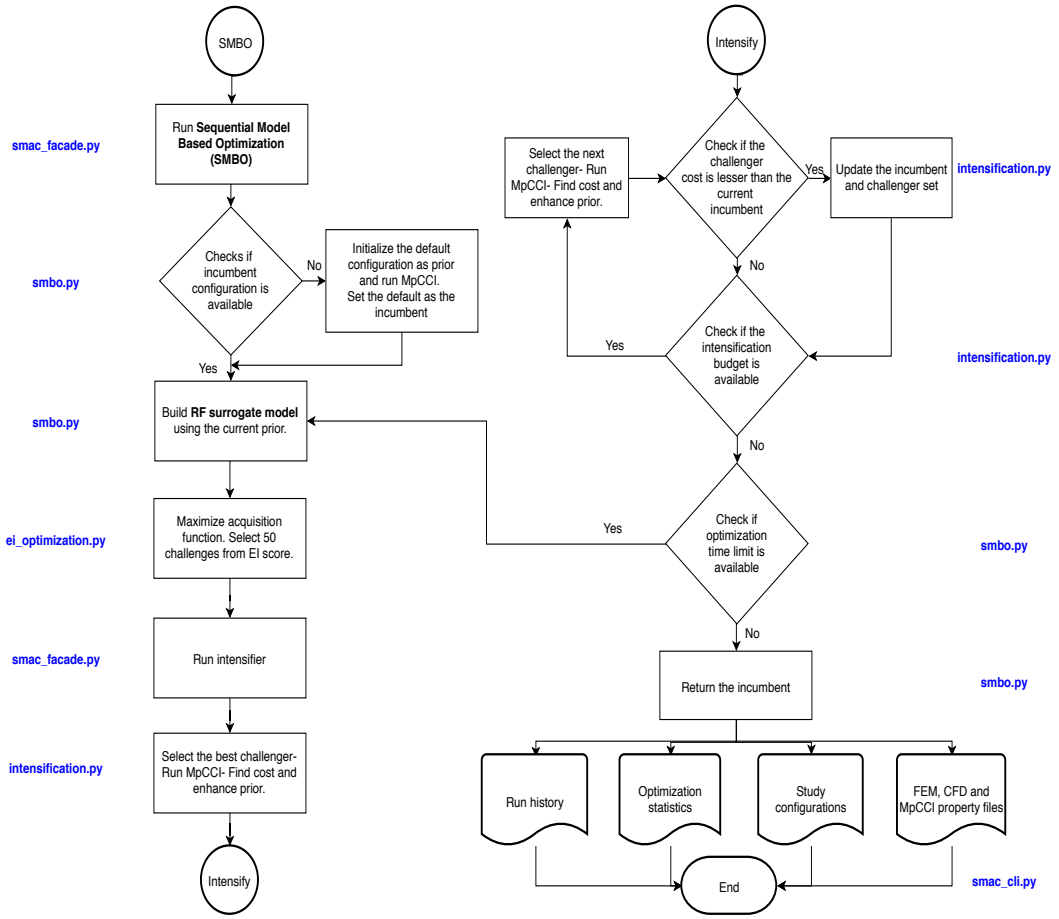


Figure C.2: Flowchart of SMAC performing single instance optimization- 2.

D

Software tools contributed

The python-based packages of the smart coupling and feature reader tools are available in a GitLab repository[1]. The package holds a private ownership. Kindly contact Mr. Klaus Wolf[2] and Mr. Hamid Arjmandi[3] for collaboration.

D.1 Smart coupling tool

The smart coupling tool provides a suitable configuration for a co-simulation given the MpCCI simulation input files. The MpCCI input files and the process of setting up MpCCI is available at [35]. The package operates in three different phases- Optimization, Training and Prediction. This package includes an implementation of the original SMAC from [65] developed by the Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp and Frank Hutter. However, the SMAC has been modified using the algorithm 2. The steps to utilize the package are enumerated below.

1. Clone the repository Smart coupling tool and feature reader.

2. The initial requirement is swig (4.0 >= 3.0, recommended 3.0.12). Next, install all the requirements to use the package using the below commands.

   $ cat requirements.txt | xargs -n 1 -L 1 pip install
   $ pip install .

3. Optimization phase: The optimization phase optimizes a given simulation problem and the output contains the run history of the optimization, the best configuration for the particular simulation instance and the features of the simulation instance.

   $ mpcci_smac –csp <csp-file.csp> –pcs <pcs-file.pcs>

[1] https://gitlab.scai.fraunhofer.de/deepan.chakravarthi.padmanabhan/smart_coupling/tree/master
[2] Email ID: klaus.wolf@scai.fraunhofer.de
[3] Email ID: hamid.reza.arjmandi.marvast@scai.fraunhofer.de

<csp-file.csp> is the MpCCI project input file. <pcs-file.pcs> is the parameter configuration space format file containing the range and default values of the parameters. The MpCCI project file holds all the information regarding the co-simulation task- the solid and fluid model files directory and coupling parameters required from the user.

4. Training phase: The training phase trains the machine learning models for all the simulation instances optimized. The dataset is created online given the run history and features of the simulation instances. The run history and features of the simulation are present in the output directory of the optimization phase. Therefore, this phase requires only the output of the optimization phase of each instance.

   $ mpcci_smac –mode TRAIN –trainstudy <trainstudy-file.txt>

   <trainstudy-file.txt> contains the mpcci_smac optimization phase output directory of all the training instances.

5. Prediction phase: The prediction phase utilize the models trained in the training phase. In addition, it uses the user provided MpCCI input csp files and the pcs files.

   $ mpcci_smac –mode PREDICT –csp <csp-file.csp> –pcs
   <pcs-file.pcs> –models <path-of-trained-models>

   The output of the prediction phase is the MpCCI input files with the configurations predicted by the smart coupling tool for the simulation instance provided by the user.

D.2 Feature reader

The feature reader package provides the features of the simulation instance. It extracts approximately 100 features from the solid and fluid simulation models, MpCCI inputs and the result log of MpCCI. For this research purpose only 6 features are used depending on the literature study [93] [94]. The package installation is similar to smart coupling tool. The following command executes the feature reader.

$ mpcci_feature_reader <csp-file.csp> <mpcci-job-file.log>
<mpcci-server-file.log>

E

COSEAL 2019 poster

This research work was selected for poster presentation at the COnfiguration and SElection of ALgorithms workshop 2019 held at Potsdam, Berlin on 26th and 27th August 2019. It was organized by experts in the field of Automated Algorithm Configuration, Prof. Dr. Holger H. Hoos and M. Sc. Marius Lindauer. The poster is included in the following page.

References

[1] Belarmino Adenso-Diaz and Manuel Laguna. Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operations Research*, 54(1): 99–114, 2006.

[2] Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A Gender-Based Genetic Algorithm for the Automatic Configuration of Algorithms. In *Principles and Practice of Constraint Programming*, pages 142–157. Springer Berlin Heidelberg, 2009.

[3] Carlos Ansótegui, Yuri Malitsky, Horst Samulowitz, Meinolf Sellmann, and Kevin Tierney. Model-based Genetic Algorithms for Algorithm Configuration. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 733–739. AAAI Press, 2015.

[4] Prasanna Balaprakash, Mauro Birattari, and Thomas Stützle. Improvement Strategies for the F-Race Algorithm: Sampling Design and Iterative Refinement. In *Hybrid Metaheuristics*, pages 108–122. Springer Berlin Heidelberg, 2007.

[5] V. Barnett and Lewis T. Outliers in Statistical Data. *Biometrical Journal*, 37(2), 1995.

[6] J. W. Bartlett and C. Frost. Reliability, repeatability and reproducibility: analysis of measurement errors in continuous variables. *Ultrasound in Obstetrics & Gynecology*, 31(4):466–475, 2008.

[7] Thomas Bartz-Beielstein. Designs for Computer Experiments. In *Experimental Research in Evolutionary Computation: The New Experimentalism*, pages 79–92. Springer-Verlag, 2006.

[8] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, pages 2546–2554, USA, 2011.

[9] André Biedenkapp, Joshua Marben, Marius Lindauer, and Frank Hutter. CAVE: Configuration Assessment, Visualization and Evaluation. In *Learning and Intelligent Optimization*, pages 115–130. Springer International Publishing, 2019.

[10] Mauro Birattari, Thomas Stützle, Luis Paquete, and Klaus Varrentrapp. A Racing Algorithm for Configuring Metaheuristics. In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation*, GECCO'02, pages 11–18, 2002.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, 2006.

[12] J Bland and Douglas Altman. Statistical-Methods For Assessing Agreement Between 2 Methods Of Clinical Measurement. *International journal of nursing studies*, 47, 2010.

[13] Leo Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[14] Eric Brochu, Vlad M. Cora, and Nando de Freitas. A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. *arXiv e-prints*, art. arXiv:1012.2599, 2010.

[15] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[16] Hans-Joachim Bungartz, Florian Lindner, Bernhard Gatzhammer, Miriam Mehl, Klaudius Scheufele, Alexander Shukaev, and Benjamin Uekermann. preCICE – A fully parallel library for multi-physics surface coupling. *Computers & Fluids*, 141: 250–258, 2016.

[17] R. Burger, M. Bharatheesha, M. van Eert, and R. Babuška. Automated tuning and configuration of path planning algorithms. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4371–4376, 2017.

[18] Christopher J. C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Min. Knowl. Discov.*, 2(2):121–167, 1998.

[19] Leslie Pérez Cáceres, Bernd Bischl, and Thomas Stützle. Evaluating Random Forest Models for Irace. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 1146–1153, 2017.

[20] Thomas Philip Cahill. Rough method of creating blood vessel, 2013. URL https://grabcad.com/library/blood-vessel-1. Accessed on: 2019-11-05. [Online].

[21] Paola Causin, Jean-Frédéric Gerbeau, and Fabio Nobile. Added-mass effect in the design of partitioned algorithms for fluid–structure problems. *Computer methods in applied mechanics and engineering*, 194(42-44):4506–4527, 2005.

[22] Marco Cavazzuti. Deterministic Optimization. In *Optimization Methods: From Theory to Design Scientific and Technological Aspects in Mechanics*, pages 77–102. Springer Berlin Heidelberg, 2013.

[23] Seng Chu. Hypothesis Testing with ANOVA in Python, 2019. URL https://codingdisciple.com/hypothesis-testing-ANOVA-python.html. Accessed on: 2019-12-09. [Online].

[24] Steven P. Coy, Bruce L. Golden, George C. Runger, and Edward A. Wasil. Using Experimental Design to Find Effective Parameter Settings for Heuristics. *Journal of Heuristics*, 7(1):77–97, 2001-01.

[25] Adele Cutler, D. Richard Cutler, and John R. Stevens. Random forests. In *Ensemble Machine Learning: Methods and Applications*, pages 157–175. Springer US, 2012.

[26] Rina Dechter. Stochastic Greedy Local Search. In *Constraint Processing*, The Morgan Kaufmann Series in Artificial Intelligence, pages 191–208. Morgan Kaufmann, 2003.

[27] Joris Degroote, Robby Haelterman, Sebastiaan Annerel, and Jan Vierendeels. Coupling techniques for partitioned fluid-structure interaction simulations with black-box solvers. In *10th MpCCI User Forum*, pages 82–91. Fraunhofer Institute SCAI, 2009.

[28] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding Up Automatic Hyperparameter Optimization of Deep Neural Networks by Extrapolation of Learning Curves. In *IJCAI*, 2015.

[29] Thomas Ludescher Dr. Marc Ratzel. Streamlining Aerodynamic CFD Analyses. *NAFEMS World Congress*, 2013.

[30] Rodrigo Pereira Duquia, João Luiz Bastos, Renan Rangel Bonamigo, David Alejandro González-Chica, and Jeovany Martínez-Mesa. Presenting data in tables and charts. *Anais brasileiros de dermatologia*, 89(2):280–285, 2014.

[31] Katharina Eggensperger, Marius Lindauer, Holger H. Hoos, Frank Hutter, and Kevin Leyton-Brown. Efficient benchmarking of algorithm configurators via model-based surrogates. *Machine Learning*, 107(1):15–41, 2018.

[32] Katharina Eggensperger, Marius Lindauer, and Frank Hutter. Pitfalls and Best Practices in Algorithm Configuration. *Journal of Artificial Intelligence Research*, 64 (1):861–893, 2019.

[33] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and Robust Automated Machine Learning. In *Advances in Neural Information Processing Systems 28*, pages 2962–2970. Curran Associates, Inc., 2015.

[34] Fraunhofer Institute for Algorithms and Scientific Computing. MpCCI - Research Community, 2018. URL https://www.mpcci.de/en/research-community.html#tabpanel-3. Accessed on: 2019-11-05. [Online].

[35] Fraunhofer Institute for Algorithms and Scientific Computing SCAI. MpCCI 4.5.2-1 Documentation, 2018. URL https://www.mpcci.de/content/dam/scai/mpcci/documents/MpCCIdoc-452-20180504.pdf. Accessed on: 2019-05-22. [Online].

[36] Peter I. Frazier. A Tutorial on Bayesian Optimization. *arXiv e-prints*, art. arXiv:1807.02811, 2018.

[37] Jerome Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29, 2000.

[38] Bernhard Gatzhammer. A partitioned approach for Fluid-Structure Interaction on cartesian grids. Master's thesis, Technische Universitüt München, 2008.

[39] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, Torsten Schaub, Marius Schneider, and Stefan Ziller. A Portfolio Solver for Answer Set Programming: Preliminary Report. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, pages 352–357. Springer, 2011.

[40] Jonathan Gratch and Steve Chien. Adaptive Problem-solving for Large-scale Scheduling Problems: A Case Study. *Journal of Artificial Intelligence Research*, 4(1):365–396, 1996.

[41] Jonathan Gratch and Gerald DeJong. COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-Up Learning. *Proceedings Tenth National Conference on Artificial Intelligence*, pages 235–240, 1992.

[42] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. KNN Model-Based Approach in Classification. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 986–996. Springer Berlin Heidelberg, 2003.

[43] Saskatchewan Highways and Transportation. Statistical Quality Control Principles-Standard Test Procedures Manual, 1996. URL http://www.highways.gov.sk.ca/304-3/. Accessed on: 2019-12-17. [Online].

[44] Holger H. Hoos. Automated Algorithm Configuration and Parameter Tuning. In *Autonomous Search*, pages 37–71. Springer Berlin Heidelberg, 2012.

[45] Frank Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Department of Computer Science, 2009.

[46] Frank Hutter, Domagoj Babic, Holger Hoos, and Alan Hu. Boosting Verification by Automatic Tuning of Decision Procedures. In *Formal Methods in Computer Aided Design, FMCAD*, pages 27–34, 2007.

[47] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Kevin P. Murphy. An Experimental Investigation of Model-based Parameter Optimisation: SPO and Beyond. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 271–278, 2009.

[48] Frank Hutter, Holger H Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An Automatic Algorithm Configuration Framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

[49] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Automated Configuration of Mixed Integer Programming Solvers. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 186–202. Springer Berlin Heidelberg, 2010.

[50] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Kevin Murphy. Time-Bounded Sequential Parameter Optimization. In *Learning and Intelligent Optimization*, pages 281–298. Springer Berlin Heidelberg, 2010.

[51] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential Model-Based Optimization for General Algorithm Configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer Berlin Heidelberg, 2011.

[52] Frank Hutter, Marius Lindauer, Adrian Balint, Sam Bayless, Holger Hoos, and Kevin Leyton-Brown. The configurable SAT solver challenge (CSSC). *Artificial Intelligence*, 243:1–25, 2017.

[53] Leyton-Brown K Hutter F, Hoos H. Sequential model-based optimization for general algorithm configuration (extended version). In *Technical Report. TR-2010-10*. University of British Columbia, 2010.

[54] Multi-Physics in Robotics Lab. Robots. URL https://mpirl.com/portfolio/robotics-2/. Accessed on: 2019-01-03. [Online].

[55] ASTM International. Repeatability test procedure. In *Standard Practice for Use of the Terms Precision and Bias in ASTM Test Methods*. ASTM International, 2013.

[56] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13 (4):455–492, 1998.

[57] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC – Instance-Specific Algorithm Configuration. In *Proceedings of the 2010 Conference on ECAI 2010: 19th European Conference on Artificial Intelligence*, pages 751–756. IOS Press, 2010.

[58] Dr Zara Kassam. CRISPR used to improve red blood cell transfusion compatibility, 2018. URL https://www.drugtargetreview.com/news/31427/crispr-blood-transfusion-compatibility/. Accessed on: 2019-11-05. [Online].

[59] Ashiqur R. KhudaBukhsh, Lin Xu, Holger H. Hoos, and Kevin Leyton-Brown. SATenstein: Automatically building local search SAT solvers from components. *Artificial Intelligence*, 232:20–42, 2016.

[60] M. D Koch. Quasi-Newton Methods for Unstable Partitioned Fluid-Structure Interactions. Master's thesis, Institut für Numerische Simulation, Rheinische Friedrich-Wilhelms-Universität Bonn., 2016.

[61] Martin Krasser. Bayesian Optimization. GitHub, 2018. URL https://github.com/krasserm/bayesian-machine-learning. Accessed on: 2019-11-12. [Online].

[62] Hendrik C. Kuhlmann and Francesco Romanò. The lid-driven cavity. In *Computational Modelling of Bifurcations and Instabilities in Fluid Dynamics*, pages 233–309. Springer International Publishing, Cham, 2019.

[63] O. Lima and R. Ventura. A case study on automatic parameter optimization of a mobile robot localization algorithm. In *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 43–48, 2017.

[64] Ming-Hua Lin, Jung-Fa Tsai, and Chian-Son Yu. A Review of Deterministic Optimization Methods in Engineering and Management. *Mathematical Problems in Engineering*, 2012.

[65] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, Stefan Falkner, André Biedenkapp, and Frank Hutter. SMAC v3: Algorithm Configuration in Python. GitHub, 2017. URL https://github.com/automl/SMAC3. Accessed on: 2019-05-22. [Online].

[66] Daniel Lizotte, Tao Wang, Michael Bowling, and Dale Schuurmans. Automatic Gait Optimization with Gaussian Process Regression. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 944–949, 2007.

[67] Yuri Malitsky, Ashish Sabharwal, Horst Samulowitz, and Meinolf Sellmann. Algorithm Portfolios Based on Cost-sensitive Hierarchical Clustering. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 608–614. AAAI Press, 2013.

[68] Colm McAlinden, Jyoti Khadka, and Konrad Pesudovs. Precision (repeatability and reproducibility) studies and sample-size calculation. *Journal of Cataract & Refractive Surgery*, 41(12):2598–2604, 2015.

[69] Nicolai Meinshausen. Quantile Regression Forests. *Journal of Machine Learning Research*, 7:983–999, 2006.

[70] Anke Meyer-Baese and Volker Schmid. Genetic Algorithms. In *Pattern Recognition and Signal Analysis in Medical Imaging (Second Edition)*, pages 135–149. Academic Press, second edition, 2014.

[71] Steven Minton. An Analytic Learning System for Specializing Heuristics. In *Proceedings of the 13th International Joint Conference on Artifical Intelligence - Volume 2*, IJCAI'93, pages 922–928, 1993.

[72] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 58:161–205, 1992.

[73] J. Mockus, Vytautas Tiesis, and Antanas Zilinskas. The application of Bayesian methods for seeking the extremum. In *Towards Global Optimization*, volume 2, pages 117–129. 2014.

[74] Hartmut Moeck. State–Space Search: Algorithms, Complexity, Extensions, and Applications. *Computer Standards & Interfaces*, 22(3):229 – 230, 2000.

[75] COMSOL Multiphysics. COMSOL Multiphysics Tutorial -Car Cabin Acoustics - Frequency Domain Analysis, . URL https://www.comsol.com/model/car-cabin-acoustics-frequency-domain-analysis-15013. Accessed on: 2019-11-05. [Online].

[76] COMSOL Multiphysics. Comsol multiphysics tutorial -magnetic signature of a submarine, . URL https://www.comsol.com/model/magnetic-signature-of-a-submarine-291. Accessed on: 2019-11-05. [Online].

[77] COMSOL Multiphysics. COMSOL Multiphysics Tutorial - Fluid-Structure Interaction in a Network of Blood Vessels, 2019. URL https://www.comsol.com/model/fluid-structure-interaction-in-a-network-of-blood-vessels-660. Accessed on: 2019-11-05. [Online].

[78] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Society for Industrial and Applied Mathematics, 2000.

[79] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[80] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, 3rd edition, 2009.

[81] Steven Sawyer. Analysis of Variance: The Fundamental Concepts. *Journal of Manual & Manipulative Therapy*, 17:27E–38E, 2009.

[82] Klaudius Scheufele. *Robust Quasi-Newton Methods for Partitioned Fluid-Structure Simulations*. PhD thesis, Institute of Parallel and Distributed Systems, University of Stuttgart, 2015.

[83] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.

[84] Galina Sieber. *Numerical simulation of fluid-structure interaction using loose coupling methods*. PhD thesis, Technische Universität Darmstadt, 2002.

[85] James C. Spall. *Introduction to Stochastic Search and Optimization*. John Wiley & Sons, Inc., 1 edition, 2003.

[86] L M Sykes, F Gani, and Z Vally. Statistical terms Part 1: The meaning of the MEAN, and other statistical terms commonly used in medical research. *South African Dental Journal* , 71:274–278, 2016.

[87] Hugo Terashima-Marín, Peter Ross, and Manuel Valenzuela-Rendón. Evolution of Constraint Satisfaction Strategies in Examination Timetabling. In *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 1*, GECCO'99, pages 635–642, 1999.

[88] Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 847–855, 2013.

[89] Dave A. D. Tompkins and Holger H. Hoos. Dynamic Scoring Functions with Variable Expressions: New SLS Methods for Solving SAT. In *Theory and Applications of Satisfiability Testing – SAT 2010*, pages 278–292. Springer Berlin Heidelberg, 2010.

[90] Mauro Vallati, Chris Fawcett, Alfonso Gerevini, Holger Hoos, and Alessandro Saetti. Automatic Generation of Efficient Domain-Optimized Planners from Generic Parametrized Planners. *Sixth Annual Symposium on Combinatorial Search, SoCS*, 2013.

[91] Lin Xu, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research*, 32: 565–606, 2008.

[92] Xin-She Yang. Genetic Algorithms. In *Nature-Inspired Optimization Algorithms*, pages 77–87. Elsevier, 2014.

[93] Qun Zhang and Song Cen. 13 - Multiphysics modeling for biomechanical problems. In *Multiphysics Modeling*, Elsevier and Tsinghua University Press Computational Mechanics Series, pages 363–373. Academic Press, 2016.

[94] O.C. Zienkiewicz, R.L. Taylor, and P. Nithiarasu. Fluid–Structure Interaction. In *The Finite Element Method for Fluid Dynamics*, pages 423–449. Butterworth-Heinemann, seventh edition, 2014.