Chapter 2

Preliminaries

This chapter summarizes some fundamental information that are essential for an in-depth comprehension of this thesis. Firstly, a gentle introduction to BBO is provided in Section 2.1, including state-of-the-art optimization algorithms and benchmark functions. In Section 2.2, an overview about the fine-tuning of optimization algorithms is briefly highlighted. This is followed by an insight into ELA for capturing the optimization landscape of BBO problems in Section 2.3. Lastly, conventional optimization approaches and recent advancements in solving real-world automotive crashworthiness optimization are discussed in Section 2.4.

2.1 Black-Box Optimization

Generally, BBO belongs to a class of optimization problems, where an analytic form is not available, derivative information is not reliable or lacking, and numerical approximation of the derivatives is costly [6, 9]. Without loss of generality, a continuous, single-objective BBO problem can be represented using Equation 2.1 [8]:

$$\begin{aligned} &\text{minimize } f : \Omega \subset \mathbb{R}^d \to \mathbb{R} \\ &\text{subject to } g_i(\boldsymbol{x}) \leq 0 \; \forall i \in \{1, \cdots, p\}; \boldsymbol{x} \in \Omega \,, \end{aligned} \tag{2.1}$$

where f is an objective function, g are some constraint functions, p is the number of constraints, d is the problem dimensionality, $\boldsymbol{x} = (x_1, \ldots, x_d)$ represents an optimization solution, and Ω denotes the search space. Unless otherwise stated, the optimization problems throughout this thesis are defined as minimization problems. In many real-world BBO problems, the objective and constraint functions are commonly evaluated using laboratory experiments or virtual simulations. Hence, such problems using expensive function evaluations are also known as *expensive* BBO problems.

Due to the fact that derivative information is not available, classical gradient-based optimization approaches are not applicable for BBO problems. Subsequently, various BBO algorithms have been developed over the years, which can perform exceptionally well on benchmark suites. In this regard, an overview for several optimization algorithms investigated in this thesis is provided in Section 2.1.1 and for the BBOB suite in Section 2.1.2.

2.1.1 State-of-the-art Optimization Algorithms

In recent developments of BBO algorithms, rapid advancements in derivative-free EAs and surrogate-based approaches have been witnessed [8]. Basically, most of the bio-inspired EAs are developed based on the principle of survival-of-the-fittest, where a population evolves over generations to produce better individuals through mutation and recombination. For instance, CMA-ES and Differential Evolution (DE) [128] are two powerful EAs for solving BBO problems. On the other hand, surrogate-based approaches like BO iteratively search for promising solutions through resampling, by balancing between exploration of unknown regions within the search space and exploitation of the best-found solution. Principally, these algorithms are performed in a few iterations until a termination condition is fulfilled, such as the function evaluation budget has been elapsed or the improvement in the best-found solutions of the last few iterations is marginal. In this thesis, we primarily focus on the following three BBO algorithms, namely CMA-ES, DE, and BO.

Covariance Matrix Adaptation Evolutionary Strategy

Considered by many as one of the top-performing BBO algorithms, CMA-ES was developed and extended based on Evolutionary Strategy (ES) [107, 108, 116]. In brief, the general framework of ES can be outlined as follows:

1. In the first generation, a parent population is initialized.
2. Next, new offspring are created through recombination of selected parents. To diversify the population, the offspring undergo mutation, which is crucial for escaping from being stuck in local optima.
3. Based on a selection strategy, survivals are then selected for the next generation.

4. Step 2 and 3 are repeated until a termination condition is fulfilled.

Compared to ES, CMA-ES has the advantage that the populations in previous generation are considered for adapting the mutation distribution, which is a covariance matrix [45, 46]. Combined with the information about evolution paths, correlated mutations that are invariant to search space rotations can be employed to update the covariance matrix. Furthermore, the step size in CMA-ES is modified using a cumulative step size adaptation mechanism based on the conjugate evolution paths [40], which is another crucial aspect for its empirical performance.

Since then, various extensions and variants with different improvements for CMA-ES have been progressively introduced, such as active update [51], mirrored sampling [17], threshold convergence [95], and weighted recombination [46]. Subsequently, this motivates the development of a modular framework for CMA-ES [23], which we refer to as Modular Covariance Matrix Adaptation Evolutionary Strategy (ModCMA) in this thesis. In brief, a collection of CMA-ES variants are integrated into ModCMA as modules that can be independently activated or deactivated, offering a high degree of flexibility for a custom instantiation of CMA-ES. Following this, the interactions between different CMA-ES variants as well as between variants and hyperparameters, e.g., population size and learning rates, can be conveniently investigated using ModCMA.

Differential Evolution

Apart from CMA-ES, DE is another widely-used BBO algorithm from the family of EAs, which is a popular choice for its performance, simplicity, and relatively few hyperparameters [128]. The general workflow of DE can be summarized as follows:

1. In the first generation, a population is initialized, and ideally, evenly spread across the search space.

2. Using a mutation strategy, several "donor" solutions (mutants) are created by adding the scaled difference between a pair of solutions to another solution, which are randomly selected.

3. Next, new "trial" solutions (offspring) can be generated through a recombination of donor and "target" solutions (parents).

4. Afterwards, the trial solutions compete against target solutions to determine survivals for the next generation.

5. Step 2 to 4 are repeated until a termination condition is fulfilled.

Fundamentally, the underlying structure of DE is modular by design, where the operations of mutation and recombination are independent, and hence, can be completely separated. Similar to ModCMA, a modular framework consisting of a variety of DE variants has been recently proposed, which we refer to as Modular Differential Evolution (ModDE) [142].

Bayesian Optimization

Unlike EAs, BO is a class of iterative optimization algorithms that explores and searches for better solutions through resampling promising regions within the search space, relying on approximating the true functions using surrogate models [85]. Later, BO was popularized for BBO problems with expensive function evaluations in engineering domain under the name Efficient Global Optimization (EGO) [52]. In fact, BO is widely applied for solving expensive BBO problems using a limited function evaluation budget. A comprehensive explanation of BO and its extensions, e.g., for multi-objective optimization and constraint handling, is available in [153]. For the sake of consistency, we refer to standard BO and EGO simply as BO in the remaining of this thesis. Basically, the general workflow of BO can be described as follows:

1. A set of initial Design of Experiments (DoE) samples is generated using a sampling method, such as Latin Hypercube Sampling (LHS) [81]. Ideally, the DoE samples are evenly distributed across the design space, for a better approximation of the true functions using surrogate models.

2. Based on the DoE samples, a surrogate model is constructed to approximate the true optimization functions with uncertainties. In practice, Gaussian Process Regression (GPR) [61] model is commonly employed as surrogate model in the standard BO approach.

3. By exploiting the information provided by the surrogate model, an acquisition function is utilized to determine promising candidates to be evaluated, which is also known as resampling. Essentially, the acquisition function is maximized through an internal optimization within the BO framework. Some popular choices of acquisition function are Expected Improvement (EI) [52], Probability of Improvement (PI) [84], Lower Confidence Bound (LCB), and Upper Confidence Bound (UCB) [124], each offering a different search strategy in balancing between the explorative and exploitative behavior of BO algorithm.

4. After evaluating the suggested candidates, the new solutions are added to the set of DoE samples. In expensive BBO problems, the candidates are evaluated using expensive function evaluations.

5. Step 2 to 4 are repeated until a termination condition is fulfilled.

One of the weaknesses of BO is that its performance suffers from the curse of dimensionality and tends to deteriorate for high-dimensional problems. Following this, different variants of BO have been introduced over the years to better handle high-dimensional problems, such as Principal Component Analysis assisted Bayesian Optimization (PCA-BO) [106], Kernel Principal Component Analysis assisted Bayesian Optimization (KPCA-BO) [5], Trust Region Bayesian Optimization (TuRBO) [32], Sparse Axis-Aligned Subspace Bayesian Optimization (SAASBO) [31], and Heteroscedastic Evolutionary Bayesian Optimization (HEBO) [21]. An extensive analysis of these BO variants is provided in [113]. Recently, it has been shown that slightly adjusted standard BO can achieve improved performance in high dimensionality, which calls for further investigations [48].

Further Optimization Algorithms

Apart from the aforementioned algorithms, we also briefly employ Genetic Programming (GP) to guide the function evolution towards specific optimization problem classes in this thesis (Section 4.3). Inspired by neo-Darwinian evolution [64], the workflow of standard GP is similar to other EAs, i.e., by evolving a population based on the principle of survival-of-the-fittest and biologically-inspired operators. Previously, GP has been commonly applied for solving symbolic regression problems [130], where GP is employed to construct an explicit mathematical expression based on a given dataset. Following this, human-interpretable surrogate models can be build to analyze the relationship between different decision variables of BBO problems.

Beyond that, we also utilize several other BBO algorithms in this thesis, such as PSO [55], Estimation of Multivariate Normal Algorithm (EMNA) [66], Constrained Optimization by Linear Approximation with Random Restart (RCobyla) [97], Simultaneous Perturbation Stochastic Approximation (SPSA) [123], NGOpt [104], and Random Search (RS). Comprehensive overviews of further state-of-the-art BBO algorithms are available in [6, 8, 9, 10, 96].

2.1.2 Black-Box Optimization Benchmarking Suite

With the exponential growth of BBO algorithms over the years, a fair performance assessment and comparison between them heavily relies on suitable benchmark suites that cover a diverse set of optimization problem classes. For example, the BBOB family of problem suites are some of the most utilized sets of problems considered for benchmarking optimization heuristics [41]. Initially proposed in [44], the BBOB suites have been integrated into the Comparing Continuous Optimizers (COCO) platform [43] and Iterative Optimization Heuristic (IOH) profiler [28]. As one of the most established platforms for benchmarking purposes, COCO facilitates a comparison between algorithms by storing detailed performance statistics, which can be easily retrieved, processed, and compared against a constantly expanding set of publicly accessible data [42]. Due to their popularity, the BBOB suites have become a popular testbed for the benchmarking of algorithm configuration fine-tuning techniques, even though they were never originally designed for this purpose.

In this thesis, we focus on the BBOB suite that consists of 24 single-objective, noiseless, and continuous functions, which we refer to as *the* BBOB functions. According to their high-level properties [82, 83], the 24 BBOB functions can be separated into five main groups, namely (i) separable problems, (ii) low or moderate conditioned problems, (iii) high conditioned and unimodal problems, (iv) multi-modal problems with adequate global structure, and (v) multi-modal problems with weak global structure, as presented in Table 2.1.

One of the main advantages of the BBOB suite is that the functions can be scaled to arbitrary dimensionality, facilitating an investigation of different problem dimensionalities. Furthermore, arbitrarily many problem instances or variants of BBOB functions can be generated through transformations of both the search space and objective values, which are controlled internally by a unique identifier, also known as instance ID (IID). Principally, problem instances of the same BBOB function belong to the same optimization problem class. For most of the BBOB functions, the search space transformations consist of a combination of translation and rotation matrices. Since the objective values can be transformed as well, the optimization performance metrics commonly used are relative to the global optimum, allowing a comparison between different instances. While the BBOB suite was originally intended to be used for unconstrained optimization, in practice, most investigations based on the BBOB functions focus on the search space $[-5, 5]^d$, where the global optimum is located inside $[-4, 4]^d$.

Table 2.1: Classification of 24 BBOB functions into five groups (separated by horizontal lines) based on their high-level properties, such as multi-modality, global structure, separability, variable scaling, homogeneity, basin-sizes, and global to local contrast. *From top to bottom:* Separable problems (F1–F5), low or moderate conditioned problems (F6–F9), high conditioned and unimodal problems (F10–F14), multi-modal problems with adequate global structure (F15–F19), and multi-modal problems with weak global structure (F20–F24). Table taken from [82].

	BBOB Function	**multim.**	**gl.-struc.**	**separ.**	**scaling**	**homog.**	**basins**	**gl.-loc.**
F1	Sphere	none	none	high	none	high	none	none
F2	Ellipsoidal separable	none	none	high	high	high	none	none
F3	Rastrigin separable	high	strong	none	low	high	low	low
F4	Bueche-Rastrigin	high	strong	high	low	high	med.	low
F5	Linear Slope	none	none	high	none	high	none	none
F6	Attractive Sector	none	none	high	low	med.	none	none
F7	Step Ellipsoidal	none	none	high	low	high	none	none
F8	Rosenbrock	low	none	none	none	med.	low	low
F9	Rosenbrock rotated	low	none	none	none	med.	low	low
F10	Ellipsoidal high-cond.	none	none	none	high	high	none	none
F11	Discus	none	none	none	high	high	none	none
F12	Bent Cigar	none	none	none	high	high	none	none
F13	Sharp Ridge	none	none	none	low	med.	none	none
F14	Different Powers	none	none	none	low	med.	none	none
F15	Rastrigin multi-modal	high	strong	none	low	high	low	low
F16	Weierstrass	high	med.	none	med.	high	med.	low
F17	Schaffer F7	high	med.	none	low	med.	med.	high
F18	Schaffer F7 mod. ill-cond.	high	med.	none	high	med.	med.	high
F19	Griewank-Rosenbrock	high	strong	none	none	high	low	low
F20	Schwefel	med.	deceptive	none	none	high	low	low
F21	Gallagher 101 Peaks	med.	none	none	med.	high	med.	low
F22	Gallagher 21 Peaks	low	none	none	med.	high	med.	med.
F23	Katsuura	high	none	none	none	high	low	low
F24	Lunacek bi-Rastrigin	high	weak	none	low	high	low	low

Recently, effort has been invested to further diversify the benchmark problem classes available in the BBOB suite using an affine combination of two selected BBOB functions [26]. In brief, new benchmark functions can be generated through interpolation between the selected BBOB functions, by employing a weighting factor to control the shifting between them. Later, this function generation approach was generalized to affine combinations of multiple BBOB functions, also known as Many-Affine Black-Box Optimization Benchmarking (MA-BBOB) functions [147, 148]. In other words, the affine combination mechanism has been extended to a combination of multiple BBOB functions, thereby allowing the generation of more complex functions.

Beyond that, the so-called Instance Space Analysis (ISA) [121] is another exciting research direction, attempting to improve the diversity of optimization problem classes provided by existing benchmark suites. Essentially, the instance space covered by benchmark suites is analyzed based on some feature-based representations of the problem instances, to identify poorly-covered regions within the instance space. Oftentimes, ISA is performed using a performance-oriented perspective w.r.t. a set of optimization algorithms, ensuring that the newly created functions are discriminative in terms of algorithm performances. For instance, GP can be employed for the generation of new benchmark functions to fill the gaps based on some ELA features [88].

2.2 Fine-tuning of Algorithm Configuration

Since the performance of BBO heuristics highly depends on their hyperparameter setting, an optimal configuration is necessary to achieve the best possible optimization performance w.r.t. the best-found solution, time, and computational resources. In short, the terminology *hyperparameter* refers to a parameter of optimization algorithms that can be defined by practitioners to control the search behavior during optimization runs, such as the learning rates in CMA-ES. The process of identifying optimal algorithm configurations is also commonly known as Hyperparameter Optimization (HPO) [37]. Apart from fine-tuning only the hyperparameters, HPO can be combined with AS, becoming Automated Algorithm Configuration (AAC) or Combined Algorithm Selection and Hyperparameter Optimization (CASH) [133]. Essentially, the main purpose of CASH is to identify the most suitable optimization algorithm from an algorithm portfolio, while simultaneously fine-tuning its hyperparameters. In literature, these terminologies are sometimes used interchangeably, since a clear definition is still lacking. Based on the descriptions provided in [114], we consider the following definitions in this thesis for the sake of consistency:

AS: Selecting an optimal algorithm from an algorithm portfolio, without fine-tuning its hyperparameters.

HPO: Fine-tuning the hyperparameters of a particular optimization algorithm.

AAC / CASH: Selecting an optimal algorithm and fine-tuning its hyperparameters.

In this context, we also refer to the term *algorithm configuration* as a set of hyperparameter settings of an optimization algorithm. To the best of our knowledge, existing HPO approaches typically require a large function evaluation budget, and hence, are less applicable for real-world expensive BBO problems.

Since there is no such a one-for-all optimizer that can perform well on all BBO problems [156], HPO is vital to fine-tune optimization algorithms for maximal optimization efficiency. Nevertheless, this task can be extremely challenging, particularly for practitioners with limited experience and domain knowledge in HPO. Consequently, various optimizers have been developed over the years to facilitate an automated HPO process, such as Tree-structured Parzen Estimator (TPE) [13], Sequential Model-based Algorithm Configuration (SMAC) [69], Iterated Racing (irace) [78], and Mixed-Integer Parallel Efficient Global Optimization (MIP-EGO) [140], where the hyperparameter search space typically consists of continuous, integer, categorical, and conditional variables. Within the scope of this thesis, we primarily focus on the surrogate model-based HPO approaches, namely TPE and SMAC.

TPE: As a variant of the BO algorithm, TPE has a similar framework as the standard BO, but utilizes Parzen estimators as surrogate models, which can handle mixed-integer search space and scale well to high dimensionality. For example, TPE has been previously applied to fine-tune the learning rates of CMA-ES [159].

SMAC: To handle the mixed-integer search space, Random Forest (RF) models are utilized in SMAC as surrogate models. Subsequently, the mean and variance computed from the individual trees in the forest can be considered as uncertainty of RF models, which is necessary for the computation of acquisition functions, as described in Section 2.1.1.

Typically in HPO, the optimal algorithm configuration identified is applied and fixed throughout an optimization run. As optimization runs progress, nevertheless, this algorithm configuration might not always be optimal. In other words, a better optimization performance could be achieved using other configurations. Hence, recent research about HPO has been extended towards dynamic algorithm configuration, where configurations are fine-tuned and adapted on the fly during an optimization run [1, 145], which is beyond the scope of this thesis.

2.3 Exploratory Landscape Analysis

In evolutionary computation, substantial work has been invested in landscape-aware ASP for the fine-tuning of algorithm configuration based on the landscape characteristics of BBO problems. Subsequently, this encourages the development of abundant techniques for fitness landscape analysis [80]. As one of the well-established landscape analysis methods, ELA facilitates a numerical quantification of the high-level properties of an optimization landscape [83], such as global structure, multi-modality, and separability (refer to Table 2.1), based on some carefully designed low-level features. Apart from the six fundamental classes of low-level features originally proposed in ELA, namely y-distribution, level set, meta-model, local search, curvature, and convexity [82], more complementary features have been gradually included [59], such as dispersion [79], Nearest Better Clustering (NBC) [56], Principal Component Analysis (PCA), linear model, and Information Content of Fitness Sequences (ICoFiS) [89]. Previously, it has been shown that ELA features are indeed sufficiently informative for a reliable classification of the BBOB functions [111], for AS purposes [14, 49, 58, 90], and for HPO tasks [12, 27].

Essentially, ELA features can be computed using a set of DoE samples $\mathcal{X} = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n\}$ and the corresponding objective values $\mathcal{Y} = \{y_1, \cdots, y_n\}$, where $f \colon \mathcal{X} \to \mathcal{Y}$, $\boldsymbol{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$, and n represents the DoE sample size. Consequently, the computation of ELA features is dependent on factors, such as DoE sample size, sampling strategy, and problem dimensionality [87, 152]. In fact, concerns regarding the following aspects of ELA features have been previously raised, namely:

- Many of the ELA features are highly correlated and redundant, particularly for those belonging to the same feature class [150];
- Some of the ELA features are insufficiently expressive in distinguishing different problem instances [110];
- The hand-crafted ELA features might suffer from potential bias in capturing certain optimization landscape characteristics [118]. Furthermore, since ELA was developed and evaluated mainly using the BBOB suite, the design and problem classes of BBOB functions might guide the development of ELA to

some extent, raising the questions as to whether ELA can generalize well to problem classes beyond the BBOB suite; and

- ELA features are in general less discriminative for high-dimensional problems [86].

Attempting to overcome the drawbacks of ELA features, feature-free landscape analysis approaches based on latent space representations computed using deep-learning techniques have been actively explored, such as Deep-ELA [118, 119] and DoE2Vec [139] (Section 3.1.2). Further relevant work is available in [2, 99, 102]. Compared to the classical ELA features, however, an interpretation and understanding of these latent space representations are not as straightforward.

Apart from handling automated AS tasks, ELA has shown promising potential in a wide range of applications. For instance, ELA has been applied to understand the optimization landscape of neural architecture search tasks [141], to analyze the problem space of different benchmark problem sets [150], to study multi-objective optimization problems [57], and to examine the landscape properties of engineering problems like vehicle dynamics control systems [131]. Furthermore, a landscape-aware fine-tuning of the acquisition function in BO algorithm based on ELA features has been recently investigated in [12]. As far as we know, no relevant work has been previously attempted to analyze the optimization landscape characteristics of real-world expensive BBO problems, such as automotive crashworthiness optimization.

2.4 Automotive Crashworthiness Optimization

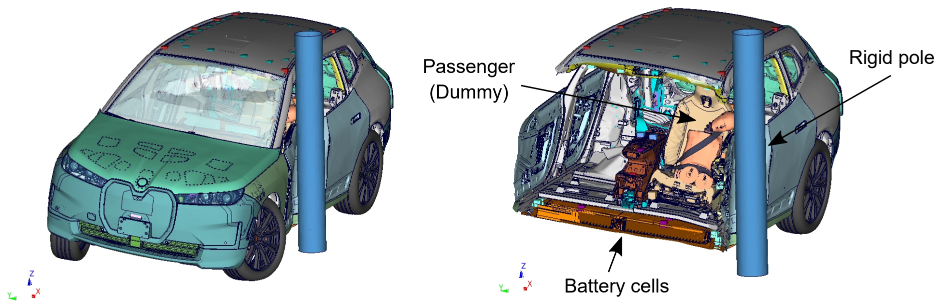
In recent years, the competitive landscape in the automotive industry has changed drastically due to the emergence of competent automobile manufacturers worldwide and the growth of market preference toward Battery Electric Vehicle (BEV). To stay competitive and relevant in the market, automobile manufacturers are facing enormous pressure to improve different aspects, such as enhancing the vehicle designs, broadening the spectra of vehicle models, shortening vehicle development cycles, and effectively lowering production costs [29]. For instance, vehicle designs nowadays are becoming increasingly sophisticated due to the stricter regulations on road safety imposed by authorities and the fact that various features and functionalities must be integrated to better retain customer satisfaction in terms of safety, comfort, driving experience, and entertainment. Subsequently, vehicle development demands a close cooperation between multiple disciplines, such as crashworthiness, structural statics, driving dynamics, and acoustics [18, 47]. Hence, an efficient strategy for the vehicle designing process during the early development phase is crucial to optimize time to market and to minimize investment costs. In this context, optimization approaches based on virtual simulations have been heavily integrated to identify vehicle designs that can optimally fulfill requirements imposed by different disciplines.

One of the ongoing, yet demanding research topics regarding vehicle design is the optimization of vehicle design w.r.t. crashworthiness, which is also commonly known as crashworthiness optimization. Belonging to the group of expensive BBO problems, automotive crashworthiness optimization problems are usually solved using numerical FE simulation runs, as shown in Figure 2.1 and Figure 2.2. Within the scope of this thesis, we focus on automotive side crash as a representative crash scenario, where the battery cells in BEV must be additionally protected from serious damages during crash. Basically, the main goal of automotive crashworthiness optimization is to identify vehicle designs that can provide sufficient protection to passengers in the event of a crash, while fulfilling other requirements, such as being durable and lightweight to reduce fuel consumption and production costs. Considering that crash problems are oftentimes strongly nonlinear, discontinuous, and high-dimensional, and the fact that FE simulations are computationally costly, solving automotive crashworthiness optimization can be tremendously challenging and time-consuming.

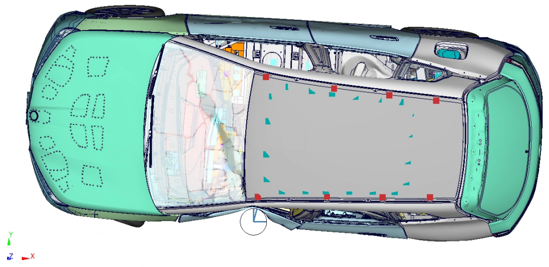
2.4.1 Surrogate-based Optimization Approaches

Classically, an automotive crashworthiness optimization problem is solved using the one-shot optimization approach, where the best vehicle design in a set of DoE samples is simply considered as the optimization solution [15, 16]. The main advantage of this approach is that multiple vehicle designs can be evaluated at once using parallel simulation runs.

Extended from the one-shot optimization approach, the Response Surface Method (RSM) has gained popularity in the automotive industry and is widely applied for automotive crashworthiness optimization, where response surfaces or data-driven surrogate models are employed to find optimal vehicle designs [35]. In short, surrogate models like polynomial function, Radial Basis Function (RBF), and GPR models are trained using a set of DoE samples to approximate the true optimization problems. With sufficiently high approximation accuracy, solutions better than the best DoE sample can be identified using RSM. Over the years, substantial work has been invested to improve the effectiveness of RSM for solving vehicle design problems [34, 93].



(a) Full-view (*left*) and cross-sectioned view (*right*) of the deformed vehicle during crash impact. To protect passengers and battery cells in BEV, the crash impact energy must be sufficiently absorbed through plastic deformation of different components in the crumple zones.



(b) Top view of the deformed vehicle. Depending on the investigation purposes, the position of side pole can be adjusted alongside the vehicle body.

Figure 2.1: Example of FE model developed for an automotive crashworthiness optimization problem with side impact against a rigid pole. Figures taken from [72, 73].

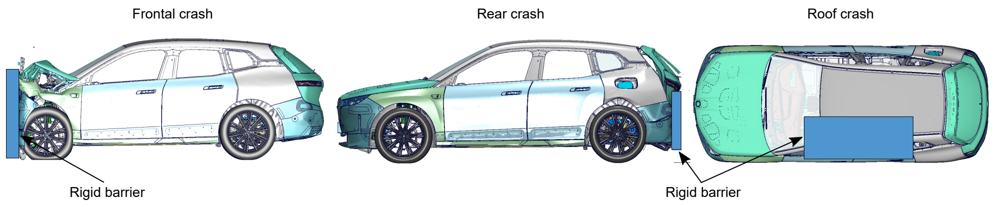


Figure 2.2: Side view of FE models developed for vehicle frontal crash (*left*), rear crash (*middle*), and roof crash (*right*) against a rigid barrier. The roof crash scenario, for instance, is designed for investigating a vehicle rollover or an overturn situation during crash.

Based on RSM, Sequential or Successive Response Surface Method (SRSM) [62, 125, 127] was developed to identify better solutions within the design space through an iterative resampling, as shown in Figure 2.3. Essentially, a new response surface is constructed to approximate the local optimization problem within a subregion of the search space in each iteration. Based on the best-found solution in each iteration, the subregion gradually shrinks and progressively moves towards regions with better solutions. Subsequently, the fitting quality of response surfaces can be improved by successively reducing the size of subregion. On top of that, engineering expertise and domain knowledge can be additionally provided to guide the adaptation of subregion towards regions that are expected to yield better solutions, potentially accelerating the solving of automotive crash problems. Nevertheless, this process is challenging, as such information might be unintentionally biased towards certain assumptions and/or experiences from the past might be less relevant for new vehicle designs. Despite SRSM has demonstrated promising potential in solving crashworthiness optimization problems [65], its application in the automotive industry is still rather limited, e.g., due to the poorly fitted response surfaces, substantial computational expenses using an iterative resampling, and challenges in sharing information between iterations [35].

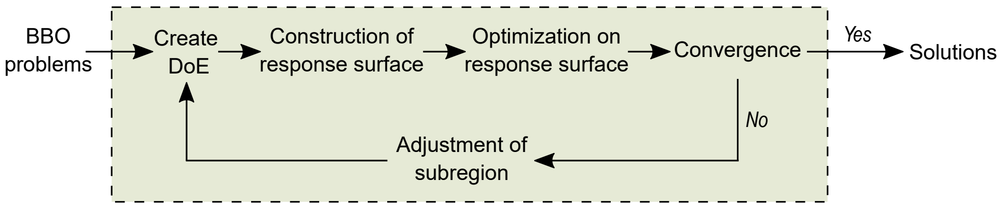


Figure 2.3: Example of typical optimization workflow based on SRSM for BBO problems.

Over the years, a variety of design optimization tools has been developed and offered to solve engineering BBO problems in the automotive industry, such as `Optimus` (Noesis Solutions) [92], `HyperStudy` (Altair) [3], `LS-OPT` (DYNAmore) [30], `optiSLang` (Ansys) [4], and `HEEDS` (Siemens) [120]. While a detailed explanation about the optimization approaches available in these commercial tools is not available due to confidentiality reasons, most of them are implemented based on RSM and/or SRSM with modifications, according to the best of our understanding. Following this, we would expect that their optimization performances would be similar or with some improvements, compared to standard RSM and SRSM.

Due to the rapid developments in vehicle designing process, both RSM and SRSM are gradually losing their effectiveness in solving complex automotive crashworthiness optimization problems, which can be briefly summarized as follows:

Low fitting quality: The performance of RSM and SRSM is heavily dependent on the fitting quality of response surfaces in approximating the true optimization problems. Since vehicle designs are getting evermore complicated, the strongly nonlinear and discontinuous characteristics in crashworthiness optimization usually lead to a poorly fitted response surface, and eventually, a sub-optimal solution; and

Expensive simulations: As the complexity of vehicle FE models rises exponentially over the years, the solving time and computational resources required for FE simulations have also increased significantly, limiting the FE simulation runs and optimization iterations that can be performed. Consequently, it is often infeasible in the automotive industry to improve the fitting quality of response surfaces in RSM and SRSM by increasing the DoE sample size.

Meanwhile, the BO algorithm that excels for solving expensive BBO problems using a small function evaluation budget (Section 2.1.1) has shown promising potential in solving automotive crashworthiness optimization problems [39, 129]. Despite of that, the problem dimensionalities investigated were relatively low, i.e., in 5-d and 6-d respectively. As the complexity of vehicle design increases nowadays, a higher problem dimensionality must be considered for optimization, e.g., commonly between 15-d and 25-d. Moreover, the hyperparameters of BO algorithm must be properly fine-tuned for an effective optimization performance, e.g., the choices of surrogate model and acquisition function, which can be challenging for practitioners lacking in domain knowledge.

In summary, optimization approaches that can effectively solve crashworthiness optimization problems have gained growing attention in the automotive industry. Beyond that, recent optimization approaches focus on relieving practitioners from the monotonous and exhausting manual tasks associated with conventional optimization approaches. This can be achieved, for example, by integrating artificial intelligence into the optimization workflow. Subsequently, practitioners can redirect their focus and resources towards enhancing other aspects of the vehicle designing process, such as a better definition of the vehicle design problem.

2.4.2 Alternative Optimization Approaches

Apart from the conventional surrogate-based optimization approaches, various alternatives have been actively explored for solving complex automotive crashworthiness optimization problems. Some of the related recent advancements are presented in the following:

- While population-based optimization heuristics, such as CMA-ES, DE, and PSO, are powerful in solving BBO problems, a large function evaluation budget is usually required for an optimization convergence. Unlike benchmark suites, where a budget of $10\,000 \cdot d$ evaluations is still relatively affordable for an optimization run, such an evaluation budget is infeasible for expensive BBO problems. Following this, these algorithms are currently less practical for real-world expensive BBO problems like automotive crashworthiness optimization [158]. Meanwhile, progressive developments have been attempted to overcome the limitation of population-based algorithms for applications in expensive BBO domains [68];
- Multi-fidelity optimization [54] and model order reduction [22, 67] are some other active research areas concerning crashworthiness optimization. Basically, BBO problems are solved at a reduced cost, by utilizing low-fidelity models with an acceptable trade-off between modeling accuracy and computational resources. While these techniques have shown good potential in solving simple crashworthiness optimization problems, an application for complex vehicle designs in the automotive industry is still rather limited;
- Another intriguing research direction is to leverage the potential of transfer learning approaches for automotive crash optimization, by learning and transferring knowledge from the past to future vehicle design problems [20]. One of the main advantages of this learning-based approach is that the historical data and information can be exploited to guide the development of future vehicle designs, potentially reducing the need for expensive simulation runs. Nevertheless, this technique is still in its early developing phase and further investigations are necessary for real-world applications in the automotive industry; and
- From the perspective of numerical modeling, the computation time and resources necessary for simulation runs can be effectively reduced through some simplifications of FE models [155]. In a so-called FE submodel, only parts of a vehicle that are relevant for the crash dynamics are retained, while the rest can be replaced using appropriate substituting mass and boundary conditions. Following

this, the computational resources necessary for a simulation run can be greatly reduced, allowing more simulation runs to be performed. While the usage of FE submodels can be beneficial in reducing computational effort, they might be less practical for complex vehicle design problems, especially when the crash dynamics can be hardly represented and/or the additional effort needed for developing submodels are not well worth it.